Dynamic Modeling, Simulation, and Control of a Series Resonant Converter with Clamped Capacitor Voltage

by

Terrence Tian-Jian Ho

B.E., Electrical Engineering, The Cooper Union (1992)

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

Master of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 1994

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
January 14, 1994

Certified by .... [illegible] ....
George C. Verghese
Professor of Electrical Engineering
Thesis Supervisor

Accepted by .... [illegible] ....
Frederic Morgenthaler
Chairman, Departmental Committee on Graduate Students

Dynamic Modeling, Simulation, and Control of a Series Resonant Converter with Clamped Capacitor Voltage

by

Terrence Tian-Jian Ho

Submitted to the Department of Electrical Engineering and Computer Science
on January 14, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Abstract

A modified series resonant converter with clamped tank capacitor voltage exhibits complex dynamic characteristics due to the clamping diodes. This thesis aims to understand the dynamic behavior of the converter in a three-dimensional state-space. Geometric features of the state trajectories are analyzed and sampled-data models are developed. Based on the small-signal model around nominal operation, a feedback controller is designed. Nonlinear control rules are established to deal with large deviations from the nominal. The study is supported by simulation results obtained using a specially constructed simulator.

Thesis Supervisor: George C. Verghese
Title: Professor of Electrical Engineering

Acknowledgments

First and foremost, I wish to express my deepest gratitude to my thesis advisor, Professor George Verghese. He gave me guidance when I needed direction. He gave me ideas when I was lost. He gave me encouragement when I was about to give up. His knowledge, his patience, and his energy have helped to make the arduous process of completing a thesis a bit less trying and a bit more pleasant for me. His advice went beyond the mere technical issues. His friendship I enjoy and appreciate tremendously. For all these, I am grateful.

I would like to thank Mr. Chiharu Osawa who started the project. His initial research was extremely valuable in laying the ground work upon which my research was built. He gave me his precious time to introduce me to the project, all during the busiest hours of his final preparation of his own thesis and return to Japan.

I would like to thank Dr. Boris Jacobson at Raytheon who partially funded my research. Our discussions provided much insight.

I would like to thank the National Science Foundation for its generous financial support. It is my honor and good fortune to receive the NSF Graduate Fellowship.

I would like to thank all members of LEES for providing a friendly working environment.

I sincerely thank my friends for being their great selves. Their diligence, their optimism, their confidence, and their energy have not only impressed me but also motivated and transformed me. They have shown me, by example, the limitless possibilities of human achievements. From them, I have learned to live a fuller, better, more balanced, and more care-free life. I feel truly fortunate to be in company of such capable a group of people.

I am forever indebted to my parents whose genuine love and unwavering support are seldom professed through words but always quietly delivered through actions. My debt to them is one that cannot be repaid in a lifetime. If any of my meager accomplishments can bring them some joy, then I shall dedicate this thesis to them.

To My Parents,

Whose Love and Support

Will See Me Through It All

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

There are many variations in the design of resonant converters, but most designs share the same operating principles. The switches in the resonant converter generate a square wave ($+E$ and $-E$) or a quasi-square wave ($+E$, 0, and $-E$) from a dc voltage source (E). This voltage waveform is applied across a resonant LC circuit tuned to approximately the switching frequency to filter out the unwanted harmonics. Output power may be controlled by adjusting the switching frequency, since the gain drops off as the switching frequency moves away from the resonant frequency. If a quasi-square wave is used as the input to the LC circuit, power can also be controlled by adjusting the duty ratio of the quasi-square wave. One common application of resonant converters is in high-frequency dc/dc power supplies. The ac current through the LC circuit is rectified and low-pass filtered to produce dc. An isolation transformer may precede the rectification.

One advantage of resonant converters is lower switching losses at high switching frequency as compared to other types of dc/dc converters, since the switching can be done when the current or the voltage is nearly zero. The higher operating frequency reduces the size of energy storage components in the power converter and thus the size of the converter itself. The disadvantage, on the other hand, is that the on-state currents and off-state voltages are higher in the switching devices.

A related disadvantage of the resonant converter is the high peak capacitor voltage in series resonant converters or the high peak inductor current in parallel resonant converters. In the series case, with increasing LC circuit filter selectivity as measured by the quality factor, Q, the peak capacitor voltage can be Q times the source voltage. A similar situation exists for the inductor current in the parallel case.

To combat this undesirable feature, the series resonant converter (SRC) under investigation in this project has four clamping diodes placed around the resonant tank capacitor to ensure that the voltage across the resonant tank capacitor can never exceed the input voltage. The inductor and the primary side transformer are split into two sections. This converter was developed at Raytheon by Jacobson and DiPerna, [2]. The introduction of the clamping diodes was intended to improve the performance of the converter under heavy load conditions. However, it resulted in a large variety of possible operating modes, some of which may not be desirable in normal operation of the converter. On the other hand, it also presented us with the challenge of understanding the complex dynamics of the SRC and thereby developing better feedback control.

Earlier numerical simulations of the steady-state behavior of the SRC based on circuit models were carried out by Raytheon [2] and by Kato and Verghese [4, 5]. They had demonstrated the complexity of the boundaries between operating modes of the circuit. Simulations based on state-space models and aimed at analyzing the dynamic behavior of the SRC were done by Osawa in his Master's thesis [7]. His simulation results in general agreed with what Raytheon and Kato had done, but a few discrepancies in the region of discontinuous conduction mode, for example, remained. The behavior of the state-space trajectories was analyzed. It was shown that only four of the nine diode conduction configurations, or topological modes, need to be analyzed. The others can be derived from these four because of symmetries in the state-space models. Of the nine possible topological modes, one was analyzed in detail in [7].

1.2 Objectives

The chief goal of this thesis is to obtain a more complete understanding of the dynamic and steady-state behavior of the SRC in the face of the added complexity of operating modes due to the clamping diodes. The analysis of the state-space trajectories is to be aided with a computer simulation program, specially tailored for this circuit in order to achieve a high degree of accuracy, speed, and flexibility.

The properties of the SRC around the nominal operating point are to be examined through derivation and analysis of sampled-data models. A feedback controller will be designed utilizing classical control methods. The goal here is to deliver constant output power to the load and to maintain steady output voltage in the presence of slow fluctuations in supply voltage, or other such disturbances, and modeling uncertainties.

1.3 Thesis Organization

We will start with a brief overview of resonant converter circuits and an introduction to the basic operation of the series resonant converter in Chapter 2. We will also discuss the justification for some simplifications in the circuit model of our SRC.

In Chapter 3, the different topological modes of the SRC, their boundary conditions, and their state-space equations are stated. Also shown are some symmetry properties that may simplify circuit analysis.

In Chapter 4, the trajectories are analyzed through their velocity fields, which provide some simple but limited understanding, especially when the motion is confined to a plane. The velocity fields are less helpful for more complicated motions, due to difficulty in visualization.

Trajectory geometry is further analyzed in Chapter 5. Because of the simple structure of the state-space equations, closed-form equations of the trajectories can be easily derived. Descriptions of the trajectories in the various modes are presented.

Steady-state operating characteristics are described in Chapter 6. The selection of a nominal operating point focuses the small-signal analysis to one specific condition.

Analytical models — large-signal and small-signal sampled-data models in particular — are derived and analyzed in Chapter 7. The results are obtained with the aid of the symbolic computation capabilities of Maple, and verified by simulation.

A controller design is presented in Chapter 8. Simulation results of the closed-loop system under various operating conditions are examined.

Chapter 9 gives a description of the structure and the design of the simulation program, which is written in Matlab. Key features and a guide to the simulation programs are outlined.

Chapter 2

SRC Circuit Operations and Simplifications

2.1 A Brief Overview of Series Resonant Circuits

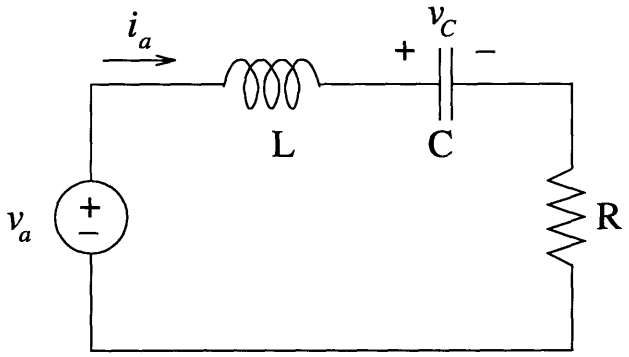
It is worthwhile to take a look at a simple series resonant converter (SRC) first, to have some basic idea of how it works. A more complete and detailed treatment on this topic can be found in Chapter 9 of [3].

Figure 2-1(a) shows the basic topology of the series resonant converter. The admittance of the series RLC circuit seen by the source is

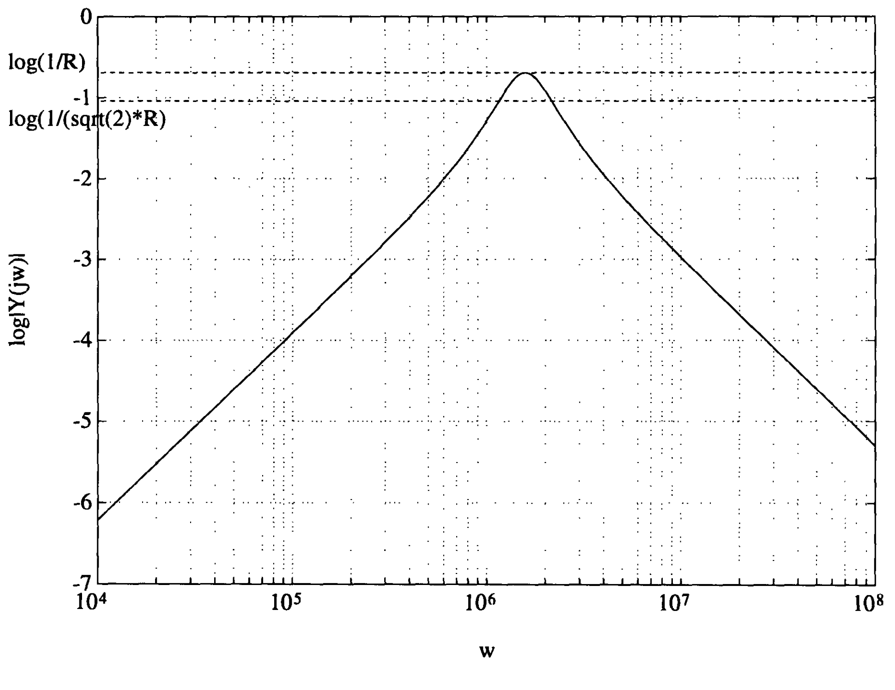
$$Y(s) = \frac{1}{\left(sL + \frac{1}{sC} + R\right)} = \frac{sC}{s^2LC + s\,RC + 1} = \frac{1}{R}\frac{2\,a\,s}{(s^2 + 2\,a\,s + \omega_0^2)} \tag{2.1}$$

where $\omega_0 = \sqrt{1/LC}$ is the resonant frequency and $2a = R/L$ is the width of the half-power or 3 dB points. The quality factor, $Q = \omega_0/2a$, is a normalized measure of the filter's selectivity. The higher the value of Q, the sharper the frequency response, $|Y(j\omega)|$. The magnitude of the frequency response of a sample $Y(j\omega)$ with $L = 2\mu\text{H}$, $C = 0.2\mu\text{F}$, and $R = 2\Omega$, is shown in Figure 2-1(b). The quality factor for this set of parameters is

$$Q = \sqrt{\frac{1}{LC}} \cdot \frac{L}{R} = 1.58 \tag{2.2}$$



(a)



(b)

Figure 2-1: A simple series resonant circuit: (a) basic topology, and (b) magnitude of the admittance $Y(j\omega)$ with $L = 2\mu$H, $C = 0.2\mu$F, and $R = 2\Omega$.

At resonance, $s = j\omega_0$, the admittance $Y(j\omega)$ is $1/R$. The load voltage across the resistor is equal to the source voltage. The capacitor voltage, however, is Q times the source at the resonant frequency. The transfer function from v_a to v_C is

$$\frac{V_C(s)}{V_a(s)} = \frac{1}{s\,C}Y(s) = \frac{1}{s^2LC + s\,RC + 1} = \frac{\omega_0^2}{(s^2 + 2\,a\,s + \omega_0^2)} \tag{2.3}$$

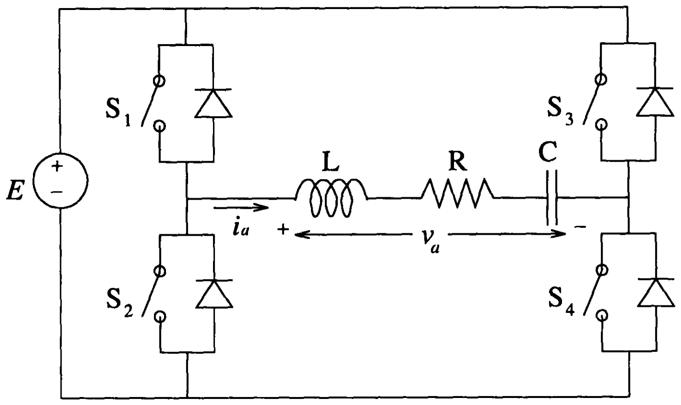
At resonance, where $\omega = \omega_0$

$$\left|\frac{V_C}{V_a}\right| = \frac{\omega_0}{2a} = Q \tag{2.4}$$

Even for a modest Q, the peak capacitor voltage, or the peak inductor current in the parallel resonant converter case, can be excessively high. This is one major disadvantage of resonant converters. The series resonant converter under study in this thesis uses diodes to clamp the capacitor so that the peak capacitor voltage is limited to within $\pm$ the source voltage. This, however, makes the dynamic behavior of the circuit more complicated, as we will see in later chapters.

When the input voltage is a square wave with an operating frequency close to ω_0, and if Q is relatively large, the filtering of the harmonics by the LC circuit is effective. Since the magnitude of $Y(j\omega)$ drops as we move away from the resonant frequency, output power can be controlled by adjusting the switching frequency. This technique presents two disadvantages. How far the switching frequency can be varied is limited by switch limitations or the presence of the third harmonic. If the Q of the circuit is not very high, the range of control is limited since a relatively large change in the switching frequency is necessary to achieve a relatively small change in output power.

An alternative approach to frequency modulation for output power control is phase modulation. A quasi-square wave, instead of a square wave, constitutes the source across the resonant circuit. This can be achieved by placing the RLC circuit inside a full bridge, as shown in Figure 2-2. The SRC analyzed in this thesis is a variation of this bridge topology, as we shall see later. Power control is achieved by varying the duration $(2\,\alpha)$ for which the input voltage is 'clamped' at zero volts, since the the magnitude of the fundamental of the quasi-square wave is $\frac{4E}{\pi}\,cos\,\alpha$. The corresponding 'duty ratio' of the applied voltage is $\phi/\pi = 1 - (2\,\alpha/\pi)$. One added



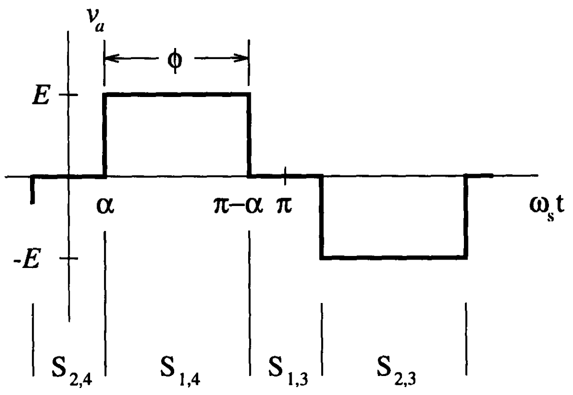


Figure 2-2: Full bridge SRC (from [3]).

advantage of phase modulation control, as suggested in [10], is that it allows the design of filters and magnetic components to be optimized at a specific frequency, which improves the efficiency of these components.

2.2 Modeling Simplifications of the SRC with Clamped Capacitor Voltage

In conventional dc/dc series resonant converters, the output ac current is rectified and filtered. An isolation transformer is often used to couple the load with the LC circuit. The modified SRC designed by Raytheon differs from the conventional SRC in two ways. One is the clamping diodes around the tank capacitor, and the other is

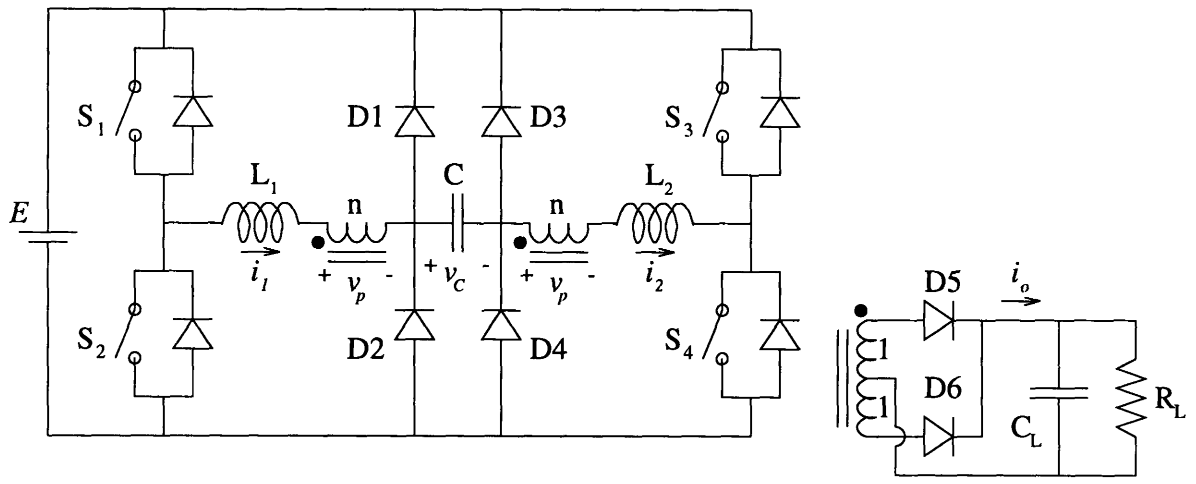


Figure 2-3: Schematic diagram of the SRC with clamped tank capacitor voltage, with $C = 0.2\mu\text{F}$, $L_1 = L_2 = 1\mu\text{H}$, $R_L = 0.0181\Omega$, $C_L = 10{,}000\mu\text{F}$, $E = 200$ to 350V, $n = 8$, and $f_s = 275\text{kHz}$.

the splitting of the inductor. Figure 2-3 shows the schematic diagram of the modified circuit with its nominal component values. The diodes and switches are assumed ideal throughout our development. If S_1, S_2 and S_3, S_4 are switched in the complementary manner shown in Figure 2-4, we may represent the voltage applied to the resonant circuit via voltage sources e_1 and e_2, which are both square waves $(0, +E)$ of the same frequency, but with a phase difference. This is shown in Figure 2-5. The voltage across the resonant circuit is $e_1 - e_2$.

For state-space descriptions, it is natural to take capacitor voltages and inductor currents as state variables. The conventional SRC can be modeled with two state variables as there is only one capacitor and one inductor. Its state-space trajectory can be easily represented in 2-D. The circuit in Figure 2-3, however, will then be a fourth-order system. This poses a problem for graphical representation and visualization. Some reasonable approximations can be made, however, so as to reduce the order of the system. The output capacitor C_L in our case is $10{,}000\mu\text{F}$, which is quite large. The output filtering is therefore highly effective, and the voltage across the load does not fluctuate much during typical operating conditions, so we may treat the load as a constant voltage source, as shown in Figure 2-5. Given that a constant output power

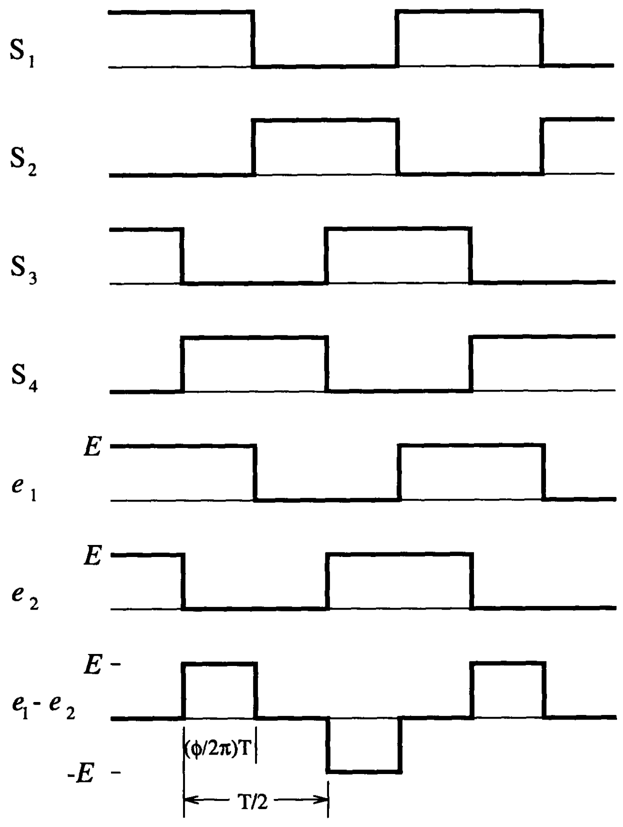


Figure 2-4: Waveforms of the switches and equivalent voltage sources.

of 4kW is to be maintained, the equivalent voltage source, V_L, is therefore roughly 8.5 volts. With this approximation, the circuit in Figure 2-5 is third-order.

We will model the output stage of the converter as a constant voltage source in most of our analysis of the converter. In the actual SRC circuit built by Raytheon, the output current is hard to measure because of physical constraints, so for the feedback controller, only the output voltage can be measured. Obviously, the constant voltage approximation is then no longer adequate. We will at that stage model the output either as a load resistor in parallel with the capacitor, as in Figure 2-3, or as a constant current source in parallel with the load capacitor.

The supply voltage, E, is not constant. It droops slowly, and its rate of change is much lower than that of the state variables. This slow variation in supply voltage will be treated as disturbance in the control design. In the analysis of converter dynamics,

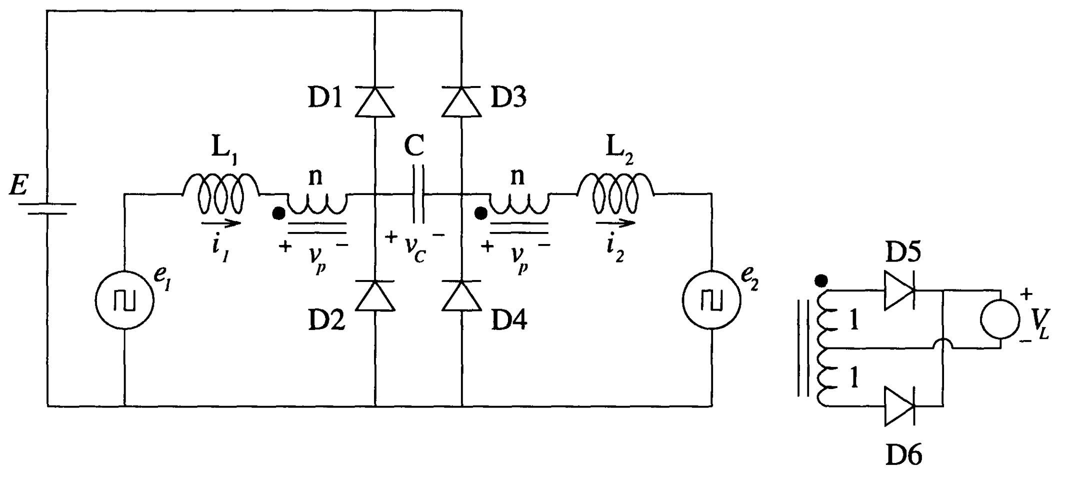


Figure 2-5: Simplified circuit diagram.

E is assumed to be constant.

The real circuit components also have parasitic capacitances and resistances, which pose various problems, particularly in ensuring well-behaved switching. We will not include any of these parasities in our circuit model. Their omission does not have a large impact on dynamic modeling for purposes of controller design.

We then arrive at the simplified circuit model in Figure 2-5 for the series resonant converter with clamped capacitor voltage. This is the model that we will use in analysis and simulation. Again, the output stage has to be modeled a little differently in control design.

Chapter 3

Topological Modes and State-Space Models

3.1 Topological Modes

The circuit shown in Figure 2-5 is piecewise LTI: we can write linear, time-invariant state-space equations for each combination of the diode conduction state configurations, *i.e.* for each *topological mode*. The natural state variables are $i_1(t)$, $i_2(t)$, and $v_C(t)$. For the six diodes in the circuit, there can be no more than $2^6 = 64$ different combinations of conduction states, but some combinations are not possible. We will consider the clamping diodes on the primary side of the transformer and the output rectifying diodes on the secondary side separately in order to keep the classification more manageable. For the clamping diodes, there are a total of nine topological modes. They are

M0	all off		
M1	D1 on	M5	D1 on and D4 on
M2	D2 on	M6	D2 on and D3 on
M3	D3 on	M7	D1 on and D3 on
M4	D4 on	M8	D2 on and D4 on

The output current is a transformed version of $i_1(t) + i_2(t)$, rectified by the two secondary-side diodes. The two diodes also determine the sign of $v_p(t)$ when in continuous conduction. Discontinuous conduction occurs when both output diodes are off and the output current is zero. The three topological modes associated with the output diodes are

S1	D5 on and D6 off
S2	D5 off and D6 on
S3	both off

3.2 Boundary Conditions and Their Representation in 3-D State-Space

Diode currents and voltages determine the conduction states of the diodes, which in turn determine the topological modes of the SRC. Diode current is positive and diode voltage is zero when the diode is on, and diode current is zero and diode voltage is negative when it is off. However, it can be shown [7] that the conditions on diode currents and voltages that determine the topological modes may be rewritten in terms of conditions on the three state variables, $i_1(t)$, $i_2(t)$, $v_C(t)$, and the input switching voltages, $e_1(t)$ and $e_2(t)$. Since the values of the state variables and input voltages are readily available in simulation, this simplifies the determination of topological modes in that diode currents and voltages need not be calculated. The boundary conditions are listed in Table 3.1. Detailed derivations of these boundary conditions are presented in Chapter 10 of [7].

A three-dimensional representation of the topological mode conditions in the state-space is shown in Figure 3-1. The topological modes that occupy planes ('plane modes') and those that occupy space region ('volume modes') are drawn separately for clarity. As can be seen from Table 3.1, conditions for M1 through M8, and for S1 and S2 involve only the state variables. Mode M0 occupies the $i_1 = i_2$ plane with the additional condition $e_1 \neq e_2$. The reason for this will be explained in Chapter 4, when we discuss the trajectory fields in M0. S3 also has an additional contraint involving

Topological Mode	Boundary Conditions
M0	$i_1 = i_2$ $-E \leq v_C \leq E$ $e_1 \neq e_2$
M1	$i_1 \geq i_2$ $0 \leq v_C \leq E$
M2	$i_1 \leq i_2$ $-E \leq v_C \leq 0$
M3	$i_1 \geq i_2$ $-E \leq v_C \leq 0$
M4	$i_1 \leq i_2$ $0 \leq v_C \leq E$
M5	$i_1 \geq 0, i_2 \geq 0$ $v_C = E$
M6	$i_1 \leq 0, i_2 \leq 0$ $v_C = -E$
M7	$i_1 \geq 0, i_2 \leq 0$ $v_C = 0$
M8	$i_1 \leq 0, i_2 \geq 0$ $v_C = 0$
S1	$i_1 + i_2 > 0$
S2	$i_1 + i_2 < 0$
S3	$i_1 + i_2 = 0$ $e_1 - e_2 - 2nV_L \leq v_C \leq e_1 - e_2 + 2nV_L$

Table 3.1: Boundary conditions for the topological modes.

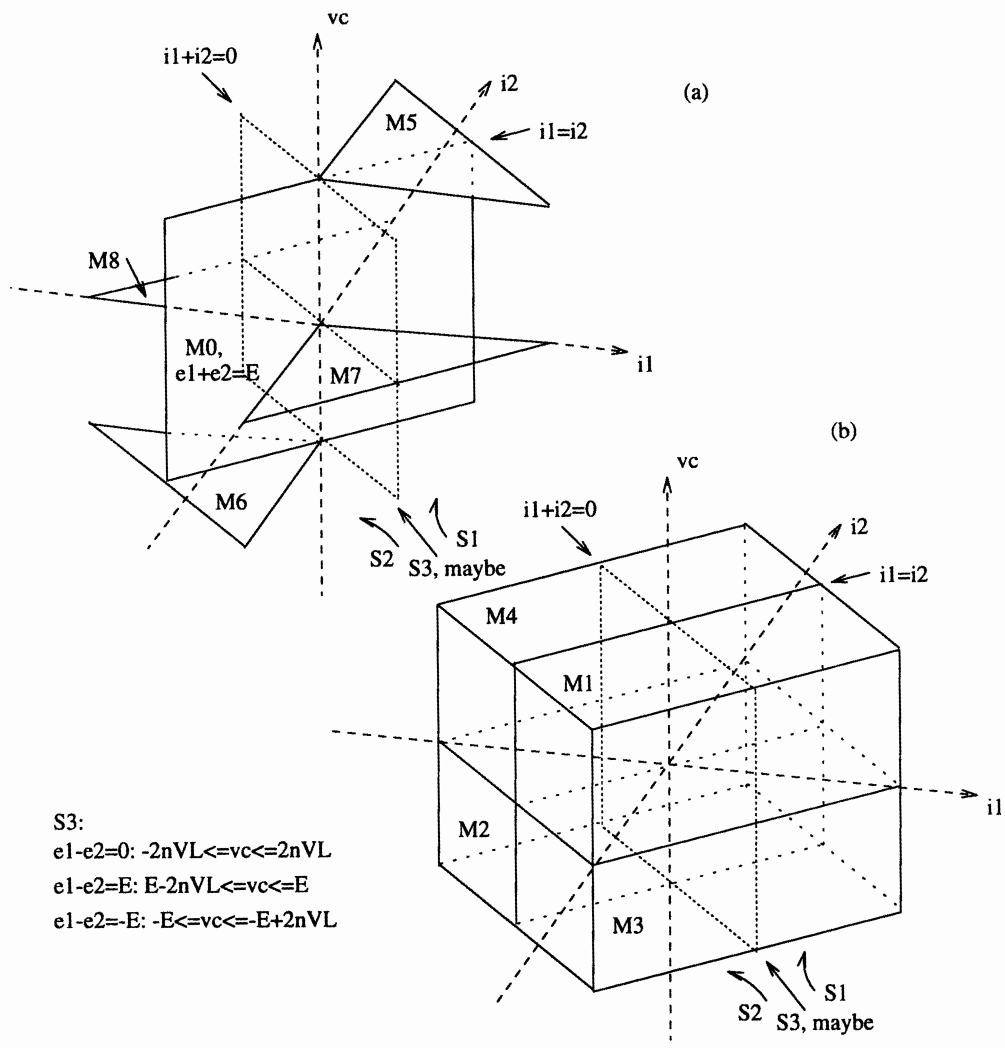


Figure 3-1: A three-dimensional representation of the topological modes of the SRC.

the input voltages and the load voltage. The $i_1 + i_2 = 0$ plane, as shown in Figure 3-1, is only a possible region for S3. The actual S3 mode may only lie in a part of that plane.

All the topological modes now hold a one-to-one correspondence between their three-dimensional representation in the state-space and their boundary conditions. Therefore, given a point in the state-space, the corresponding topological mode can be determined immediately from Figure 3-1. (At the boundaries between two modes, the converter can be considered to be in either or both modes.) The conditions specified in Table 3.1 are conditions for sustained operation in each mode, so the trajectory will remain within the particular mode until it reaches the edge and enters a different mode. A change in topological mode as a result of a change in the input voltages occurs only with M0 and S3, since only their boundary conditions involve e_1 and e_2. The input voltages do influence the directions of the trajectories in all cases, though.

3.3 State-Space Equations

For each of the topological modes of the SRC, the state-space equations can be written in the form

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

where $\mathbf{x}(t) = [i_1(t)\ i_2(t)\ v_C(t)]^T$, $\mathbf{B}_j\mathbf{u}(t)$ is a vector and is a function of $e_1(t)$, $e_2(t)$, and $v_p(t)$, and j denotes the topological mode. Detailed derivations of these equations are shown in [7]. They are listed in Table 3.2 for reference.

3.4 Symmetry Among Topological Modes

The symmetry in the structure of the SRC suggests possible symmetry in its trajectories in the state-space. The similarities among the various state-space matrices listed in Table 3.2 are also an indication. Three types of symmetry were found by Osawa [7]: symmetry about the origin, about the M0 plane, and about the line

Continuous Conduction (S1 or S2) $v_p = nV_L \mathrm{sign}(i_1 + i_2)$	Discontinuous Conduction (S3) $v_p = (e_1 - e_2 - v_C)/2$
$\mathbf{A}_0 = \begin{bmatrix} 0 & 0 & -\frac{1}{2L} \\ 0 & 0 & -\frac{1}{2L} \\ \frac{1}{C} & 0 & 0 \end{bmatrix}, \mathbf{B}_0\mathbf{u} = \begin{bmatrix} \frac{e_1-e_2-2v_p}{2L} \\ \frac{e_1-e_2-2v_p}{2L} \\ 0 \end{bmatrix}$	$\mathbf{A}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{C} & 0 & 0 \end{bmatrix}, \mathbf{B}_0\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
$\mathbf{A}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L} \\ 0 & \frac{1}{C} & 0 \end{bmatrix}, \mathbf{B}_1\mathbf{u} = \begin{bmatrix} \frac{e_1-E-v_p}{L} \\ \frac{-e_2+E-v_p}{L} \\ 0 \end{bmatrix}$	$\mathbf{A}_1 = \begin{bmatrix} 0 & 0 & \frac{1}{2L} \\ 0 & 0 & -\frac{1}{2L} \\ 0 & \frac{1}{C} & 0 \end{bmatrix}, \mathbf{B}_1\mathbf{u} = \begin{bmatrix} \frac{e_1+e_2-2E}{2L} \\ \frac{-e_1-e_2+2E}{2L} \\ 0 \end{bmatrix}$
$\mathbf{A}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L} \\ 0 & \frac{1}{C} & 0 \end{bmatrix}, \mathbf{B}_2\mathbf{u} = \begin{bmatrix} \frac{e_1-v_p}{L} \\ \frac{-e_2-v_p}{L} \\ 0 \end{bmatrix}$	$\mathbf{A}_2 = \begin{bmatrix} 0 & 0 & \frac{1}{2L} \\ 0 & 0 & -\frac{1}{2L} \\ 0 & \frac{1}{C} & 0 \end{bmatrix}, \mathbf{B}_2\mathbf{u} = \begin{bmatrix} \frac{e_1+e_2}{2L} \\ \frac{-e_1-e_2}{2L} \\ 0 \end{bmatrix}$
$\mathbf{A}_3 = \begin{bmatrix} 0 & 0 & -\frac{1}{L} \\ 0 & 0 & 0 \\ \frac{1}{C} & 0 & 0 \end{bmatrix}, \mathbf{B}_3\mathbf{u} = \begin{bmatrix} \frac{e_1-E-v_p}{L} \\ \frac{-e_2+E-v_p}{L} \\ 0 \end{bmatrix}$	$\mathbf{A}_3 = \begin{bmatrix} 0 & 0 & -\frac{1}{2L} \\ 0 & 0 & \frac{1}{2L} \\ \frac{1}{C} & 0 & 0 \end{bmatrix}, \mathbf{B}_3\mathbf{u} = \begin{bmatrix} \frac{e_1+e_2-2E}{2L} \\ \frac{-e_1-e_2+2E}{2L} \\ 0 \end{bmatrix}$
$\mathbf{A}_4 = \begin{bmatrix} 0 & 0 & -\frac{1}{L} \\ 0 & 0 & 0 \\ \frac{1}{C} & 0 & 0 \end{bmatrix}, \mathbf{B}_4\mathbf{u} = \begin{bmatrix} \frac{e_1-v_p}{L} \\ \frac{-e_2-v_p}{L} \\ 0 \end{bmatrix}$	$\mathbf{A}_4 = \begin{bmatrix} 0 & 0 & -\frac{1}{2L} \\ 0 & 0 & \frac{1}{2L} \\ \frac{1}{C} & 0 & 0 \end{bmatrix}, \mathbf{B}_4\mathbf{u} = \begin{bmatrix} \frac{e_1+e_2}{2L} \\ \frac{-e_1-e_2}{2L} \\ 0 \end{bmatrix}$
$\mathbf{A}_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{B}_5\mathbf{u} = \begin{bmatrix} \frac{e_1-E-v_p}{L} \\ \frac{-e_2-v_p}{L} \\ 0 \end{bmatrix}$	$i_1 = i_2 = 0$ $v_C = E$ $e_1 = E$ $e_2 = 0$ $v_p = 0$
$\mathbf{A}_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{B}_6\mathbf{u} = \begin{bmatrix} \frac{e_1-v_p}{L} \\ \frac{-e_2+E-v_p}{L} \\ 0 \end{bmatrix}$	$i_1 = i_2 = 0$ $v_C = -E$ $e_1 = 0$ $e_2 = E$ $v_p = 0$
$\mathbf{A}_7 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{B}_7\mathbf{u} = \begin{bmatrix} \frac{e_1-E-v_p}{L} \\ \frac{-e_2+E-v_p}{L} \\ 0 \end{bmatrix}$	$\mathbf{A}_7 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{B}_7\mathbf{u} = \begin{bmatrix} \frac{e_1+e_2-2E}{2L} \\ \frac{-e_1-e_2+2E}{2L} \\ 0 \end{bmatrix}$
$\mathbf{A}_8 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{B}_8\mathbf{u} = \begin{bmatrix} \frac{e_1-v_p}{L} \\ \frac{-e_2-v_p}{L} \\ 0 \end{bmatrix}$	$\mathbf{A}_8 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{B}_8\mathbf{u} = \begin{bmatrix} \frac{e_1+e_2}{2L} \\ \frac{-e_1-e_2}{2L} \\ 0 \end{bmatrix}$

Table 3.2: Matrices for state-space representation of the SRC.

$i_1 + i_2 = 0,\ v_C = 0$. Two trajectories $(i_{1a}(t), i_{2a}(t), v_{Ca}(t))$ and $(i_{1b}(t), i_{2b}(t), v_{Cb}(t))$ are symmetric about the origin if

$$(i_{1b}, i_{2b}, v_{Cb}) = (-i_{1a}, -i_{2a}, -v_{Ca}) \tag{3.2}$$

and

$$\left(\frac{di_{1b}}{dt}, \frac{di_{2b}}{dt}, \frac{dv_{Cb}}{dt}\right) = \left(-\frac{di_{1a}}{dt}, -\frac{di_{2a}}{dt}, -\frac{dv_{Ca}}{dt}\right) \tag{3.3}$$

for all t. The trajectories are symmetric about the M0 plane if

$$(i_{1b}, i_{2b}, v_{Cb}) = (i_{2a}, i_{1a}, v_{Ca}) \tag{3.4}$$

and

$$\left(\frac{di_{1b}}{dt}, \frac{di_{2b}}{dt}, \frac{dv_{Cb}}{dt}\right) = \left(\frac{di_{2a}}{dt}, \frac{di_{1a}}{dt}, \frac{dv_{Ca}}{dt}\right) \tag{3.5}$$

They are symmetric with respect to the line $i_1 + i_2 = 0,\ v_C = 0$ if

$$(i_{1b}, i_{2b}, v_{Cb}) = (-i_{2a}, -i_{1a}, -v_{Ca}) \tag{3.6}$$

and

$$\left(\frac{di_{1b}}{dt}, \frac{di_{2b}}{dt}, \frac{dv_{Cb}}{dt}\right) = \left(-\frac{di_{2a}}{dt}, -\frac{di_{1a}}{dt}, -\frac{dv_{Ca}}{dt}\right) \tag{3.7}$$

Table 3.3 lists the symmetric regions among different topological modes, with their corresponding combinations of input voltages. The proofs are relatively simple and the one for the M1, M2 pair is illustrated below. The others are stated here without proof. Refer to [7] for more detail.

At a point in M1, $\mathbf{x}_1$,

$$\dot{\mathbf{x}}_1 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{B}_1\mathbf{u} \tag{3.8}$$

At the corresponding point in M2, $\mathbf{x}_2 = -\mathbf{x}_1$,

$$\dot{\mathbf{x}}_2 = -\mathbf{A}_2\mathbf{x}_1 + \mathbf{B}_2\mathbf{u} \tag{3.9}$$

Mode	with (e_1, e_2)	$w.r.t$	Mode	with (e_1, e_2)
M2	$(0,0)$	origin	M1	(E,E)
	$(0,E)$			$(E,0)$
	$(E,0)$			$(0,E)$
	(E,E)			$(0,0)$
M4	$(0,0)$	origin	M3	(E,E)
	$(0,E)$			$(E,0)$
	$(E,0)$			$(0,E)$
	(E,E)			$(0,0)$
M6	$(0,0)$	origin	M5	(E,E)
	$(0,E)$			$(E,0)$
	$(E,0)$			$(0,E)$
	(E,E)			$(0,0)$
M8	$(0,0)$	origin	M7	(E,E)
	$(0,E)$			$(E,0)$
	$(E,0)$			$(0,E)$
	(E,E)			$(0,0)$
M4	$(0,0)$	M0	M1	(E,E)
	$(0,E)$	plane		$(E,0)$
	$(E,0)$			$(0,E)$
	(E,E)			$(0,0)$
M3	$(0,0)$	M0	M2	(E,E)
	$(0,E)$	plane		$(E,0)$
	$(E,0)$			$(0,E)$
	(E,E)			$(0,0)$
M3	$(0,0)$	line	M1	$(0,0)$
	$(0,E)$	$i_1 + i_2 = 0$,		$(E,0)$
	$(E,0)$	$v_C = 0$		$(0,E)$
	(E,E)			(E,E)
M4	$(0,0)$	line	M2	$(0,0)$
	$(0,E)$	$i_1 + i_2 = 0$,		$(E,0)$
	$(E,0)$	$v_C = 0$		$(0,E)$
	(E,E)			(E,E)

Table 3.3: Symmetric regions among topological modes.

Mode	with (e_1, e_2)	$w.r.t$	Mode	with (e_1, e_2)
M0	$(0, E)$	origin	M0	$(E, 0)$
M5	$(0, E)$ $(0, 0)$ (E, E)	$i_1 = i_2$	M5	$(E, 0)$ $(0, 0)$ (E, E)
M6	$(0, E)$ $(0, 0)$ (E, E)	$i_1 = i_2$	M6	$(E, 0)$ $(0, 0)$ (E, E)
M7	$(0, E)$ $(0, 0)$ (E, E)	$i_1 + i_2 = 0$	M7	$(E, 0)$ $(0, 0)$ (E, E)
M7	$(0, E)$ $(0, 0)$ (E, E)	$i_1 + i_2 = 0$	M7	$(E, 0)$ $(0, 0)$ (E, E)

Table 3.4: Symmetric regions within topological modes.

For the two points to be symmetric with respect to the origin,

$$\dot{\mathbf{x}}_1 = -\dot{\mathbf{x}}_2 \tag{3.10}$$

Since $\mathbf{A}_1$ is equal to $\mathbf{A}_2$, this means that $\mathbf{B}_1\mathbf{u}$ must be equal to $-\mathbf{B}_2\mathbf{u}$. Note that

$$\mathbf{B}_1\mathbf{u} = \left.\begin{bmatrix} \frac{e_1 - E - v_p}{L} \\ \frac{-e_2 + E - v_p}{L} \\ 0 \end{bmatrix}\right|_{\mathbf{x}_1}, \quad -\mathbf{B}_2\mathbf{u} = -\left.\begin{bmatrix} \frac{e_1 - v_p}{L} \\ \frac{-e_2 - v_p}{L} \\ 0 \end{bmatrix}\right|_{\mathbf{x}_2} \tag{3.11}$$

Since $\mathbf{x}_2 = -\mathbf{x}_1$ and $v_p = nV_L \text{sign}(i_1 + i_2)$, it follows that $v_p|_{\mathbf{x}_1} = -\,v_p|_{\mathbf{x}_2}$. The combinations of e_1 and e_2 at $\mathbf{x}_1$ and $\mathbf{x}_2$ that will satisfy (3.11) are

$$e_1 = 0, e_2 = 0 \text{ at } \mathbf{x}_1 \text{ and } e_1 = E, e_2 = E \text{ at } \mathbf{x}_2$$

$$e_1 = 0, e_2 = E \text{ at } \mathbf{x}_1 \text{ and } e_1 = E, e_2 = 0 \text{ at } \mathbf{x}_2$$

$$e_1 = E, e_2 = 0 \text{ at } \mathbf{x}_1 \text{ and } e_1 = 0, e_2 = E \text{ at } \mathbf{x}_2$$

$$e_1 = E, e_2 = E \text{ at } \mathbf{x}_1 \text{ and } e_1 = 0, e_2 = 0 \text{ at } \mathbf{x}_2$$

Symmetry properties also exist within a topological mode itself. Again, without proof, a list of the symmetric regions is shown in Table 3.4. This type of symmetry is most evident in the velocity fields shown in Chapter 4.

These symmetry properties effectively reduce the analysis of the state-space trajectories to only four modes: M0, M1, M5, and M7. The trajectories in other modes can be derived from them. The symmetry of the circuit suggests that when the switching is also done symmetrically, the steady-state limit cycle will have two half-cycles that are symmetric with respect to the origin. This will be confirmed in Chapter 6.

Chapter 4

Velocity Fields of State-Space Trajectories

The velocity vector of the state trajectory at any point in the state-space, for some combination of input voltages and load voltage, can be found directly from the state-space equations in Table 3.2. A graphical representation can be obtained by plotting out the trajectories for a short period of time from a number of different state-space locations. The resulting diagrams, which will be called the trajectory *velocity fields*, give a clear picture as to where and how fast the trajctories will move at various points in the state-space. The plots shown in this chapter are generated using a value of $E = 200$V and $V_L = 8.5$V for 0.05μsec.

4.1 Topological Mode M0

Topological mode M0 was analyzed in some detail in [7]. None of the clamping diodes is conducting in this mode, so the two inductor currents, i_1 and i_2, must be equal. To stay in M0, v_C must satisfy

$$\begin{aligned} -(e_1+e_2) &\leq v_C \leq 2E-(e_1+e_2) \\ -2E+(e_1+e_2) &\leq v_C \leq (e_1+e_2) \end{aligned} \tag{4.1}$$

When $e_1 = e_2$, $v_C = 0$ is the only value that can satisfy both conditions. However, with a nonzero inductor current, v_C cannot be maintained at zero and the above conditions are immediately violated. The trajectory will then leave mode M0. The combination of $v_C = 0$, $i_1 = 0$, and $i_2 = 0$ is a trivial solution. This situation is illustrated in Figure 4-1[1] where i_1 and i_2 are plotted against v_C for the four combinations of input voltages. The trajectories in (c) and (d) for the case of $e_1 = e_2$ start on the $i_1 = i_2$ plane, but i_1 *vs.* v_C and i_2 *vs.* v_C do not coincide as time goes on, showing a difference in i_1 and i_2. The trajectories obviously do not stay on the $i_1 = i_2$ plane except for the starting point. Thus, to sustain operation in M0, the input voltages must be different. M0 is, in fact, the only M mode in which a condition involving the input voltages e_1 and e_2 is required.

Given that the two input voltages are not the same, topological mode M0 represents the $i_1 = i_2$ plane in state-space with v_C limited to between $+E$ and $-E$ because of the clamping diodes. Since the two inductor currents, i_1 and i_2, are equal, the order of the circuit is reduced to two, and a plot of either i_1 *vs.* v_C or i_2 *vs.* v_C is adequate for analysis of the trajectories. Both i_1 *vs.* v_C and i_2 *vs.* v_C are plotted in Figures 4-1(a) and (b), but because the two overlap each other, only one graph is visible. The velocities of the trajectories depend on circuit parameters and the initial locations. It is a simple matter to derive them from the state-space equations listed in Table 3.2.

From Figures 4-1(a) and (b), some general properties of the trajectory in this mode can be seen. When the inductor current is positive, it charges up the capacitor, and v_C increases. Negative current reduces the capacitor voltage. If the initial current is large enough, the capacitor voltage will eventually reach $+E$ or $-E$, at which time the SRC enters M5 or M6, respectively. Figures 4-1(a) and (b) are symmetric with respect to the origin, a property described also in [7]. The mode numbers with parentheses, shown in Figure 4-1 and the velocity fields on the next few pages, are the topological modes the trajectories can move into once they reach the boundaries.

One condition for discontinuous conduction, S3, is $i_1 + i_2 = 0$. On the M0 plane,

[1]Simulated with *sim0.m*, see Appendix A.

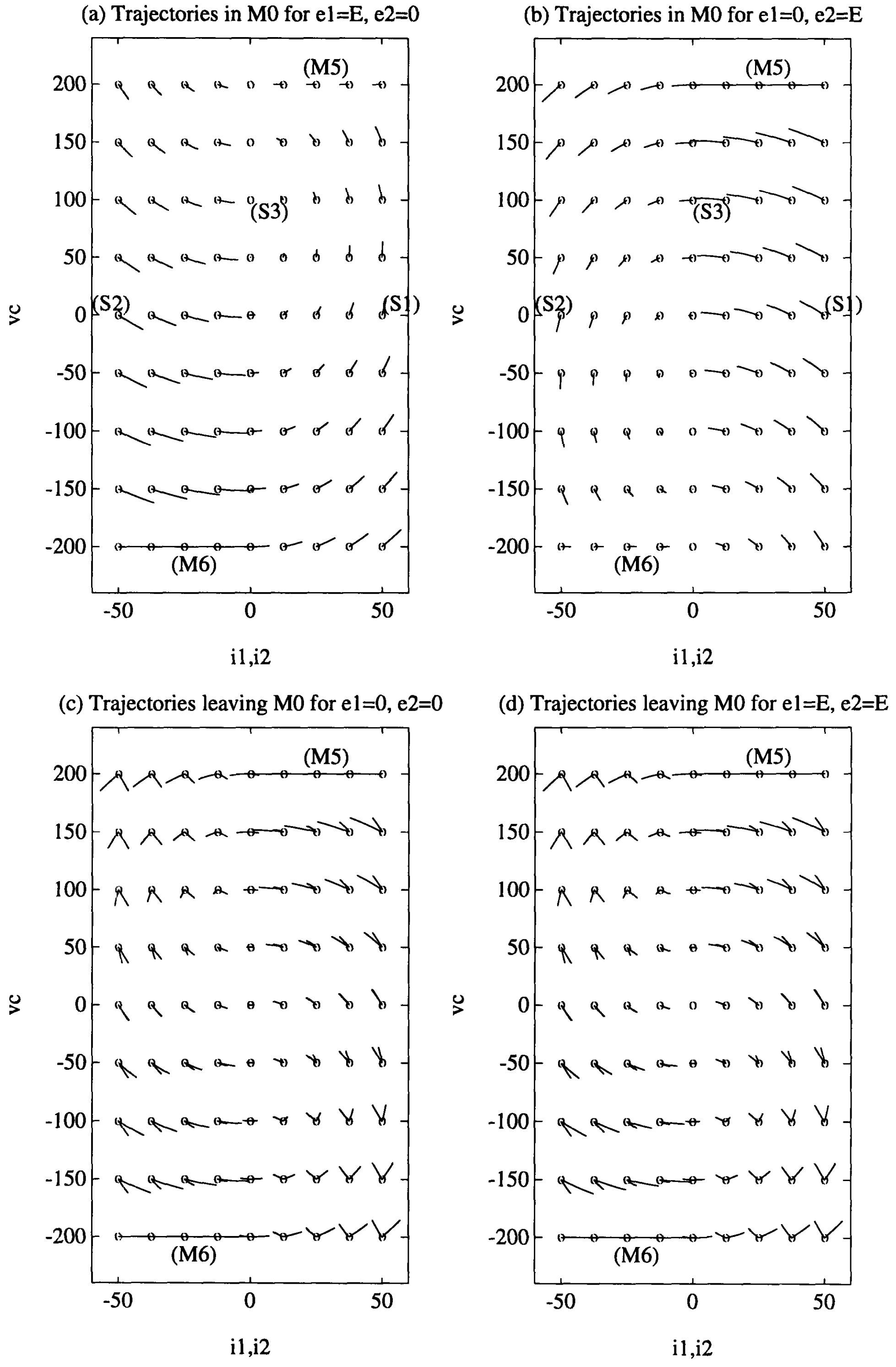


Figure 4-1: Velocity fields in mode M0 for (a) $e_1 = E$ and $e_2 = 0$, (b) $e_1 = 0$ and $e_2 = E$, (c) $e_1 = 0$ and $e_2 = 0$, (d) $e_1 = E$ and $e_2 = E$.

S3 can occur only on the v_C axis, where $i_1 = i_2 = 0$. The additional condition on v_C says that the region for S3 is also limited to $E - 2nV_L \leq v_C \leq E$ for $e_1 = E$ and $e_2 = 0$, and $-E \leq v_C \leq -E + 2nV_L$ for $e_1 = 0$ and $e_2 = E$. With the state-space matrices $\mathbf{A}_0$ and $\mathbf{B}_0\mathbf{u}$ in Table 3.2, it can be shown that the derivatives of all state variables in S3 are zero. Therefore, once the trajectory reaches S3, it will stop moving until input conditions change. This effect is shown on the top part of the v_C axis in Figure 4-1(a) and on the bottom part in Figure 4-1(b). Osawa referred to this as the 'wall' on the v_C axis, where the trajectory cannot move from one side to the other [7]. On the remaining part of the v_C axis where S3 cannot be sustained, the trajectory simply passes through. Osawa called this the 'window.'

4.2 Topological Modes M5 and M6

When the capacitor voltage attempts to increase beyond E, diodes D1 and D4 turn on so that v_C is clamped to E. The circuit operation in this mode M5 is relatively simple. As the inductor currents must be positive, the circuit is guaranteed to be in continuous conduction mode S1. The load voltage referred to the primary side, v_p, is positive, so the voltages across the two inductors are negative for all of the four combinations of input voltages. For any particular set of input voltages, the inductor voltages are constant. As a result, i_1 and i_2 decrease linearly at a constant rate, independent of the values of i_1 or i_2, until one or both of them drop to zero. Depending on which current drops to zero first, D1 or/and D4 turn off and the SRC enters M4, M1, or M0. The rate of change in the inductor current depends only on the input voltage combination and not on the location of the trajectory, as can be seen in Figure 4-2[2].

Topological mode M6 is the dual of M5. The two are symmetric with respect to the origin. Plots of trajectories in M6 will not be repeated here.

[2]Simulated with *sim5.m*, see Appendix A.

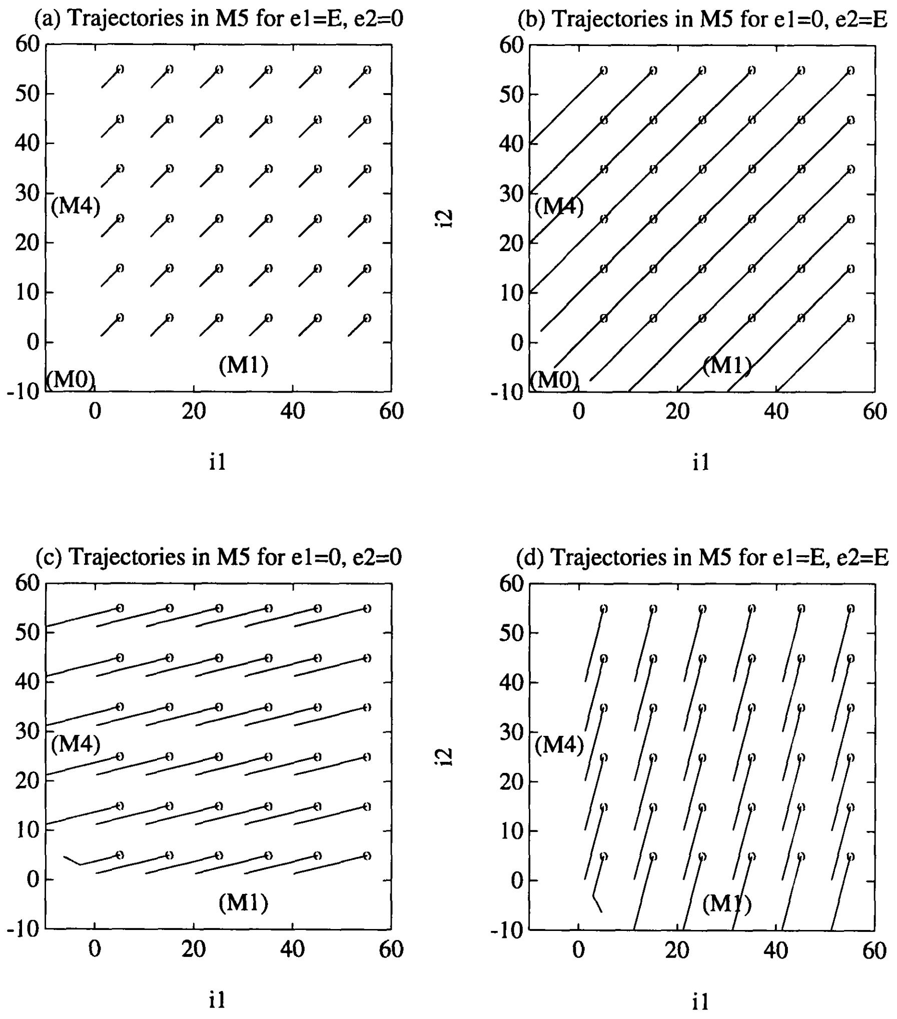


Figure 4-2: Velocity fields in mode M5 for (a) $e_1 = E$ and $e_2 = 0$, (b) $e_1 = 0$ and $e_2 = E$, (c) $e_1 = 0$ and $e_2 = 0$, (d) $e_1 = E$ and $e_2 = E$.

4.3 Topological Modes M7 and M8

In topological mode M7, the two upper diodes D1 and D3 are on, so the capacitor voltage is zero. M7 is entered from either M1 or M3. Transition between M1 and M3 without entering M7 occurs when the two inductor currents are in the same direction. Note that v_C changes sign during the transition, but cannot stay at zero. Only when i_1 is positive and i_2 is negative do both D1 and D3 turn on. The voltage at both ends of the tank capacitor is $+E$ now.

Once the combination of input voltages and the sign of v_p are known, the voltages across the inductors are constant. The trajectories should then move at a constant rate, independent of the magnitude of the inductor currents. The S3 plane divides the M7 plane into two regions, S1 and S2. Trajectories in S1 and S2 are symmetric with respect to the line $i_1 + i_2 = 0$ for input voltage combinations $e_1 = 0, e_2 = 0$ and $e_1 = E, e_2 = E$. Trajectories for $e_1 = E, e_2 = 0$ in Figure 4-3[3](a) and those for $e_1 = 0, e_2 = E$ in Figure 4-3(b) are also symmetric. For the values used in simulating Figure 4-3, $E - 2nV_L > 0$. The voltage condition for sustainable S3 operation is not satisfied on the M7 plane for $e_1 = E, e_2 = 0$ and $e_1 = 0, e_2 = E$, so S3 does not exist at all in these two cases. One the other hand, for $e_1 = 0, e_2 = 0$ and $e_1 = E, e_2 = E$, the voltage condition for S3 is satisfied, and S3 is the $i_1 + i_2 = 0, v_C = 0$ line on the M7 plane. As can be seen in Figures 4-3(c) and (d), the trajectories tend towards S3. Once they reach S3, depending on the input voltages, they either move towards the origin or stop moving altogether.

Topological mode M8 is the dual of M7. The two are symmetric with respect to the origin.

4.4 Topological Modes M1, M2, M3, and M4

The velocity vectors in volume modes can be derived from the state-space equations as before. Unlike where the trajectories travel on planes, in M1, M2, M3, and M4,

[3]Simulated with *sim7.m*, see Appendix A.

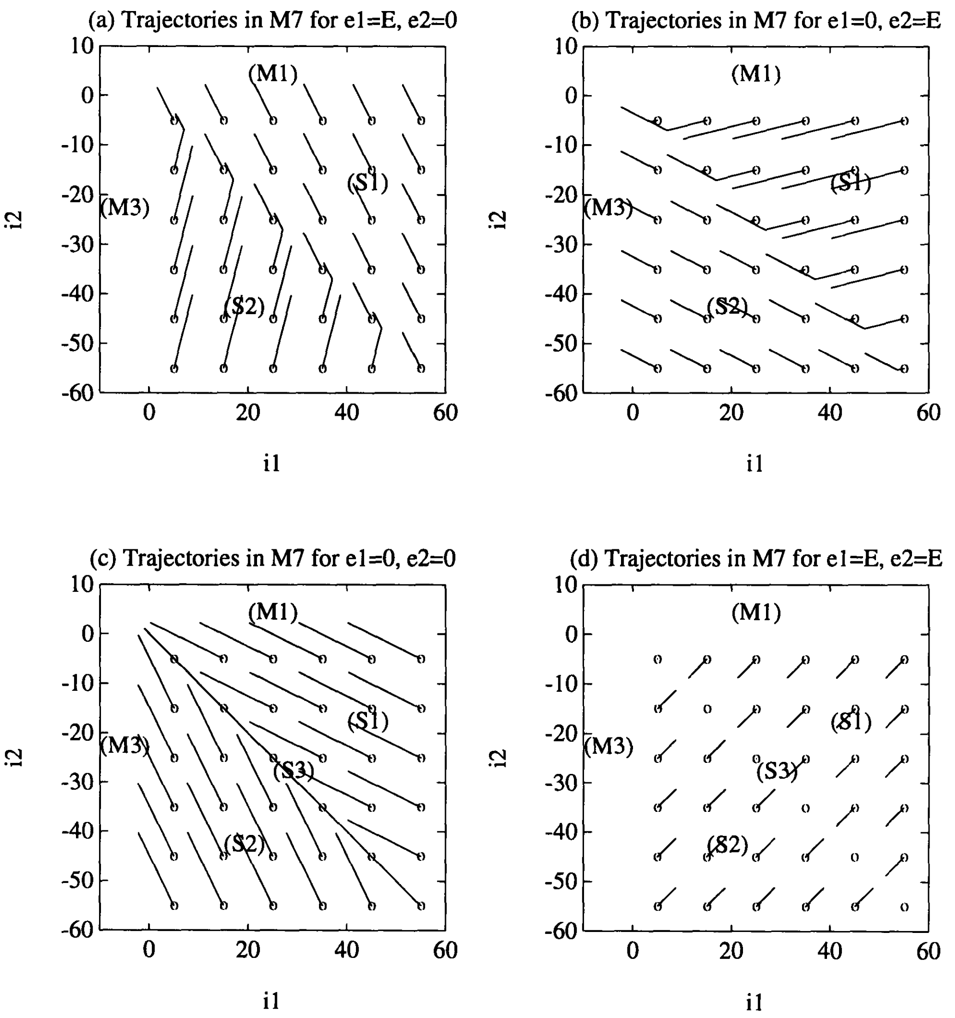


Figure 4-3: Velocity fields in mode M7 for (a) $e_1 = E$ and $e_2 = 0$, (b) $e_1 = 0$ and $e_2 = E$, (c) $e_1 = 0$ and $e_2 = 0$, (d) $e_1 = E$ and $e_2 = E$.

they move through space. This makes the visualization of the trajectory fields difficult even if 3-D imaging is used. Not only is it difficult to visualize a bunch of short curves in space from their projections, it is equally difficult to see them in a 3-D plot from any one perspective, due to the lack of depth. Fortunately, the v_C component of the trajectory velocity vector in modes M1 to M4 is either $\frac{dv_C}{dt} = \frac{1}{C} i_1$ or $\frac{dv_C}{dt} = \frac{1}{C} i_2$ (See Table 3.2). Therefore, not much information is lost by projecting the trajectory field onto the i_1 *vs.* i_2 plane at various levels of v_C.

In M1, the v_C component of the trajectory vector is directly proportional to i_2. In Figures 4-4[4] and 4-5, projections of the trajectories starting at four different values of v_C are plotted. For positive i_2, the trajectory comes out of the paper towards the reader in the direction of the v_C axis, and for negative i_2, it goes into the paper. To indicate the direction of the v_C component, an **o** is marked at the starting location in the former case, and an **x** is marked in the latter case. Moving from one value of v_C to the next, we can see the gradual changes in the i_1 and i_2 components of the velocity vectors.

The condition on v_C for sustaining S3, stated in Table 3.1, dictates what part of the $i_1 + i_2 = 0$ plane belongs to S3. For the simulations done here, $E - 2nV_L > 0$. S3 is then confined to a band near $v_C = 0$, or $v_C = E$, or $v_C = -E$, depending on the input voltages. In M1, where v_C is positive, when e_1 is equal to e_2, S3 occurs in the lower range of v_C around 0 from $-2nV_L$ to $2nV_L$. When $e_1 = E, e_2 = 0$, S3 occurs in the upper range of v_C near E from $E - 2nV_L$ to E. For $e_1 = 0, e_2 = E$, S3 does not exist at all since in this case v_C would have to be negative. This is why sometimes in Figures 4-4 and 4-5, S3 is not labeled because it does not exist.

The trajectories in M2, M3, and M4 are related to those in M1 through the symmetry properties stated in Table 3.3. The analysis is similar and thus will not be repeated here.

[4]Figure 4-4 and Figure 4-5 are simulated with *sim1.m*. See Appendix A.

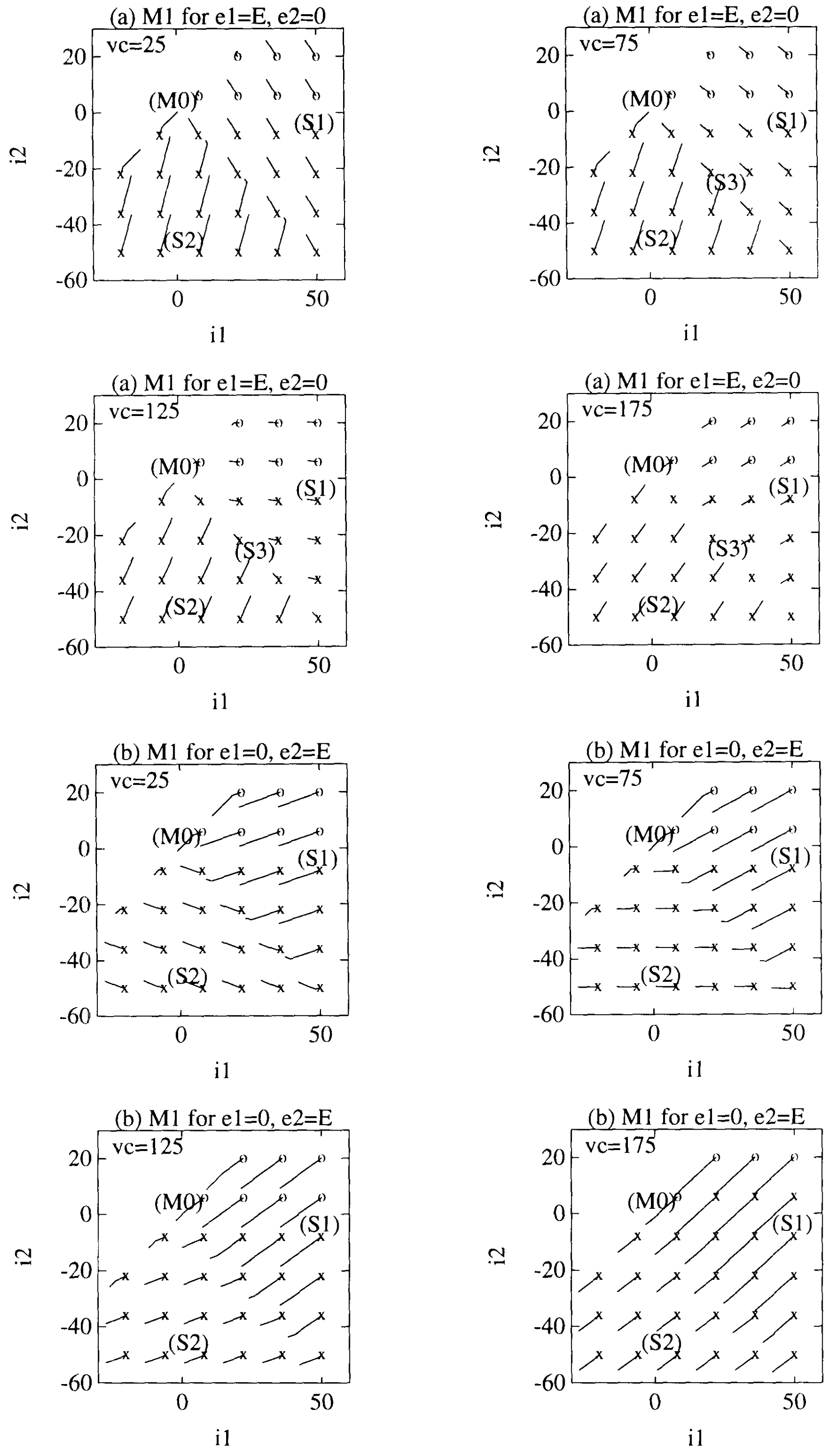


Figure 4-4: Velocity fields in mode M1 for (a) $e_1 = E$ and $e_2 = 0$, (b) $e_1 = 0$ and $e_2 = E$ at various levels of v_C and with $E = 200\text{V}$.

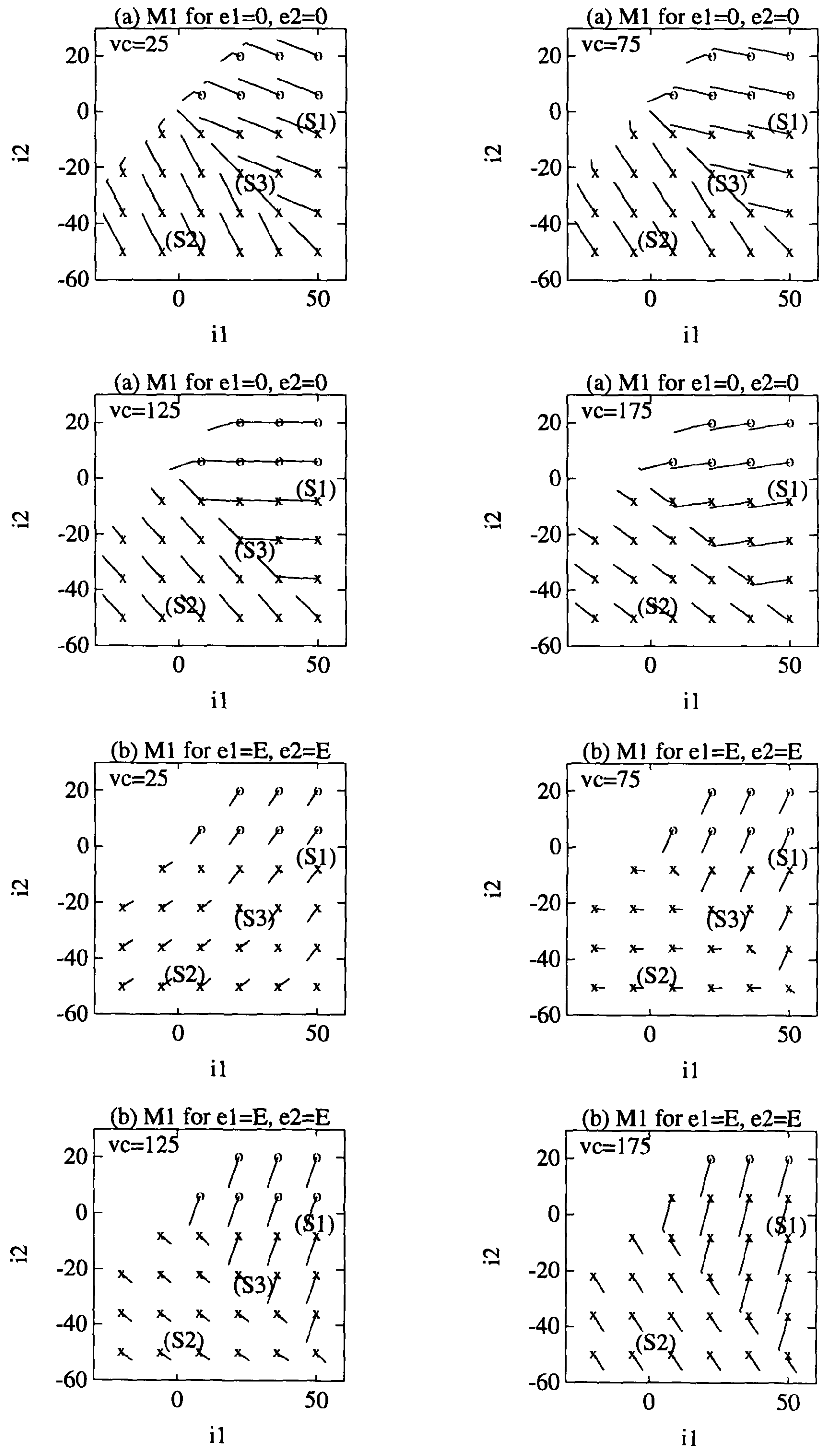


Figure 4-5: Velocity fields in mode M1 for (a) $e_1 = 0$ and $e_2 = 0$, (b) $e_1 = E$ and $e_2 = E$ at various levels of v_C and with $E = 200$V.

Chapter 5

Trajectory Geometry

The trajectory velocity fields give a good picture of the speed and direction of the trajectories at different points in the state-space. However, they are not adequate for topological modes M1 to M4, which occupy space regions, because of difficulty with 3-D representation and visualization.

The trajectories of the conventional series resonant converter in 2-D consist of segments of circular arcs when plotted against normalized axes. Even with the additional inductor current in our SRC, we may guess that the 3-D trajectories in our case will be arcs on spheres or cylinders when plotted against normalized axes. The trajectory velocity fields in the last chapter have already given some indication of this.

5.1 Topological Modes M1, M2, M3, and M4

In Table 3.2, we listed the state-space equations for each of the topological modes. The differential equations may be solved to obtain closed-form expressions for the trajectories. The relatively sparse state-space matrices suggest relatively simple solutions. Trajectory behaviors in Modes M1, M2, M3, and M4 are similar, as indicated by the symmetry properties. A more detailed analysis of mode M1 in continuous conduction is presented here first. Trajectory behavior in the other modes and in discontinuous conduction mode S3 can be derived in the same manner, and are briefly described.

5.1.1 Analysis of Trajectories in Mode M1

The state-space equations for M1 in continuous conduction mode are

$$\begin{aligned}\frac{di_1}{dt} &= \frac{e_1 - E - v_p}{L}\\ \frac{di_2}{dt} &= -\frac{1}{L}v_C + \frac{-e_2 + E - v_p}{L}\\ \frac{dv_C}{dt} &= -\frac{1}{C}i_2\end{aligned} \tag{5.1}$$

Inductor current i_1 is completely decoupled from the other two state variables. Its evolution in time is linear. The interaction between i_2 and v_C can be seen by cross multiplying the second and the third equations:

$$\frac{1}{C}i_2\frac{di_2}{dt} = -\frac{1}{L}v_C\frac{dv_C}{dt} + \frac{-e_2 + E - v_p}{L}\frac{dv_C}{dt} \tag{5.2}$$

Integrating this, we get

$$\frac{1}{2C}i_2^2 + \frac{1}{2L}v_C^2 - \frac{-e_2 + E - v_p}{L}v_C = K' \tag{5.3}$$

where is K' is a constant determined by the initial conditions. This is true for any set of constant values of e_1, e_2, and v_p, that is, as long as we remain in the same switching state and the same output S mode. Rearranging (5.3), we have

$$\left(\sqrt{L}\,i_2\right)^2 + \left[\sqrt{C}\,v_C - \sqrt{C}\,(-e_2 + E - v_p)\right]^2 = K \tag{5.4}$$

where K is a constant. This is the equation for an ellipse. The trajectory plotted against normalized axes of $\sqrt{L}\,i_2$ *vs.* $\sqrt{C}\,v_C$ is an arc of a circle centered at $\sqrt{C}\,(-e_2 + E - v_p)$ with radius $\sqrt{K}$. It is clear that the center is determined by the input switching voltage, which is just e_2 in this case, and the transformer voltage, v_p, whose sign depends on the output diode configuration. The radius is determined by the

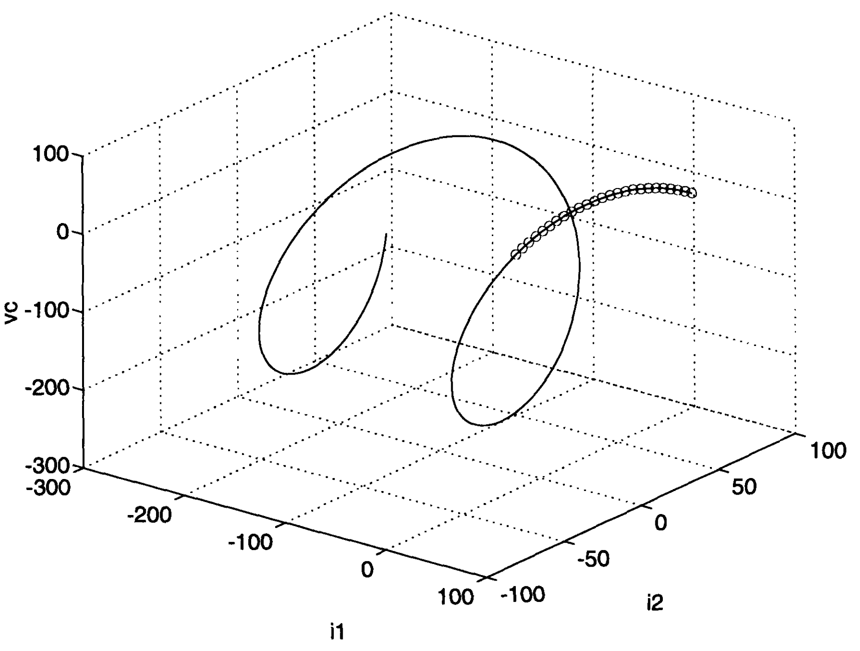


Figure 5-1: A sample trajectory in M1 with $e_1 = e_2 = E = 250\text{V}$, $v_p = nV_L = 8{\cdot}8.5\text{V}$, $i_1(0) = 90\text{A}$, $i_2(0) = 40\text{A}$, $v_C(0) = 60\text{V}$.

starting position of the trajectory. Since $\sqrt{L}\, i_1$ is a linear function of time, the state-space trajectory is then a spiral on the surface of a cylinder whose axis is parallel to the $\sqrt{L}\, i_1$ axis and intersects the $\sqrt{C}\, v_C$ axis. Figure 5-1[1] shows an example of the spiraling trajectory in M1 for $e_1 = e_2 = E = 250\text{V}$, $v_p = nV_L = 8 \cdot 8.5\text{V}$, and the initial conditions $i_1(0) = 90\text{A}$, $i_2(0) = 40\text{A}$, $v_C(0) = 60\text{V}$. Because of boundary conditions, only the portion of the graph dotted with circles is actually in M1. The rest of the spiral is drawn as a visual cue.

The solution to the state-space equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{5.5}$$

is

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\, d\tau \tag{5.6}$$

[1]Drawn with *spiralm1.m*, see Appendix A.

Since we are looking at the trajectory in a particular topological mode for a particular combination of e_1, e_2, and v_p, $\mathbf{Bu}$ is a constant vector, which makes the evaluation a little simpler. However, because the $\mathbf{A}$ matrices are sparse, it is easier and more intuitive to simply solve the set of differential equations directly to obtain closed-form expressions for the state variables as a function of time. Doing this for M1, we get

$$\begin{aligned} i_1(t) &= \frac{e_1 - E - v_p}{L} t + i_1(0) \\ i_2(t) &= \frac{A}{\sqrt{L}} \cos\left(\frac{1}{\sqrt{LC}} t + \theta\right) \\ v_C(t) &= \frac{A}{\sqrt{C}} \sin\left(\frac{1}{\sqrt{LC}} t + \theta\right) + (-e_2 + E - v_p) \end{aligned} \tag{5.7}$$

where

$$A = \sqrt{L\, i_2^2(0) + C\,[v_C^2(0) - (-e_2 + E - v_p)]^2} \tag{5.8}$$

and

$$\theta = \tan^{-1}\left[\frac{v_C(0) - (-e_2 + E - v_p)}{i_2(0)} \sqrt{\frac{C}{L}}\right] \tag{5.9}$$

By expanding the sine and cosine terms, we may write the state variables in vector form as a function of time and initial conditions. This is the state transition matrix form

$$\mathbf{x}(t) = \mathbf{\Phi}_j(t)\mathbf{x}(0) + \mathbf{\Psi}_j(t) \tag{5.10}$$

where j denotes the topological mode. For M1, we have

$$\mathbf{\Phi}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\frac{t}{\sqrt{LC}}) & -\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ 0 & \sqrt{\frac{L}{C}}\sin(\frac{t}{\sqrt{LC}}) & \cos(\frac{t}{\sqrt{LC}}) \end{bmatrix} \tag{5.11}$$

and

$$\mathbf{\Psi}_1 = \begin{bmatrix} \frac{e_1 - E - v_p}{L} t \\ \sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}})(-e_2 + E - v_p) \\ \left(1 - \cos(\frac{t}{\sqrt{LC}})\right)(-e_2 + E - v_p) \end{bmatrix} \tag{5.12}$$

The results obtained by solving Equation 5.6 are exactly the same as Equation 5.10. It is not hard to verify, for example, that $\mathbf{\Phi}_1(t) = e^{\mathbf{A}_1 t}$.

5.1.2 Trajectory Geometry in M2, M3 and M4

Trajectories in M2 behave in very much the same way as in M1, barring the difference in the location of the axis of the cylinder. This is consistent with the symmetry property between M1 and M2 with respect to the origin for various combinations of switching voltages. The state transition matrices are listed in Table 5.1.

The state-space equations for M3 are similar to those for M1 except that now i_2 varies linearly with time, and only i_1 and v_C are coupled. The equation relating i_1 and v_C, similar to Equation 5.4, is

$$\left(\sqrt{L}\,i_1\right)^2 + \left[\sqrt{C}\,v_C - \sqrt{C}\,(e_1 - E - v_p)\right]^2 = K \qquad (5.13)$$

where K is a constant determined by the initial conditions. When plotted against normalized axes, the state-space trajectory is a spiral on the surface of a cylinder whose axis is parallel to the $\sqrt{L}\,i_2$ axis and intersects the $\sqrt{C}\,v_C$ axis. An example is shown in Figure 5-2[2] for $e_1 = 0\text{V}$, $e_2 = E = 250\text{V}$, $v_p = nV_L = 8 \cdot 8.5\text{V}$, and the initial conditions $i_1(0) = 90\text{A}$, $i_2(0) = 40\text{A}$, $v_C(0) = -60\text{V}$. Again, because of boundary conditions, only the portion of the graph marked with circles is actually in M3.

The trajectory geometry in mode M4 exhibits similar characteristics to that in M3.

5.1.3 Discontinuous Conduction Mode S3

In the previous discussion about trajectory geometry, we have assumed continuous conduction, so the transformer primary-side voltage v_p is equal to $\pm nV_L$ (with the sign depending on the output diode conduction state). When the circuit is in discontinuous conduction mode, v_p is no longer a function of the output load voltage, and $i_1 + i_2 = 0$.

[2]Drawn with *spiralm3.m*, see Appendix A.

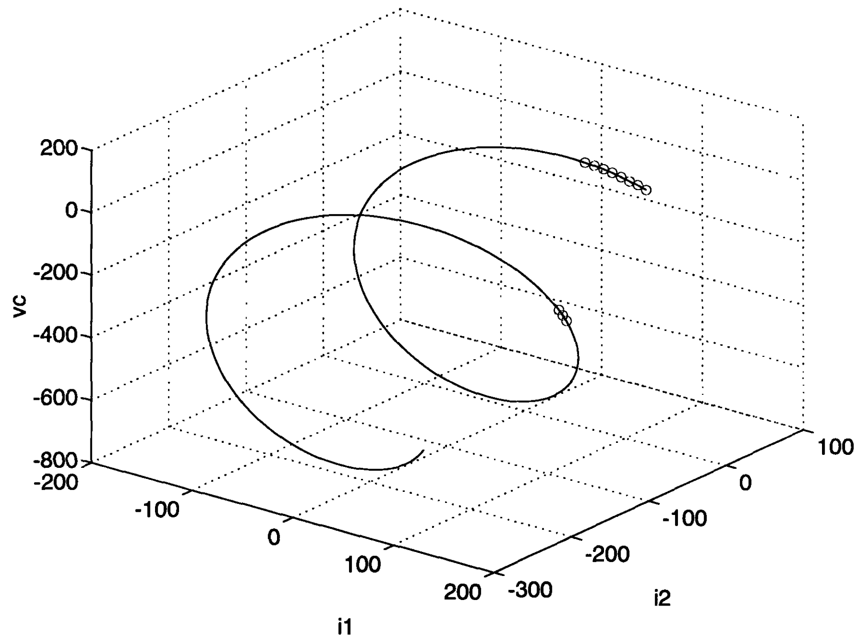


Figure 5-2: A sample trajectory in M3 with $e_1 = 0\text{V}$, $e_2 = E = 250\text{V}$, $v_p = nV_L = 8 \cdot 8.5\text{V}$, $i_1(0) = 90\text{A}$, $i_2(0) = 40\text{A}$, $v_C(0) = -60\text{V}$.

The trajectory is then confined to the $i_1 = -i_2$ or the S3 plane. We are then actually dealing with a second-order system.

Assuming that the conditions for S3 are satisfied, and using the state-space equations for M1, S3:

$$\begin{aligned}\frac{di_1}{dt} &= \frac{1}{2L}v_C + \frac{e_1 + e_2 - 2E}{2L} \\ \frac{di_2}{dt} &= -\frac{1}{2L}v_C - \frac{e_1 + e_2 - 2E}{2L} \\ \frac{dv_C}{dt} &= -\frac{1}{C}i_2\end{aligned} \tag{5.14}$$

we get

$$\left(\sqrt{2L}\,i_2\right)^2 + \left[\sqrt{C}\,v_C - \sqrt{C}\,(-e_1 - e_2 + 2E)\right]^2 = K \tag{5.15}$$

where K is a constant. The normalization factor for the current is $\sqrt{2L}$. When plotted against normalized axes of $\sqrt{L}\,i_2$ *vs.* $\sqrt{C}\,v_C$, the above equation still represents an ellipse. However, the $\sqrt{L}\,i_1 = -\sqrt{L}\,i_2$ plane is at a 45° angle to both the $\sqrt{L}\,i_1$

and the $\sqrt{L}\,i_2$ axis. Taking the additional factor of $\sqrt{2}$ into account, the trajectory projected onto the $\sqrt{L}\,i_1 = -\sqrt{L}\,i_2$ plane follows a circle.

The trajectories in M2, M3, and M4 in discontinuous conduction mode similarly project as circles on the $\sqrt{L}\,i_1 = -\sqrt{L}\,i_2$ plane.

5.2 Topological Mode M0

The trajectory geometry in mode M0 is in fact very similar to the situation in discontinuous mode for M1, except that the plane to which the trajectory is confined is now the $i_1 = i_2$ plane. Since i_1 is equal to i_2, the effective inductance is $2L$, and the system is second-order. The trajectory on the M0 plane follows a circle when plotted against normalized axes of $\sqrt{L}\,i_1$ *vs.* $\sqrt{C}\,v_C$. The velocity fields in Chapter 4 have already demonstrated this circular pattern. In Figures 4-1(a) and (b), the pattern on the left side of the v_C axis is different from that on the right side due to the different signs of v_p on either side of this axis. The sign of v_p affects the center of the circle since the equation governing the relation between $i = i_1 = i_2$ and v_C is

$$\left(\sqrt{2L}\,i\right)^2 + \left[\sqrt{C}\,v_C - \sqrt{C}\,(e_1 - e_2 - 2v_p)\right]^2 = K \tag{5.16}$$

The state transition matrix is really second order, that is,

$$\mathbf{\Phi}_0 = \begin{bmatrix} \cos(\frac{t}{\sqrt{2LC}}) & -\sqrt{\frac{C}{2L}}\sin(\frac{t}{\sqrt{2LC}}) \\ \sqrt{\frac{2L}{C}}\sin(\frac{t}{\sqrt{2LC}}) & \cos(\frac{t}{\sqrt{2LC}}) \end{bmatrix} \tag{5.17}$$

and

$$\mathbf{\Psi}_0 = \begin{bmatrix} (e_1 - e_2 - 2v_p)\sqrt{\frac{C}{2L}}\sin(\frac{t}{\sqrt{2LC}}) \\ (e_1 - e_2 - 2v_p)\left(1 - \cos(\frac{t}{\sqrt{2LC}})\right) \end{bmatrix} \tag{5.18}$$

However, we may write them as a pseudo third-order, as listed in Table 5.1.

Discontinuous conduction mode in M0 is the intersection of the M0 plane and the S3 plane, which is just a part of the v_C axis, as can be seen in Figure 3-1, so

$\mathbf{\Phi}_j(t)$	$\mathbf{\Psi}_j(t)$
$\mathbf{\Phi}_0 = \begin{bmatrix} 0 & \cos(\frac{t}{\sqrt{2LC}}) & -\sqrt{\frac{C}{2L}}\sin(\frac{t}{\sqrt{2LC}}) \\ 0 & \cos(\frac{t}{\sqrt{2LC}}) & -\sqrt{\frac{C}{2L}}\sin(\frac{t}{\sqrt{2LC}}) \\ 0 & \sqrt{\frac{2L}{C}}\sin(\frac{t}{\sqrt{2LC}}) & \cos(\frac{t}{\sqrt{2LC}}) \end{bmatrix}$	$\mathbf{\Psi}_0 = \begin{bmatrix} (e_1 - e_2 - 2v_p)\sqrt{\frac{C}{2L}}\sin(\frac{t}{\sqrt{2LC}}) \\ (e_1 - e_2 - 2v_p)\sqrt{\frac{C}{2L}}\sin(\frac{t}{\sqrt{2LC}}) \\ (e_1 - e_2 - 2v_p)\left(1 - \cos(\frac{t}{\sqrt{2LC}})\right) \end{bmatrix}$
$\mathbf{\Phi}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\frac{t}{\sqrt{LC}}) & -\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ 0 & \sqrt{\frac{L}{C}}\sin(\frac{t}{\sqrt{LC}}) & \cos(\frac{t}{\sqrt{LC}}) \end{bmatrix}$	$\mathbf{\Psi}_1 = \begin{bmatrix} \frac{e_1 - E - v_p}{L}\, t \\ (-e_2 + E - v_p)\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ (-e_2 + E - v_p)\left(1 - \cos(\frac{t}{\sqrt{LC}})\right) \end{bmatrix}$
$\mathbf{\Phi}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\frac{t}{\sqrt{LC}}) & -\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ 0 & \sqrt{\frac{L}{C}}\sin(\frac{t}{\sqrt{LC}}) & \cos(\frac{t}{\sqrt{LC}}) \end{bmatrix}$	$\mathbf{\Psi}_2 = \begin{bmatrix} \frac{e_1 - v_p}{L}\, t \\ (-e_2 - v_p)\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ (-e_2 - v_p)\left(1 - \cos(\frac{t}{\sqrt{LC}})\right) \end{bmatrix}$
$\mathbf{\Phi}_3 = \begin{bmatrix} \cos(\frac{t}{\sqrt{LC}}) & 0 & -\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ 0 & 1 & 0 \\ \sqrt{\frac{L}{C}}\sin(\frac{t}{\sqrt{LC}}) & 0 & \cos(\frac{t}{\sqrt{LC}}) \end{bmatrix}$	$\mathbf{\Psi}_3 = \begin{bmatrix} (e_1 - E - v_p)\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ \frac{-e_2 + E - v_p}{L}\, t \\ (e_1 - E - v_p)\left(1 - \cos(\frac{t}{\sqrt{LC}})\right) \end{bmatrix}$
$\mathbf{\Phi}_4 = \begin{bmatrix} \cos(\frac{t}{\sqrt{LC}}) & 0 & -\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ 0 & 1 & 0 \\ \sqrt{\frac{L}{C}}\sin(\frac{t}{\sqrt{LC}}) & 0 & \cos(\frac{t}{\sqrt{LC}}) \end{bmatrix}$	$\mathbf{\Psi}_4 = \begin{bmatrix} (e_1 - v_p)\sqrt{\frac{C}{L}}\sin(\frac{t}{\sqrt{LC}}) \\ \frac{-e_2 - v_p}{L}\, t \\ (e_1 - v_p)\left(1 - \cos(\frac{t}{\sqrt{LC}})\right) \end{bmatrix}$
$\mathbf{\Phi}_5 = \mathbf{I}$	$\mathbf{\Psi}_5 = \begin{bmatrix} \frac{e_1 - E - v_p}{L}\, t \\ \frac{-e_2 - v_p}{L}\, t \\ 0 \end{bmatrix}$
$\mathbf{\Phi}_6 = \mathbf{I}$	$\mathbf{\Psi}_6 = \begin{bmatrix} \frac{e_1 - v_p}{L}\, t \\ \frac{-e_2 + E - v_p}{L}\, t \\ 0 \end{bmatrix}$
$\mathbf{\Phi}_7 = \mathbf{I}$	$\mathbf{\Psi}_7 = \begin{bmatrix} \frac{e_1 - E - v_p}{L}\, t \\ \frac{-e_2 + E - v_p}{L}\, t \\ 0 \end{bmatrix}$
$\mathbf{\Phi}_8 = \mathbf{I}$	$\mathbf{\Psi}_8 = \begin{bmatrix} \frac{e_1 - v_p}{L}\, t \\ \frac{-e_2 - v_p}{L}\, t \\ 0 \end{bmatrix}$

Table 5.1: State transition matrices in continuous conduction mode.

the trajectory is confined to the axis. Figures 4-1(a) and (b) had shown that the trajectory velocity is zero there.

5.3 Topological Modes M5, M6, M7, and M8

The state-space matrices $\mathbf{A}_j(t)$ for M5, M6, M7, and M8 are all zeros. The trajectories on these mode planes are all linear. The velocity field analysis in Chapter 4 has shown this very clearly. The solution for the state variables in these modes is simply

$$\mathbf{x}(t) = \mathbf{x}(0) + \mathbf{Bu} \cdot t \tag{5.19}$$

where is $\mathbf{Bu}$ is a constant vector for any particular set of values for e_1, e_2, and v_p. Therefore,

$$\mathbf{\Phi}_j(t) = \mathbf{I}, \; j = 5, 6, 7, 8 \tag{5.20}$$

and

$$\mathbf{\Psi}_j(t) = \mathbf{Bu} \cdot t, \; j = 5, 6, 7, 8 \tag{5.21}$$

5.4 A Summary

The trajectory geometry in each individual topological mode is quite simple. When plotted against the normalized axes, they are circles or lines in plane modes. In volume modes, they are spirals riding on cylinders. The cylinders are oriented with axes either parallel to the $\sqrt{L}\,i_1$ axis or the $\sqrt{L}\,i_2$ axis. The center of the circle and the location of the axis of the cylinder are determined by the switching voltages and the transformer primary-side voltage, which also decide the velocity of the trajectory. From any point in the state-space, knowing the topological mode it is in and the switching voltages, we may calculate the trajectory to any future time with Equation 5.10 and the state transition matrices. The complete trajectory is simply a concatenation of the individual pieces in each topological mode and switching state. Complexity arises in the ordering of this sequence of topological modes subject to changes in the

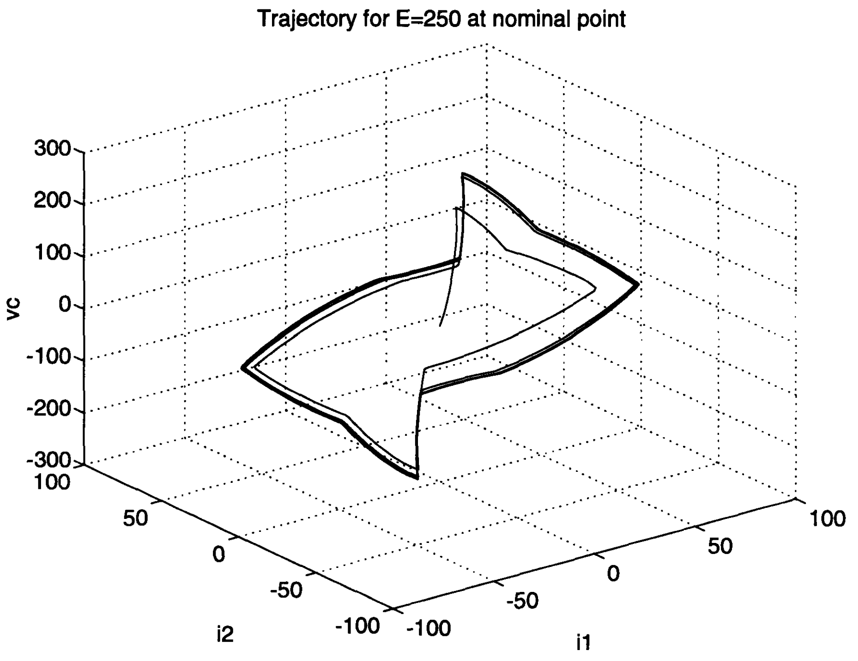


Figure 5-3: 3-D trajectory from zero initial state under nominal operating conditions.

switching voltages.

Figure 5-3[3] shows a typical trajectory from zero initial conditions under nominal operating conditions with $E = 250$ V, $nV_L = 8 \cdot 8.5$ V, $f_s = 275$ kHz, and $\phi = 115.6°$. The graph is generated with the simulation program, which will be discussed later in Chapter 9.

[3]Simulated with *simnom.m*, see Appendix A.

Chapter 6

Steady-State Characteristics

From Figure 5-3, we can see that at least for the circuit parameters used there the series resonant converter settles into a steady-state fairly quickly after only a few cycles from start-up. This rapid settling occurs for almost any values of switching phase angle ϕ, supply voltage E, and load voltage V_L. This fact makes the steady-state simulation of the SRC easy to perform, because if we simply start simulation from the origin, we do not need to wait for an extraordinarily long period of time for the SRC to settle. The steady-state behavior of the SRC was extensively simulated and analyzed by Kato and Verghese in [5]. Osawa verified some of the results and examined some discrepancies [7]. Kato also presented a numerical algorithm for mapping the steady-state operating modes in [4].

6.1 Average Output Current

A primary aim in the design of the converter and a feedback control is to maintain a constant output power. With the assumption that the output voltage, V_L, is relatively constant because of the large load capacitance, the average output current alone determines the output power. The output current is a function of both the supply voltage, E, and the switching phase angle or duty ratio, ϕ. As E and ϕ increase, the output current and thus output power increase. To attain an output power of 4kW with a load voltage of 8.5V, the average output current $\overline{i_o}$, referred to the primary side

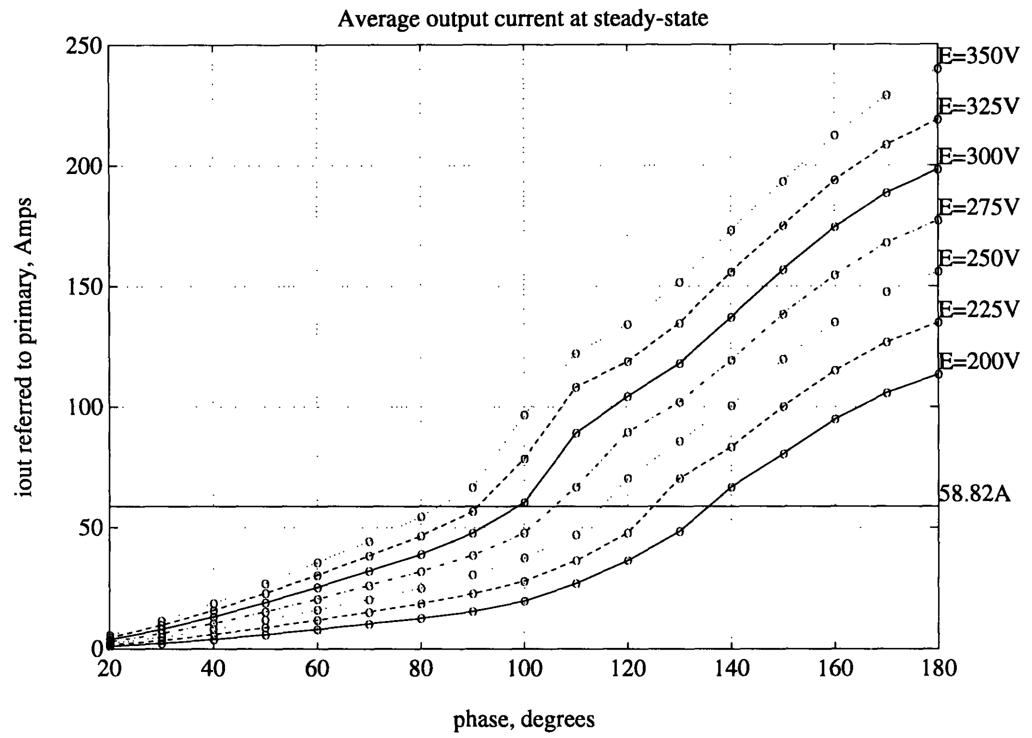


Figure 6-1: Average output current characteristics for various supply voltage levels with $nV_L = 8 \cdot 8.5$V.

of the isolation transformer with a turns ratio of $n = 8$, is $\overline{i_o}/n = 58.82$A. Figure 6-1[1] shows the simulation results of average output current *vs.* phase for various supply voltages. For different combinations of E and ϕ, the steady-state limit cycle may go through different sequences of topological modes. This information, however, is not shown in the figure.

6.2 Operating Modes

The sequence of topological modes that the steady-state limit cycle goes through, *i.e.* the *operating mode*, is dependent not only on supply voltage and switching phase, but also on the output voltage. Although we have approximated the output stage as a constant voltage source, its fluctuation still has some effect on the operating mode.

[1]Simulated with *simsss.m* and drawn by *simsssi.m*, see Appendix A.

However, it is the ratio between E and V_L that influences the mode sequence and not the individual variables. The state-space equations for the topological modes are

$$\dot{\mathbf{x}}(t) = \mathbf{A}_j\mathbf{x}(t) + \mathbf{B}_j\mathbf{u}(t) \tag{6.1}$$

where $\mathbf{B}_j\mathbf{u}(t)$ is a vector whose elements are functions of $e_1(t)$, $e_2(t)$, and $v_p(t)$. Referring to Table 3.2, we see that E and V_L do not enter the state matrices $\mathbf{A}_j$, while $\mathbf{B}_j\mathbf{u}$ is a linear combination of $e_1(t)$, $e_2(t)$, and $v_p(t)$. Normalizing $i_1(t)$, $i_2(t)$, $v_C(t)$, $e_1(t)$, $e_2(t)$, and $v_p(t)$ by a factor K amounts to dividing the state-space equations and boundary conditions by K. That is, the normalized state variables

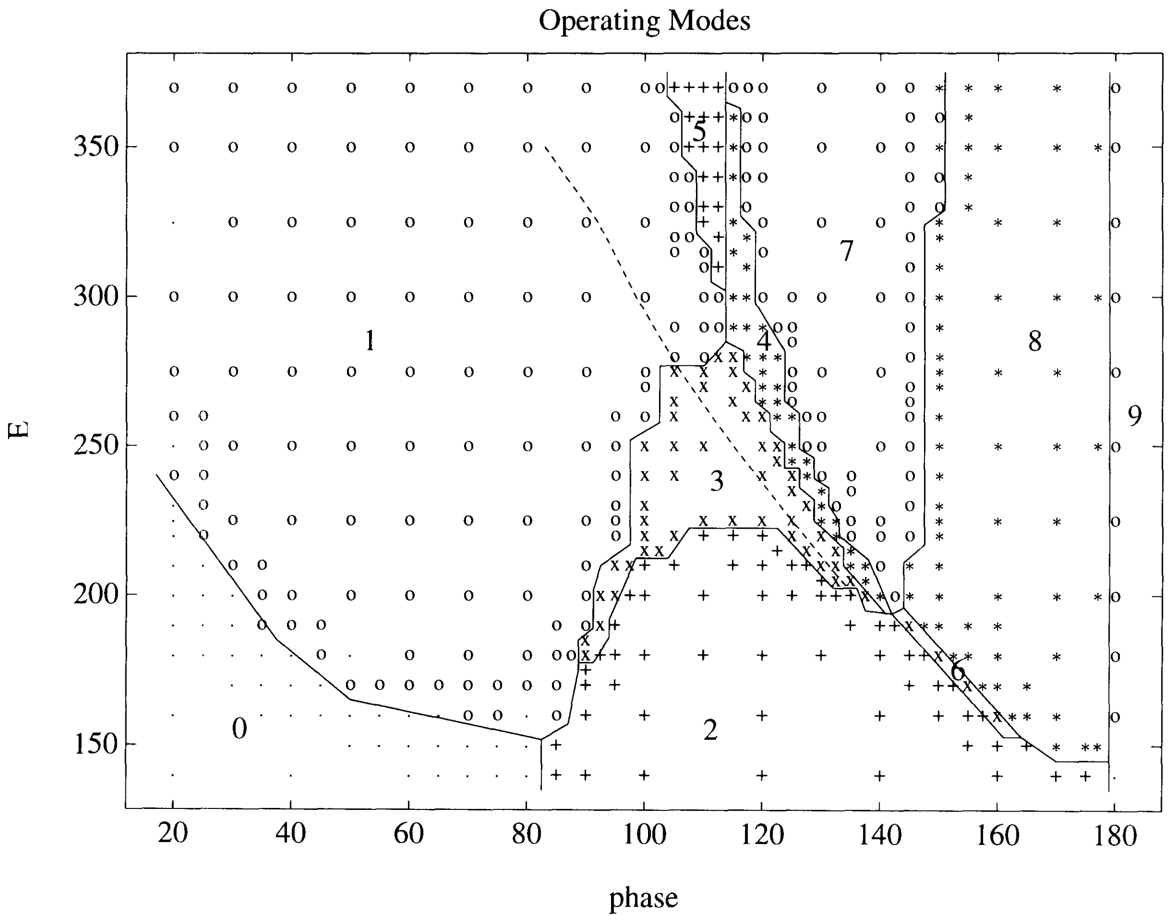
$$\mathbf{x}_{nor}(t) = \begin{bmatrix} i_1(t)/K \\ i_2(t)/K \\ v_C(t)/K \end{bmatrix} \tag{6.2}$$

along with the normalized inputs, $e_1(t)/K$, $e_2(t)/K$, and $v_p(t)/K$, are governed by the same state-space equations and boundary conditions. The topological sequence the trajectory goes through is unaffected by the normalizing factor. Since we have assumed the value of E to be slowly varying and V_L to be a constant, it makes more sense to normalize all the variables by V_L although normalizing by E is also a viable alternative and is done in [4].

In the simulations, we use a nominal value of $V_{L_{nom}} = 8.5$V, and examine the operating modes for different values of E and ϕ. The fluctuation in the load voltage can be translated into a factor onto the value of E. That is, the operating modes for any V_L, E, and ϕ, are the same as with $V_{L_{nom}} = 8.5$V, $E \cdot \frac{V_{L_{nom}}}{V_L}$, and ϕ.

Figure 6-2[2] shows the different operating modes at various combinations of E and ϕ. A binary box numerical method for mapping the operating modes is presented in [4]. Simulations are first done at a relatively sparse grid of points and the operating modes are checked at these points. More grid points near the boundaries are picked and more simulations are done. The boundaries between operating modes can then

[2]Simulated with ***simsss.m*** and ***simssa.m***, and generated by ***simssm.m***, see Appendix A.



	·	Mode 0
	o	Modes 1, 7, 9
Legend:	+	Modes 2, 5
	x	Modes 3, 6
	$*$	Modes 4, 8

Figure 6-2: Operating modes for $nV_L = 8 \cdot 8.5$V.

Operating Mode	Topological Mode Sequence
0	M0-M1-M7-M0-M2-M8
1	M0-M1-M7-M3-M0-M2-M8-M4
2	M0-M1-M0-M2
3	M0-M1-M3-M0-M2-M4
4	M0-M1-M5-M1-M3-M0-M2-M6-M2-M4
5	M0-M1-M5-M1-M7-M3-M0-M2-M6-M2-M8-M4
6	M0-M1-M5-M1-M0-M2-M6-M2
7	M0-M5-M1-M3-M0-M6-M2-M4
8	M0-M5-M1-M0-M6-M2
9	M0-M5-M0-M6

Table 6.1: Operating modes and corresponding topological mode sequences.

be found with successively higher resolution. In [4] and [5], the operating modes of the SRC were examined for a different set of circuit parameters and normalizing factor. The rules of the binary box method were not strictly adhered to in mapping the operating modes shown in Figure 6-2. In addition, the operating modes are labeled according to the associated sequence of M modes only, *i.e.*, the conduction states of the clamping diodes alone. The secondary-side diode modes, S, are not taken into account, so there are fewer operating modes as compared to what was presented in [4, 5]. Understandably, the operating mode map found here does not agree entirely with that in [4, 5].

For the range of E and ϕ simulated and examined, we have found ten steady-state operating modes, listed in Table 6.1 along with their corresponding topological mode sequences. Since the circuit is symmetric and switching is done symmetrically, it is expected that the first half-cycle is symmetric with respect to the second half-cycle. Of particular interest are the operating modes along the $\overline{i_o}/n = 58.82$A line shown in Figure 6-1. This is the dashed line in Figure 6-2. As E changes slowly over time, the switching phase will have to be moved along that line to maintain the constant 4kW output power. Assuming that the output voltage is constant and the supply voltage varies between 350V and 200V, Figure 6-2 indicates that the $\overline{i_o}/n = 58.82$A line cuts through only two steady-state operating modes, 1 and 3. It may enter operating modes 2, 4, or 6 at the low supply voltage end if there are some variations in the output voltage or the supply voltage. Representative plots of the trajectories from zero initial conditions for each of the operating modes except mode 3 are included in Appendix C. The nominal operating point we select is in operating mode 3 and its trajectories are shown in Figures 6-3 and 6-4.

6.3 The Selection of a Nominal Operating Point

Through simulation, we find that a switching phase angle of $\phi = 115.6°$ at $E = 250$V gives us approximately 58.82A of average output current for a nominal load voltage of $V_L = 8.5$V. We have chosen this point to be our nominal operating point since it gives us approximately the desired output power. Figure 6-3 shows the time responses from zero initial conditions of the voltage across the resonant circuit $v_t = e_1 - e_2$, the primary-side transformer voltage v_p, the state variables, i_1, i_2, v_C, and the topological mode numbers (M drawn on the positive part of the axis and S drawn on the negative part). Figure 6-4 shows the projections of this start-up trajectory onto the state-space planes. The 3-D plot of the same trajectory is shown in Figure 5-3. The circuit in steady-state is in operating mode 3 at this nominal point. The topological modes that the steady-state trajectory goes through are the following:

$$\text{M0,S1} \rightarrow \text{M1,S1} \rightarrow \text{M1,S2} \rightarrow \text{M3,S2} \rightarrow \text{M0,S2} \rightarrow \text{M2,S2} \rightarrow \text{M2,S1} \rightarrow \text{M4,S1}$$

During each cycle, two of the transitions in the switching voltage v_t cause a transition in topological modes. They occur at M0$\rightarrow$M1 when v_t transitions from E to 0 and at M0$\rightarrow$M2 when v_t transitions from $-E$ to 0. The trajectories in both cases are on the M0 plane before the transition. When e_1 becomes equal to e_2, the trajectory immediately leaves the M0 plane, as we have said earlier in Chapter 4. The other two transitions only cause the trajectory to change direction. They occur in M1 and M2 when v_t transitions from 0 to $-E$ and from 0 to E.

If we take the switch configuration into consideration, then the steady-state trajectory at the selected nominal point will go through the following cycle, described in terms of the *complete configuration state* of the circuit, (M,S,e_1,e_2), which includes

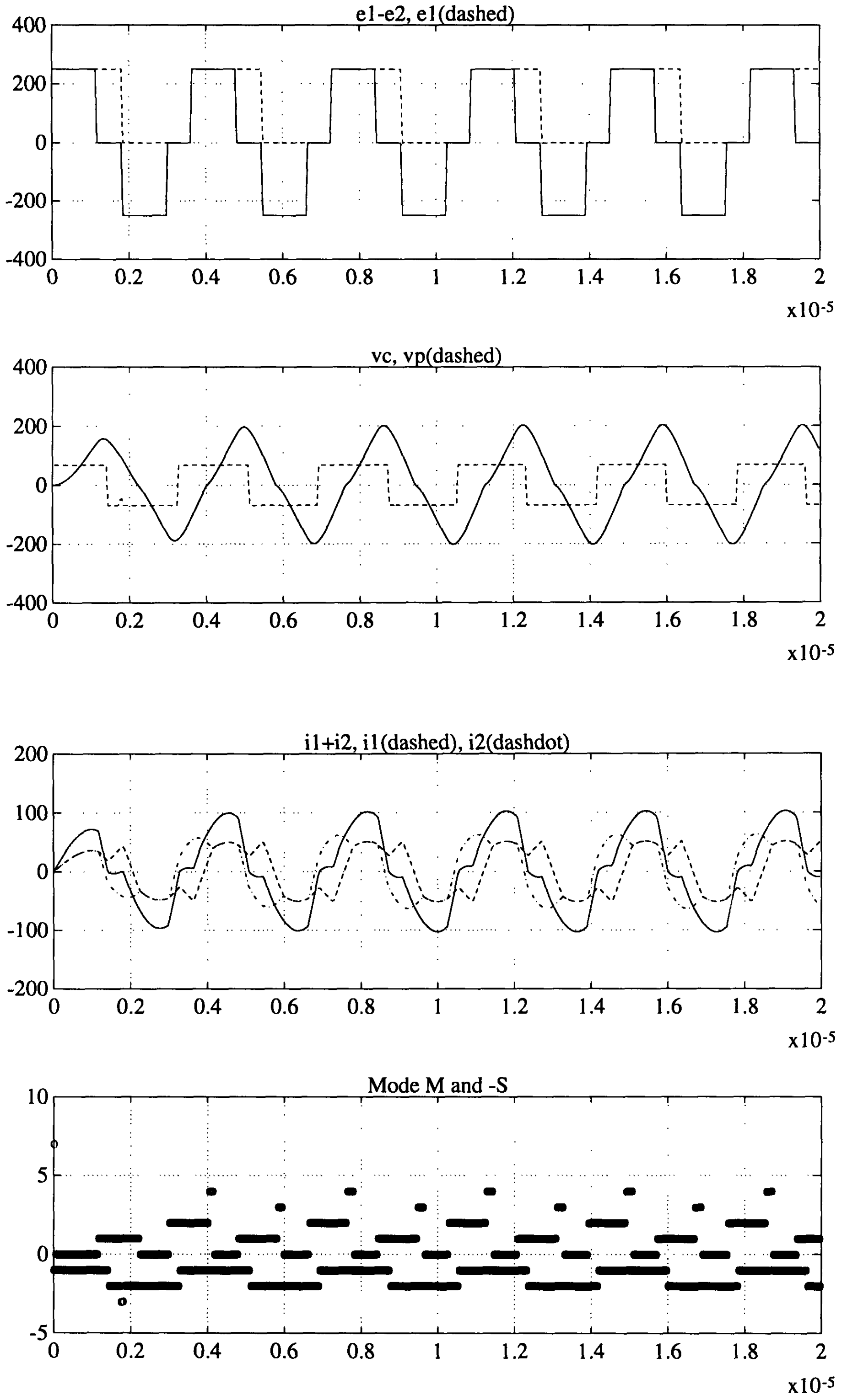


Figure 6-3: Time responses from zero initial state at nominal operating point of $\phi = 115.625°$, $E = 250V$, and $nV_L = 8 \cdot 8.5V$.

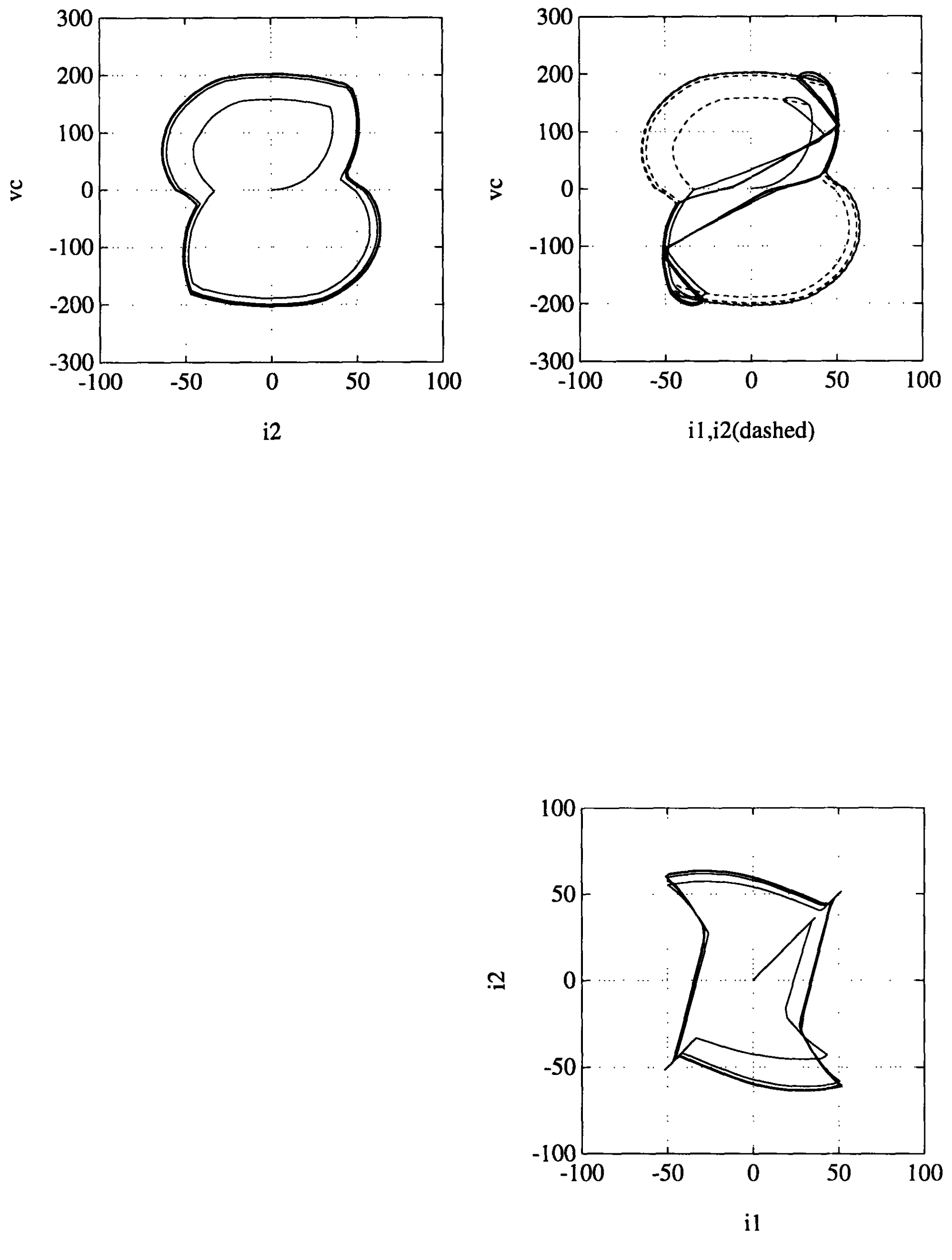


Figure 6-4: Projections of state variables at nominal operating point of $\phi = 115.625°$, $E = 250\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

the clamping diode configuration M, the output diode configuration S, and switch configuration:

$$\text{M0,S1,}E\text{,0} \rightarrow \text{M1,S1,}E,E \rightarrow \text{M1,S2,}E,E \rightarrow \text{M1,S2,0,}E \rightarrow \text{M3,S2,0,}E \rightarrow$$
$$\text{M0,S2,0,}E \rightarrow \text{M2,S2,0,0} \rightarrow \text{M2,S1,0,0} \rightarrow \text{M2,S1,}E\text{,0} \rightarrow \text{M4,S1,}E\text{,0}$$

The duration of a steady-state limit cycle is the period of the switching action. In principle, we may define the beginnings of operating cycles to be at any regularly spaced points in time with intervals equal to the period. However, it is more convenient to define the beginning of a cycle to be at one of the transition points of v_t.

Chapter 7

Analytical Models

After a nominal operating point has been selected at a particular phase angle for some nominal values of supply voltage and output voltage, we may find the steady-state operating mode using simulation. The steady-state trajectory can then be computed with more accuracy using closed-form expressions based on the transition matrices, as shown in Equation 5.10, subject to the boundary conditions listed in Table 3.1. A small-signal sampled-data model can also be derived for the nominal operating point. The small-signal transition matrices may be calculated via simulation or computed by differentiating the large-signal sampled-data model.

7.1 Large-Signal Sampled-Data Model

The state transition matrices derived in Chapter 5 form the basis of the large-signal sampled-data model. Since the circuit is piecewise LTI and closed-form expressions are available for the state variables in any particular topological mode under any particular switching conditions, the complete trajectory in a switching cycle can be explicitly written as a concatenation of the trajectory segments in each topological mode it traverses through. The state of the circuit at the end of a switching cycle can be expressed in terms of its state at the beginning of the cycle, subject to some constraining conditions. The concept is presented in [11] and [1]. It is briefly introduced here before we show its application to our circuit.

7.1.1 Formulation

Suppose that a system operates cyclically, and that the system goes through N switch and diode configurations in the kth cycle. The system in each of the configurations can be described with a linear time-invariant state-space equation

$$\dot{\mathbf{x}}(t) = \mathbf{A}_i\mathbf{x}(t) + \mathbf{B}_i\mathbf{u}(t), \quad i = 1, 2, \ldots, N \tag{7.1}$$

Denote the time at the start of the kth cycle by t_k, and let the transition times from one configuration to the next (relative to the start of the cycle) be put into a vector:

$$\mathbf{T}_k = \begin{bmatrix} T_{k,1} \\ \vdots \\ T_{k,N} \end{bmatrix} \tag{7.2}$$

where $T_{k,N}$ is the duration of the kth cycle, so $t_{k+1} = t_k + T_{k,N}$. All independent controlling parameters may be put into a vector, $\mathbf{p}_k$. These controlling parameters may include some externally controlled switching times and some circuit parameters. The controlling parameters also determine all the source waveforms in $\mathbf{u}$ in the state-space equations. The transition times, $T_{k,i}$, may be directly controlled by external control action or may be indirectly controlled when the states reach some boundaries or threshold conditions. The N equations that govern the relationship between $\mathbf{T}_k$, $\mathbf{p}_k$, and the initial state $\mathbf{x}(t_k)$ can be put into a constraint equation of the form

$$\mathbf{c}(\mathbf{x}(t_k), \mathbf{p}_k, \mathbf{T}_k) = \mathbf{0} \tag{7.3}$$

The large-signal sampled-data description of the circuit is

$$\mathbf{x}(t_{k+1}) = \mathbf{f}(\mathbf{x}(t_k), \mathbf{p}_k, \mathbf{T}_k) \tag{7.4}$$

For piecewise linear time-invariant systems, as in our case, the sampled-data equation takes the form

$$\mathbf{x}(t_{k+1}) = \mathbf{F}(\mathbf{p}_k, \mathbf{T}_k)\mathbf{x}(t_k) + \mathbf{g}(\mathbf{p}_k, \mathbf{T}_k) \tag{7.5}$$

and

$$\mathbf{C}(\mathbf{p}_k, \mathbf{T}_k)\mathbf{x}(t_k) + \mathbf{d}(\mathbf{p}_k, \mathbf{T}_k) = \mathbf{0} \tag{7.6}$$

where $\mathbf{F}$ and $\mathbf{g}$ can be expressed in terms of the $\mathbf{A}_i$, $\mathbf{B}_i$, and $\mathbf{u}(t)$. In steady-state,

$$\mathbf{x}(t_k) = \mathbf{x}(t_{k+1}) = \mathbf{x} \tag{7.7}$$

Further simplification can be made when the circuit is half-cycle symmetric. Let the first half-cycle be governed by

$$\mathbf{x}(t_{2m+1}) = \mathbf{f}_h(\mathbf{x}(t_{2m}), \mathbf{p}_{2m}, \mathbf{T}_{2m}) \tag{7.8}$$

and

$$\mathbf{c}_h(\mathbf{x}(t_{2m}), \mathbf{p}_{2m}, \mathbf{T}_{2m}) = \mathbf{0} \tag{7.9}$$

If the symmetry between the second half-cycle and the first half-cycle can be expressed with a matrix transformation, then

$$\mathbf{W}\mathbf{x}(t_{2m+2}) = \mathbf{f}_h(\mathbf{W}\mathbf{x}(t_{2m+1}), \mathbf{p}_{2m+1}, \mathbf{T}_{2m+1}) \tag{7.10}$$

and

$$\mathbf{c}_h(\mathbf{W}\mathbf{x}(t_{2m+1}), \mathbf{p}_{2m+1}, \mathbf{T}_{2m+1}) = \mathbf{0} \tag{7.11}$$

where

$$\mathbf{W}^2 = \mathbf{I} \tag{7.12}$$

7.1.2 Application to SRC at the Selected Nominal Point

For the nominal point we have selected, we have found through simulation that the complete topological mode sequence, (M,S,e_1,e_2), is

$$\text{M1,S1},E,E \rightarrow \text{M1,S2},E,E \rightarrow \text{M1,S2,0},E \rightarrow \text{M3,S2,0},E \rightarrow \text{M0,S2,0},E \rightarrow$$
$$\text{M2,S2,0,0} \rightarrow \text{M2,S1,0,0} \rightarrow \text{M2,S1},E,0 \rightarrow \text{M4,S1},E,0 \rightarrow \text{M0,S1},E,0$$

as we have shown previously. Let us take the start of a cycle to be at the point where e_2 switches from 0 to E while e_1 is at E. For the case shown in Figure 6-3, in which e_2 lags e_1 by ϕ, this is the point where v_t falls from E to 0. This is also the point where the trajectory leaves the M0 plane and enters M1. Since the system is piecewise linear time-invariant, we may shift the time axis and let the time at the beginning of the cycle be zero. The duration of a cycle, T_N, is equal to the switching period, $T_s = 1/f_s$. Indexing the above configuration sequence from $i = 1$ to $i = 10 = N$, we have

$$\mathbf{x}(T_1) = \mathbf{\Phi}_1(T_1)\mathbf{x}(0) + \mathbf{\Psi}_1(T_1, nV_L, E, E) \tag{7.13}$$

where T_i is the time at the end of the ith configuration state, and $\mathbf{\Phi}_j(t)$, $\mathbf{\Psi}_j(t, v_p, e_1, e_2)$ are the transition matrices (in mode Mj) derived in Chapter 5. At T_1, the trajectory moves from S1 to S2, so the constraint equation is the boundary condition between S1 and S2,

$$i_1(T_1) + i_2(T_1) = 0 \tag{7.14}$$

or in vector form

$$[\,1\;1\;0\,]\begin{bmatrix} i_1(T_1) \\ i_2(T_1) \\ v_C(T_1) \end{bmatrix} = [\,1\;1\;0\,]\cdot\mathbf{x}(T_1) = 0 \tag{7.15}$$

We continue the computation into $i = 2$:

$$\begin{aligned}
\mathbf{x}(T_2) &= \mathbf{\Phi}_1(T_2 - T_1)\mathbf{x}(T_1) + \mathbf{\Psi}_1(T_2 - T_1, -nV_L, E, E) \qquad (7.16)\\
&= \mathbf{\Phi}_1(T_2 - T_1)\left[\mathbf{\Phi}_1(T_1)\mathbf{x}(0) + \mathbf{\Psi}_1(T_1, nV_L, E, E)\right] + \mathbf{\Psi}_1(T_2 - T_1, -nV_L, E, E)\\
&= \mathbf{\Phi}_1(T_2)\mathbf{x}(0) + \mathbf{\Phi}_1(T_2 - T_1)\mathbf{\Psi}_1(T_1, nV_L, E, E) + \mathbf{\Psi}_1(T_2 - T_1, -nV_L, E, E)
\end{aligned}$$

T_2 is a known switching time where e_1 changes from 0 to E. The constraint there is simply

$$T_2 - \frac{\pi - \phi}{2\pi}T_s = 0 \tag{7.17}$$

For $i = 3$, in (M1,S2,0,E),

$$\begin{aligned} \mathbf{x}(T_3) &= \mathbf{\Phi}_1(T_3 - T_2)\mathbf{x}(T_2) + \mathbf{\Psi}_1(T_3 - T_2, -nV_L, 0, E) \\ &= \mathbf{\Phi}_1(T_3)\mathbf{x}(0) + \mathbf{\Phi}_1(T_3 - T_1)\mathbf{\Psi}_1(T_1, nV_L, E, E) \\ &+ \mathbf{\Phi}_1(T_3 - T_2)\mathbf{\Psi}_1(T_2 - T_1, -nV_L, E, E) + \mathbf{\Psi}_1(T_3 - T_2, -nV_L, 0, E) \end{aligned} \tag{7.18}$$

The transition at T_3 is a transition from M1 to M3, so the the constraint equation is

$$v_C(T_3) = 0 \tag{7.19}$$

or

$$[\,0\;0\;1\,] \cdot \mathbf{x}(T_3) = 0 \tag{7.20}$$

The size of the expression for $\mathbf{x}$ gets large very quickly. We will continue with two more steps without expanding the terms. For $i = 4$, in (M3,S2,0,E),

$$\mathbf{x}(T_4) = \mathbf{\Phi}_3(T_4 - T_3)\mathbf{x}(T_3) + \mathbf{\Psi}_3(T_4 - T_3, -nV_L, 0, E) \tag{7.21}$$

with the constraint

$$i_1(T_4) = i_2(T_4) \tag{7.22}$$

or

$$[\,1\;{-1}\;0\,] \cdot \mathbf{x}(T_4) = 0 \tag{7.23}$$

For $i = 5$, in (M0,S2,0,E),

$$\mathbf{x}(T_5) = \mathbf{\Phi}_0(T_5 - T_4)\mathbf{x}(T_4) + \mathbf{\Psi}_0(T_5 - T_4, -nV_L, 0, E) \tag{7.24}$$

with constraint

$$T_5 - \frac{T_s}{2} = 0 \tag{7.25}$$

Since the two half-cycles of the steady-state trajectory are symmetric with respect to the origin, the description for the second half-cycle can be written entirely in terms of the first half-cycle. The matrix, $\mathbf{W}$, that relates the two halves in this case is simply $-\mathbf{I}$. In the formulation of the sampled-data model, we are mostly concerned with the state at the beginning of each cycle. The state at T_i is hidden in the equations. However, the way we have set up the equations for finding the state at the end of the half-cycle, $\mathbf{x}(T_5)$, we can also find the state at all the transition times in the process. This is an advantage since a comparison of simulation results and analytical computation results can be better made when we have all these points.

When the appropriate $\mathbf{\Phi}_j$ and $\mathbf{\Psi}_j$ are substituted into Equation 7.24 and the terms expanded, the symbolic expression will be quite large. It clearly is to our advantage to use a computer program with symbolic manipulation capabilities to work out the details. The actual computation using this large-signal sampled-data model is carried out in Maple. The source codes and the results are listed in Appendix B for reference.

7.1.3 Simulation and Analytical Computation Results

The state-space trajectory may be calculated using the method of assumed state. We first assume that the circuit is in one particular topological mode. We then do some calculations, and check if the conditions for the assumed topological mode are satisfied. If not, we pick another topological mode. This is continued until the right mode is found. However, we have said in Chapter 3 that once we know where the trajectory is in the state-space along with a few conditions as illustrated in Figure 3-1, we know what topological mode it is in. Little effort is needed to find the correct topological mode. Computation of the trajectory can be carried out with Equation 5.10 while checking for changes in topological modes and switching voltages. The simulator written for the converter follows this latter method, except that the trajectory is computed with a trapezoidal integration scheme. Using a numerical integration method, the simulator inevitably has some numerical errors in its computation of the trajectory, so we use it to find the approximate values of the state variables at the beginning of a cycle before we use the analytical model to find more accurate values.

Simulation starts at the origin, and the trajectory quickly settles down to an approximate steady-state within a few cycles. However, the states at the start of the subsequent cycles do not converge to a precise value quickly. Therefore, simulations are run starting from a grid of points near $\mathbf{x}(0)$. The states at the end of the cycle are compared with those at the beginning of the cycle and the one with the smallest deviation is chosen. This process is repeated with finer resolution until we find the $\mathbf{x}(0)$ that closes onto itself after a cycle, with very small error. A similar approach is taken in the search for the steady-state $\mathbf{x}(0)$ using the analytical model. This search process is done at two places: at the beginning of the cycle, where v_t drops from E to 0, and at the point where v_t drops from 0 to $-E$ (see Figure 6-3).

At $v_t : E \longrightarrow 0$, simulation results[1] show that in the steady-state

$$\mathbf{x}(0) = \begin{bmatrix} 46.821 \\ 46.821 \\ 180.088 \end{bmatrix}, \quad \mathbf{x}(T_5) = \begin{bmatrix} -46.832 \\ -46.832 \\ -180.056 \end{bmatrix}, \text{ and } \mathbf{x}(T_{10}) = \begin{bmatrix} 46.821 \\ 46.8195 \\ 180.089 \end{bmatrix}$$

Computation via the large-signal sampled-data model shows that[2]

$$\mathbf{x}(0) = \begin{bmatrix} 46.83790 \\ 46.83790 \\ 179.97310 \end{bmatrix}, \quad \mathbf{x}(T_5) = \begin{bmatrix} -46.83787 \\ -46.83787 \\ -179.97312 \end{bmatrix}, \text{ and } \mathbf{x}(T_{10}) = \begin{bmatrix} 46.83787 \\ 46.83787 \\ 179.97311 \end{bmatrix}$$

At $v_t : 0 \longrightarrow -E$, or T_2, simulation results show that

$$\mathbf{x}(T_2) = \begin{bmatrix} 52.616 \\ -60.116 \\ 108.405 \end{bmatrix}$$

[1]Simulated with *simsyo.m*, see Appendix A.
[2]See Appendix B for the source codes and results.

while computation using the sampled-data model in Maple shows that

$$\mathbf{x}(T_2) = \begin{bmatrix} 52.6590 \\ -60.1024 \\ 108.4017 \end{bmatrix}$$

A more detailed listing of the data points is included in Appendix B.

As we can see, simulation results obtained with a trapezoidal numerical integration method do not differ too much from the results obtained using the closed-form expressions in the analytical model. This validates the accuracy of the simulator. More will be said about the simulator in Chapter 9. The states found for the nominal operating cycle are used next in the computation of the small-signal sampled-data model and the small-signal transition matrix.

7.2 Small-Signal Sampled-Data Model

7.2.1 Formulation

Again, detailed formulation of the procedure for deriving small-signal sampled-data models for power electronic circuits can be found in [11] and [1]. Only an outline of the concept and key equations are presented here.

Once we have the cyclic steady-state description,

$$\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{p}, \mathbf{T}) \tag{7.26}$$

and the constraint equation,

$$\mathbf{c}(\mathbf{x}, \mathbf{p}, \mathbf{T}) = \mathbf{0} \tag{7.27}$$

let us represent the perturbations of the variables as follows:

$$\begin{aligned} \tilde{\mathbf{x}}(t_k) &= \mathbf{x}(t_k) - \mathbf{x} \\ \tilde{\mathbf{p}}_k &= \mathbf{p}_k - \mathbf{p} \end{aligned} \tag{7.28}$$

$$\tilde{\mathbf{T}}_k = \mathbf{T}_k - \mathbf{T}$$

From the large-signal sampled-data model, we have

$$\tilde{\mathbf{x}}(t_{k+1}) = \mathbf{f}(\mathbf{x} + \tilde{\mathbf{x}}(t_k), \mathbf{p} + \tilde{\mathbf{p}}_k, \mathbf{T} + \tilde{\mathbf{T}}_k) - \mathbf{x} \tag{7.29}$$

and the constraint equation,

$$\mathbf{c}(\mathbf{x} + \tilde{\mathbf{x}}(t_k), \mathbf{p} + \tilde{\mathbf{p}}_k, \mathbf{T} + \tilde{\mathbf{T}}_k) = \mathbf{0} \tag{7.30}$$

Retaining only the linear terms from the Taylor expansions gives us

$$\tilde{\mathbf{x}}(t_{k+1}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\tilde{\mathbf{x}}(t_k) + \frac{\partial \mathbf{f}}{\partial \mathbf{p}}\tilde{\mathbf{p}}_k + \frac{\partial \mathbf{f}}{\partial \mathbf{T}}\tilde{\mathbf{T}}_k \tag{7.31}$$

and

$$\frac{\partial \mathbf{c}}{\partial \mathbf{x}}\tilde{\mathbf{x}}(t_k) + \frac{\partial \mathbf{c}}{\partial \mathbf{p}}\tilde{\mathbf{p}}_k + \frac{\partial \mathbf{c}}{\partial \mathbf{T}}\tilde{\mathbf{T}}_k = \mathbf{0} \tag{7.32}$$

where the partial derivatives are Jacobians evaluated at the nominal steady-state $\mathbf{x}$, $\mathbf{T}$, and $\mathbf{p}$. Solve the second equation for $\tilde{\mathbf{T}}_k$,

$$\tilde{\mathbf{T}}_k = -\left(\frac{\partial \mathbf{c}}{\partial \mathbf{T}}\right)^{-1}\left[\frac{\partial \mathbf{c}}{\partial \mathbf{x}}\tilde{\mathbf{x}}(t_k) + \frac{\partial \mathbf{c}}{\partial \mathbf{p}}\tilde{\mathbf{p}}_k\right] \tag{7.33}$$

Substitute it into the first equation, and we have

$$\tilde{\mathbf{x}}(t_{k+1}) = \mathbf{F}_o\tilde{\mathbf{x}}(t_k) + \mathbf{G}_o\tilde{\mathbf{p}}_k \tag{7.34}$$

where

$$\mathbf{F}_o = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} - \frac{\partial \mathbf{f}}{\partial \mathbf{T}}\left(\frac{\partial \mathbf{c}}{\partial \mathbf{T}}\right)^{-1}\frac{\partial \mathbf{c}}{\partial \mathbf{x}} \tag{7.35}$$

and

$$\mathbf{G}_o = \frac{\partial \mathbf{f}}{\partial \mathbf{p}} - \frac{\partial \mathbf{f}}{\partial \mathbf{T}}\left(\frac{\partial \mathbf{c}}{\partial \mathbf{T}}\right)^{-1}\frac{\partial \mathbf{c}}{\partial \mathbf{p}} \tag{7.36}$$

The system is locally asymptotically stable if and only if the eigenvalues of $\mathbf{F}_o$ are within the unit circle.

If we perform Taylor expansions on the large-signal equations for a system with half-cycle symmetry as in Equations 7.8 to 7.11, we have the following.

$$\tilde{\mathbf{x}}(t_{2k+1}) = \mathbf{F}_o\tilde{\mathbf{x}}(t_{2k}) + \mathbf{G}_o\tilde{\mathbf{p}}_{2k} \tag{7.37}$$

and

$$\tilde{\mathbf{x}}(t_{2k+2}) = \mathbf{W}\mathbf{F}_o\mathbf{W}\tilde{\mathbf{x}}(t_{2k+1}) + \mathbf{W}\mathbf{G}_o\tilde{\mathbf{p}}_{2k+1} \tag{7.38}$$

If $\mathbf{W} = -\mathbf{I}$ and $\tilde{\mathbf{p}}_{2k+1} = \tilde{\mathbf{p}}_{2k}$ as in our SRC, then

$$\tilde{\mathbf{x}}(t_{2k+2}) = \mathbf{F}_o^2\tilde{\mathbf{x}}(t_{2k}) + (\mathbf{F}_o\mathbf{G}_o - \mathbf{G}_o)\tilde{\mathbf{p}}_{2k} \tag{7.39}$$

This is the full-cycle model written in terms of matrices derived for the half-cycle model.

7.2.2 Application to SRC at the Selected Nominal Point

Since the SRC circuit is half-cycle symmetric with respect to the origin, the small-signal model in Equation 7.37 is used. The nominal steady-state values of **x**, **T**, and **p** for the first half-cycle have been found both through simulation and with the large-signal model. It is fairly straightforward to set up the equations for the small-signal model. In particular,

$$\mathbf{f}_h = \mathbf{x}(T_5) \tag{7.40}$$

and

$$\mathbf{c}_h = \begin{bmatrix} [\,1\ 1\ 0\,]\cdot\mathbf{x}(T_1) \\ T_2 - \frac{\pi-\phi}{2\pi}T_s \\ [\,0\ 0\ 1\,]\cdot\mathbf{x}(T_3) \\ [\,1\ \text{-}1\ 0\,]\cdot\mathbf{x}(T_4) \\ T_5 - \frac{T_s}{2} \end{bmatrix} \tag{7.41}$$

Expanding the two functions will put them in the form of Equations 7.8 and 7.9, but the messy expressions will only obscure the meaning of the two functions. They are, of course, functions of $\mathbf{x}(T_0)$, and T_i. The other terms in them such as nV_L, E, and ϕ can be taken as the independent controlling parameters, $\mathbf{p}$. We will take one of them, the switching phase angle ϕ, to be the controlling parameter. It is then very easy to compute the Jacobians and set up Equation 7.34 in Maple as shown in Appendix B.

Equation 7.34 and Equation 7.37, however, only calculate the perturbation at the end of a cycle or a half-cycle. Perturbations in transition times can be found with Equation 7.33, but to trace the perturbations in the state variables at these transition times, we need to redefine the small-signal model. For example, to find $\tilde{\mathbf{x}}(T_i)$, we may let

$$\mathbf{f}_h = \mathbf{x}(T_i), \quad i < 5 \tag{7.42}$$

and retain the first i rows in $\mathbf{c}_h$.

7.2.3 Simulation and Analytical Computation Results

The point that we have defined to be the beginning of a cycle is where the trajectory leaves the M0 plane and v_t goes from E to 0. At this point, the circuit is really second-order, so we would expect one of the eigenvalues of $\mathbf{F}_o$ to be 0. However, this does not necessarily mean that if we sample the states at a different point in the cycle, $\mathbf{F}_o$ will also have an eigenvalue of 0. Only 2-D perturbations make sense in the first case while perturbations can be in all three directions in the second case. For this reason, small-signal models are also developed for cycles beginning at the point where v_t drops from 0 to $-E$.

The transition matrix can be constructed numerically by perturbing the state in several directions and finding the perturbations at the end of the cycle while holding the independent controlling parameters constant. Suppose that we use three linearly independent perturbations, $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$, and $\tilde{\mathbf{x}}_3$, at the beginning of the nominal cycle. The perturbations at the end of the cycle are $\mathbf{F}_o\tilde{\mathbf{x}}_1$, $\mathbf{F}_o\tilde{\mathbf{x}}_2$, and $\mathbf{F}_o\tilde{\mathbf{x}}_3$. Putting the vectors into a matrix form, we have

$$\begin{bmatrix} \mathbf{F}_o\tilde{\mathbf{x}}_1 & \mathbf{F}_o\tilde{\mathbf{x}}_2 & \mathbf{F}_o\tilde{\mathbf{x}}_3 \end{bmatrix} = \mathbf{F}_o \begin{bmatrix} \tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \tilde{\mathbf{x}}_3 \end{bmatrix} \tag{7.43}$$

Since $\tilde{\mathbf{x}}_1$, $\tilde{\mathbf{x}}_2$, and $\tilde{\mathbf{x}}_3$ are linearly independent, we have

$$\mathbf{F}_o = \begin{bmatrix} \mathbf{F}_o\tilde{\mathbf{x}}_1 & \mathbf{F}_o\tilde{\mathbf{x}}_2 & \mathbf{F}_o\tilde{\mathbf{x}}_3 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_1 & \tilde{\mathbf{x}}_2 & \tilde{\mathbf{x}}_3 \end{bmatrix}^{-1} \tag{7.44}$$

Sampled at $v_t : E \to 0$, the small-signal transition matrix for the full cycle is

$$\mathbf{F}_o = \begin{bmatrix} -0.0779 & 0.1274 & 0.0318 \\ -0.0779 & 0.1274 & 0.0318 \\ -0.3259 & 0.4910 & 0.1192 \end{bmatrix}$$

with eigenvalues

$$0.1647, \;\; 0.00393, \;\; 0$$

and

$$\mathbf{g}_o = \begin{bmatrix} 0.3432 \\ 0.3432 \\ 5.6156 \end{bmatrix}$$

The eigenvalues found by simulation[3] for the second-order transition matrix at the same starting point are

$$0.165, \;\; 0.0041$$

and the third eigenvalue is taken to be 0.

Sampled at $v_t : 0 \to -E$, the small-signal transition matrix for the full cycle is

$$\mathbf{F}_o = \begin{bmatrix} -0.0877 & -0.0588 & 0.0300 \\ 0.2419 & 0.1604 & -0.0796 \\ -0.2739 & -0.1849 & 0.0959 \end{bmatrix}$$

[3]Simulated with *simsyq.m*, see Appendix A.

with the same eigenvalues

$$0.1647, \quad 0.00393, \quad 0$$

and

$$\mathbf{g}_o = \begin{bmatrix} -0.1158 \\ -1.4622 \\ 4.6304 \end{bmatrix}$$

The transition matrix found via simulation is

$$\mathbf{F}_o = \begin{bmatrix} -0.087 & -0.058 & 0.0299 \\ 0.243 & 0.159 & -0.0793 \\ -0.272 & -0.180 & 0.0954 \end{bmatrix}$$

with eigenvalues

$$0.163, \quad 0.0048, \quad 0.0000$$

A more detailed listing of source codes and data points can be found in Appendix B.

As we can see, simulation results agree well with computed results from the small-signal sampled-data model. Since the nominal cycle goes through the M0 plane, the small-signal transition matrix is only second-order with one additional eigenvalue at 0. The two nonzero eigenvalues have magnitudes smaller than 1. We may conclude that the system is locally stable.

One of the nonzero eigenvalues is almost forty times as large as the other one. This indicates that the system dynamics are mostly first-order. We will exploit this approximation in our small-signal controller design. The dynamics of the system are also quite fast. The largest eigenvalue is 0.1647, so the error is reduced to less than 20% within a single cycle. This is consistent with what we have seen in the (large-signal) start-up simulations in Chapter 6, where the steady-state is reached in only a few cycles.

Chapter 8

Controller Design

Although the series resonant converter with clamped capacitor voltage (SRC) is LTI in each topological mode, it is a nonlinear circuit. As we have seen in the last chapter, the small-signal transition matrix has a dominant eigenvalue of 0.165, which is more than one order of magnitude larger than the other nonzero eigenvalue. As the dynamics of the system is mostly first-order, we may approximately model the converter as a first-order LTI system, design a feedback controller around the nominal operating point, and test the controller under various operating conditions.

8.1 Transfer Function of the Plant

The average output current depends on the switching phase angle, ϕ, and the supply voltage, E. While ϕ is available as a control input, E is not. The output current, i_o, and the load current, i_L, are not available for measurements or feedback because current sensing is difficult in the actual implementation of the circuit. What is available for feedback is the output voltage, v_L. This means that the constant voltage source model for the output stage is too simple for control design. The load may be modeled as either a current source or a resistor in parallel with the output capacitor. The plant to be controlled consists of the converter/load dynamics, with control input ϕ and output v_L (or $n \cdot v_L$ if computation and design are done with variables referred to the primary side of the transformer, as we shall do here for convenience).

The SRC aims to provide a constant output power to the load. We have so far assumed the output to be a voltage source since the output capacitance is large and the output voltage is relatively constant. This translates to an aim for a constant average output current. With the voltage source model becoming invalid, providing constant power to the load now involves controlling the output voltage. Output voltage regulation becomes an important issue.

8.1.1 Approximate Converter Transfer Function

The converter is approxmated as a first-order system with a small-signal continuous-time transfer function from the input signal, $\Delta\phi$, to the output, $\Delta\overline{i_o}/n$, in the form

$$P_C(s) = \frac{\beta}{s + p_1} \tag{8.1}$$

From the small-signal transition matrix, we have found the discrete-time pole to be at the eigenvalue, $\lambda_1 = 0.165$. This translates approximately to a continuous-time pole at

$$\begin{aligned} p_1 &= -\frac{1}{T_s}\ln\lambda_1 \\ &= -275 \times 10^3 \cdot \ln 0.165 \\ &= 495.5 \times 10^3 \end{aligned} \tag{8.2}$$

based on the discrete-time/continuous-time relationship

$$z = e^{-s} \tag{8.3}$$

To find the numerator of the transfer function, we perturb the steady-state system with a step change in ϕ and numerically find the change in the average output current after the circuit settles into steady-state again:

$$\lim_{t\to\infty} \frac{\Delta\overline{i_o}/n}{\Delta\phi} = \lim_{s\to 0} s\frac{\beta}{s+p_1}\frac{1}{s} = \frac{\beta}{p_1} \tag{8.4}$$

Simulation results[1] show that

$$\frac{\beta}{p_1} \approx 2.53$$

Therefore, we have found the approximate transfer function from switching phase angle perturbations, $\Delta\phi$, to the average output current perturbations, $\Delta\overline{i_o}/n$:

$$P_C(s) = \frac{2.53 \cdot 495.5 \times 10^3}{s + 495.5 \times 10^3} \tag{8.5}$$

The supply voltage in the actual system droops at a slow rate over time, from about 350V to 200V. This voltage is treated as a disturbance, and the transfer function from ΔE to $\Delta\overline{i_o}/n$ is assumed to have a similar form as $P_C(s)$:

$$P_E(s) = \frac{\gamma}{s + p_1} \tag{8.6}$$

The constant in the numerator, γ, is found with simulation in the same way as β, by perturbing the nominal point with a step change in E and finding the change in $\overline{i_o}/n$. Simulation results show that

$$\frac{\gamma}{p_1} \approx 0.91$$

so the transfer function from supply voltage perturbations, ΔE, to the average output current perturbations, $\Delta\overline{i_o}/n$, is

$$P_E(s) = \frac{0.91 \cdot 495.5 \times 10^3}{s + 495.5 \times 10^3} \tag{8.7}$$

These two transfer functions are the approximate description of the SRC circuit around the nominal operating point of $\phi = 115.625°$ and $E = 250$V. They serve as a guide to the design of a small-signal feedback controller.

[1]Simulated with ***simplt.m***, see Appendix A.

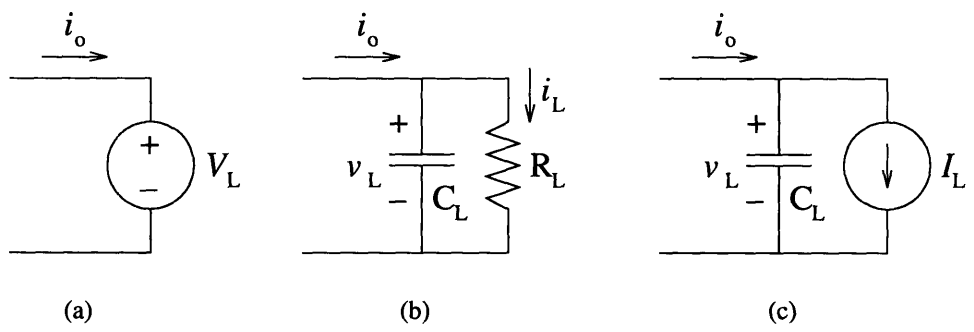


Figure 8-1: Three different models for the output stage: (a) constant voltage source, (b) capacitor in parallel with resistor, and (c) capacitor in parallel with current source.

8.1.2 Output Stage and Load Requirements

The converter needs to supply a constant power of 4kW at full-load. The power demand on the SRC alternates between full-load and no-load, but at a rate that is much slower than the dynamics of the converter. As the load capacitance is very large, the average load voltage does not change very fast over time. It is reasonable to assume that the load is a voltage source, as shown in Figure 8-1(a) when we examine the dynamic behavior of the converter, especially since the converter exhibits very fast dynamics and converges to the steady-state in only a few switching cycles, before either the supply voltage variation or the load voltage variation become appreciable. However, since the output current is not available for measurement and the output voltage has to be used as the feedback signal, the constant voltage source model for the load becomes invalid for control design. It becomes necessary to include the dynamics of the output stage, particularly at the transitions between full-load and no-load. At such transitions, the sudden application or withdrawal of current demand by the load will cause an appreciable drop or rise in the output capacitor voltage even though it has a large capacitance value. Since the behavior of the next stage in the power system is not entirely clear, whether the load is better modeled as a resistor or as a current source is uncertain. The simulator of the SRC is able to use any of the three types of output models shown in Figure 8-1. While the dynamics of the converter have been simulated assuming the constant voltage source model, the design of the feedback controller and the performance testing of the complete system must be done for both the resistor model and the current source model.

For the resistor model in Figure 8-1(b), the transfer function from i_o to v_L is simply

$$P_O(s) = \frac{R_L}{s\,R_L C_L + 1} = \frac{1/C_L}{s + 1/R_L C_L} \tag{8.8}$$

With component values, $R_L = 0.018625\Omega$ and $C_L = 0.01$F, we have

$$P_O(s) = \frac{100}{s + 5.536 \times 10^3} \tag{8.9}$$

The small-signal transfer function from Δi_o to Δv_L is the same. The transfer function of the averaged quantities from $\Delta \overline{i_o}$ to $\Delta \overline{v_L}$ is also the same since the output is LTI [3]. The transfer function under no-load condition can be obtained by letting R go to ∞, which yields $1/sC_L$.

For the current source model in Figure 8-1(c), the small-signal transfer function from Δi_o to Δv_L is

$$P_O(s) = \frac{1}{sC_L} \tag{8.10}$$

The transfer function of the averaged quantities and that under no-load condition are the same.

In the constant voltage source model, the output current of the SRC is the same as the load current. Controlling the output power to the load is the same as controlling the output current of the converter. With the resistor model for the load, the load current is proportional to the load voltage. Controlling the power delivered to the load is equivalent to controlling the load voltage. With the current source model, controlling power delivered to the load is again equivalent to controlling the load voltage since now the load current is assumed to be constant. In either of the latter two cases, maintaining a constant power to the load hinges upon maintaining a constant load voltage. Of course, variations in the load voltage depend on the output current of the SRC. An increase or decrease in the output current from nominal will raise or

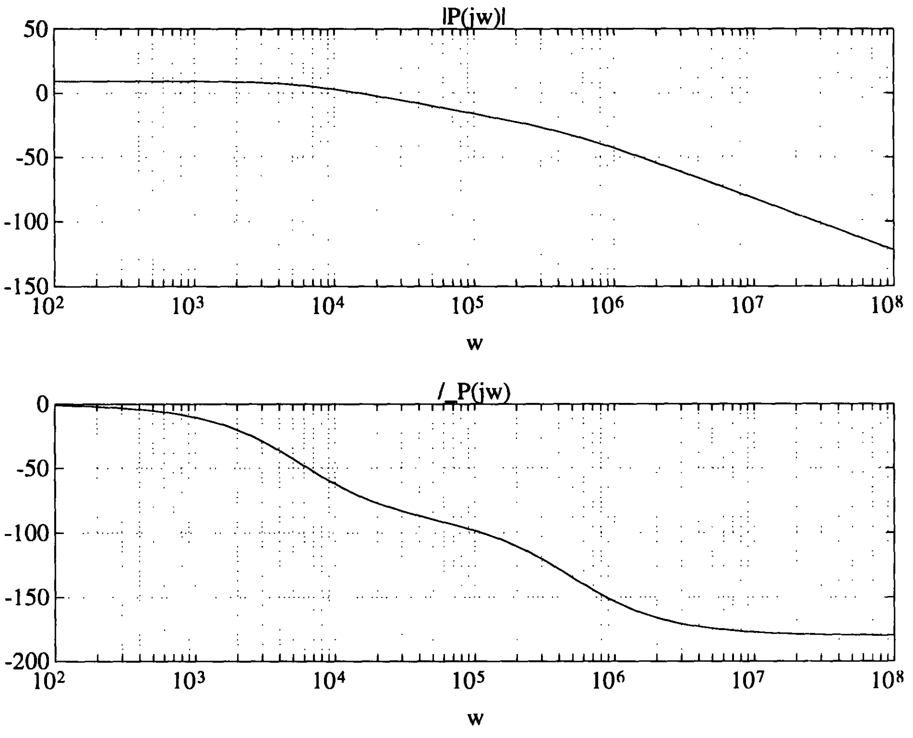


Figure 8-2: Bode plots of the plant transfer function.

lower the load voltage and thus affect the power delivered to the load.

8.1.3 Approximate Plant Transfer Function

The plant transfer function is simply the product of $P_O(s)$ and $P_C(s)$. All variables are referred to the primary side of the transformer, so the combined transfer function will have a factor of n^2, where n is the turns ratio of the transformer.

$$P(s) = n^2 P_O(s) P_C(s) \tag{8.11}$$

Based on this transfer function, we will design a small-signal LTI feedback controller. Figure 8-2 shows the Bode plots of the plant when the load is modeled as a resistor. For both resistive and current-source loads, the dominant dynamics is that of the load; the converter current dynamics reflected in $P_C(s)$ is much faster.

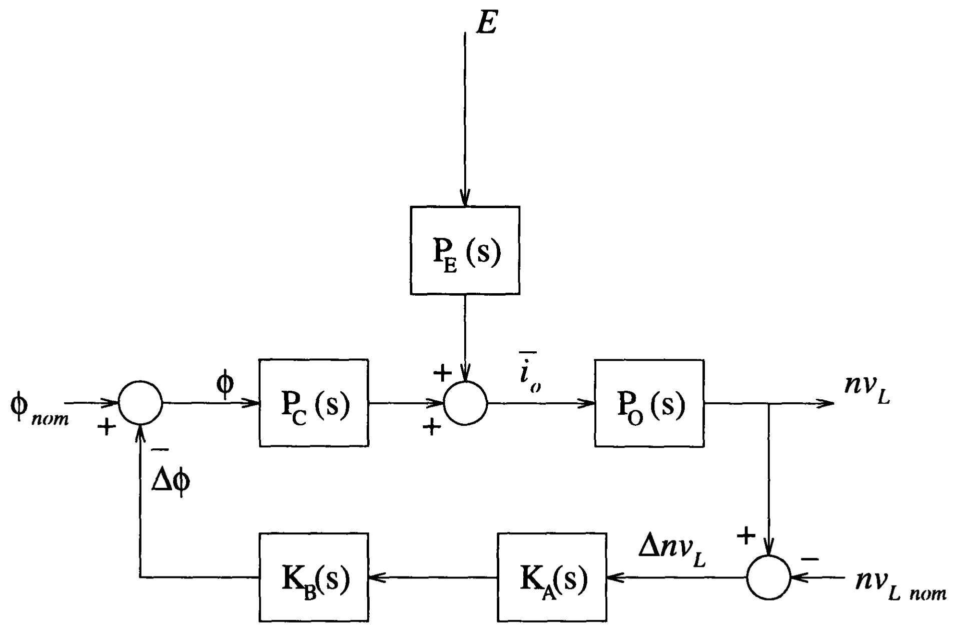


Figure 8-3: System with small-signal feedback controller.

8.2 Small-Signal Feedback Controller and Performance Evaluation

A small-signal feedback controller is designed for the nominal operating point using classical control theory. The controller takes the continuous-time Δnv_L (the deviation in the output voltage from the nominal value) as its input, and generates a continuous-time phase correction $\Delta\phi$. A sampled version of the continuous time $\Delta\phi$ at the beginning of each switching period is subtracted from the nominal switching phase angle.

The continuous-time output voltage, nv_L, is available for feedback. However, it is the average output voltage that we seek to control, so the averaged deviation of the output voltage from its nominal value is of interest. The first piece of the feedback controller is an averager in the form of a low-pass filter, with transfer function

$$K_A(s) = \frac{p_A}{s + p_A} \tag{8.12}$$

The switching frequency of the circuit is at 275kHz, so the rectified output current and thus the output voltage is at twice this frequency, at 3.46×10^6 radians/sec. For a corner frequency at $p_A = 2.3 \times 10^6$, the attenuation above 3.46×10^6 is roughly more than 1/2. More substantial low-pass filtering will require us to take the dynamics of the averaging circuit into account during the design process, so we settle for this. Therefore, we take

$$K_A(s) = \frac{2.3 \times 10^6}{s + 2.3 \times 10^6} \tag{8.13}$$

The output of the averager, $n\overline{v_L}$, is compared to the nominal reference output voltage, $nv_{L_{nom}}$, and the difference is the input to the next stage of the feedback controller. However, since the reference is a constant value, the comparison can be done before averaging, as shown in Figure 8-3. The averager, $K_A(s)$, takes the difference between the output voltage and the reference as its input.

The simplest small-signal controller is a proportional-gain feedback controller. However, to have better disturbance rejection at low frequencies, a proportional-plus-integral (PI) controller is used. This gives a higher loop-gain at lower frequencies and thus a lower sensitivity value. The transfer function of the feedback controller is of the form

$$K_B(s) = \frac{k_B(s + z_B)}{s} \tag{8.14}$$

Combined with the low-pass filter $K_A(s)$, the complete feedback controller transfer function is

$$K(s) = K_B(s)K_A(s) \tag{8.15}$$

The dynamics of the feedback controller should be slower than the dynamics of the converter current dynamics in $P_C(s)$, so that the two do not interact closely. We know that the converter has a pole at $p_1 = 495.5 \times 10^3$. While we need to maximize the cross-over frequency of the loop gain, $T(s) = P(s)K(s)$, and maintain sufficient phase margin, we also need to keep the cross-over frequency much lower than p_1. One of the closed-loop poles tends towards the open-loop zero at z_B, so for faster convergence of the closed-loop system, we also wish to push z_B higher while still retaining sufficient phase margin. Since the plant transfer function is only an approximation of the actual

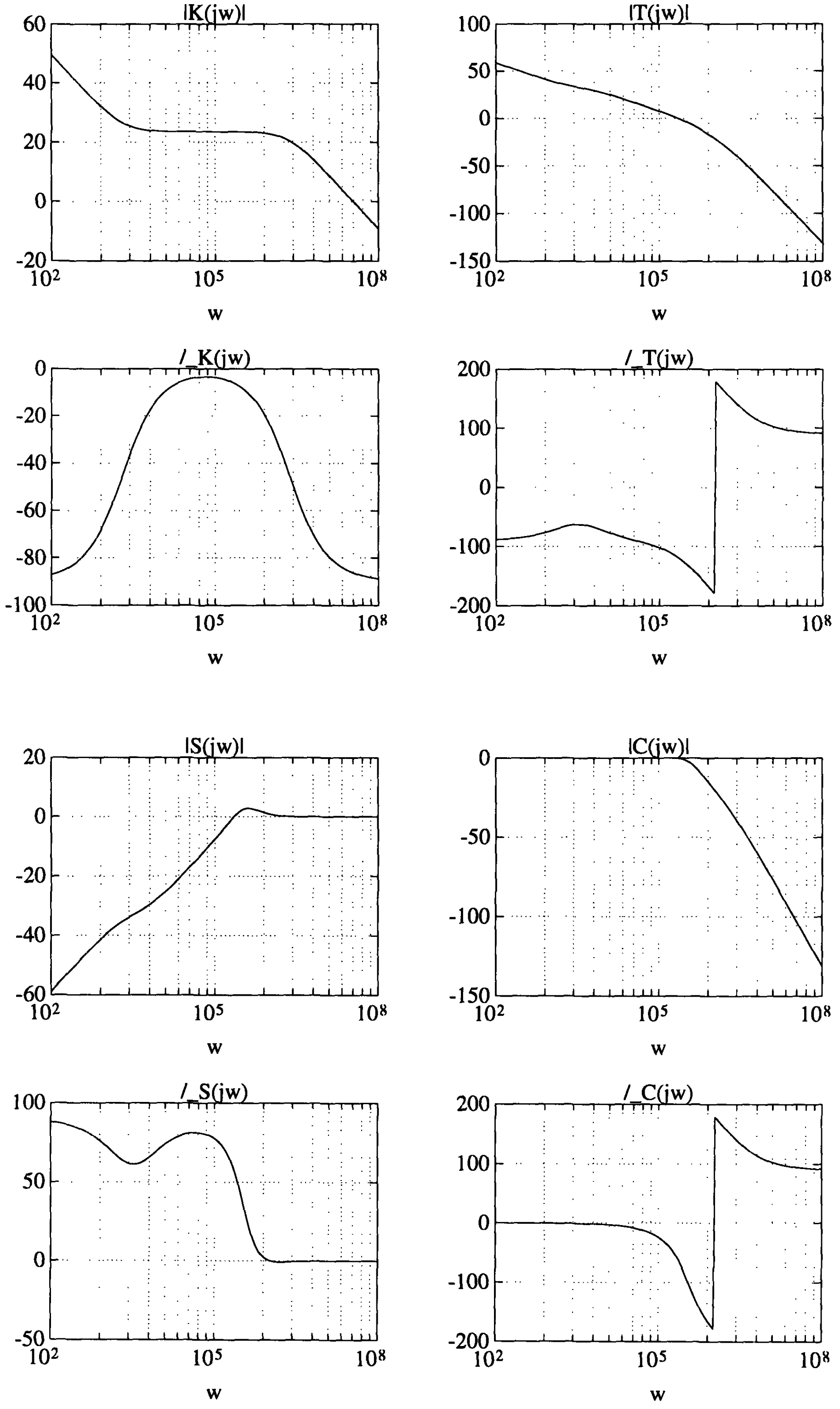


Figure 8-4: Bode plots of transfer functions $K(s), T(s), S(s)$, and $C(s)$.

system, we must have enough phase margin in order to avoid possible instability. For

$$K(s) = \frac{15(s+2000)}{s}\frac{2.3\times10^6}{s+2.3\times10^6} \tag{8.16}$$

Figure 8-4 shows the Bode plots of the transfer functions of the feedback controller, $K(s)$, the loop gain, $T(s)$, the sensitivity, $S(s) = 1/(1+T(s))$, and the closed-loop $C(s) = T(s)/(1+T(s))$. From the Bode plots, we see that the cross-over frequency of $|T(jw)|$ is at about 2.1×10^5 with a phase margin of about 63°. The closed-loop system has poles at $s = -2.36\times10^6, s = -2.18\times10^5 \pm j2.67\times10^5, s = -1.97\times10^3$. The complex conjugate pair has an imaginary part equal to 42.5kHz, which is about 6.5 times slower than the switching frequency. The pole at $s = -1.97\times10^3$ is very close to the open-loop zero at $s = -2000$. A lower p_A will average better and a higher z_B will let the closed-loop settle to steady-state faster. However, as we can see from the phase plots of $K(s)$ and $T(s)$, they will reduce the phase margin. A higher gain, k_B, will also reduce the phase margin. Although a phase margin of 30° to 40° may be sufficient, the modeling errors and approximations we have made in designing the controller may necessitate a larger margin. Simulations actually show that when the gain, k_B, is increased three times to 45, the closed-loop system becomes unstable in some cases, even though there is substantial phase margin left in the Bode plots.

Simulations[2] of the SRC with the feedback control system are run to validate and verify the controller design. For the resistor load model, Figure 8-5(a) shows the output voltage after the SRC has settled into the steady-state without feedback control. Since the nominal phase angle, $\phi = 115.625°$ is only an approximation, the output current it generates does not exactly equal the desired output current for 4kW output power. The output voltage in Figure 8-5(a) slowly settles to a lower value than the nominal $8\cdot8.5$V, indicating that $\phi = 115.625°$ is smaller than the phase angle need to produce 4kW of output power. Figure 8-5(b) shows the same situation with the feedback controller. With feedback, the average output voltage is maintained closely around the nominal, without the drooping seen in (a). The smooth curve for

[2]Simulated with *simctr.m*, see Appendix A.

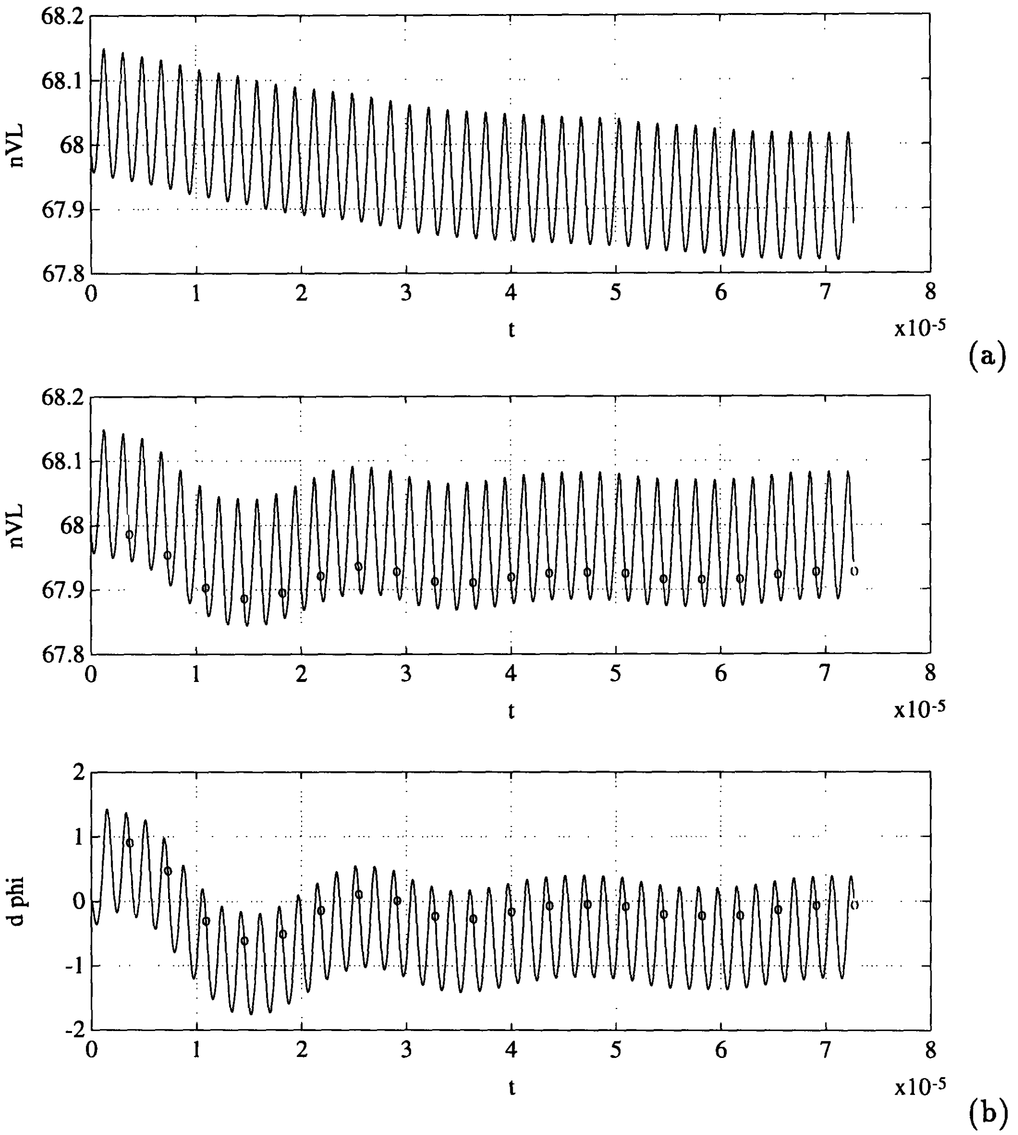


Figure 8-5: Output voltage, nv_L, (a) without feeback and (b) with feedback, and feedback signal $\Delta\phi$.

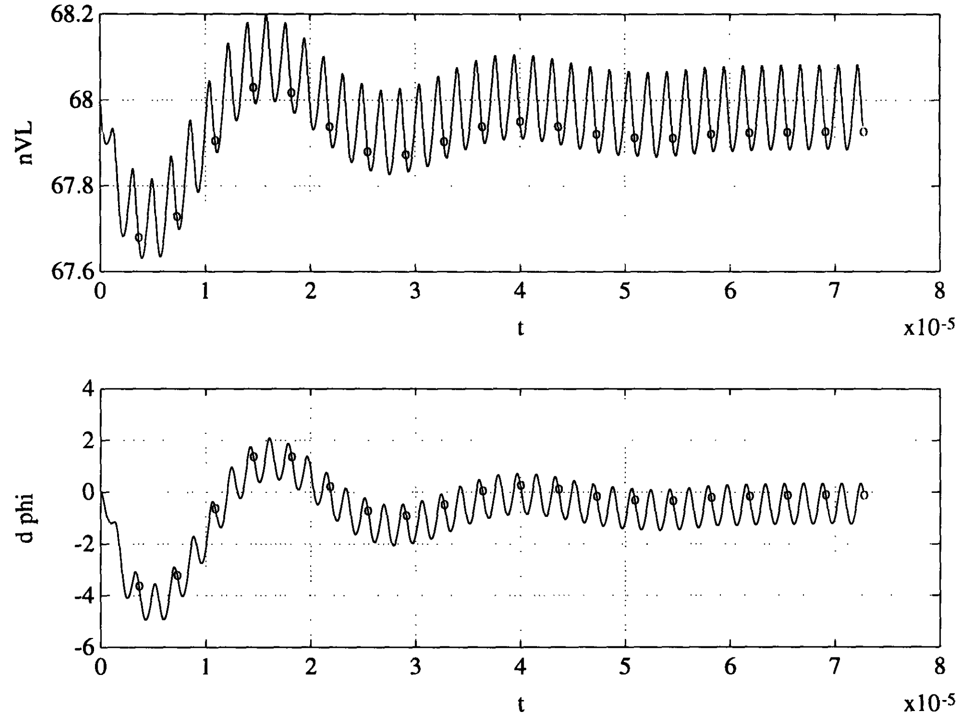


Figure 8-6: Output voltage, nv_L, and feedback signal, $\Delta\phi$, for system starting with $i_1(0) = 0, i_2(0) = 0, v_C(0) = 0$.

nv_L shows the continuous-time output voltage, while the circle marks overlaid are its values at the sample times, which are the beginnings of switching cycles. The smooth curve for $\Delta\phi$ is the output of the feedback controller while the circle marks are the values used to correct the phase for the switching cycle. The values of $\Delta\phi$ do not necessarily agree with the actual output from the feedback controller when control limits, nonlinear control, or disturbance feedforward come into play. Simulations for the current model show very similar results and are not repeated here.

The behavior of the output voltage for the resistor output model when the SRC starts from the zero initial state (but with the load voltage at nominal) is shown in Figure 8-6. The oscillation in nv_L and $\Delta\phi$ is roughly 6 times the period of the switching frequency, as can be seen from the graphs in Figure 8-6 as well as in Figure 8-5(b), and is due to the two complex poles at $s = -2.18 \times 10^5 \pm j2.67 \times 10^5$. For the number of cycles simulated, the average output voltage does not reach precisely the

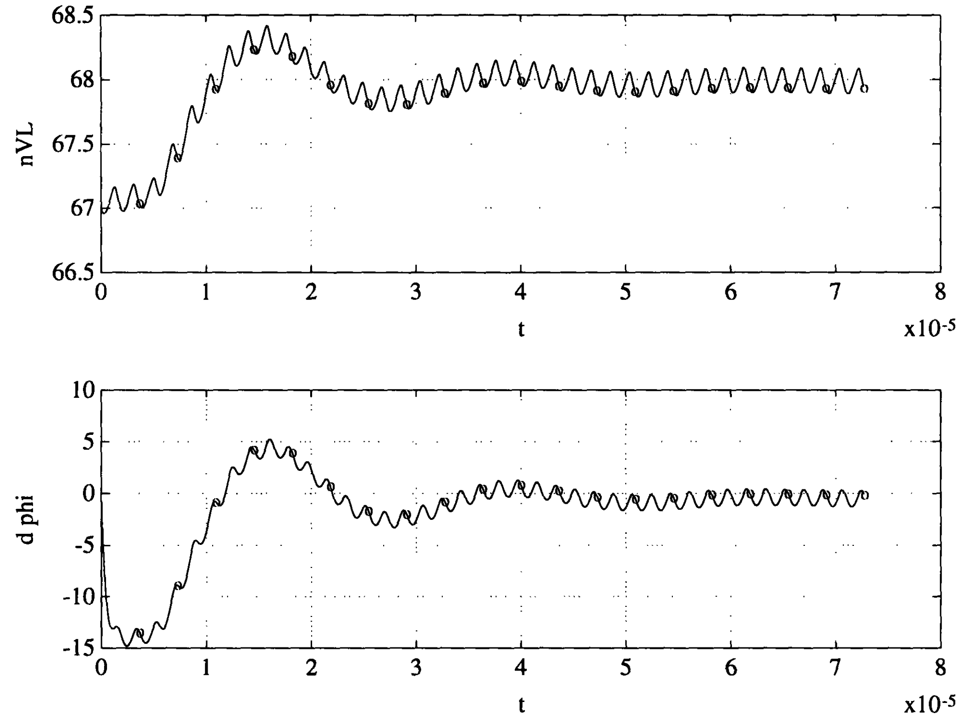


Figure 8-7: Output voltage, nv_L, and feedback signal, $\Delta\phi$, with initial $\Delta nv_L = -1\text{V}$.

desired nominal value, but recovers to the value at a rate determined by the slow pole at $s = -1.97 \times 10^3$.

The behavior of the output voltage for the resistor output model with an initial deviation of -1V in the output voltage is shown in Figure 8-7. The small-signal feedback controller is able to bring the output voltage to near the nominal value in roughly 10 cycles.

In all these cases, the output voltage does not deviate from the nominal by too much. Simulation results for the resistor load model or the current source load model, therefore, differ only slightly. We have presented only the graphs for the resistor load model here.

8.3 Nonlinear Control Rules and Performance Evaluation

To deal with large deviations from the nominal, nonlinear control rules are added to the small-signal feedback controller, to achieve overall large-signal and small-signal control.

The phase correction signal $\Delta\phi$ that is subtracted from the nominal phase angle may be outside of the allowed range as a result of large deviation in output voltage. So the first rule is that $\phi_{nom} - \Delta\phi$ has to be within the allowed range of 0 to π. The maximum possible output current is generated at $\phi = 180°$ and the minimum, or no, current is generated at $\phi = 0°$. This means that $\Delta\phi$ is clamped to between $-64.375°$ and $115.625°$.

Another situation where nonlinear control is necessary is when the output goes from full-load to no-load or vice versa. When the load demand is withdrawn, the output current charges up the output capacitor. The phase angle will be reduced from the nominal value. However, the design for the small-signal feedback controller does not have a gain high enough for this large-signal case. The phase correction is not enough to quickly counter the rise in output voltage. If it is possible to sense the load current, then a change of the reference phase angle to zero at the transition from full-load to no load can deal with the problem. With only voltage sensing, the nonlinear control rule will be to maximize output current when the sampled output voltage drops below a threshold, and to minimize the current when the voltage rises above a threshold. At nominal, the average output current $\overline{i_o}/n$ is 58.82A. In one switching cycle, $T_s = 1/f_s = 3.63 \times 10^6$sec, the change in the output voltage is

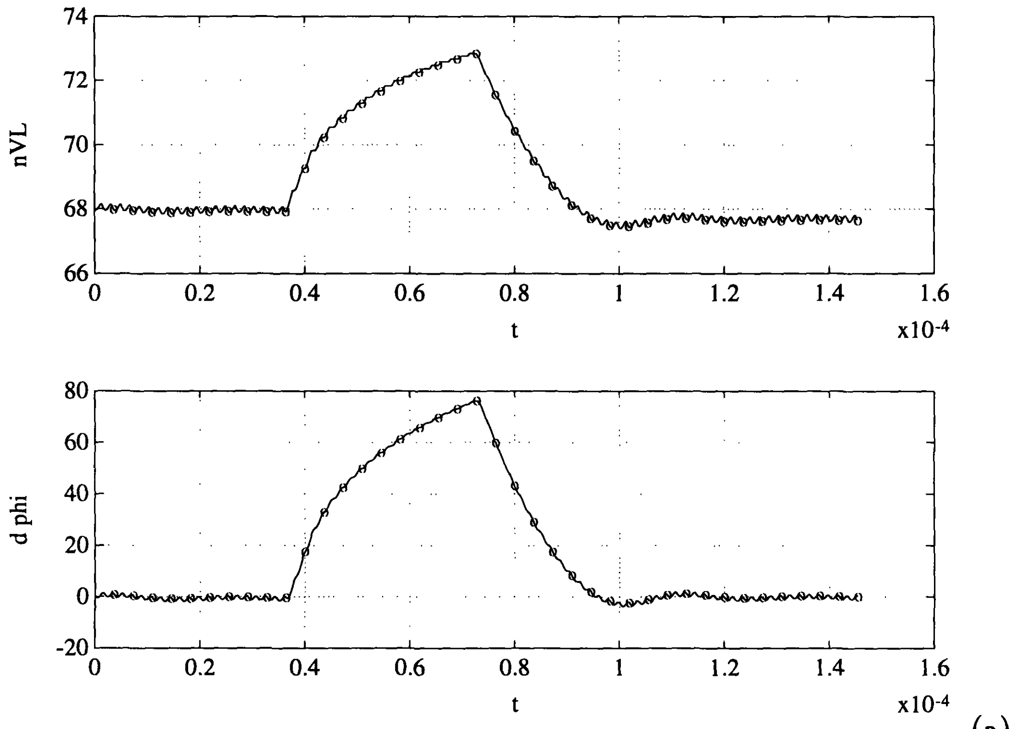
$$\Delta v_L = \frac{1}{C_L}\,\overline{i_o}\,T_s = \frac{1}{0.01} \cdot 8 \cdot 58.82 \cdot 3.63 \times 10^6 = 0.17\text{V} \qquad (8.17)$$

When referred to the primary side, $\Delta n v_L = 1.37$V. The nonlinear control threshold above which minimum current will be delivered is set at 1.4V above the nominal $nv_{L_{nom}} = 8{\cdot}8.5 = 68$V. The threshold below which maximum current will be delivered is set at 1.4V below the nominal.

In Figure 8-8(a), after the transition from full-load to no-load, the output voltage and the phase correction grow exponentially to some limiting values, but the phase correction is insufficient to curb the rise in the output voltage. (The subsequent return to nominal values corresponds to the full load being reapplied.) Ideally, the switching phase should go to zero when the converter enters no-load. The simulation is done with the resistor output model. No-load is modeled by increasing the load resistance by 100 times so that the quiescent current at no-load is roughly 1/100 of the full-load current. The comparison shown in Figure 8-8(b) with nonlinear control shows that the output voltage is limited to a much smaller range dictated by the threshold value. Since the converter cannot absorb power, we must rely on the quiescent current at no-load to slowly discharge the capacitor. Once the load reverts back to full-load and draws the nominal current again, the controller is able to stablize the output voltage within 10 to 15 cycles. In the actual power supply system, the no-load duration is probably much longer than simulated here and the output capacitor will be able to discharge fully to the nominal voltage. The simulation shown in Figure 8-8(b) demonstrates that even if the quiescent current at no-load is not sufficient to discharge the capacitor, the system is still well behaved. Again, simulation assuming a current source output model shows very similar behavior because the deviation in the output voltage from its nominal is not large. Only results for the resistor output model are presented here.

How the feedback controller deals with even larger deviation in the output voltage is shown in Figure 8-9. In this case, $\Delta n v_L = -18$V, such that the initial voltage is at 50V, well below the nonlinear control threshold. This causes the phase correction to dip below $-250°$ initially, so the phase correction is clamped to $-64.375°$. With maximum phase, the output voltage quickly rises to its nominal value. With a large deviation in the output voltage, the difference between the responses of a resistor output model and a current source output model becomes noticeable.

From the examples we have shown, we can see that the small-signal feedback controller with nonlinear large-signal control rules can deal adequately with deviations



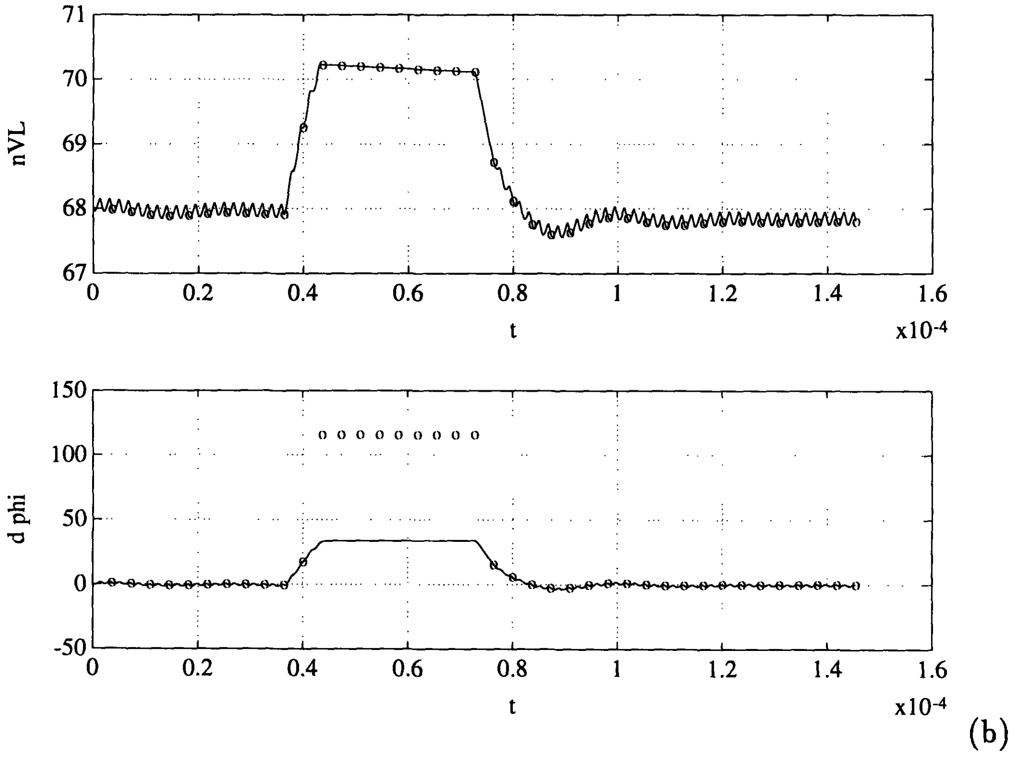


Figure 8-8: Output voltage, nv_L, and feedback signal, $\Delta\phi$, (a) from full-load to no-load with only linear feedback (b) from full-load to no-load to full-load with nonlinear control.

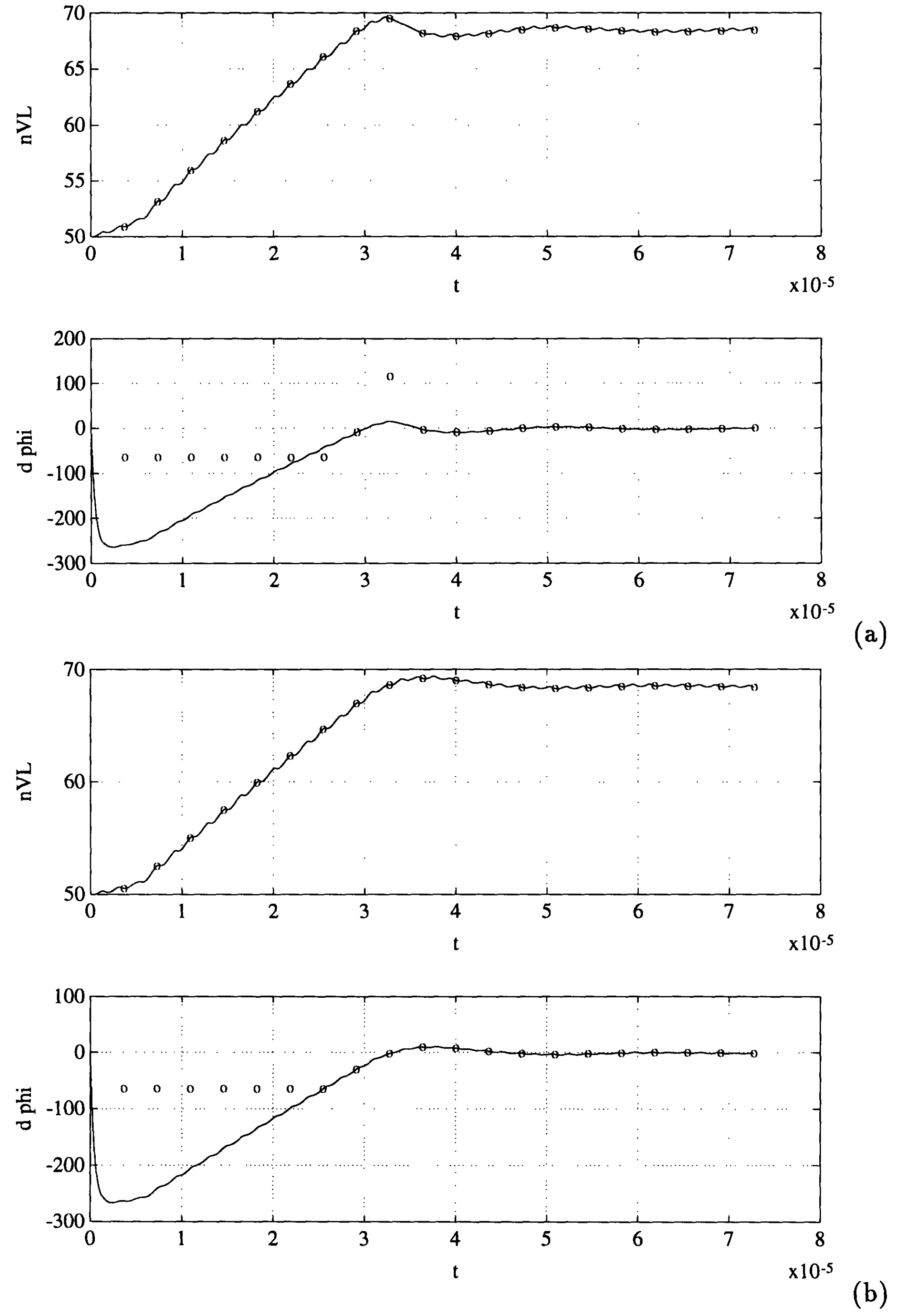


Figure 8-9: Output voltage, nv_L, and feedback signal, $\Delta\phi$, from initial $nv_L = 50$V (a) with a resistor output model and (b) with a current source model.

in the output voltage and the transition between full-load and no-load conditions. Whether the output is modeled as a resistor or a current source does not effect system performance in any fundamental way.

8.4 Disturbance Feedforward and Performance Evaluation

The supply voltage of the converter droops over time at a slow rate. It has been treated as a disturbance to the system. However, the small-signal feedback system is not adequate in handling a drooping supply voltage. Figure 8-10(a) shows the response of the output voltage from nominal operation but with a supply voltage, E, that ramps down from 250V at -1V per cycle. As the supply voltage decreases, a larger phase angle is needed to maintain the nominal output current. However, the feedback controller is unable to produce enough phase correction. A second PI stage was added to the feedback controller in an attempt to improve sensitivity characteristics at low frequencies and to combat ramp function disturbances, but little improvement was obtained.

The supply voltage as a disturbance is measurable, so a disturbance feedforward scheme is feasible. The block diagram of the complete system with the feedforward controller and nonlinear control is shown in Figure 8-11. We have assumed the transfer function from disturbance to output is in a similar form to that from control input to output. The feedforward controller, therefore, is simply a constant gain,

$$K_E(s) = \frac{\gamma}{\beta} \tag{8.18}$$

With simulation results, we have

$$K_E = \frac{0.91}{2.53} = 0.36$$

Note that the K_E found here is only for the nominal operating point. From Figure 6-

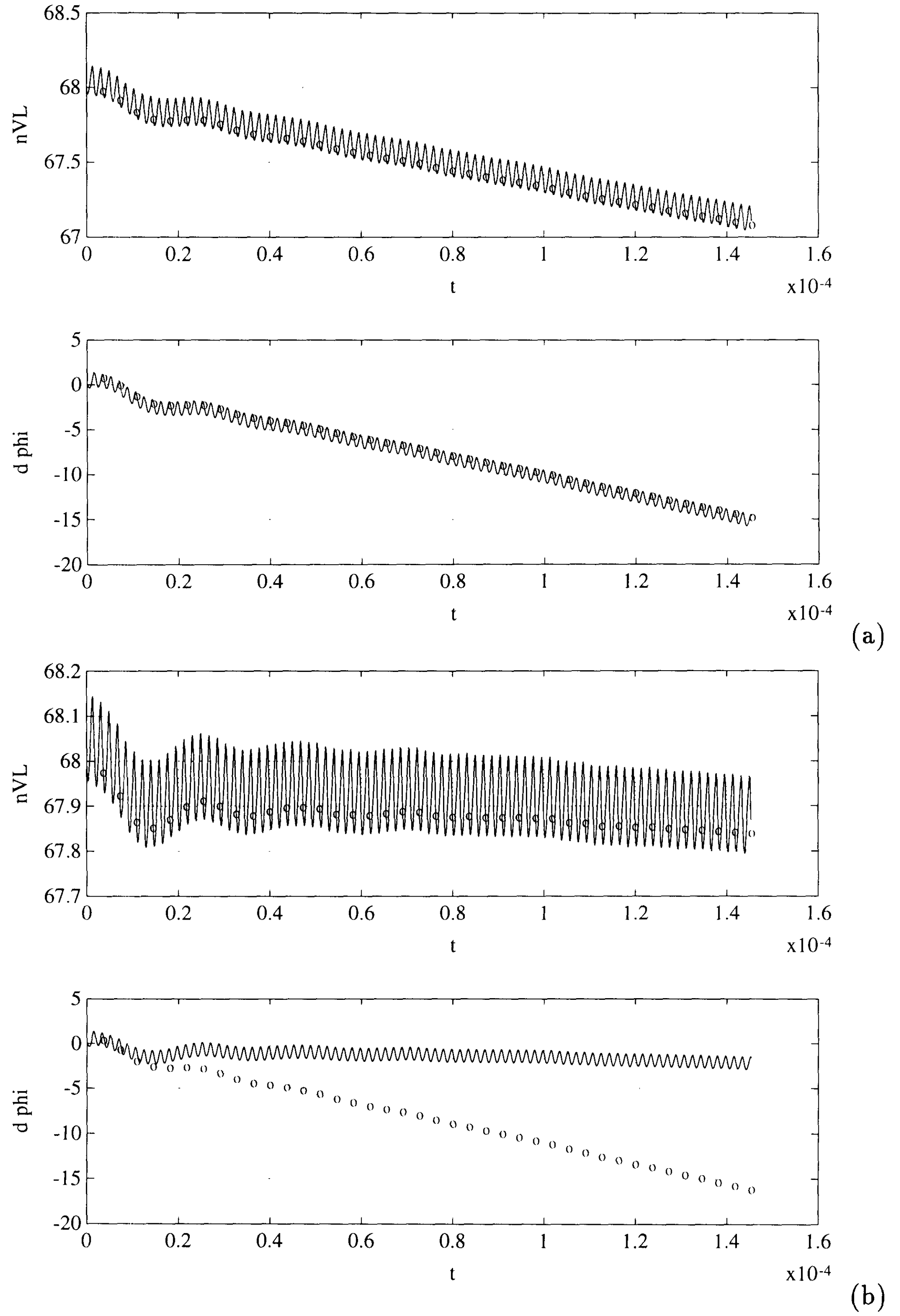


Figure 8-10: Output voltage, nv_L, and feedback signal, $\Delta\phi$, with supply voltage, E, droops from $E_{nom} = 250$V at -1V/cycle (a) without disturbance feedforward, and (b) with disturbance feedforward.

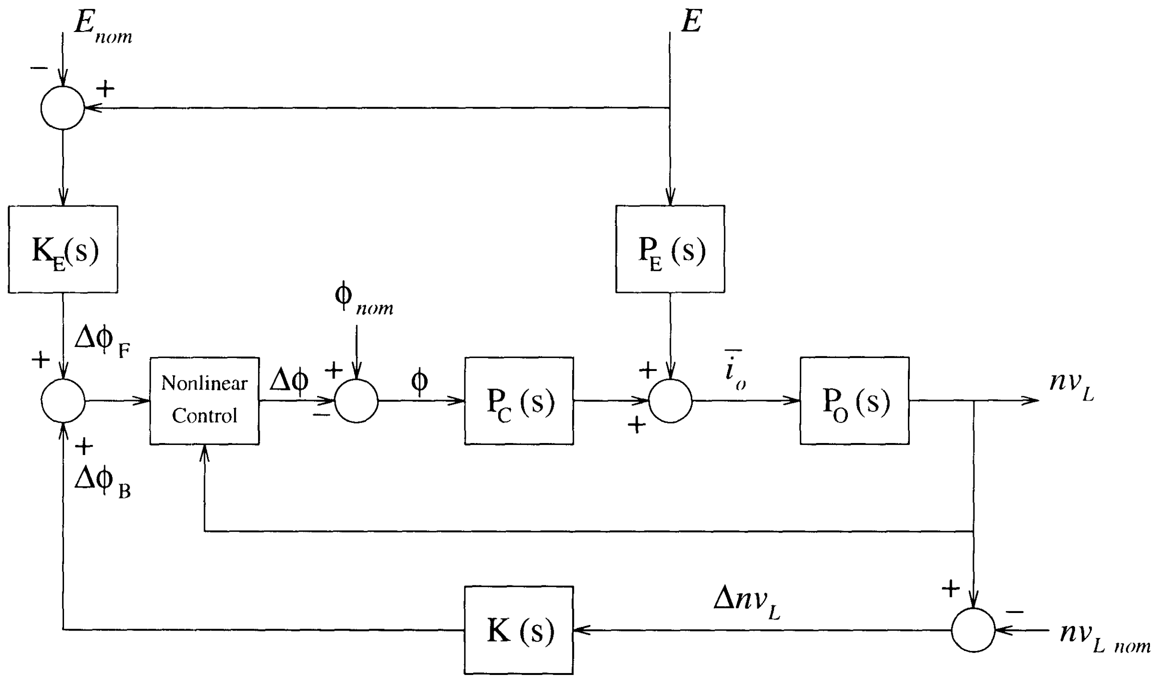


Figure 8-11: System block diagram.

1, we can see that a given variation in the supply voltage, E, causes smaller variations in output current at smaller phase angles, and larger variations at larger phase angles. When E is far away from its nominal, E_{nom}, we should expect errors in the feedforward phase correction signal. However, even with this error, we can still see the improved disturbance rejection in Figure 8-10(b). We have assumed in this case that the supply droops at -1V/cycle. At this rate, the full range of supply voltage between 350V and 200V is covered in 150 switching cycles or 0.545 msec. This rate may be higher than what can be expected in the actual system. The smooth curve for $\Delta\phi$ is the output of the feedback controller while the circle marks are the phase correction values at the start of a switching cycle. The difference between the two is due to the feedforward correction. Again, since the deviation in the output voltage is not large, the responses are similar for the two output models and only the responses for the resistor model are presented here.

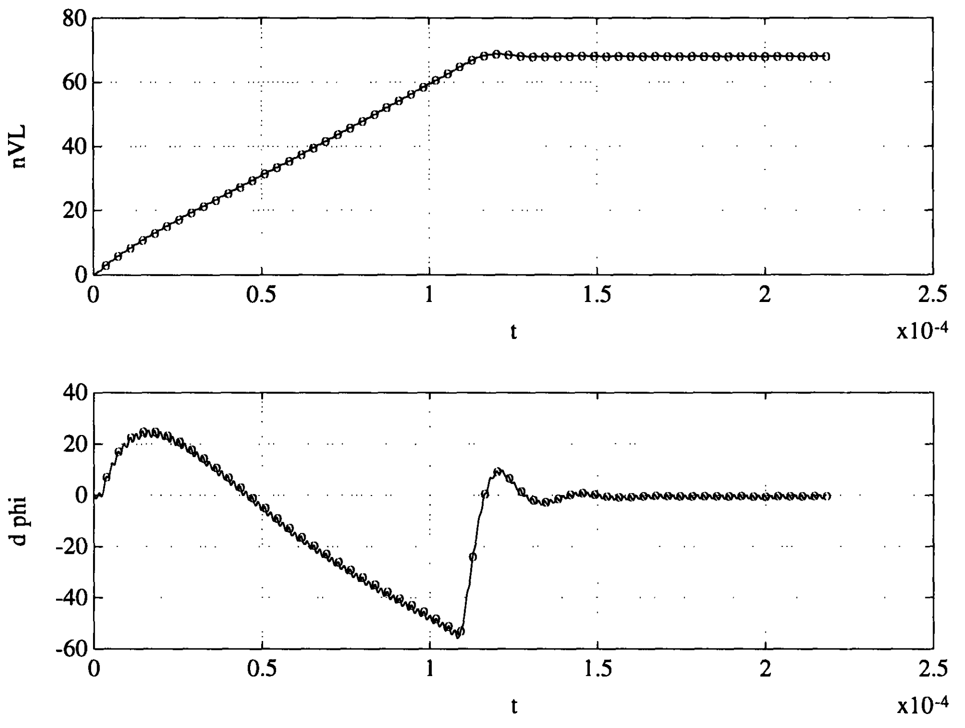
8.5 Initial Start-Up Behavior

At initial start-up the state variables of the converter are all zero, and the output voltage is zero since the output capacitor is uncharged. (We have used the term start-up elsewhere to mean zero initial state variables but with the output voltage at the nominal; the difference should be clear in the context where the terms are used.) The supply voltage may be assumed to be at the nominal value or at the higher end of its range. With a large deviation in output voltage from the desired nominal, the integrator in the feedback controller causes its phase correction signal to rapidly move outside of the allowed range, below its lower bound. Since maximum output current should be used to charge up the output capacitor quickly, this response would appear to be satisfactory. However, after the output voltage has come close to its nominal value and the phase correction should be close to zero, integrator windup [3] due to the initial transient causes the phase correction to remain large on the negative side for an extended period of time. The output current thus becomes excessively high and the output voltage continues to rise until nonlinear control kicks in and reduces the switching phase to zero. Output voltage then drops into the region close to its nominal, but the large phase correction still remaining from the feedback controller will quickly bring the voltage up again. Although with nonlinear threshold control the output voltage never goes out of the bounds by too much, it does not settle down to its nominal value either. The small-signal feedback control never gets a chance to work properly because of integrator windup.

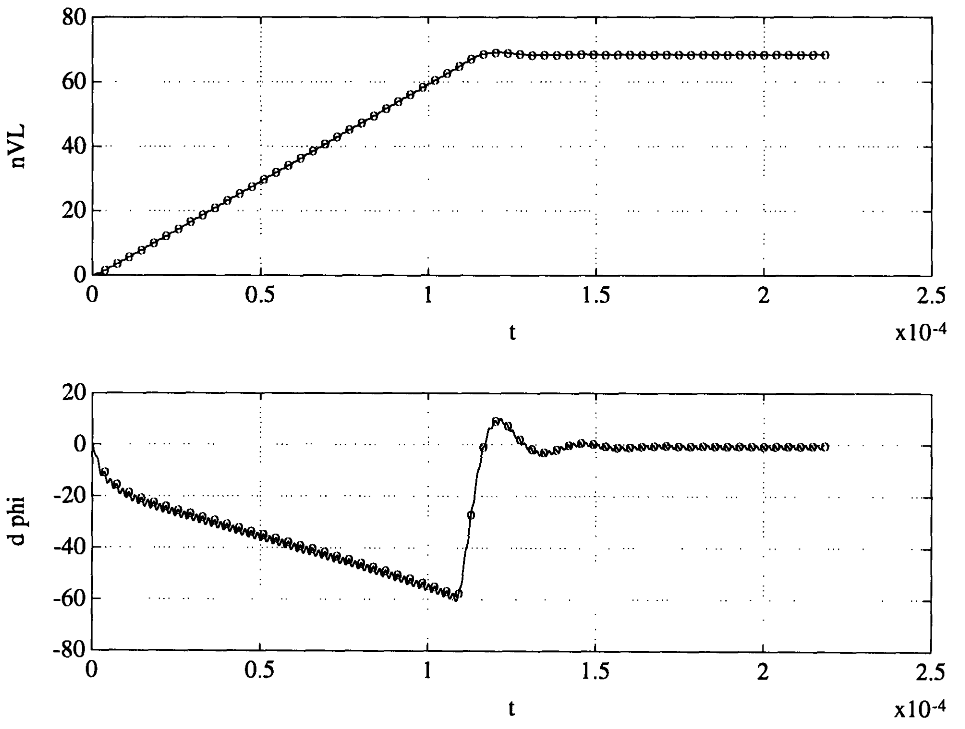
To correct this problem, we have implemented a soft-start scheme. Instead of having a fixed reference output voltage $nv_{L_{nom}}$ throughout, it will be gradually increased from zero to the desired nominal value of 68V in about 30 cycles[3]. Figure 8-12 shows the resulting start-up behavior of the simulator. The supply voltage is assumed to be constant at its nominal value of 250V without drooping. The response for the resistor output model and the current source model are quite different initially. In the resistor output model, little load current is drawn initially and the output capacitor charges up faster than the rise in the reference. The phase correction is actually positive to slow down the rise in output voltage. In our simulation, the reference nominal output voltage increases linearly from zero to 68V and then flattens out. This sharp corner causes a sharp change in $\Delta\phi$ as can be seen in Figure 8-12. A smoother reference may produce a smoother $\Delta\phi$. Figure 8-13 shows the start-up behavior with a supply voltage, E, that droops from 300V at -1V/cycle. For the 60 cycles simulated, the supply voltage drops from 300V to 240V. For Figure 8-13(a), with a higher supply voltage, nv_L actually charges up too fast at the beginning so that it exceeds the reference by more than the nonlinear threshold of 1.4V. Nonlinear control kicks in and the switching phase is reduced to zero for a cycle.

The simulation results shown in this chapter have demonstrated that in addition to small-signal control near the nominal operating point, the augmented controller is adequate in dealing with a variety of operating conditions including transitions between full-load and no-load, and the initial start-up.

[3]Simulated with *simstr.m*, see Appendix A.

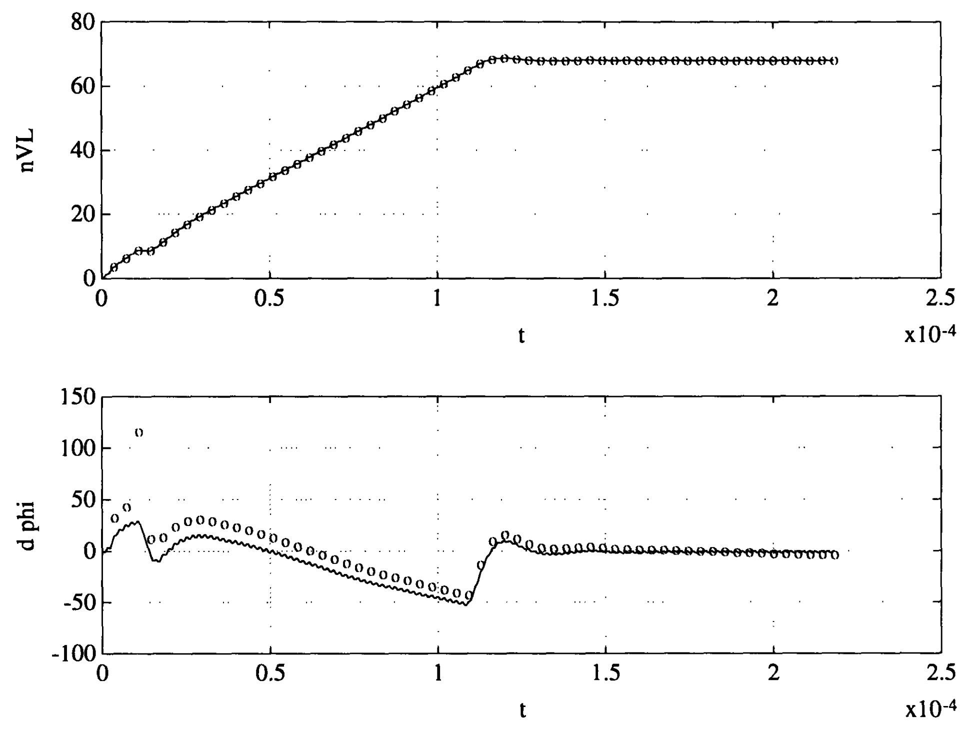


(a)

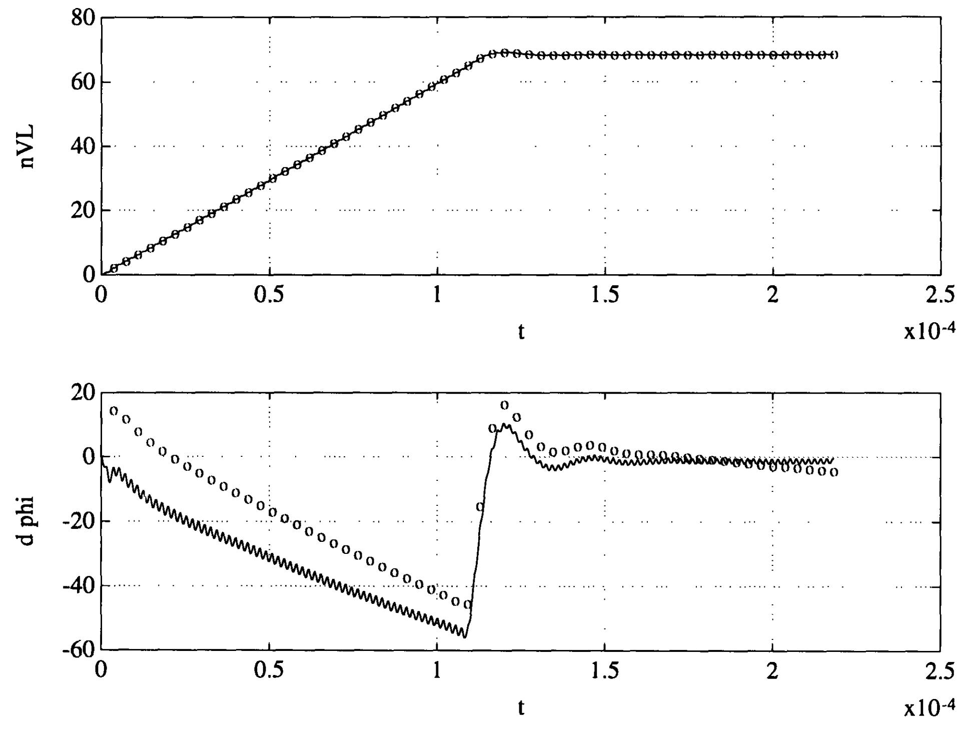


(b)

Figure 8-12: Output voltage, nv_L, and feedback signal, $\Delta\phi$, from start-up with constant supply voltage, E, at 250V (a) with resistor output model, and (b) with current source output model.



(a)



(b)

Figure 8-13: Output voltage, nv_L, and feedback signal, $\Delta\phi$, from start-up with drooping supply voltage, E, from 300V at -1V/switching cycle (a) with resistor output model, and (b) with current source output model.

Chapter 9

Simulator Design

To aid the analysis of the series resonant converter (SRC) with clamped tank capacitor voltage, a computer simulator based on the state-space model was written in Matlab. It is a very important and useful tool in the numerical analysis of the converter and is a major component of this project. Simulation results have been referred to and presented throughout the previous chapters.

A simulation program of the SRC circuit based on the state-space model and using trapezoidal numerical integration was written by Osawa [7] in his thesis. The fundamental approach of the current simulator here is similar to that of Osawa's. However, the current simulator has been completely redesigned and rewritten to facilitate numerous modifications and improvements, to resolve some of the simulation errors he had encountered, and to enhance the accuracy of the simulation results. A whole new set of functions has been added and the simulator has been expanded to include new capabilities, such as the output of critical points at mode and switching transitions (in addition to the regularly spaced time points), the simulation of the dynamics of the load, the design of feedback controllers, *etc.* With the 3-D imaging capability of Matlab release 4.0, animation of the evolution of the trajectories can be done to assist visualization.

9.1 Simulator Structure

The principal task of the simulator is to generate the time response of the state variables of the circuit, given some initial state and certain other operating parameters. The state variables are the inductor currents, i_1, i_2, and the tank capacitor voltage, v_C, if the load is modeled as a voltage source. When the load is modeled as a capacitor in parallel with a resistor or a current source, the output capacitor voltage becomes an additional state variable. The simulation of the SRC circuit is centered around the function *srctrans*(), which monitors the transitions in topological modes and computes the value of the state at the next time point, using the trapezoidal integration method. Some of the relatively self-contained subtasks such as numerical integration, topological mode determination, switching signal generation, *etc.* are handled by other functions. Together, they form the basic simulation program that generates the time response of the SRC.

A higher-level simulation usually simulates the SRC in a certain way in order to determine certain characteristics. It may include such tasks as the generation of the trajectory fields, or the simulation of start-up dynamics, or the simulation of the steady-state behavior, or the search for the nominal switching phase angle. It often consists of several calls to *srctrans*() while changing some parameters between calls. The returned data on the state variables are processed and relevant information extracted and/or displayed with some auxiliary functions.

The auxiliary functions perform subtasks that are common to many simulation programs. They could be functions that compute the average output current, or check the operating mode of the circuit, or generate the feedback control signal, or display the output waveforms, or carry out an animation of the trajectory in 3-D.

This division of tasks into functions allows the top-level simulation programs to be relatively simple. On the other hand, if any part of the simulator needs to be changed, may it be the integration method, or the modeling of the output stage, or the feedback controller design, modifications can be implemented with ease in the functions with little effect on the higher-level programs or the overall structure of the simulator.

Most of the Matlab source codes are listed in Appendix A.

9.2 Core Simulation Programs

The core simulation program that generates the time response of the SRC comprises the function *srctrans*() and several functions it calls. The usage of these functions can be found in the source codes and the comments.

9.2.1 SRC Transient Response, *srctrans.m*

Usage: [*t,x,m,s,tm,xm,mm,sm,nvl*]= *srctrans*(*x0,para1,para2*)

srctrans() is a self-contained function that generates the complete trajectory of the SRC from any starting point in the state-space, based on some circuit parameters such as capacitor and inductor values, operating parameters such as switching frequency, supply voltage, load voltage, and simulation parameters such as starting time, ending time, and time-step size, *etc.* These parameters are usually specified in the higher-level calling program, and they are passed in vector form, although some functions only use a subset of the parameters contained in the vectors.

The vector *para1* contains twelve parameters. They are the resonant circuit tank capacitance C, inductance $L = L_1 = L_2$, supply voltage E, load voltage nV_L referred to the primary side of the transformer, switching frequency $\omega_s = 2\pi f_s$ in radians per second, phases of e_1 and e_2 in radians, error tolerance in determination of equality, integration time-step size, starting time of simulation, ending time of simulation, and output compression ratio.

The vector *para2* was added at a later stage in the development of the simulator. It contains the parameters needed to model the dynamics of the output stage. It is optional if the output is modeled as a voltage source. There are six parameters in *para2*. They are the load capacitance C_L, load resistance R_L, load current I_L, transformer turns ratio n, and load model number. When the load is modeled as a capacitor in parallel with a resistor, the load current value is ignored; when the load is modeled as a capacitor in parallel with a current source, the load resistance value is ignored.

The first four variables in the output of *srctrans*() are the regularly spaced time points, and the state variables, topological mode M, topological mode S of the SRC at those points. The next four variables are the critical points at mode transitions, switching transitions, and other extra points calculated in addition to the regularly spaced time points. The last output variable is the response of the load voltage referred to the primary side of the transformer, with values at only the regular time points.

A major component of *srctrans*() is in essence a look-up table for the various topological mode transitions. This helps to overcome one major problem with Osawa's simulator, namely that the penetration of a plane by the trajectory's moving from one volume mode to another was not always detected. Tolerance levels were set to determine if the trajectory was close enough to a plane for it to be counted as being on the plane, so that equations for that plane could be used to carry out the calculations. The tolerance level could not be set too low or a penetration of the plane would be missed, but the tolerance could not be set too high or the accuracy of computation would be compromised. The tolerance level had to be adjusted heuristically according to the circuit parameters and the integration step size. However, once it was set, depending on the magnitude of the state variables and other factors, detection of mode transition was still not guaranteed.

From the 3-D representation of the topological modes of the SRC in Figure 3-1, it is possible to list all the allowed topological mode transitions. The program uses this information as follows. Knowing the topological modes and values of the state variables at the current time point, *srctrans*() calls *nextval*() to find the state variables at the next time point. It then calls *ckcfg*() to find the topological mode for the new point. There are several possibilities here. When there is no change in the topological modes, no action is required, and numerical integration carries on. When a change in topological modes is detected, the function determines whether that transition is valid. If the change is a valid one, it does a linear interpolation to find the time and place of transition, saves the data as a critical point, and then recalculates the trajectory from this critical point using equations for the new mode to the next regular time point. When the trajectory moves from one volume mode to another, boundary conditions have to be checked to see if the trajectory has penetrated a plane mode.

One example is the transition from M1 to M3. The planar M7 sector is only a part of the plane separating M1 and M3. When the trajectory seems to move from M1 to M3, it might be captured by the M7 plane and continue to move on M7 instead of going into M3. If the appropriate boundary conditions for M7 are satisfied, then the equations for M7 will be used to carry on the computation. Otherwise, the equations for M3 will be used. An invalid transition is usually caused by a large time step at a place near the boundary of more than two modes in the state-space, for example, near the origin. There are instances where the correct sequence of mode changes should go, say, from mode X to Y to Z, but a large time step may result in a mode change that goes directly from X to Z. For example, M1 cannot go to M2 without going through either M3 or M4. Of course, a semi-degenerate case can be that M1 goes to M2 directly through the line $i_1 = i_2$, but this should not happen in general. Upon finding an invalid transition and after checking for some of the semi-degenerate conditions, *srctrans*() reduces the time step and recalculates until a valid mode transition is attained. The list of allowed mode transitions can be derived by inspection most easily from the 3-D representation of the topological modes in Figure 3-1, although the number of possible transitions is not small.

Switching changes, like mode transitions, usually do not occur exactly at the regular time points. When a change in the switching voltages e_1 or e_2 is detected, *srctrans*() calls *edtexact*() to find the exact time of the change and computes the state variables at that critical point.

The capturing of these critical points is an important feature of the simulator, especially since a comparison between the simulated trajectory and the computed results using the analytical models are made at these points. The ability of *srctrans*() to vary internally the integration time step, Δt, when necessary at critical points or when invalid mode transitions occur, makes it possible to specify larger time steps to obtain a fast simulation without losing the details at transition points.

Depending on the load model specified, *srctrans*() also updates the load voltage at the regular time points, using a trapezoidal integration method. The output current and voltage are referred to the primary side in computation, so the turns ratio has to be specified in *para2*.

Long simulations with small time steps may generate unreasonably large numbers of data points. The user is provided with the option to specify the compression ratio, k, in the last entry of *para1*, so that only the first and every k-th point thereafter are returned. The function *dilute*() is called to perform this task. We describe *dilute*() as an auxiliary function, since it can be used in many other places.

In the rest of this section, we will discuss the functions that *srctrans*() calls. Although in principle these functions can be used elsewhere as auxiliary functions, they are almost exclusively used by *srctrans*(), and are considered more as a part of *srctrans*().

9.2.2 Evaluation of State at the Next Time Point, *nextval.m*

From the current state at time t, *nextval*() calculates the value of the state at time $t + \Delta t$, where Δt is the integration time-step size, using the trapezoidal integration method:

$$\mathbf{x}((k+1)\Delta t) \approx \mathbf{x}(k\Delta t) + [\dot{\mathbf{x}}((k+1)\Delta t) + \dot{\mathbf{x}}(k\Delta t)]\,\frac{\Delta t}{2} \tag{9.1}$$

Combining it with the state-space equation of the system,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{9.2}$$

yields the integration formula

$$\mathbf{x}((k+1)\Delta t) \approx \left(\mathbf{I} - \mathbf{A}\frac{\Delta t}{2}\right)^{-1} \left\{\left(\mathbf{I} + \mathbf{A}\frac{\Delta t}{2}\right)\mathbf{x}(k\Delta t) + \frac{\Delta t}{2}\left[\mathbf{B}\mathbf{u}(k\Delta t) + \mathbf{B}\mathbf{u}((k+1)\Delta t)\right]\right\} \tag{9.3}$$

Depending on the current topological mode the SRC is in, *nextval*() picks the appropriate state-space equations in its computation. The generation of the input voltages and the determination of the topological mode of the SRC are handled by *eswitch*() and *ckcfg*(), respectively. The values are passed to *nextval*() from *srctrans*().

It is worth noting that most of the code in this function is used to find the right state-space matrix for the given topological mode. Computation of the state takes only one line of code using the formula shown above. The computation of the output voltage referred to the primary side, nv_L, is done in *srctrans*(), since its computation does not depend on the topological mode.

9.2.3 Determination of Topological Modes, *ckcfg.m*

The topological mode of the SRC is determined by *ckcfg*() based on the boundary conditions set out in Table 3.1. The effects of numerical errors must be taken into consideration. For example, v_C is clamped to $+E$, but numerical integration may lead the trajectory to a point above the $v_C = E$ plane. This point above the $v_C = E$ plane is counted as being in mode M5. The function *ckcfg*() does not modify the trajectory; it only reports the possible mode the SRC may be in. It is left to *srctrans*() to recognize this penetration of the M5 plane into an impossible region and to fix the trajectory accordingly.

The conditions in Table 3.1 are stated with $\geq$ and $\leq$ signs, so a point on the boundary is ambiguous when a mode number has to be assigned to it. Since the state-space equations for either of the modes are the same on the boundary, it makes no difference which mode number is assigned. In assigning mode numbers, the conditions in Table 3.1 are modified so that a point in the state-space can have a unique mode number. The decisions in some cases are made for a specific reason, while in other cases they are rather arbitrary. One example is the boundary between modes M1 and M3 outside of the M7 plane. On the side where $i_1 > 0$, mode number M1 is assigned, while on the other side where $i_1 < 0$, mode number M3 is assigned. The reason is that the trajectory naturally tends towards M1 in the former case and towards M3 in the latter case, if we simply take a look at the state-space equations. Of course, if the mode numbers are assigned the other way around, the program would still work. The origin is located on the boundary of seven modes, namely M0, M1, M2, M3, M4, M7, and M8. Any one of these can be assigned, but M7 was chosen to represent the origin purly for convenience. M0 may have been a better choice because simulations have shown that the trajectory usually goes into M0 from the origin.

9.2.4 Switching Signals, *eswitch.m*

The voltage across the resonant circuit, v_t, is the difference $e_1 - e_2$, where e_1 and e_2 are two square waves of the same switching frequency but with a phase shift. One way to specify and generate the two input voltages conveniently is through the use of two sine waves of the same frequency and phase difference as the square waves. The value of the input voltage is then assigned according to the sign of the sine wave.

In certain cases, we may want e_1 and e_2 to be held at a constant dc value. This is the case when we examine the trajectory velocity fields for the different combinations of input voltages in Chapter 4. This can be done by setting the switching frequency to zero and substituting in place of the phase values in *para1* the values of dc voltages for e_1 and e_2.

9.2.5 Locating Exact Switching Times, *edtexact.m*

The function *edtexact*() was written out of the need to find the state variables at the precise switching times. However, the values of e_1 and e_2 are ill-defined at the transition times because we have modeled the input voltages as ideal square waves. Given that switching has taken place between t_1 and t_2 at $t_1 + \Delta t$, *edtexact*() returns the value of Δt plus a very small amount ϵ so that at $t_1 + \Delta t + \epsilon$, the value of the switching voltage is defined. This value of $\Delta t + \epsilon$ is return to *srctrans*(). Hence, the so-called precise value of switching time is really off by a very small percentage. In the codes, ϵ is set to 10^{-7} of Δt. Given that there are approximations and errors from many other sources, this error is entirely acceptable.

9.3 Auxiliary Functions

After the time response of the SRC has been obtained from *srctrans*(), the raw data may need to be processed in some fashion. For example, we may want to find the average current, or we may want to display and print the trajectories. The auxiliary functions perform some of these tasks.

9.3.1 Average Output Current, *iavg.m*

One of the important quantities that characterizes the SRC circuit is its average output current. The output current referred to the primary side is the rectified version of the sum of the inductor currents, *i.e.*, $i_o/n = |i_1 + i_2|$. If the SRC is truly in the steady-state, a moving average of the output current with a window size equal to the switching period should be constant in time. The size of the ripples in the windowed moving average may serve as an indication of the degree to which the SRC is in steady-state. Discretization in time may introduce some errors, since the window size stated in number of points does not correpond exactly to the duration of a period in continuous time. This may artificially introduce some ripples in the moving average. However, if the number of points in a period is sufficiently large, such errors are small.

The number of computed points in a switching period of duration T_s is equal to $T_s/\Delta t$. The output of *srctrans*() is a diluted version, so the window size in number of points is $T_s/(k\,\Delta t)$, where k is the compression ratio. The moving average at any time t is the average of the output current in the preceding period. Note that *iavg*() only generates output at points after the first period, or $t > T_s$. If the input is not regularly spaced, *iavg*() attempts to adjust the window size at every point, t, so that if the window size is W, W points will be less than one period while $W + 1$ points will be more than one period. This, however, will weigh closely spaced points more heavily. Therefore, the input to the function should be regularly spaced. The function also outputs the average of the last period and the ripple size of the last period. This is based on the assumption that usually the trajectory settles towards some steady-state, so the last period should be closest to steady-state and the average output current of the last period should be closest to the true value. This assumption, however, does not always hold.

9.3.2 Output Display, *plsrc.m*

The function *plsrc*() handles the screen display and plotting of the trajectories. Two types of graphs can be displayed. One is the projections of the trajectories, as in the standard method of displaying a 3-D object by projecting it onto three orthogonal planes. This amounts to plotting one state variable versus another: v_C *vs.* i_2, v_C *vs.* i_1, and i_2 *vs.* i_1. The second type of graph consists of four plots of responses of various variables *vs.* time. The first plot is the voltage across the resonant circuit, $v_t = e_1 - e_2$, along with one of the switching voltages, e_1. The second plot displays the tank capacitor voltage, v_C, and the primary-side transformer voltage, v_p. The third displays the two inductor currents, i_1 and i_2, and the sum of the two currents. A rectified version of the sum of the two currents is the output current. The last plot traces the topological modes of the SRC as time goes on. It puts the M number on the positive side of the axis and the S number on the negative side.

The state variables are obtained from *srctrans*() while the values of the switching voltages and transformer voltage can be obtained from *evp*(). The function *plsrc*() has the ability to 'dilute' long input time series to under 1000 points, since the resolution of either screen display or hard copy printing does not need more points. This avoids generating unnecessarily large output files. The user also has the option of large-size plots or small-size ones, and the option of screen display or saving the graphs to an output file for hard copy printing.

Simpler output display routines are often written as part of the main simulation programs if the display manipulation is not too complicated.

9.3.3 3-D Animation, *anim.m*

Animation of the trajectory in 3-D requires Matlab 4.0 and above for its 3-D displaying capabilities. As we have said in Chapter 5, the trajectories are spirals on cylinders in topological modes M1, M2, M3, and M4, and are circles or lines on the other mode planes. The function *anim*() displays the trajectory in 3-D with a moving window. To assist visualization, it also plots the projections of the trajectory alone with a sketch of the cylinders and the planes it rides on. It calls on *cylin*() and *planes*() to sketch the cylinders and the planes. The trajectory is not plotted against normalized axes, so the projections of the trajectory do not follow circles but ellipses. Being able to see the trajectories evolve in time as it moves in 3-D state-space gives us a better feel for how the trajectories behave.

9.3.4 Feedback Controller, *fbctrl.m*

The feedback controller is simulated in *fbctrl*(). The feedback controller is a linear system and its simulation can be handled by the Matlab function *lsim*(). The input to the feedback controller is the difference between the output voltage, nv_L, and the its nominal value, $nv_{L_{nom}}$, which can be a constant in most cases or a time series like nv_L for soft-start. The transfer function of the feedback controller is transformed into a state-space description with the Matlab function *tf2ss*(). Each design of the controllers is given a controller number, providing an easy way for the calling program to pick any one of them in the design process.

9.3.5 Other Auxiliary Functions

Other useful auxiliary functions include the following.

srccomb.m

The data points at the regularly spaced time points and those at the critical points are placed in separate variables by *srctrans*(). The two can be combined into one sequence in order of time by *srccomb*().

dilute.m

While a small time-step size and a large number of points may be necessary for computation accuracy, they are not needed for output. The function *dilute*() picks the first and every k-th column thereafter out of a time series, where k is the specified factor.

evp.m

The switching voltages, e_1 and e_2, can readily be found for any time point, so there is no need for *srctrans*() to include them as part of the output. The primary-side transformer voltage, v_p, can also be calcuated when we know the switching voltages, the tank capacitor voltage, and the output mode S. When in S1, $v_p = n \cdot v_L$; when in S2, $v_p = -n \cdot v_L$; and when in S3, $v_p = e_1 - e_2 - v_C$. This is true whether v_L is assumed to be constant or not. Note that *evp*() should be run before *plsrc*() since the switching and transformer voltages are needed in the plots.

cylin.m

The function *cylin*() is mostly used by *anim*() to handle the sketching of cylinders. Using specified parameters such as the orientation of the cylinder, its radius, center, *etc.*, *cylin*() sketches the cylinder.

planes.m

The function *planes*() is also mostly used by *anim*(). It sketches the planes the trajectory rides on.

mseq.m

Given the time response of mode M numbers, *mseq*() finds the topological mode sequence in the last period of the time response and the corresponding number of points for each mode. It finds only the mode sequence for the clamping diode modes M.

9.4 Simulation Calling Programs

To perform a specific simulation, a higher-level program calls *srctrans*(), often many times with varying parameters, and uses some of the auxiliary functions to process the data in order to extract useful information. Many of the graphs and data shown in the previous chapters are generated by these simulation programs. These programs have names starting with *sim*.

9.4.1 Trajectory Fields, *sim0.m, sim1.m, sim5.m, sim7.m*

The trajectory velocity fields shown in Chapter 4 are generated by repeatedly calling *srctrans*() from a lattice of initial points. While holding the input switching voltages at some combination of constant values, the trajectories are simulated for a short period of time. The resulting trajectories approximately show the relative velocities of the trajectories at these initial points.

9.4.2 Searching for the Nominal Phase Angle, *simnom.m*

To achieve a constant output power of 4kW, the average output current referred to the primary side of the transformer has to be maintained at 58.82A. Given a particular supply voltage E, and output voltage V_L, a search for the switching phase angle ϕ, that will achieve the desired average output current is performed with the simple method of bisection. A fairly loose range of phase angles is set at first, with the upper bound giving an average output current larger than the desired value and the lower bound giving a smaller value. Simulation is run at the midpoint until the SRC settles into steady-state. The average output current is computed and the range of phase angles is updated. Another midpoint phase angle is tested and the process is repeated until the range is reduced to a small size, yielding the approximate nominal phase angle.

9.4.3 Steady-State Operating Modes and Average Output Current, *simsss.m, simsssi.m simssa.m, simssm.m*

Assuming that the output can be modeled as a voltage source, we wish to find the average output current for a range of supply voltages, E, and a range of phase angles, ϕ. We also wish to find the operating mode at each of these points. The program *simsss* simulates the trajectories at each E, ϕ combination until the trajectories settle into steady-state. The average output current values are computed with *iavg*() and the M mode sequences are found with *mseq*(). All the data is saved, to be processed by *simssm.* The program *simsssi* uses the data saved by *simsss* and plots the average output current *vs.* phase angle for various supply voltages.

To establish a map of operating modes at various E and ϕ, it is necessary to simulate more points in areas bordering different operating modes in addition to the grid of points simulated in *simsss.* This is done with *simssa.* The binary box method for determining the operating mode regions is briefly discussed in Chapter 6. The program *simssa* is a similar to *simsss*, and builds on top of the data obtained by *simsss* to simulate at additional points. The locations of the new points are entered manually, since it is not worthwhile to write an automated scheme for locating the additional points (because the simulation time is much larger than the time needed to modify the *simssa* file a little and enter a few points).

The program *simssm* sifts through the M sequences at each E, ϕ point, compares the mode sequence with the mode sequences of known operating modes, determines its operating mode number if it matches one of the known operating modes, or assigns a new operating mode number if a match cannot be found. It then plots the operating mode table.

9.4.4 Nominal States at Switching Times, *simsyo.m*

The nominal phase angle found by *simnom* delivers the desired average output current only approximately. However, for the angle selected, we need to know the trajectory more precisely in order to find the small-signal transition matrices. Although the trajectory of the SRC converges to a steady-state very fast, the precise values at the switching times do not converge fast. Numerical errors also accumulate if we run the simulation for a very long period of time. Therefore, after the trajectory has roughly settled into steady-state and we know the approximate values of the state at the switching times, simulations are run from a grid of points near the approximate state for about one period. The starting point that closes onto itself after one cycle with the least amount of deviation is chosen to be the nominal starting point. This process is repeated with finer resolution. The search for the nominal values of the states are performed at two switching times, $v_t : E \rightarrow 0$ and $v_t : 0 \rightarrow -E$.

9.4.5 Small-Signal Transition Matrices, *simsyq.m*

Once we have found the precise states at the two switching times that will close onto themselves after a cycle, we may perturb the states by a little in different directions, and find the perturbations at the end of the cycle. Small-signal transition matrices can be constructed according to the formula shown in Chapter 7. Perturbations of different magnitudes and directions are applied at the two switching times and the transition matrices at the two places are computed. Averages of the transition matrices are taken and the eigenvalues are found. The output is written to a diary file and is listed in Appendix B for comparison with values computed in Maple using the analytical model.

9.4.6 System with Feedback Controller, *simplt.m, simctr.m simstr.m*

In *simplt* the switching phase angle, ϕ, or the supply voltage, E, is perturbed by a little at the beginning of a switching cycle, after the circuit is already in steady-state. The program waits for the circuit to settle into steady-state again and finds the new average output current. We may then calculate the numerators, β and γ, of the transfer functions for the converter, as set out in Chapter 8.

The closed-loop system with feedback controller is simulated in *simctr*. It is simulated on a cycle-by-cycle basis, because the phase angle correction is sampled at the beginning of every switching cycle. A variety of different operating conditions can be specified by changing the parameters, such as initial state variables, load model, nominal values for phase angle, supply voltage, output voltage, their initial values, *etc.* The small-signal feedback controller is simulated with *fbctrl*(), but the nonlinear control rules and the feedforward controller codes are written here. Operating conditions, such as load demand and supply voltage, can also be changed from cycle to cycle, to simulate conditions such as the transition between full-load and no-load, or the droop in supply voltage. Since the droop in E is assumed to be slow, it is approximated by a step decrease from cycle to cycle. Parts of the control system such as disturbance feedforward, or the nonlinear control rules, can be disabled by taking out the corresponding lines of code in the program. The program output consists of a plot of the output voltage and the control signal from the feedback controller as continuous waveforms. The sampled control signal actually used to correct the phase at the beginning of each switching cycle is plotted as a discrete-time series. It is not necessarily the same as the control signal from the output of the feedback controller, due to saturation, nonlinear control, and disturbance feedforward. Most of the simulations shown in Chapter 8 are produced with this program.

The program *simstr* is very similar to *simctr*. It is used to simulate the start-up conditions of the converter. To correct integrator windup in the feedback controller due to large initial deviation in output voltage from its nominal value, a soft-start strategy is applied by specifying the reference $nv_{L_{nom}}$ as a smooth ramp function, starting at near zero and reaching the nominal value in about 30 cycles. This is a special case because in almost all other situations, the nominal values are used.

Chapter 10

Conclusion

We have first examined individually the topological modes of the series resonant converter with clamped tank capacitor voltage. In each of the topological modes, the circuit is LTI, so state-space equations can be written to represent them. State-space representation also forms the basis for the simulation program. Although the number of topological modes is not small, by separating the clamping diode modes and the output diode modes, we have kept the analysis manageable.

Analysis of the state-space trajectories in each of the topological modes is first carried out via trajectory velocity fields. Trajectory geometry is further examined by solving the state-space equations. Depending on the topological mode, the trajectories can be lines or circular arcs in plane modes or spirals in volume modes, when plotted against normalized axes. A complete trajectory is simply a concatenation of the segments in the topological modes it goes through.

With simulations, we have found ten operating modes for the range of supply voltages and switching phase angles for normal operations of the converter. A nominal operating point that produces the desired 4kW output power is selected. From the solutions to the state-space equations, large-signal transition matrices are derived for the nominal operating point. Large-signal and small-signal sampled-data models around the nominal point are developed, and the results are verified by simulation. The small-signal model shows that the SRC has fast dynamics and is essentially first-order.

By approximating the SRC as a first-order system, a small-signal feedback controller is developed using classical control methods. Nonlinear control rules are added to handle large-signal errors, and a disturbance feedforward scheme is added to improve disturbance rejection characteristics. Simulations under various circuit operating conditions validate the controller design.

Most of the modeling and analysis have been done with the assumption that the supply voltage and the output voltage are constant. Consideration of the output model has been incorporated in the controller design process, and the simulation program is able to model the output dynamics. Further studies of this SRC circuit may include more accurate modeling of the converter by taking into acount the dynamics of the output and the supply voltage variation. More detailed analysis of the SRC at other operating points should be performed to see if the first-order approximation of the SRC is indeed justified. Although the controller design based on a first-order approximation of the SRC proved to be quite adequate in performance, a higher-order model of the dynamics of the converter may give us a better controller design, particularly in terms of better disturbance rejection.

The simple controller designed in this thesis performed well. However, it is still of interest to see if our analysis of trajectory geometry can lead to nonlinear controllers whose large-signal performance can be theoretically guaranteed (in contrast to the empirical 'guarantees' provided by more simulation of a variety of operating conditions, as done in this thesis). One such design of a nonlinear controller for the conventional series resonant converter is presented in [6]. Another challenge for future work is to provide a simple explanation for why the SRC dynamics (with constant-voltage load) is essentially first-order. Related to this is the task of directly deriving approximate continuous-time first-order models of the dynamics, perhaps in the style of [8] or [9].

Appendix A

Simulator Source Codes

A.1 Core Programs

srctrans.m

```
function [tall,xall,mall,sall,tmor,xmor,mmor,smor,nVLall]...
 =srctrans(x0,para,para2)
% [t,x,m,s,tm,xm,mm,sm,nvl]=srctrans(x0,para1,para2)
%
% [t,s,m,s]=srctrans(x0,para1,para2)
% generates the transient response of the state trajectory given the
% initial state x0 and parameters contained in para and para2.
% The initial x0 is assumed to be a valid state so that no fixing is
% needed on x0.
%
% [t,x,m,s,tm,xm,mm,sm]=srctrans(x0,para1,para2)
% With eight output arguments, both the regularly spaced points and
% the critical points are outputed.
% Use SRCCOMB to insert the critical points into the regular points.
%
% [t,x,m,s,tm,xm,mm,sm,nvl]=srctrans(x0,para1,para2)
% With nine output arguments, the output voltage is also available
% but only at reguarly spaced time points.
%
% para1=[C; L; E; initial nVL; freq,dc; ph1; ph2; err;
%       dt; tlower; tupper; compression ratio]
% para2 is optional. It specify additional parameters where default
% values are substituted if it is not given
% para2=[CL; RL; IL; n; VLcase];
% with default para2=[.01; 0.0181; 8; 0; 0];
% VLcase=0: VL is constant
% VLcase=1: VL in parallel with RL
% VLcase=2: Output is a current source, must specify IL

% Written Spring 93, Terrence Ho
% Modified 6/23/93 to make output sequence equally spaced in time
% Modified 6/25/93 to make time reference tlower and tupper absolute
% Modified 7/20/93 to fix undifferentiable transition points between modes
% Modified 8/2/93 to fix transition in control inputs
% Modified 8/5/93 to enable output of critical points
% Modified 12/8/93 to enable special switching cycle case number ecase
% Modified 12/9/93 to enable variable VL and its output and additional para2
% Modified 12/20/93 to take out ecase as it is not used at all

% ===== default constant parameters ==========
if nargin==2                % When no parameters on the output model is
 CL=.01;                    % specified, assume constant voltage output model.
 RL=.0180625;
 IL=8.5/0.0180625/8;
 nt=8;
 VLcase=0;
% ecase=0;
else
 CL=para2(1);              % Read the individual parameters from the
 RL=para2(2);              % parameter vectors.
 IL=para2(3);
 nt=para2(4);
 VLcase=para2(5);
% ecase=para2(6);
end
E=para(3);
err=para(8);
dtorg=para(9);
tlower=para(10);
tupper=para(11);
compress=round(para(12)); % Make sure compression ratio is an integer
if compress<=0            % and it should be made positive.
 compress=1;
end;

% ===== Initial conditions ==================
dt=dtorg;
tall=tlower:dtorg:tupper; % Generate the regularly spaced time points
tlen=length(tall);
pctmark=fix(tlen/10);     % A variable used to signal "percent completed"
k=1;                      % Initializing some indices
kk=1;
xct=x0;                   % Set initial state
i1ct=x0(1);
i2ct=x0(2);
vcct=x0(3);
[e1ct e2ct]=eswitch(tall(k),para);
[e1nt e2nt]=eswitch(tall(k)+dt,para);
[mct,sct]=ckcfg(xct,e1ct,e2ct,para);
xall=xct;                 % Fill the first point with initial values
mall(1)=mct;
sall(1)=sct;
nVLall(1)=para(4);

% ===== Start the iteration =================
disp('Message, srctrans(): Percentage completed: ');
while k<=tlen-1           % k+1 is filled in the loop
 if rem(k,pctmark)==0     % Display percentage completed
  disp(round(100*(tall(k)-tlower)/(tupper-tlower)))
 end;
 act=0;                   % Initialize some variables and flags for
 itcount=0;               % each new point
 dtcum=0;                 % Cumulative time from last time point
 next=0;
 nVL=nVLall(k);  % nVL is assumed to be constant at each point
 para(4)=nVL;    % Reset para.  Some subrountines use nVL
 while next~=1   % Repeat until ready for next regular time point
  if act==2, itcount=itcount+1;      % Display the number of times of
   if itcount>=2, itcount, end; end  % halving dt.
  if e1ct~=e1nt | e2ct~=e2nt         % If there is a change in switching
   dt=edtexact(tall(k)+dtcum, tall(k)+dtcum+dt, para);
  end;                               % find the precise time
  xnt=nextval(xct,e1ct,e2ct,e1nt,e2nt,mct,sct,dt,para);
  i1ct=xct(1);            % Find the next point
  i2ct=xct(2);            % dt is not necessarily equal to dtorg
  vcct=xct(3);
  i1nt=xnt(1);
  i2nt=xnt(2);
  vcnt=xnt(3);
  [mnt,snt]=ckcfg(xnt,e1nt,e2nt,para);
```

```
% === Determining transition in modes M and S ===========================
  if mnt==mct & snt==sct                    % No change in modes ==========
   act=1;

  elseif mnt==mct & snt~=sct                % Fix discontinous conduction ==
   if(sct==1 & snt==2) | (sct==2 & snt==1)  % S1->S2 or S2->S1
    f=-(i1ct+i2ct)/(i1nt-i1ct+i2nt-i2ct);   % den can't be zero
    xnw=xct+f*(xnt-xct);           % or the vector will be // i1+i2=0
    xnt=xnw;                       % xnw is the crossing pt on i1+i2=0 plane
    xnt(2)=-xnt(1);                % force i1+i2=0 just in case
    dt=dt*f;
    act=3;
    if xnw(3)>=e1ct-e2ct-2*nVL & xnw(3)<=e1ct-e2ct+2*nVL  % vcnw ok
     snt=3;       % If the crossing pt ill stay in S3, set snt
    end           % otherwise, snt stay the same as before
   else           % Other changes in S are fine
    act=1;        % This may need fixing.
   end

                                            % Change in mode M ===========
  elseif mnt~=mct % & ~(sct==1 & snt==2) & ~(sct==2 & snt==1)
   if mct==5 & (mnt==1 | mnt==4 | mnt==0)           % leaving M5
    act=1;
   elseif mnt==5 & (mct==0 | mct==1 | mct==4 )      % going onto M5
    if vcnt==E            % vcnt is guaranteed to be >=E
     act=1;
    elseif vcnt~=vcct     % dx not // to vc=E
     f=(E-vcct)/(vcnt-vcct);
     xnt=xct+f*(xnt-xct);
     xnt(3)=E;  % Force vc to E just in case there is a small error
     dt=dt*f;
     act=3;
    else
     disp('Error.  dx parallel to M5, not on M5 but going onto M5.');
     disp('Error partially recovered.');
     xnt(3)=E;
     act=1;
    end;
   elseif mct==6 & (mnt==2 | mnt==3 | mnt==0)       % leaving M6
    act=1;
   elseif mnt==6 & (mct==0 | mct==2 | mct==3)       % going onto M6
    if vcnt==-E           % vcnt is guaranteed to be <= -E
     act=1;
    elseif vcnt~=vcct     % dx not // to vc=-E
     f=(-E-vcct)/(vcnt-vcct);
     xnt=xct+f*(xnt-xct);
     xnt(3)=-E;  % Force vc to -E just in case there is a small error
     dt=dt*f;
     act=3;
    else
     disp('Error.  dx parallel to M6, not on M6, but going onto M6.');
     disp('Error partially recovered.');
     xnt(3)=-E;
     act=3;
    end;
   elseif mct==7 & (mnt==1 | mnt==3 )               % leaving M7
    act=1;
   elseif (mct==1 & mnt==7) | (mct==3 & mnt==7)     % M1 or M3 onto M7
    act==1;
   elseif (mct==1 & mnt==3) | (mct==3 & mnt==1)     % bet. M1 M3, ck M7
    f=-vcct/(vcnt-vcct); % den won't be zeros or dx // vc=0
    xnw=xct+f*(xnt-xct);
    xnt=xnw;
    xnt(3)=0;   % force vc=0 just in case
    dt=dt*f;
    act=3;
    if xnw(1)>=0 & xnw(2)<=0
     mnt=7;     % If the crossing pt is in M7, fix mnt
    end;
   elseif mct==8 & (mnt==2 | mnt==4 )              % leaving M8
    act=1;
   elseif (mct==2 & mnt==8) | (mct==4 & mnt==8)     % M2 or M4 onto M8
    act=1;
   elseif (mct==2 & mnt==4) | (mct==4 & mnt==2)     % bet. M2 M4, ck M8
    f=-vcct/(vcnt-vcct);  % den won't be zeros or dx // vc=0
    xnw=xct+f*(xnt-xct);
    xnt=xnw;
    xnt(3)=0;   % force vc=0 just in case
    dt=dt*f;
    act=3;
    if xnw(1)<=0 & xnw(2)>=0
     mnt=8;     % If the crossing pt is in M8, fix mnt
    end;
   elseif mct==0 & (mnt==1 | mnt==2 | mnt==3 | mnt==4 | mnt==7 | mnt==8)
    act=1;                                          % leaving M0
   elseif mnt==0 & (mct==1 | mct==2 | mct==3 | mct==4 | mct==7 | mct==8)
    act=1;       % onto M0;  !!! There may be complications here !!!
   elseif (mct==1 & mnt==4) | (mct==4 & mnt==1) | (mct==2 & mnt==3) ...
     | (mct==3 & mnt==2)
    f=(i1ct-i2ct)/(-i1nt+i1ct+i2nt-i2ct);  % den can't be zero
    xnw=xct+f*(xnt-xct);             % or the vector will be // i1=i2
    xnw(2)=xnw(1);        % force i1=i2 just in case
    [e1nw e2nw]=eswitch(tall(k)+dt*f,para);
    [mnw,snw]=ckcfg(xnw,e1nw,e2nw,para);
    if mnw==0
     mnt=0;      % fix numbers
     xnt=xnw;
     dt=dt*f;
     act=3;
    else
     act=1;      % dx doesn't go through M0, leave it
    end
                                     % semidegenerate case =============
   elseif (mct==7 & mnt==8) | (mct==8 & mnt==7)
    f=-i1ct/(i1nt-i1ct);      % vc=0 and find pt goint through i1=0
    xnw=xct+f*(xnt-xct);
    if abs(xnw(2))<err        % check for i2=0
     act=1;
    else
     act=2;
    end;
   elseif (mct==1 & mnt==2) | (mct==2 & mnt==1) | (mct==3 & mnt==4) ...
     | (mct==4 & mnt==3)
    f=-vcct/(vcnt-vcct);      % den won't be zeros or dx // vc=0
    xnw=xct+f*(xnt-xct);
    if xnw(2)==xnw(1)         % going through i1=i2 is allowed
     act=1;
    else
     act=2;
    end
   else          % The transition can't be done without an intermiate step
    disp('Info.  Transition not listed or not allowed.  (act=2)');
    act=2;       % half the time step dt
   end;

  else                    % Impossible cases  ==========================
%   disp('Message, srctrans(): Change in M and S1<->S2.')
%   act=2;       % Causes problems because the boundary is assigned to S1
                 % Reduction in dt does not solve some cases
   disp('Should never get here. ');
  end

  if act==1
   dtcum=dtcum+dt;
   dt=dtorg-dtcum;        % dt is set to the remaining portion to next t pt
   xct=xnt;
```

```
    mct=mnt;
    sct=snt;
   elseif act==3          % same code for act==3 and act==1
    dtcum=dtcum+dt;       %  with mtest section take out
    dt=dtorg-dtcum;       % adjust dt and rerun
    xct=xnt;
    mct=mnt;
    sct=snt;
%    [mtest stest]=ckcfg(xnt,e1nt,e2nt,para); % just checking ...
%    if mnt~=mtest                             % may take these lines out
%     disp('discrepency in M');
%     [mct sct xct' e1ct e2ct;
%      mnt snt xnt' e1nt e2nt;
%      mtest stest zeros(1,5)]
%    end;
%    if snt~=stest
%     disp('discrepency in S');
%     [mct sct xct' e1ct e2ct;
%      mnt snt xnt' e1nt e2nt;
%      mtest stest zeros(1,5)]
%    end;
   elseif act==2          % else half dt and recalculate as much as necessary
    dt=dt/2;
    disp('Reduction in dt at');
    disp([mct sct xct' e1ct e2ct; mnt snt xnt' e1nt e2nt]);
   else
    disp('Error.  Null action number.  Error in programming.');
   end

   if abs(dtorg-dtcum)<dtorg*1e-9
    k=k+1;                % increment k and store data only if
    xall(:,k)=xct;        %  at equally space time interval
    mall(k)=mct;          % k=1 is for x0, new data start at k=2
    sall(k)=sct;
    [e1ct e2ct]=eswitch(tall(k),para);
    [e1nt e2nt]=eswitch(tall(k)+dtorg,para);
    dt=dtorg;
    next=1;               % Set flag and ready for next regular t pt
    if VLcase==0          % load is VL
     nVLall(k)=nVLall(k-1);
    elseif VLcase==1      % load is RL in parallel with CL
%     nVLall(k)=nVLall(k-1)+(nt*nt*(abs(xall(1,k-1)+xall(2,k-1))) ...
%        -nVLall(k-1)/RL)*dtorg/CL;                  % Forward Euler
     nVLall(k)=((1-dtorg/(2*RL*CL))*nVLall(k-1)+(dtorg*nt*nt/(2*CL)) ...
       *(abs(xall(1,k-1)+xall(2,k-1))+abs(xall(1,k)+xall(2,k)))) ...
       /(1+dtorg/(2*RL*CL));                         % trapezoidal
    elseif VLcase==2      % load is current source in paralle with CL
     nVLall(k)=nVLall(k-1)+dtorg*(nt*nt*(abs(xall(1,k-1)+xall(2,k-1)) ...
      +abs(xall(1,k)+xall(2,k)))/2-nt*nt*IL)/CL;   % trapezoidal
    end                   % nvl is available only at regular t pts
   else
    tmor(kk)=tall(k)+dtcum;    % Save the critical points
    xmor(:,kk)=xct;
    mmor(kk)=mct;
    smor(kk)=sct;
    kk=kk+1;
    [e1ct e2ct]=eswitch(tall(k)+dtcum,para);
    [e1nt e2nt]=eswitch(tall(k)+dtcum+dt,para);
   end
  end
end;

disp('Message, srctrans(): Transient response computation completed.');

% ============ Compress the output to reduce the number of points =======
tall=dilute(tall,compress);
xall=dilute(xall,compress);
mall=dilute(mall,compress);
```

```
sall=dilute(sall,compress);
nVLall=dilute(nVLall,compress);
```

nextval.m

```
function xnt=nextval(x,e1,e2,e1nt,e2nt,mcfg,scfg,dt,para)
% xnt=nextval(x,e1,e2,e1nt,e2nt,mcfg,scfg,dt,para)
% Takes x(t), u(t), u(t+dt), M mode, S mode, dt
% and finds x(t+dt) using trapezoid method

% Written Spring 1993, Terrence Ho

C=para(1);
L=para(2);
E=para(3);
nVL=para(4);
if scfg==1
 vp=nVL;
elseif scfg==2
 vp=-nVL;
end;

                 % Get the right A matrix
if scfg==3       % <--- discontinuous conduction S3
 if mcfg==0
  A=[0 0 0; 0 0 0; 1/C 0 0];
 elseif mcfg==1 | mcfg==2
  A=[0 0 .5/L; 0 0 -.5/L; 0 1/C 0];
 elseif mcfg==3 | mcfg==4
  A=[0 0 -.5/L; 0 0 .5/L; 1/C 0 0];
 elseif mcfg==5 | mcfg==6
  disp('Error in nextval. M5, M6 should not occur with S3');
 else
  A=zeros(3,3);
 end
else             % <--- normal conduction S1 or S2
 if mcfg==0
  A=[0 0 -.5/L; 0 0 -.5/L; 1/C 0 0];
 elseif mcfg==1 | mcfg==2
  A=[0 0 0; 0 0 -1/L; 0 1/C 0];
 elseif mcfg==3 | mcfg==4
  A=[0 0 -1/L; 0 0 0; 1/C 0 0];
 else
  A=zeros(3,3);
 end
end
                 % Now get the u's
if scfg==3       % <--- discontinuous conduction S3
 if mcfg==0
  u=zeros(3,1);
  unt=zeros(3,1);
 elseif mcfg==1 | mcfg==3 | mcfg==7
  u=[(e1+e2)/2-E; -(e1+e2)/2+E; 0]/L;
  unt=[(e1nt+e2nt)/2-E; -(e1nt+e2nt)/2+E; 0]/L;
 elseif mcfg==2 | mcfg==4 | mcfg==8
  u=[(e1+e2)/2; -(e1+e2)/2; 0]/L;
  unt=[(e1nt+e2nt)/2; -(e1nt+e2nt)/2; 0]/L;
 else
  disp('Error in nextval. M5, M6 should not occur with S3');
 end
else             % <--- normal conduction S1 or S2
 if mcfg==0
  u=.5*(e1-e2-2*vp)*[1;1;0]/L;
  unt=.5*(e1nt-e2nt-2*vp)*[1;1;0]/L;
 elseif mcfg==1 | mcfg==3 | mcfg==7
```

```
  u=[e1-E-vp; -e2+E-vp; 0]/L;
  unt=[e1nt-E-vp; -e2nt+E-vp; 0]/L;
 elseif mcfg==2 | mcfg==4 | mcfg==8
  u=[e1-vp; -e2-vp; 0]/L;
  unt=[e1nt-vp; -e2nt-vp; 0]/L;
 elseif mcfg==5
  u=[e1-E-vp; -e2-vp; 0]/L;
  unt=[e1nt-E-vp; -e2nt-vp; 0]/L;
 else
  u=[e1-vp; -e2+E-vp; 0]/L;
  unt=[e1nt-vp; -e2nt+E-vp; 0]/L;
 end
end

xnt=inv(eye(3)-A*dt/2)*((eye(3)+A*dt/2)*x+(unt+u)*dt/2);
```

ckcfg.m

```
function [m,s]=ckcfg(x,e1,e2,para);
% [m,s]=ckcfg(x,e1,e2,para)
% Finds possible M mode and S mode for a point in state-space given
% the switching voltages.

% Written Spring 93, Terrence Ho

E=para(3);
nVL=para(4);
err=para(8);      % Tolerance used for determination of equality
i1=x(1);
i2=x(2);
vc=x(3);

if abs(i1-i2)<err
 i1=(i1+i2)/2;
 i2=i1;
end

if abs(i1+i2)<err
 i1=(i1-i2)/2;
 i2=-i1;
end


% ================ Determination of Mode M ====================
m=10;            % error trap
if vc>=E         % On the top plane (and everything above) =======
 if i1>0 & i2>0
  m=5;           % i1=0 or i2=0 is not included in M5 to avoid S3
 elseif i1>i2    % the region not of M5 must be M1 or M4
  m=1;
 elseif i1<i2
  m=4;
 else
  m=0;           % or possibly M0
 end;
elseif vc<=-E    % On the bottom plane (and everything below) =====
 if i1<0 & i2<0  % i1=0 or i2=0 is not included in M6 to avoid S3
  m=6;
 elseif i1>i2    % the region not of M6 must be M3 or M2
  m=3;
 elseif i1<i2
  m=2;
 else
  m=0;           % or possibly M0
 end;
```

```
elseif vc>0      % In the upper half, 0<vc<E ====================
 if i1>i2
  m=1;
 elseif i1<i2
  m=4;
 else
  m=0;           % for i1=i2 it may be M0
 end;
elseif vc<0      % In the bottom half, 0>vc>-E ===================
 if i1>i2
  m=3;
 elseif i1<i2
  m=2;
 else
  m=0;           % for i1=i2 it may be M0
 end;
else             % On the middle plane of vc=0 ====================
 if i1>=0 & i2<=0   % Unlike in M5 or M6, the equal signs includes
  m=7;           % the i1=0 or i2=0 boundaries for convenience in M7 & M8
 elseif i1<=0 & i2>=0
  m=8;           % Starting at the origin, M7 will be assigned !!!!****
 elseif i1>i2    % if not in either M7 or M8 and on boundary of M1 & M3
  if i1>0        % assign half to M1 since the trajectory leads to M1
   m=1;          % in that region; it tends towards M5 on the top plane
  elseif i1<0    % assign the other half to M3 since the trajectory leads
   m=3;          % to M3; it tends towards M6 on the bottom plane
  end;
 elseif i1<i2    % if not in M7 or M8 and on boundary of M2 & M4
  if i1>0        % assign half to M4 since the trajectory leads to M4
   m=4;          % in that region; it tends towards M5 on the top plane
  elseif i1<0    % assign the other half to M2 since the trajectory leads
   m=2;          % to M2; it tends towards M6 on the bottom plane
  end;
 else            % The remaining case of i1=i2 ====================
  m=0;           % It is only a possibility that it's M0
 end;
end;

if m==0          % Further determination of M0 ====================
 vc1=(e1+e2+vc)/2;
 vc2=(e1+e2-vc)/2;
 if vc1>E
  m=1;
 elseif vc1<0
  m=2;
 elseif vc2>E
  m=3;
 elseif vc2<0
  m=4;
 end;
end;

if m==10         % If no mode is found to fit the state variables ====
 disp('Error. No mode assigned to the set of state variables.');
end;

% =============== Determination of Mode S ===================
if i1+i2>0
 s=1;
elseif i1+i2<0
 s=2;
else             % i1+i2=0
 if vc>=e1-e2-2*nVL & vc<=e1-e2+2*nVL
  s=3;
 else
  s=1;           % cannot sustain S3 so arbitrarily assigned to S1
 end;
end;
```

eswitch.m

```
function [e1,e2]=eswitch(t,para)
% [e1 e2]=eswitch(t,para)
% For nonzero frequency rad/sec, ph1 and ph2 are restricted to between
% -pi and pi in radians.
% Positive phase shift the control to the left and negative to the right.
% To get a dc value for e, set freq=0, phase=0 or E
% t must be a scalar

% Written, Spring 93, Terrence Ho

E=para(3);
freq=para(5);
ph1=para(6);
ph2=para(7);

if freq==0
 e1=ph1;
 e2=ph2;
else
 e1=E*(sin(freq*t+ph1)>=0);  % E if sin >=0; 0 if sin <0
 e2=E*(sin(freq*t+ph2)>=0);
end
```

edtexact.m

```
function dt=edtexact(tct,tnt,para)
% dt=edtexact(tct,tnt,para)
% Finds the (almost) exact time of change in e1 and e2

% Written 8/4/93, Terrence Ho

E=para(3);
freq=para(5);  % freq, ph1, and ph2 are in radians
ph1=para(6);
ph2=para(7);
T=2*pi/freq;

e1ct=E*(sin(freq*tct+ph1)>=0); % E if sin >=0; 0 if sin <0
e1nt=E*(sin(freq*tnt+ph1)>=0); % same method as in eswtich.m
e2ct=E*(sin(freq*tct+ph2)>=0); % can use eswitch to find these values.
e2nt=E*(sin(freq*tnt+ph2)>=0);
dtmax=tnt-tct;

if (freq==0) | (e1ct==e1nt & e2ct==e2nt)
 disp('Message from edtexact(). No change in e1 or e2. ');
 dt=dtmax;
else
 if e1ct~=e1nt
  dtp=rem((freq*tnt+ph1),pi)/freq;  % w*(t+dtp)+ph=n*pi+delta; dtp=delta/w;
%  if e1ct<e1nt   % going from neg to pos no need to add extra dt
%   dt1=dtmax-dtp;
%  else
   dt1=(dtmax-dtp)*(1+1e-7);    % add a little to dt to avoid boundary
%  end                           % to ensure a change in e1 is maintained
 else
  dt1=dtmax;
 end
 if dt1>dtmax      % When dtmax is so close to the exact switching time
```

```
  disp('Warning, edtexact(): dt1>dtmax'); disp([dt1 dtmax]);
  dt1=dtmax;      % just use dtmax
 elseif dt1<=0
  disp('Warning, edtexact(): dt1<=0'); disp([dt1 dtmax]);
  dt1=dtmax;
 end
 if e2ct~=e2nt
  dtp=rem((freq*tnt+ph2),pi)/freq;
  if dtp>dtmax | dtp<0
   disp('Error, edtexact(): dt>dtmax or dt<0.');
  end
%  if e1ct<e1nt   % going from neg to pos no need to add extra dt
%   dt1=dtmax-dtp;
%  else
   dt2=(dtmax-dtp)*(1+1e-7);
%  end
 else
  dt2=dtmax;
 end
 if dt2>dtmax       % When dtmax is so close to the exact switching time
  disp('Warning, edtexact(): dt2>dtmax'); disp([dt2 dtmax]);
  dt2=dtmax;        % just use dtmax
 elseif dt2<=0
  disp('Warning, edtexact(): dt2<=0'); disp([dt2 dtmax]);
  dt2=dtmax;
 end
 dt=min([dt1 dt2]);   % If there is a change in both e1 and e2, take smaller
end
```

A.2 Auxiliary Programs

iavg.m

```
function [iav,iavt,ripple]=iavg(t,x,para)
% [iav,iavt,ripple]=iavg(t,x,para)
% iav=moving average with window size approximately equal to switching period
% iavt=average of last period, defined as the precalculated window size
% ripple=peak-to-peak ripple in last period

% Written Spring 93, Terrence Ho

freq=para(5);
dt=para(9);
compress=round(para(12));    % Make sure compression ratio is an integer
if compress<=0               % and it should be made positive.
 compress=1;
end;
T=2*pi/freq;
window=round(T/dt/compress); % only an approximation

iout=abs(x(1,:)+x(2,:));
if window>=length(iout)      % If there is less than one period of data
 disp('Warning, iavg(): Not enough points for the window size.');
 iav=zeros(iout);
 iavt=sum(iout)/length(iout);
 ripple=max(iout)-min(iout);
else                         % Otherwise,
 iav=zeros(1:window);        % iav is zero for the first period of data
% for i=1:window             % The other option is a variable window size
% iav(i)=sum(iout(1:i))/i;   % for the first period
 end
 for i=window+1:length(iout)
  pts=window;                % Dynamic adjustment of window size to best
  while (i-pts>=1)          % fit one period
   if (t(i)-t(i-pts+1)) > T  % window too large
    pts=pts-1;
   elseif (t(i)-t(i-pts))<T  % window too small
    pts=pts+1;
   else                      % If window is smaller than T, but window+1
    break                    % is greater than T, this is the right size.
   end                       % This is not necessary if time points are
  end                        % regularly spaced.  Dynamic adjustment will
  iav(i)=sum(iout(i-pts+1:i))/pts;    % weigh closely spaced points more.
 end
end
                             % The following codes assume no dynamic
iavlastT=iav(length(iav)-window+1:length(iav));  % window size adjustment
iavt=sum(iavlastT)/length(iavlastT);      % iavt is not necessarily equal
ripple=max(iavlastT)-min(iavlastT);       % to iav(lenght(iav))
```

plsrc.m

```
function plsrc(tin,xin,vpin,e1in,e2in,min,sin,tmin,tmax,task,size,prt)
% plsrc(t,x,vp,e1,e2,m,s,tmin,tmax,task,size,prt)
% If the number of points to be plotted exceeds 1000, they will be diluted
% to ensure fast display and printing.
% To perform multiple tasks, enter the product of the task numbers
% as the task number.
% task 2: projections of trajectories onto 1) vc vs. i2; 2) vc vs. i1;
%         and 3) i2 vs. i1
% task 3: transient response of 1) e1-e2,e1; 2)vc; 3)i1+i2,i1,i2;
%         and 4)Configuration mode number M and -S
% size==0: small plots (default)
% size==1: large plots
% prt==0: no printing (default)
% prt==1: printing, output meta file appended to plotsrc.met

for kmin=1:length(tin)          % Find the index for the beginning of
 if tin(kmin)>=tmin             % the time series
  break
 end
end
for kmax=length(tin):-1:1       % Find the index for the end of the
 if tin(kmax)<=tmax             % time series
  break
 end
end

if kmax-kmin+1<=1000            % If there are less than 1000 points
 t=tin(kmin:kmax);              % to be plotted, fine
 x=xin(:,kmin:kmax);
 vp=vpin(kmin:kmax);
 e1=e1in(kmin:kmax);
 e2=e2in(kmin:kmax);
 m=min(kmin:kmax);
 s=sin(kmin:kmax);
else                            % Otherwise, dilute the time series so
 dilute=round((kmax-kmin+1)/500);     % that there less than 1000 points
 jj=1;                          % Screen display or hard copy plotting
 kk=kmin;                       % does not need anywhere near 1000 points
 while (kk>=kmin & kk<=kmax)
  t(jj)=tin(kk);
  x(:,jj)=xin(:,kk);
  vp(jj)=vpin(kk);
  e1(jj)=e1in(kk);
  e2(jj)=e2in(kk);
  m(jj)=min(kk);
  s(jj)=sin(kk);
  jj=jj+1;
  kk=kk+dilute;
 end
end

if round(task/2)==task/2             % ====================== trajectory
 clg;
 axis('square');

 if size==1, subplot(121), else subplot(221), end
 plot(x(2,:),x(3,:),'w');
 grid; xlabel('i2'); ylabel('vc');
 demi2vc=axis; axis;                 % save the dimensions of i2 vs. vc

 if size==1, subplot(122), else subplot(222), end
 plot(x(1,:),x(3,:),'-w',x(2,:),x(3,:),'--w');
 grid; xlabel('i1,i2(dashed)'); ylabel('vc');
 demi1vc=axis; axis;                 % save the dimensions of i1 vs. vc

 if size==1
  if prt==1, meta plotsrc, else, pause, end
  clg
 end;

 if size==1, subplot(122), else subplot(224), end
 axis([demi1vc(1:2) demi2vc(1:2)]);  % use the saved dimensions so that
 plot(x(1,:),x(2,:),'w');            % the projections agree in dimensions
 grid; xlabel('i1'); ylabel('i2');
 axis;

 if prt==1, meta plotsrc, else, pause, end

 axis('normal');
end;


if round(task/3)==task/3             % ==================== transient
 clg;

 if size==1, subplot(211), else subplot(221), end
 plot(t,e1-e2,'-w', t,e1,'--w');      % Switching voltages
 grid; title('e1-e2, e1(dashed)');

 if size==1, subplot(212), else subplot(222), end
 plot(t,x(3,:),'-w',t,vp,'--w');      % vc and vp
 grid; title('vc, vp(dashed)');

 if size==1
  if prt==1, meta plotsrc, else, pause, end
  clg
 end;

 if size==1, subplot(212), else subplot(223), end
 plot(t,m,'ow',t,-s,'ow');            % topological modes
 grid; title('Mode M and -S');

 if size==1, subplot(211), else subplot(224), end   % currents
 plot(t,x(1,:)+x(2,:),'-w',t,x(1,:),'--w', t,x(2,:),'-.w');
 grid; title('i1+i2, i1(dashed), i2(dashdot)');

 if prt==1, meta plotsrc,
%else, pause,
 end
end;
```

anim.m

```
function anim(t,x,m,s,para,wsize,wspace,tpause,e1,e2,vp)
%function anim(t,x,m,s,para,wsize,wspace,tpause,e1,e2,vp)
% wsize=window size
% wspace=number of points the window should move up after each screen
% tpause=time delay between each window display
% e1,e2,vp are optional.  If not specified, they will be calculated here
% Requires Matlab 4.0 or above with 3-D plotting capabilities

% Written 10/93, Terrence Ho

C=para(1);
L=para(2);
E=para(3);
if nargin<11
 disp('Message, anim(): Computing e1, e2, and vp.');
 [e1,e2,vp]=evp(t,x,m,s,para);
end

imin=min(min(x(1:2,:))); imax=max(max(x(1:2,:)));   % setting axis limits
idif=imax-imin; iavg=.5*(imin+imax);
iabmax=max(abs(imax),abs(imin));
%vmin=min(x(3,:)); vmax=max(x(3,:));
vmin=-E; vmax=E;
vdif=vmax-vmin; vavg=.5*(vmin+vmax);
iwmin=iavg-.75*idif; iwmax=iavg+.75*idif;
vwmin=vavg-.75*vdif; vwmax=vavg+.75*vdif;

hold off; clg;
wbeg=1; wend=wbeg+wsize-1;
while wend<=length(t)
 axis([iwmin,iwmax,iwmin,iwmax,vwmin,vwmax]); hold on;
 xlabel('i1'); ylabel('i2'); zlabel('vc'); title('3-D Trajectory');
 plot3(x(1,wbeg:wend),x(2,wbeg:wend),x(3,wbeg:wend)); grid; hold on;
 plot3(iwmax*ones([1,wsize]),x(2,wbeg:wend),x(3,wbeg:wend),'w--'); hold on;
 plot3(x(1,wbeg:wend),iwmax*ones([1,wsize]),x(3,wbeg:wend),'w--'); hold on;
 plot3(x(1,wbeg:wend),x(2,wbeg:wend),vwmin*ones([1,wsize]),'w--'); hold on;
% disp(m(wend));
 i=wend;
 if m(i)==1                    % Find the center and radius of the cylinders
  sh1=-e2(i)+E-vp(i);          % for M=1,2,3,4 and plot the cylinders
  r1=sqrt(L*x(2,i)^2+C*(x(3,i)-sh1)^2);
  cylin(1,r1,sh1,sqrt(L),sqrt(C),2,0,iwmin,iwmax,30);
 elseif m(i)==2
  sh2=-e2(i)-vp(i);
  r2=sqrt(L*x(2,i)^2+C*(x(3,i)-sh2)^2);
  cylin(1,r2,sh2,sqrt(L),sqrt(C),2,0,iwmin,iwmax,30);
 elseif m(i)==3
  sh3=e1(i)-E-vp(i);
  r3=sqrt(L*x(1,i)^2+C*(x(3,i)-sh3)^2);
  cylin(2,r3,sh3,sqrt(L),sqrt(C),2,0,iwmin,iwmax,0);
 elseif m(i)==4
  sh4=e1(i)-vp(i);
  r4=sqrt(L*x(1,i)^2+C*(x(3,i)-sh4)^2);
  cylin(2,r4,sh4,sqrt(L),sqrt(C),2,0,iwmin,iwmax,0);
 elseif m(i)==0
%  planes(0,5,iwmin,iwmax,vwmin,vwmax);
 elseif m(i)==5
  planes(5,5,iabmax,iabmax,E,E);
 elseif m(i)==6
  planes(6,5,-iabmax,-iabmax,-E,-E);
 elseif m(i)==7
  planes(7,5,iabmax,-iabmax,0,0);
 elseif m(i)==7
  planes(8,5,-iabmax,iabmax,0,0);
 end
 pause(tpause);
 hold off; clg;
 wbeg=wbeg+wspace;
 wend=wend+wspace;
end
```

cylin.m

```
function [x,y,z,xl,yl,zl]=cylin(orient,r,shift,iif,vcf,noc,nol,llow,lhigh,phi)
% function [x,y,z,xl,yl,zl]=cylin(orient,r,shift,iif,vcf,noc,nol,llow,lhigh,phi)
% orient=1: axis of cylinder parallel to x-axis
% orient=2: axis of cylinder parallel to y-axis
% r=radius of the cylinder
% shift=distance between axis of cylinder and the xy-plane
% iif=factor in current
% vcf=factor in voltage
% noc=number of circles to be plotted in illustrating the cylinder
% nol=number of lines to be plotted in illustrating the cylinder
% llow=the lower limit of the cylinder
% lhigh=the upper limit of the cylinder
% phi=the shift in angle for placing the lines
% Use with Matlab 4 and up only

% Written 10/93, Terrence Ho

nopt = 50;               % use this number of points to draw the circles.
theta=(0:nopt)/nopt*2*pi;             % data for circles
x=r*cos(theta)/iif;
y=r*sin(theta)/vcf+shift;
z=linspace(llow,lhigh,noc);

if nol~=0
 thetal=((0:nol-1)/nol+phi/360)*2*pi;  % data for lines
 xl=r*cos(thetal)/iif;
 yl=r*sin(thetal)/vcf+shift;
 zl=[llow,lhigh];
end

if nargout == 0
if orient==2
 for ii=1:noc;           % draw circles
  plot3(x,z(ii)*ones(1,nopt+1),y,':w');
  hold on
 end
 if nol~=0
  for ii=1:nol;           % draw lines
   plot3([xl(ii),xl(ii)],zl,[yl(ii),yl(ii)],':w');
   hold on
  end
 end
% hold off
elseif orient==1
 for ii=1:noc;           % draw circles
  plot3(z(ii)*ones(1,nopt+1),x,y,':w');
  hold on
 end
 if nol~=0
  for ii=1:nol;           % draw lines
   plot3(zl,[xl(ii),xl(ii)],[yl(ii),yl(ii)],':w');
   hold on
  end
 end
% hold off
end
end
```

planes.m

```
function planes(mode,nol,imin,imax,vmin,vmax)
%function planes(mode,nol,imin,imax,vmin,vmax)
%For mode=0, imin,imax,vmin,vmax
%Discontinuous conduction mode plane mode=12
%For mode=5, i1=imin>0 and i2=imax>0 and vc=vmin>0
%For mode=6, i1=imin<0 and i2=imax<0 and vc=vmin<0
%For mode=7, i1=imin>0 and i2=imax<0
%For mode=8, i1=imin<0 and i2=imax>0
%Use with Matlab 4 and up only

%Written 10/93, Terrence Ho

i1=imin; i2=imax;  % for modes 5,6,7,8
vc=vmin;

if mode==0
 x=linspace(imin,imax,nol);
 z=linspace(vmin,vmax,nol);
 for i=1:length(x)
  plot3([x(i) x(i)],[x(i) x(i)],[vmin vmax],':w');
  hold on;
 end
 for i=1:length(z)
  plot3([imin imax],[imin imax],[z(i) z(i)],':w');
  hold on;
 end;
elseif mode==12
 x=linspace(imin,imax,nol);
 z=linspace(vmin,vmax,nol);
 for i=1:length(x)
  plot3([x(i) x(i)],[-x(i) -x(i)],[vmin vmax],':w');
  hold on;
 end
 for i=1:length(z)
  plot3([imin imax],[-imin -imax],[z(i) z(i)],':w');
  hold on;
 end;
elseif mode==5 | mode==6
 x=linspace(0,i1,nol);
 y=linspace(0,i2,nol);
 plot3([0 i1],[0 0],[vc vc],':w'); hold on;
 plot3([0 0],[i2 0],[vc vc],':w'); hold on;
 for i=1:length(x)
  plot3([x(i) 0],[0 x(i)],[vc vc],':w');
  hold on
 end
elseif mode==7 | mode==8
 x=linspace(0,i1,nol);
 y=linspace(0,i2,nol);
 plot3([0 i1],[0 0],[0 0],':w'); hold on;
 plot3([0 0],[i2 0],[0 0],':w'); hold on;
 for i=1:length(x)
  plot3([x(i) 0],[0 y(i)],[0 0],':w');
  hold on
 end
end;
```

fbctrl.m

```
function [phi,state]=fbctrl(t,nvl,nvlnom,init,nctr)
% [phi,state]=fbctrl(t,nvl,nvlnom,nctr)
% t has to be regularly spaced.
% nctr specifies the feedback controller desired

% Feedback controller design
% Written 12/11/93, Terrence Ho

if nctr==1          % Proportional gain
 phi=15*(nvl-nvlnom);
 state=phi;
elseif nctr==2      % PI
 if init==0
  [phi,state]=lsim(15*[1 1e3],[1 0],nvl'-nvlnom',t');
 else
  [phi,state]=lsim(15*[1 1e3],[1 0],nvl'-nvlnom',t',init');
 end;
 phi=phi';
 state=state';
elseif nctr==3      % PI with averager
 B=23e5;
 [a,b,c,d]=tf2ss(B*15*[1 2e3],[1 B 0]);
 if init==0
  [phi,state]=lsim(a,b,c,d,nvl'-nvlnom',t');
 else
  [phi,state]=lsim(a,b,c,d,nvl'-nvlnom',t',init);
 end;
 phi=phi';
 state=state';
elseif nctr==4      % Double PI with averager
 B=23e5;
 [a,b,c,d]=tf2ss(B*15*conv([1 2e3],[1 1e3]),[1 B 0 0]);
 if init==0
  [phi,state]=lsim(a,b,c,d,nvl'-nvlnom',t');
 else
  [phi,state]=lsim(a,b,c,d,nvl'-nvlnom',t',init);
 end;
 phi=phi';
 state=state';
else
 disp('Error, fbctrl(): Invalid controller number specified.');
end
```

srccomb.m

```
function [tret,xret,mret,sret]=srccomb(tall,xall,mall,sall,tmor,xmor,mmor,smor)
% [t,x,m,s]=srccomb(t1,x1,m1,s1,t2,x2,m2,s2)
% Points in the two series are combined in order of time.
% They should share no common point in time.

% Taken out from srctrans.m and written as a separate file
% 8/5/93, Terrence Ho

ii=1;
jj=1;
kk=1;
while(ii<=length(tall) & jj<=length(tmor))   % Before reach the end of
 if tall(ii)<tmor(jj)                      % the two time series
  tret(kk)=tall(ii);                       % add to the output the appropriate
  xret(:,kk)=xall(:,ii);                   % value from the series with a
  mret(kk)=mall(ii);                       % smaller time value.
  sret(kk)=sall(ii);
  ii=ii+1; kk=kk+1;
 elseif tall(ii)>tmor(jj)
  tret(kk)=tmor(jj);
  xret(:,kk)=xmor(:,jj);
  mret(kk)=mmor(jj);
  sret(kk)=mmor(jj);
  jj=jj+1; kk=kk+1;
 else
  disp('Error, srccomb(): tall(ii)==tmor(jj) should not occur.');
  disp('Break if necessary.');
 end
end
if jj==length(tmor)+1                   % If tmor ends first
 tret=[tret tall(ii:length(tall))];    % add the rest of tall to tret
 xret=[xret xall(:,ii:length(tall))];
 mret=[mret mall(ii:length(tall))];
 sret=[sret sall(ii:length(tall))];
elseif ii==tlen+1                       % Otherwise, if tall ends first
 tret=[tret tmor(jj:length(tmor))];    % add the rest of tmor to tret
 xret=[xret xmor(:,jj:length(tmor))];
 mret=[mret mmor(jj:length(tmor))];
 sret=[sret smor(jj:length(tmor))];
 disp('Message, srccomb(): t2(last)>t1(last) should be avoided.');
else
 disp('Error, srccomb(): Error in programming.');
end
```

dilute.m

```
function out=dilute(in,factor)
% out=dilute(in,factor)
% picks out the 1+n*factor-th column

factor=round(factor);
if factor==0 | factor==1
 out=in;
else
 j=1;
 k=1;
 while j<=length(in)
  out(:,k)=in(:,j);
  k=k+1;
  j=j+factor;
 end;
end;
```

evp.m

```
function [e1,e2,vp]=evp(t,x,m,s,para,nvl)
% [e1,e2,vp]=evp(t,x,m,s,para,nvl)
% nvl is optional.  Needed only when nVL is not constant
% Evaluates the input voltages e1, e2 and transformer primary voltage vp

% Written 6/11/93, Terrence Ho
% Modified 12/9/93, to enable variable nvl

for ind=1:length(t)
 [e1(ind) e2(ind)]=eswitch(t(ind),para);
 if nargin==5
  if s(ind)==1, vp(ind)=para(4);
  elseif s(ind)==2, vp(ind)=-para(4);
  else, vp(ind)=(e1(ind)-e2(ind)-x(3,ind))/2;
```

```
   end;
  else
   if s(ind)==1, vp(ind)=nvl(ind);
   elseif s(ind)==2, vp(ind)=-nvl(ind);
   else, vp(ind)=(e1(ind)-e2(ind)-x(3,ind))/2;
   end
  end
end
```

mseq.m

```
function [mseq,mcnt]=mseq(m,para)
% [mseq,mcnt]=mseq(m,para)
% mseq contains the topological mode M sequence extracted.
% mcnt contains the number of points belong to each mode
% This function extracts the mode sequence of m for its last cycle

% Written Summer 93, Terrence Ho

freq=para(5);
dt=para(9);
compress=para(12);
T=2*pi/freq;
window=round(T/dt/compress);  % only an approximation

len=length(m);
mT=m(len-window+1:len);       % take the last cycle
i=1;
start=1;
mseq(1)=mT(1);
for k=1:length(mT)
 if mT(k)~=mseq(i)
  mcnt(i)=k-start;
  start=k;
  i=i+1;
  mseq(i)=mT(k);
 end
end
mcnt(i)=k-start+1;   % take care of the last element when k=length(m)
mseq=mseq';
mcnt=mcnt';
```

A.3 Simulation Calling Programs

sim0.m

```
% SRC Simulation program, sim0.m
% Generates the trajectory field for M0
% Written April 22, 1993, Terrence Ho

clear;
clg;

!delete src.d fldm0*.met
diary src.d

%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
% parameters for actual circuit
```

```
for e=1:4
 if e==1
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 200; 0; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  clg; subplot(121)
 elseif e==2
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 0; 200; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  subplot(122)
 elseif e==3
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 0; 0; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  clg; subplot(121)
 else
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 200; 200; 1e-11; 0.005e-6; 0; .05e-6; 1];
  subplot(122)
 end

 first=1;
 axis(1.2*[-50,50,-200,200]);
 for vc=linspace(-200,200,9)
  for il=linspace(-50,50,9)
   x0=[il;il;vc];
   clear t x m s
   [t,x,m,s]=srctrans(x0,para);
   plot(x(1,:),x(3,:),'w',x(2,:),x(3,:),'w');
   if first==1, first=0; hold, end;
   plot(il,vc,'ow');
  end
 end
 xlabel('i1,i2'); ylabel('vc');  %grid;
 text(20,205,'(M5)'); text(-30,-225,'(M6)');
 if e==1| e==2, text(-60,0,'(S2)'); text(50,0,'(S1)'); text(0,80,'(S3)'); end;
 if e==1,     title('(a) Trajectories in M0 for e1=E, e2=0');
 elseif e==2, title('(b) Trajectories in M0 for e1=0, e2=E');
  meta fldm0e
 elseif e==3, title('(c) Trajectories leaving M0 for e1=0, e2=0');
 elseif e==4  title('(d) Trajectories leaving M0 for e1=E, e2=E');
  meta fldm00
 end;
 hold
end;

diary off
%!gpp fldm0e -dps
%!gpp fldm00 -dps
```

sim1.m

```
% SRC Simulation program, sim1.m
% Generates the trajectory fields in M1
% Written April 22, 1993, Terrence Ho

clear;
clg;

!delete src.d fldm1*.met
diary src.d

%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
% parameters for actual circuit

axis('square');
for e=1:4
 if e==1
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 200; 0; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
 elseif e==2
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 0; 200; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
```

```
elseif e==3
 para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 0; 0; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
else
 para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 200; 200; 1e-11; 0.005e-6; 0; .05e-6; 1];
end

clg;
vc=linspace(25,175,4);
for vci=1:4
 if     vci==1, subplot(221)
 elseif vci==2, subplot(222)
 elseif vci==3, subplot(223)
 else          , subplot(224), end
 first=1;
 axis([-30 60 -60 30]);
 for i1=linspace(-20,50,6)
  for i2=linspace(20,-50,6)
   if i1>=i2
    x0=[i1;i2;vc(vci)];
    clear t x m s
    [t,x,m,s]=srctrans(x0,para);
    plot(x(1,:),x(2,:),'w');
    if first==1, first=0; hold, end;
    if x(3,length(x)) > x(3,1)
     plot(i1,i2,'ow');
    else
     plot(i1,i2,'xw');
    end;
   end;
  end
 end
 xlabel('i1'); ylabel('i2');  %grid;
 if     vci==1, text(-25,20,'vc=25');
 elseif vci==2, text(-25,20,'vc=75');
 elseif vci==3, text(-25,20,'vc=125');
 else          , text(-25,20,'vc=175');
 end
 text(42,-8,'(S1)');  text(-5,-50,'(S2)');
 if     e==1, title('(a) M1 for e1=E, e2=0');
  text(-10,0,'(M0)');
  if vci==2 | vci==3 | vci==4, text(20,-30,'(S3)'); end;
 elseif e==2, title('(b) M1 for e1=0, e2=E');
  text(-10,0,'(M0)');
 elseif e==3, title('(a) M1 for e1=0, e2=0');
  if vci==1 | vci==2 | vci==3, text(20,-30,'(S3)'); end;
 elseif e==4, title('(b) M1 for e1=E, e2=E');
  if vci==1 | vci==2 | vci==3, text(20,-30,'(S3)'); end;
 end;
 hold
end
if     e==1, meta fldm1e0
elseif e==2, meta fldm10e
elseif e==3, meta fldm100
elseif e==4, meta fldm1ee
end;
end;

axis('normal');
diary off
%!gpp fldm1e0 -dps
%!gpp fldm10e -dps
%!gpp fldm100 -dps
%!gpp fldm1ee -dps
```

sim5.m

```
% SRC Simulation program, sim5.m
% Generates the trajectory fields in M5
% Written April 22, 1993, Terrence Ho

clear;
clg;

!delete src.d fldm5*.met
diary src.d

%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
% parameters for actual circuit

axis('square');
for e=1:4
 if e==1
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 200; 0; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  clg; subplot(121)
 elseif e==2
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 0; 200; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  subplot(122)
 elseif e==3
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 0; 0; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  clg; subplot(121)
 else
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 200; 200; 1e-11; 0.005e-6; 0; .05e-6; 1];
  subplot(122)
 end

 first=1;
 axis([-10 60 -10 60]);
 for i1=linspace(5,55,6)
  for i2=linspace(5,55,6)
   x0=[i1;i2;200];
   clear t x m s
   [t,x,m,s]=srctrans(x0,para);
   plot(x(1,:),x(2,:),'w');
   if first==1, first=0; hold, end;
   plot(i1,i2,'ow');
  end
 end
 xlabel('i1'); ylabel('i2');  %grid;
 text(25,-7,'(M1)'); text(-9,25,'(M4)');
 if e==1 | e==2, text(-10,-10,'(M0)'); end
 if e==1,     title('(a) Trajectories in M5 for e1=E, e2=0');
 elseif e==2, title('(b) Trajectories in M5 for e1=0, e2=E');
  meta fldm5e
 elseif e==3, title('(c) Trajectories in M5 for e1=0, e2=0');
 elseif e==4  title('(d) Trajectories in M5 for e1=E, e2=E');
  meta fldm50
 end;
 hold
end;

axis('normal')
diary off
%!gpp fldm5e -dps
%!gpp fldm50 -dps
```

sim7.m

```
% SRC Simulation program, sim7.m
% Generates the trajectory fields in M7
% Written April 22, 1993, Terrence Ho

clear;
clg;

!delete src.d fldm7*.met
diary src.d

%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
% parameters for actual circuit

axis('square');
for e=1:4
 if e==1
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 200; 0; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  clg; subplot(121)
 elseif e==2
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 0; 200; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  subplot(122)
 elseif e==3
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 0; 0; 1e-11; 0.005e-6; 0; .05e-6; 1 ];
  clg; subplot(121)
 else
  para=[.2e-6; 1e-6; 200; 1*8.5*8; 0; 200; 200; 1e-11; 0.005e-6; 0; .05e-6; 1];
  subplot(122)
 end

 first=1;
 axis([-10 60 -60 10]);
 for i1=linspace(5,55,6)
  for i2=linspace(-5,-55,6)
   x0=[i1;i2;0];
   clear t x m s
   [t,x,m,s]=srctrans(x0,para);
   plot(x(1,:),x(2,:),'w');
   if first==1, first=0; hold, end;
   plot(i1,i2,'ow');
  end
 end
 xlabel('i1'); ylabel('i2');  %grid;
 text(-10,-25,'(M3)'); text(20,2,'(M1)');
 text(40,-20,'(S1)'); text(15,-45,'(S2)');
 if e==3 | e==4,  text(25,-30,'(S3)'); end;
 if e==1,     title('(a) Trajectories in M7 for e1=E, e2=0');
 elseif e==2, title('(b) Trajectories in M7 for e1=0, e2=E');
  meta fldm7e
 elseif e==3, title('(c) Trajectories in M7 for e1=0, e2=0');
 elseif e==4  title('(d) Trajectories in M7 for e1=E, e2=E');
  meta fldm70
 end;
 hold
end;

axis('normal');
diary off
%!gpp fldm7e -dps
%!gpp fldm70 -dps
```

simnom.m

```
% SRC Simulation program, simnom.m
% Searches and finds the nominal operating phase angle for 4kW output power
% Modified 6/17/93, Terrence Ho

clear;
```

```
clg;
!delete nom.d
diary nom.d

% parameters for actual circuit
%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
para=[.2e-6; 1e-6; 250; 8.5*8; 2*pi*275e3; 0; -pi*135/180; 1e-11;
      0.005e-6; 0; 30e-6; 3 ];

x0=[0; 0; 0];
phasehi=116;                % These two values obtained by running more
phaselo=114;                % coarse simulations or from simsss.m
iouttarget=4000/8.5/8;      % The desired average output current
iout=0;
ite=1;
while (phasehi-phaselo)*2/(phasehi+phaselo) > .01 | ...
  abs((iout-iouttarget)/iouttarget) > .01    % within 1% accuracy
  % More stringent accuracy requirement can and will screw up the iterations
 phase=(phasehi+phaselo)/2;         % Numerical search method of bisection
 para(7)=-pi*phase/180;
 [t,x,m,s]=srctrans(x0,para);
 [iav,iout,ripple]=iavg(t,x,para);
 disp('ripple '); disp(ripple/iout);
 plot(t,iav,'w');
 if iout>iouttarget
  phasehi=phase;
 elseif iout<iouttarget
  phaselo=phase;
 else
  break;
 end
 disp('At iteration '); disp(ite);
 disp('iout= '); disp(iout);
 disp('phase= '); disp(phase);
 ite=ite+1;
end;

disp('The nominal phase for iouttarget= '); disp(iouttarget);
disp('is '); disp(phase');
disp('with iout= '); disp(iout);

save nom                             % Saves the trajectory
[e1,e2,vp]=evp(t,x,m,s,para);        % Plots the found nominal trajectory
!delete plotsrc.met
plsrc(t,x,vp,e1,e2,m,s,0,20e-6,6,1,1)
!delete plotnom.met
!mv plotsrc.met plotnom.met

diary off
```

simsss.m

```
% SRC Simulation program, simsss.m
% Finds the steady-state operating modes for an array of E and phase
% Run simssm.m after this to determine the mode and to plot
% Written 6/17/93, Terrence Ho
% Modified 12/4/93

clear;
clg;
!delete src.d
diary src.d

%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
```

```
E=200:25:350;
phase=20:10:180;

index=1;
for e=1:length(E)
 for i=1:length(phase)
  % The first 119 points belong to the grid of E and phase in all the
  % variables ??all.  Other programs may append data point after these
  % but should write to a differnt mat file as a good practice
  % parameters for actual circuit
  para=[.2e-6; 1e-6; E(e); 1*8.5*8; 2*pi*275e3; 0; -pi*phase(i)/180; 1e-11;
      0.005e-6; 0; 20e-6; 2 ];
  x0=[0; 0; 0];
  freq=para(5);
  dt=para(9);
  compress=para(12);
  T=2*pi/freq;
  window=round(T/dt/compress);         % only an approximation

  clear t x m s
  [t,x,m,s]=srctrans(x0,para);
  [iav,iavt,ripple]=iavg(t,x,para);    % find average current
  ss=[ripple<.01*iavt; ripple<.02*iavt; ripple<.05*iavt];
  if ss==zeros(3,1)                    % If ripple is more than 5%
   para(9)=0.002e-6;                   % run a longer and finer simulation
   para(11)=40e-6;
   para(12)=10;
   clear t x m s
   [t,x,m,s]=srctrans(x0,para);
   [iav,iavt,ripple]=iavg(t,x,para);
   ss=[ripple<.01*iavt; ripple<.02*iavt; ripple<.05*iavt];
  end
  Eall(index)=E(e);                    % save all the variables
  phall(index)=phase(i);
  xall(:,index)=x(:,length(t));
  paraall(:,index)=para;
  ssall(:,index)=ss;
  [msequ,mcnt]=mseq(m,para);           % Find the M mode sequence
  mlen=length(msequ);
  msequall(1:mlen,index)=msequ;
  mcntall(1:mlen,index)=mcnt;
  rippleall(index)=ripple;
  igavall(index)=iavt;
  igav(i,e)=iavt;
  disp('Steady-state? '); disp(ss);
%  clg; plot(t,iav,'w'); grid; title('Moving average iav at primary');
  totalpt=index;
  index=index+1;

  save sssa    % temporary file
 end;
 save sssb     % temporary file
end;
save sss

diary off
```

simsssi.m

```
% Simulation Program, simsssi.m
% Plots the stead-state output currents vs. phase for various E
% Written Summer 93, Terrence Ho
% Modified 11/27/93

clear
```

```
clg
load sss
iouttarg=ones(phase)*4000/para(4);
plot(phase,igav,'w',phase,igav,'ow',phase,iouttarg,'-w'); grid;
text(phase(i),igav(i,1),'E=200V');
text(phase(i),igav(i,2),'E=225V');
text(phase(i),igav(i,3),'E=250V');
text(phase(i),igav(i,4),'E=275V');
text(phase(i),igav(i,5),'E=300V');
text(phase(i),igav(i,6),'E=325V');
text(phase(i),igav(i,7),'E=350V');
text(phase(i),58.82,'58.82A');
xlabel('phase, degrees'); ylabel('iout referred to primary, Amps');
title('Average output current at steady-state')
!delete iavpe.met
meta iavpe
```

simssa.m

```
% SRC Simulation program, simssa.m
% Adds more data points to the basic grid used in simsss.m
% Run simssm.m after this to determine the mode and to plot
% Written based on simsss.m, 12/6/93, Terrence Ho

clear;
clg;
!delete src.d
diary src.d

%!cp sss.mat ssa.mat    % first time use sss.mat
load ssa

% The following new points are added to the grid in simsss.m ============
%E=210;  phase=95:10:145;
%E=240;  phase=[105 135 145];
%E=265;  phase=[105 115 125 145];
%E=290;  phase=[115 125 145];
%E=[315 340]; phase=[105 115];
%E=[340 315 290 270 220]; phase=[110 120];
%E=[285 275 270 250 240 225]; phase=125;
%E=[210 240]; phase=130;
%E=[220 235]; phase=[125 130 135];
%E=[180 370]; phase=20:10:180;
%E=140; phase=20:20:180;
%E=160; phase=20:20:180;
%E=350:-50:150; phase=177;
%E=190; phase=[85 95 140 145 150 155];
%E=[225 160 140]; phase=90;
%E=210; phase=[90 100 140];
%E=220; phase=105;
%E=225; phase=[95 115 135];
%E=250; phase=105;
%E=[370 350 275]; phase=105;
%E=[370 350 325 300 275]; phase=115;
%E=[180 170 150]; phase=85;
%E=190; phase=90;
%E=[220 200 170]; phase=95;
%E=[260 240]; phase=100;
%E=[360 280 220]; phase=115;
%E=[340 330 320 280 250 230 220 200 170]; phase=145;
%E=260; phase=120;
%E=200; phase=135;
%E=[190 190 160 160 150 160 140 170 180 230 290 360 360 360 280 ...
%   330 210 160 360 170 160];
%phase=[35 45 50 70 80 85 85 90 95 95 105 105 110 120 120 ...
%   150 150 150 155 155 170];
%E=[260 260 250 240 230 220 210 210 200 180 190 180 170 170 170 ...
%   150 140 150 240 230 270 280 330 240 260 260 220 300 270 260 ...
%   180 160 220 230 240 260 270 280 290 310 320 340 360 370 350 ...
%   340 180 160 150 170 190 170 160 150];
%phase=[20 25 25 25 25 25 25 35 35 35 40 45 45 55 65 ...
%   65 70 75 95 100 100 105 105 120 125 130 140 145 145 145 ...
%   145 150 150 150 150 150 150 150 150 150 150 150 150 155 155 ...
%   155 155 155 160 160 160 165 165 165];
%E=[240 230 220 210 200 210 190 170 170 160 170 160 170 160 150 ...
%   140 280 300 310 170 150 330 150 140 140];
%phase=[20 20 20 20 25 30 30 35 40 45 50 55 60 65 70 ...
%    75 110 125 145 150 155 150 165 170 170];
%E=360:-10:260;
%phase=117.25;
%E=[190 190 180 170 160 150 150 150 140 330 150 150 140];
%phase=[20 25 25 30 35 50 55 60 65 155 170 175 175];
%E=[170 170 170 160 320 330 350 360 370 220 210 200 190 265];
%phase=[70 75 80 75 105 110 145 145 145 100 120 125 135 120];
%E=[370 370 370 370 360 360 350 350 340 340 ...
%   330 330 320 320 310 310 300 290 280 260 ...
%   260 250 215 215 210 200 200 190 180 185 ...
%   175 180 290 280 270 265 260 250 245 215 ...
%   260 250 245 245 240 230 215 210 220 215 ...
%   215 210 210 200 200 190 190 180 180 170 ...
%   170 160 160];
%phase=[102.5 107.5 112.5 117.5 107.5 112.5 107.5 112.5 107.5 112.5 ...
%   107.5 112.5 107.5 112.5 110   112.5 112.5 112.5 112.5 105   ...
%    95    95   102.5 100    97.5  97.5  92.5  92.5  92.5  90   ...
%    90    87.5 122.5 122.5 122.5 122.5 122.5 122.5 122.5 122.5 ...
%   127.5 127.5 125   127.5 127.5 127.5 127.5 127.5 132.5 132.5 ...
%   135   132.5 137.5 137.5 142.5 142.5 147.5 147.5 152.5 152.5 ...
%   157.5 157.5 162.5];
%E=[205 205 205 205 200];
%phase=[130 132.5 135 137.5 132.5];
%E=[230 230 225];
%phase=[130 132.5 132.5];

% ================================= Simulation codes starts here =======
index=totalpt+1;
for e=1:length(E)      % These two lines for single loop when length of
 i=e;                  % E and length of phase are equal
%for e=1:length(E)      % These two lines for double loop when length of
% for i=1:length(phase)% E and length of phase are not equal

  % C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; csompression
  para=[.2e-6; 1e-6; E(e); 1*8.5*8; 2*pi*275e3; 0; -pi*phase(i)/180; 1e-11;
      0.005e-6; 0; 20e-6; 1 ];
  x0=[0; 0; 0];

  [t,x,m,s]=srctrans(x0,para);
  [iav,iavt,ripple]=iavg(t,x,para);
  ss=[ripple<.01*iavt; ripple<.02*iavt; ripple<.05*iavt];
  if ss==zeros(3,1)
%   para(9)=0.002e-6; para(11)=40e-6; para(12)=10;
   para(11)=60e-6; para(12)=1;
   [t,x,m,s]=srctrans(x0,para);
   [iav,iavt,ripple]=iavg(t,x,para);
   ss=[ripple<.01*iavt; ripple<.02*iavt; ripple<.05*iavt];
  end
  Eall(index)=E(e);
  phall(index)=phase(i);
  xall(:,index)=x(:,length(t));
  paraall(:,index)=para;
  ssall(:,index)=ss;
  [msequ,mcnt]=mseq(m,para);
  mlen=length(msequ);
  msequall(1:mlen,index)=msequ;
  mcntall(1:mlen,index)=mcnt;
  rippleall(index)=ripple;
  igavall(index)=iavt;
  disp('Steady-state? '); disp(ss);
  clg; plot(t,iav,'w'); grid; title('Moving average iav at primary');
  totalpt=index;
  index=index+1;
%  save sstmp t x m s para
  clear t x m s
  save ssa
 end
%end                % uncomment this line for double loop
diary off
```

simssm.m

```
% Simulation Program, simssm.m
% Builds a operating mode map based on M sequence using data in sss or ssa
% Written 6/20/93, Terrence Ho
% Modified 12/5/93 to eliminate the shortest mode in sequences with
%  odd number of modes
% Modified 12/23/93 to eliminate modes of length=1

clear;
clg;
%load sss   % examine data points generated by simsss.m
load ssa   % examine data points generated by simssa.m
!delete msq.d
diary msq.d

%        #, length, sequence
knownseq=[0  6   0 1 7 0 2 8  0 0 0 0 0 0;
          1  8   0 1 7 3 0 2 8 4  0 0 0 0;
          2  4   0 1 0 2  0 0 0 0 0 0 0 0;
          3  6   0 1 3 0 2 4  0 0 0 0 0 0;
          4 10   0 1 5 1 3 0 2 6 2 4  0 0;
          5 12   0 1 5 1 7 3 0 2 6 2 8 4 ;
          6  8   0 1 5 1 0 2 6 2  0 0 0 0;
          7  8   0 5 1 3 0 6 2 4  0 0 0 0;
          8  6   0 5 1 0 6 2  0 0 0 0 0 0;
          9  4   0 5 0 6  0 0 0 0 0 0 0 0];

mseqsize=size(msequall);
for index=1:totalpt
  knownsize=size(knownseq);         % New entry may be added to it
  mcnt=mcntall(:,index);
  msequ=msequall(:,index);

  for k=1:mseqsize(1)               % Find the length of the sequence
   if mcnt(k)==0                    %  to be examined
    k=k-1;
    break
   end
  end
  if k~=1 & msequ(k)==msequ(1)      % the last m may be
   mcnt(1)=mcnt(1)+mcnt(k);         % same as 1st
   k=k-1;
  end                               % k is the length

  if round(k/2)~=k/2                % if k is odd, there may be a problem
   [tmp,least]=min(mcnt(1:k));      % find the shortest mode
   disp('Odd k at index='); disp(index);% This doesn't work all the time!!!
   disp('with '); disp(mcnt(least));    % If the modes is very short
   if mcnt(least)<=5                    % (arbitrary number, maybe okay to
    disp('point(s). Mode deleted: ');   % delete the mode if under this # )
    disp(msequall(least,index));        % take the mode out.
    mcnt=[mcnt(1:least-1);mcnt(least+1:k)];
    seq=[msequall(1:least-1,index);msequall(least+1:k,index)];
    k=k-1;
   else
    disp('points. Mode not deleted.');
    seq=msequall(1:k,index);
   end
  else
   seq=msequall(1:k,index);          % Otherwise, pick out the sequence
  end

  jj=1;                              % Get rid of all the length=1 modes
  k2=round(k/2);                     % in a symmetric way
  for ii=1:k2
   if mcnt(ii)>=2 | mcnt(ii+k2)>=2
    msequtmp1(jj,1)=msequ(ii); msequtmp2(jj,1)=msequ(ii+k2);
    mcntmp1(jj,1)=mcnt(ii);    mcntmp2(jj,1)=mcnt(ii+k2);
    jj=jj+1;
   end
  end
  if (jj-1)~=k2
   disp('Modes of length <2 taken out at index= '); disp(index);
   mcnt=[mcntmp1(1:jj-1);mcntmp2(1:jj-1)];
   seq=[msequtmp1(1:jj-1);msequtmp2(1:jj-1)];
   k=(jj-1)*2;
  end

  found=0;
  for n=1:knownsize(1)
   if knownseq(n,2)==k               % Compare to = length known seq
    for shift=0:k-1
     for ind=1:k                     % Get every possible order of code
      code(ind+shift)=knownseq(n,ind+2);
     end
     if shift~=0
      code(1:shift)=code(k+1:k+shift);
     end
     if code(1:k)==seq'              % and compare to seq
      found=1;
      seqmap(index)=knownseq(n,1);   % save the operating mode number
      break;
     end
    end
   end
   if found==1
    break
   end
  end

  if found==0                        % If finds a new operating mode
   disp('New sequence found at E= '); disp(Eall(index));
   disp('phase= '); disp(phall(index));
   disp('index= '); disp(index);     % add to the known table
   knownseq(knownsize(1)+1,1)=knownseq(knownsize(1))+1;
   knownseq(knownsize(1)+1,2)=k;
   knownseq(knownsize(1)+1,3:2+k)=seq';
   seqmap(index)=knownseq(knownsize(1)+1);
  end

end

disp('The known sequences are: ');
disp(knownseq);
knownsize=size(knownseq);
save msq                              % Save the information

% ==================== Plotting the operating mode map =========
%contour(mseqmap(:,length(E):-1:1)',knownsize(1),phase,E,'w')
%contour(mseqmap(:,length(E):-1:1)',1,phase,E,'i'); hold on
%for e=1:length(E)
% for i=1:length(phase)
%  plot(phase(i),E(e),'+w');
%  text(phase(i),E(e),48+mseqmap(i,e));
% end
%end
Emin=min(Eall); Emax=max(Eall); Eavg=.5*(Emin+Emax); Edev=.55*(Emax-Emin);
El=Eavg-Edev; Eh=Eavg+Edev;
pmin=min(phall); pmax=max(phall); pavg=.5*(pmin+pmax); pdev=.55*(pmax-pmin);
pl=pavg-pdev; ph=pavg+pdev;
axis([pl ph El Eh]);
for i=1:totalpt
 if seqmap(i)==0
  plot(phall(i),Eall(i),'.w'); hold on;
 elseif seqmap(i)==2 | seqmap(i)==5
  plot(phall(i),Eall(i),'+w'); hold on;
 elseif seqmap(i)==1 | seqmap(i)==9 | seqmap(i)==7
  plot(phall(i),Eall(i),'ow'); hold on;
 elseif seqmap(i)==3 | seqmap(i)==6
  plot(phall(i),Eall(i),'xw'); hold on;
 elseif seqmap(i)==4 | seqmap(i)==8
  plot(phall(i),Eall(i),'*w'); hold on;
 else
  plot(phall(i),Eall(i),'.w'); hold on;
 end
% if ssall(3,i)~=1                       % For nonconvergent or slowly
%  text(phall(i),Eall(i),'D');           % convergent points, use symbol 'D'
% end
% text(phall(i),Eall(i),48+seqmap(i));
end
axis; hold off;
xlabel('phase'); ylabel('E');
title('Operating Modes');
                                % Draw the boundary lines manually
line1=[179 179;       375 135];
line2=[151 151 147.5 147.5 144 144 164 170 179;
       375 329 324   217   211 196 153 145 145];
line3=[164 161 142;       153 153 194];
line4=[144 142 137.5 136 132   122.5 107.5 104  98.5   94 94 91.25  88.75;
       196 194 195 202.5 202.5 222.5 222.5 212.5 212.5 192 186 177.5 177.5];
line5=[113.75 113.75;       375 285];
line6=[103.75 103.75 106.25 106.25 108.75 108.75 111.25 111.25 113.75;
       375    367    362    347    342    321    316    305    302];
line7=[113.75 110 102.5 102.5 97.5 97.5 92.5 91.25 91.25 88.75 88.75 87 ...
       82.5 82.5;
       285    277 277   258 252 217 210 202 189 185 175 157 152 135];
line8=[113.75 116.75 116.75 118.75 118.75 121.25 121.25 123.75 123.75 ...
       126.25 126.25 128.75 128.75 133.75 133.75 141;
       285    282     275    272    265    261    255    251    242.5 ...
       242.5  236    232    225    216    210  194];
line9=[142 138 133 133 131.25 131.25 128.75 128.75 126.25 126.25 123.75 ...
       123.75 118.75 118.75 116.25 116.25 113.75;
       194 212 220 225 230    236    239     246   249     261   265  ...
       282 298 322 327 363 365];
line0=[17  37.5 50 82.5;       240 185 165 152];
linei=[135.5 125 115 106 98.5 92 83; 200 225 250 275 300 325 350];
hold on
plot(line1(1,:),line1(2,:),'w'); plot(line2(1,:),line2(2,:),'w');
plot(line3(1,:),line3(2,:),'w'); plot(line4(1,:),line4(2,:),'w');
plot(line5(1,:),line5(2,:),'w'); plot(line6(1,:),line6(2,:),'w');
plot(line7(1,:),line7(2,:),'w'); plot(line8(1,:),line8(2,:),'w');
plot(line9(1,:),line9(2,:),'w'); plot(line0(1,:),line0(2,:),'w');
plot(linei(1,:),linei(2,:),'--w');
text(30,150,'0'); text(52,280,'1');
text(110,150,'2'); text(111,233,'3');
text(119,280,'4'); text(108,350,'5');
text(152,170,'6'); text(133,310,'7');
text(165,280,'8'); text(182,255,'9');
hold off
!delete mseqmap.met
meta mseqmap

diary off
```

simsyo.m

```
% SRC Simulation program, simsyo.m
% Poincare plane in synchronization with control inputs
% Searching for the nominal point at two places
% 8/7/93 Terrence Ho

clear;
clg;

para=[.2e-6; 1e-6; 250; 8.5*8; 2*pi*275e3; 0; -pi*115.625/180; 1e-11;
      0.001e-6; 0; 30e-6; 2];
freq=para(5);  % freq, ph1, and ph2 are in radians
ph1=para(6);
ph2=para(7);
T=2*pi/freq;
para(10)=(2*pi-ph2)/freq;
para(11)=para(10)+1.1*T;

dothis=0;
if dothis==1
N=1000;   % N and n used to enable running the program from the middle
n=1;                                    % Locate the nominal starting
%for i1=46.79:.01:46.85                 % point at E->0
% for vc=179.90:.05:180.30
for i1=46.819:.001:46.825               % Simulate from a grid of points
 for vc=180.05:.01:180.10
  if n>N
   n
   x0=[i1;i1;vc];
   x10(:,n)=x0;
   [tr,xr,mr,sr,tm,xm,mm,sm]=srctrans(x0,para);
   t1=[t1 tm];
   x1=[x1 xm];
   n1=[n1 length(tm)];
   save syo1
  end
  n=n+1;
 end
end

%axis([46.7 46.9 179.8 180.4])
axis([46.818 46.826 180.04 180.11])
for ii=1:length(n1)                      % Plot the points and manually
 b=sum(n1(1:ii))-n1(ii)+1;               % determine the best point from
 e=sum(n1(1:ii));                        % the graphs
 plot(x10(2,ii),x10(3,ii),'ow',x1(2,b:e),x1(3,b:e),'xw'); grid;
 x10(:,ii)'
end
xlabel('i2'), ylabel('vc');
axis;
% Nominal found to be close to (46.820->46.821,180.08->180.09)

load syo2
para(10)=(3*pi-ph1)/freq;
para(11)=para(10)+1.1*T;
M=40;
```

```
m=1;
%for i1=52.61:.01:52.63
% for i2=-60.11:.01:-60.09
%  for vc=108.39:.01:108.41
for i1=52.612:.002:52.618             % Locating the nominal starting
 for i2=-60.118:.002:-60.112          % point at 0-> -E
  for vc=108.40:.002:108.41           % Simulate from a grid of points
   if m>M
    m
    x0=[i1;i2;vc]
    x20(:,m)=[i1;i2;vc];
    [tr,xr,mr,sr,tm,xm,mm,sm]=srctrans(x0,para);
    t2=[t2 tm];
    x2=[x2 xm];
    n2=[n2 length(tm)];
    save syo2
   end
   m=m+1;
  end
 end
end

end  % of dothis  ==========

load syo2
for ii=1:length(n2)                   % Manually search for the best
 b=sum(n2(1:ii))-n2(ii)+1;            % starting point from the graphs
 e=sum(n2(1:ii));
 x20(:,ii)'
 clg
 subplot(221); axis([-60.119 -60.111 108.39 108.42]);
 plot(x20(2,ii),x20(3,ii),'ow',x2(2,b:e),x2(3,b:e),'xw'); grid;
 xlabel('i2'), ylabel('vc');
 subplot(222); axis([52.611 52.619 108.39 108.42]);
 plot(x20(1,ii),x20(3,ii),'ow',x2(1,b:e),x2(3,b:e),'xw'); grid;
 xlabel('i1'), ylabel('vc');
 subplot(224); axis([52.611 52.619 -60.119 -60.111]);
 plot(x20(1,ii),x20(2,ii),'ow',x2(1,b:e),x2(2,b:e),'xw'); grid;
 xlabel('i1'), ylabel('i2');
% pause
end
axis;
% Nominal near 52.616 -60.116 108.405
```

simsyq.m

```
% SRC Simulation program, simsyq.m
% Poincare plane in synchronization with control inputs
% Finding the small-signal transition matrix at M0->M1
% Written 8/9/93, Terrence Ho
% Rerun on 10/24/93

clear;
clg;
!delete syq.d
diary syq.d

load syq

dothis=0;
para=[.2e-6; 1e-6; 250; 8.5*8; 2*pi*275e3; 0; -pi*115.625/180; 1e-11;
      0.001e-6; 0; 30e-6; 2];
freq=para(5);  % freq, ph1, and ph2 are in radians
ph1=para(6);
ph2=para(7);
```

```
T=2*pi/freq;
para1=para;
para1(10)=(2*pi-ph2)/freq;
para1(11)=para1(10)+1.1*T;
x01=[46.8208;46.8208;180.088];
para2=para;
para2(10)=(3*pi-ph1)/freq;
para2(11)=para2(10)+1.1*T;
x02=[52.616; -60.116; 108.405];

if dothis==1
P1=eye(3);
P2=[1 1 0;
    1 0 1;
    0 1 1];
[tr,xr,mr,sr,t1,x1,mm,sm]=srctrans(x01,para1);
[tr,xr,mr,sr,t11,x11,mm,sm]=srctrans(x01+P1(:,1),para1);
[tr,xr,mr,sr,t12,x12,mm,sm]=srctrans(x01+P1(:,2),para1);
[tr,xr,mr,sr,t13,x13,mm,sm]=srctrans(x01+P1(:,3),para1);
[tr,xr,mr,sr,t21,x21,mm,sm]=srctrans(x01+P2(:,1),para1);
[tr,xr,mr,sr,t22,x22,mm,sm]=srctrans(x01+P2(:,2),para1);
[tr,xr,mr,sr,t23,x23,mm,sm]=srctrans(x01+P2(:,3),para1);
clear tr xr mr sr, save syq

[tr,xr,mr,sr,tt1,xx1,mm,sm]=srctrans(x02,para2);
[tr,xr,mr,sr,tt11,xx11,mm,sm]=srctrans(x02+P1(:,1),para2);
[tr,xr,mr,sr,tt12,xx12,mm,sm]=srctrans(x02+P1(:,2),para2);
[tr,xr,mr,sr,tt13,xx13,mm,sm]=srctrans(x02+P1(:,3),para2);
[tr,xr,mr,sr,tt21,xx21,mm,sm]=srctrans(x02+P2(:,1),para2);
[tr,xr,mr,sr,tt22,xx22,mm,sm]=srctrans(x02+P2(:,2),para2);
[tr,xr,mr,sr,tt23,xx23,mm,sm]=srctrans(x02+P2(:,3),para2);
clear tr xr mr sr, save syq

P3=.5 * eye(3);
P4=.5 * [1 1 0;
         1 0 1;
         0 1 1];

[tr,xr,mr,sr,t31,x31,mm,sm]=srctrans(x01+P3(:,1),para1);
[tr,xr,mr,sr,t32,x32,mm,sm]=srctrans(x01+P3(:,2),para1);
[tr,xr,mr,sr,t33,x33,mm,sm]=srctrans(x01+P3(:,3),para1);
[tr,xr,mr,sr,t41,x41,mm,sm]=srctrans(x01+P4(:,1),para1);
[tr,xr,mr,sr,t42,x42,mm,sm]=srctrans(x01+P4(:,2),para1);
[tr,xr,mr,sr,t43,x43,mm,sm]=srctrans(x01+P4(:,3),para1);
clear tr xr mr sr, save syq

[tr,xr,mr,sr,tt31,xx31,mm,sm]=srctrans(x02+P3(:,1),para2);
[tr,xr,mr,sr,tt32,xx32,mm,sm]=srctrans(x02+P3(:,2),para2);
[tr,xr,mr,sr,tt33,xx33,mm,sm]=srctrans(x02+P3(:,3),para2);
[tr,xr,mr,sr,tt41,xx41,mm,sm]=srctrans(x02+P4(:,1),para2);
[tr,xr,mr,sr,tt42,xx42,mm,sm]=srctrans(x02+P4(:,2),para2);
[tr,xr,mr,sr,tt43,xx43,mm,sm]=srctrans(x02+P4(:,3),para2);
clear tr xr mr sr, save syq

P5=[.6 .7 .9;
    .5 .8 1;
    .4 .5 .3];

[tr,xr,mr,sr,t51,x51,mm,sm]=srctrans(x01+[P5(2,2);P5(2:3,2)],para1);
[tr,xr,mr,sr,t52,x52,mm,sm]=srctrans(x01+[P5(2,3);P5(2:3,3)],para1);

[tr,xr,mr,sr,tt51,xx51,mm,sm]=srctrans(x02+P5(:,1),para2);
[tr,xr,mr,sr,tt52,xx52,mm,sm]=srctrans(x02+P5(:,2),para2);
[tr,xr,mr,sr,tt53,xx53,mm,sm]=srctrans(x02+P5(:,3),para2);
clear tr xr mr sr, save syq

para1(7)=para(7)+1*pi/180;
[tr,xr,mr,sr,t6,x6,mm,sm]=srctrans(x01,para1);
```

```
para1(7)=para(7)+.5*pi/180;
[tr,xr,mr,sr,t7,x7,mm,sm]=srctrans(x01,para1);
clear tr xr mr sr, save syq

para2(7)=para(7)+1*pi/180;
[tr,xr,mr,sr,tt6,xx6,mm,sm]=srctrans(x02,para2);
para2(7)=para(7)+.5*pi/180;
[tr,xr,mr,sr,tt7,xx7,mm,sm]=srctrans(x02,para2);
clear tr xr mr sr, save syq
end    % of dothis =======================================

disp('Simulation from E=>0: States at transition points at time');
disp((t1-para1(10))')
disp('from initial state ')
disp(x01')
disp('are ')
disp(x1')

T1=[x11(:,10)-x01 x12(:,10)-x01 x13(:,10)-x01];
[v1,d1]=eig(T1);  eig(T1)';
 T2=[x11(:,10)-x1(:,10) x12(:,10)-x1(:,10) x13(:,10)-x1(:,10)];
 [v2,d2]=eig(T2);   eig(T2)';
T3=[x21(:,10)-x01 x21(:,10)-x01 x13(:,10)-x01];
T3=T3(2:3,2:3);
[v3,d3]=eig(T3);  eig(T3)';
Ta=[x21(:,10)-x01 x22(:,10)-x01 x23(:,10)-x01]*inv(P2);
[va,da]=eig(Ta);  eig(Ta)';

Tb=[x31(:,10)-x01 x32(:,10)-x01 x33(:,10)-x01]*inv(P3); eig(Tb)';
Tc=[x41(:,10)-x01 x41(:,10)-x01 x33(:,10)-x01];
Tc=Tc(2:3,2:3)*inv(.5*eye(2));  eig(Tc)';
Td=[x41(:,10)-x01 x42(:,10)-x01 x43(:,10)-x01]*inv(P4); eig(Td)';
Te=[x51(:,10)-x01 x52(:,10)-x01];
Te=Te(2:3,:)*inv(P5(2:3,2:3));   eig(Te)';

disp('Transition matrix and eigenvalues at E=>0: ');
Tavg1=(T3+Tc+Te)/3; disp(Tavg1)
disp(eig(Tavg1))

T4=[xx11(:,11)-x02 xx12(:,11)-x02 xx13(:,11)-x02];
[v4,d4]=eig(T4); eig(T4)';
T5=[xx21(:,11)-x02 xx22(:,11)-x02 xx23(:,11)-x02]*inv(P2);
[v5,d5]=eig(T5);  eig(T5)';
T6=[xx31(:,11)-x02 xx32(:,11)-x02 xx33(:,11)-x02]*inv(P3); eig(T6)';
T7=[xx41(:,11)-x02 xx42(:,11)-x02 xx43(:,11)-x02]*inv(P4); eig(T7)';
T8=[xx51(:,11)-x02 xx52(:,11)-x02 xx53(:,11)-x02]*inv(P5); eig(T8)';

disp('Transition matrix and eigenvalues at 0=>-E: ');
Tavg2=(T4+T5+T6+T7+T8)/5; disp(Tavg2)
disp(eig(Tavg2)')

load syq
B6=x6(:,10)-x01;
B7=2*(x7(:,10)-x01);
Ba=xx6(:,11)-x02;
Bb=2*(xx7(:,11)-x02);

diary off
```

simplt.m

```
% SRC Simulation program, simplt.m
% Finds the numerator in plant transfer function
% Written 12/8/93, Terrence Ho
% Modified 12/19/93
```

```
clear;
clg;
!delete src.d
diary src.d

%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
para=[.2e-6; 1e-6; 250; 8.5*8; 2*pi*275e3; 0; -pi*115.625/180; 1e-11;
      0.002e-6; 0; 30e-6; 2 ];

x0=[-52.616;60.116;-108.405];   % values from simsyo.m

[t1,x1,m1,s1,tt1,xx1,mm1,ss1]=srctrans(x0,para);
[iav,iavt1,ripple1]=iavg(t1,x1,para);
clg; plot(t1,iav,'w'); grid; title('Moving average iav at primary');
disp('Average current at nominal is '); disp(iavt1);

para2=para;
para2(7)=-pi*116.625/180;       % Increase the phase angle
[t2,x2,m2,s2,tt2,xx2,mm2,ss2]=srctrans(x0,para2);
[iav,iavt2,ripple2]=iavg(t2,x2,para2);
clg; plot(t2,iav,'w'); grid; title('Moving average iav at primary');
disp('Average current after step change in phase by +1 degree is ');
disp(iavt2);

para3=para;
para3(7)=-pi*114.625/180;       % Decrease the phase angle
[t3,x3,m3,s3,tt3,xx3,mm3,ss3]=srctrans(x0,para3);
[iav,iavt3,ripple3]=iavg(t3,x3,para3);
clg; plot(t3,iav,'w'); grid; title('Moving average iav at primary');
disp('Average current after step change in phase by -1 degree is ');
disp(iavt3);

para4=para;
para4(3)=252;
[t4,x4,m4,s4,tt4,xx4,mm4,ss4]=srctrans(x0,para4);
[iav,iavt4,ripple4]=iavg(t4,x4,para4);
clg; plot(t4,iav,'w'); grid; title('Moving average iav at primary');
disp('Average current after step change in supply voltage by 2V is ');
disp(iavt4);

para5=para;
para5(3)=248;
[t5,x5,m5,s5,tt5,xx5,mm5,ss5]=srctrans(x0,para5);
[iav,iavt5,ripple5]=iavg(t5,x5,para5);
clg; plot(t5,iav,'w'); grid; title('Moving average iav at primary');
disp('Average current after step change in supply voltage by -2V is ');
disp(iavt5);

save plt para x0 iavt1 iavt2 iavt3 iavt4 iavt5

diary off
```

simctr.m

```
% SRC Simulation program, simctr.m
% System with feeback controller
% Specify output model in param
% Specify feedback controller in fbctrl()
% Specify initial value of nVL in para
% Written 12/11/93, Terrence Ho
% Modified 12/93

clear;
clg;
!delete src.d
diary src.d
```

```
CTRno=3;                % set feedback controller, final design is 3
OUTno=1;                % V=0; R//C=1; I//C=2;
CYCno=40;               % set number of cycles to be simulated
nvlnom=8.5*8;           % set nominal nVL
phinom=115.625;         % phinom is the duty ratio in degrees
Enom=250;               % set Enom
nvlinit=8.5*8;          % set initial nVL
dE=0;                   % specify if there is drooping in E
Einit=250;              % set initial E
%x0=[0;0;0];                        % Set initial states: zero initial
x0=[-52.616;60.116;-108.405];       % nominal values from simsyo.m

%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
para=[.2e-6; 1e-6; Einit; nvlinit; 2*pi*275e3; 0; -pi*115.625/180; 1e-11;
      0.005e-6; 0; 20e-6; 1 ];
%  CL, RL, ILn, nvlcase
param=[0.01; 0.0180625; 8.5/0.0180625/8; 8; OUTno];

freq=para(5);
T=2*pi/freq;

for ite=1:CYCno
 if dE==1, para(3)=para(3)-1; end   % Drooping in E at -1V/cycle
                        % Comment out the following block if no load change
 if ite>10 & ite <=20               % Full->no load
  if OUTno==1, param(2)=1.80625;
  elseif OUTno==2, param(3)=(8.5/0.0180625/8)*.01; end
  if dE==1, para(3)=para(3)+2; end  % E rises at +1/cyc in no load
 elseif ite>20                      % No->full load
  if OUTno==1, param(2)=0.0180625;
  elseif OUTno==2, param(3)=8.5/0.0180625/8; end
 end

 para(10)=(ite-1)*T;      % Start at the beginning of a cycle
 para(11)=para(10)+1.1*T;% Simulate for a cycle
 [t,x,m,s,tt,xx,mm,ss,nvl]=srctrans(x0,para,param);
 [e1,e2,vp]=evp(t,x,m,s,para,nvl);
 for ii=1:length(tt)      % Check for the end of the cycle
  if abs(tt(ii)-ite*T) < para(8)
   break
  end
 end
 if ii==1
  disp('Warning: the precise switching may not have been found.');
 end
 x0=xx(:,ii);             % Reset starting point for next cycle
 for jj=length(t):-1:1    % Find the length of the current cycle
  if t(jj)<=ite*T
   break
  end
 end

 if ite==1, initstate=0; end
 [dp,state]=fbctrl(t,nvl,nvlnom,initstate,CTRno);
 initstate=state(:,jj);  % Reset controller initial condition for next cycle
 para(4)=nvl(jj);         % Reset initial nvl for next cycle
 tDT(ite)=t(jj);          % Save sampled points
 dpDT(ite)=dp(jj)+(0.91/2.53)*(para(3)-Enom);  % Disturbance feedforward
 nvlDT(ite)=nvl(jj);
                          % Nonlinear control rule:
 if (dpDT(ite)>phinom) %| (nvlDT(ite)>nvlnom+1.4)
  disp('Control saturation. phi is set to 0');
  dpDT(ite)=phinom;
 elseif (dpDT(ite)<phinom-180) %| (nvlDT(ite)<nvlnom-1.4)
  disp('Control saturation.  phi is set to 180');
  dpDT(ite)=phinom-180;
 end
 para(7)=-(phinom-dpDT(ite))*pi/180;
 disp(dpDT);

 if ite==1              % put the whole thing together
  ta=t(1:jj);   xa=x(:,1:jj);
  ma=m(1:jj);   sa=s(1:jj);
  e1a=e1(1:jj); e2a=e2(1:jj);
  vpa=vp(1:jj); va=nvl(1:jj);
  dpa=dp(1:jj);
 else
  ta=[ta t(1:jj)];    xa=[xa x(:,1:jj)];
  ma=[ma m(1:jj)];    sa=[sa s(1:jj)];
  e1a=[e1a e1(1:jj)]; e2a=[e2a e2(1:jj)];
  vpa=[vpa vp(1:jj)]; va=[va nvl(1:jj)];
  dpa=[dpa dp(1:jj)];
 end

% plsrc(ta,xa,vpa,e1a,e2a,ma,sa,ta(1),ta(length(ta)),3,0,0);
 clg;                   % Draw a few graphs
 ff=round(ite/2);
 subplot(211); plot(dilute(ta,ff),dilute(va,ff),'w',tDT,nvlDT,'ow');
 grid; xlabel('t'); ylabel('nVL');
 subplot(212); plot(dilute(ta,ff),dilute(dpa,ff),'w',tDT,dpDT,'ow');
 grid; xlabel('t'); ylabel('d phi');
 !delete plotvl.met
 meta plotvl
end

diary off
```

simstr.m

```
% SRC Simulation program, simstr.m
% System with feeback controller, Start-up simulation
% Specify output model in param
% Specify feedback controller in fbctrl()
% Specify initial value of nVL in para
% Written 12/11/93, Terrence Ho
% Modified 12/93

clear;
clg;
!delete str.d
diary str.d

CTRno=3;                % set feedback controller, final design is 3
OUTno=1;                % V=0; R//C=1; I//C=2;
CYCno=60;               % set number of cycles to be simulated
phinom=115.625;         % phinom is the duty ratio in degrees
Enom=250;               % set Enom
nvlinit=0.1;            % set initial nVL
dE=1;                   % specify if there is drooping in E
Einit=300;              % set initial E
x0=[0;0;0];             % zero initial state

%  C; L; E; nVL; freq,dc; ph1; ph2; err;  dt; tlower; tupper; compression
para=[.2e-6; 1e-6; Einit; nvlinit; 2*pi*275e3; 0; -pi*115.625/180; 1e-11;
      0.005e-6; 0; 20e-6; 1 ];
%  CL, RL, ILn, nvlcase
param=[0.01; 0.0180625; 8.5/0.0180625/8; 8; OUTno];

freq=para(5);
T=2*pi/freq;

nvlnombeg=nvlinit;
```

```
for ite=1:CYCno
 if dE==1, para(3)=para(3)-1; end   % Drooping in E at -1V/cycle

 para(10)=(ite-1)*T;     % Start at the beginning of a cycle
 para(11)=para(10)+1.1*T;% Simulate for a cycle
 [t,x,m,s,tt,xx,mm,ss,nvl]=srctrans(x0,para,param);
 for ii=1:length(tt)     % Check for the end of the cycle
  if abs(tt(ii)-ite*T) < para(8)
   break
  end
 end
 if ii==1
  disp('Warning: the precise switching may not have been found.');
 end
 x0=xx(:,ii);            % Reset starting point for next cycle
 for jj=length(t):-1:1   % Find the length of the current cycle
  if t(jj)<=ite*T
   break
  end
 end
                         % Soft start
 dnvlnom=2.27/(jj-1);    % nvlnom rises at 2.27/cycle to nominal
 nvlnom=nvlnombeg:dnvlnom:nvlnombeg+dnvlnom*(length(nvl)-1);
 for ii=1:length(nvlnom)
  if nvlnom(ii)>8.5*8
   nvlnom(ii)=8.5*8;
  end
 end
 nvlnombeg=nvlnom(jj);

 if ite==1, initstate=0; end
 [dp,state]=fbctrl(t,nvl,nvlnom,initstate,CTRno);
 initstate=state(:,jj);  % Reset controller initial condition for next cycle
 para(4)=nvl(jj);        % Reset nvl for next cycle
 tDT(ite)=t(jj);         % Save sampled points
 dpDT(ite)=dp(jj)+(0.91/2.53)*(para(3)-Enom);  % Disturbance feedforward
 nvlDT(ite)=nvl(jj);
                         % Nonlinear control rule:
 if dpDT(ite)>phinom | (nvlDT(ite)>nvlnom+1.4)
  disp('Control saturation. phi is set to 0');
  dpDT(ite)=phinom;
 elseif dpDT(ite)<phinom-180 | (nvlDT(ite)<nvlnom-1.4)
  disp('Control saturation.  phi is set to 180');
  dpDT(ite)=phinom-180;
 end
 para(7)=-(phinom-dpDT(ite))*pi/180;
 disp(dpDT);

 if ite==1               % put the whole thing together
  ta=t(1:jj);   xa=x(:,1:jj);
  va=nvl(1:jj); dpa=dp(1:jj);
 else
  ta=[ta t(1:jj)];    xa=[xa x(:,1:jj)];
  va=[va nvl(1:jj)];  dpa=[dpa dp(1:jj)];
 end

 clg;                    % Draw a few graphs
 ff=round(ite/2);
 subplot(211); plot(dilute(ta,ff),dilute(va,ff),'w',tDT,nvlDT,'ow');
 grid; xlabel('t'); ylabel('nVL');
 subplot(212); plot(dilute(ta,ff),dilute(dpa,ff),'w',tDT,dpDT,'ow');
 grid; xlabel('t'); ylabel('d phi');
 !delete plotstr.met
 meta plotstr
end

diary off
```

A.4 Other Programs

spiralm1.m

```
% spiralm1.m
% Draws the spiral for M1
% For matlab4
% Written, 11/22/93 Terrence Ho

clear;
clg;

C=.2e-6;
L= 1e-6;
E=250;
w=1/sqrt(L*C);
e1=E;
e2=E;
vp=8.5*8;

x0=[90; 40; 60];
t=linspace(0,5e-6,200);
for i=1:length(t);
 Phi1=[1 0 0; 0 cos(w*t(i)) -sqrt(C/L)*sin(w*t(i));
   0 sqrt(L/C)*sin(w*t(i)) cos(w*t(i))];
 Psi1=[t(i)*(e1-E-vp)/L; sqrt(C/L)*sin(w*t(i))*(-e2+E-vp);
      (1-cos(w*t(i)))*(-e2+E-vp)];
 x(:,i)=Phi1*x0+Psi1;
end

plot3(x(1,:),x(2,:),x(3,:),'w');
xlabel('i1'); ylabel('i2'); zlabel('vc');
view(37.5,30); grid;

hold on
for i=1:length(t)
 if x(1,i)+x(2,i)>0 & x(1,i)>x(2,i) & x(3,i)>0 & x(3,i)<E
  plot3(x(1,i),x(2,i),x(3,i),'ow');
 end
end
hold off

print spiralm1
```

spiralm3.m

```
% spiralm3.m
% Draws the spiral for M3
% For matlab4
% Written, 11/22/93 Terrence Ho

clear;
clg;

C=.2e-6;
L= 1e-6;
E=250;
w=1/sqrt(L*C);
e1=0;
```

```
e2=E;
vp=8.5*8;

x0=[90; 40; -60];
t=linspace(0,5e-6,200);
for i=1:length(t);
 Phi3=[cos(w*t(i)) 0 -sqrt(C/L)*sin(w*t(i));
   0 1 0; sqrt(L/C)*sin(w*t(i)) 0 cos(w*t(i))];
 Psi3=[sqrt(C/L)*sin(w*t(i))*(e1-E-vp); t(i)*(-e2+E-vp)/L;
      (1-cos(w*t(i)))*(e1-E-vp)];
 x(:,i)=Phi3*x0+Psi3;
end

plot3(x(1,:),x(2,:),x(3,:),'w');
xlabel('i1'); ylabel('i2'); zlabel('vc');
view(37.5,30); grid;

hold on
for i=1:length(t)
 if x(1,i)+x(2,i)>0 & x(1,i)>x(2,i) & x(3,i)<0 & x(3,i)>-E
  plot3(x(1,i),x(2,i),x(3,i),'ow');
 end
end
hold off

print spiralm3
```

Appendix B

Sampled-Data Model Computation and Results

B.1 Source Codes and Results in Maple

```
> with(linalg):
> EE:=250:
> L:=1.*10^(-6):
> C:=.2*10^(-6):
> nVL:=8.5*8:
> w:=1/sqrt(L*C):
> w0:=1/sqrt(2*L*C):
> freq:=275000:
> .5/freq:
> phase:=115.625:
--------------------------------------------------------------------------------
> T0:=0:
> xT0:=matrix(3,1,[46.8379,46.8379,179.9731]);

                                        [  46.8379 ]
                                        [          ]
                                 xT0 := [  46.8379 ]
                                        [          ]
                                        [ 179.9731 ]
> phi1:=t->matrix([[1, 0, 0],[0, cos(w*t), -sqrt(C/L)*sin(w*t)],
>    [0, sqrt(L/C)*sin(w*t), cos(w*t)]]):
> psi1:=(t,e1,e2,vp) ->matrix(3,1,[(e1-EE-vp)*t/L,
>    sqrt(C/L)*sin(w*t)*(-e2+EE-vp), (1-cos(w*t))*(-e2+EE-vp)]):
> f1:=(t,xinit,e1,e2,vp)->add(multiply(phi1(t),xinit),psi1(t,e1,e2,vp)):
> f1(t,xT0,EE,EE,nVL):
> T1:=fsolve(row(f1(t,xT0,EE,EE,nVL),1)[1]+row(f1(t,xT0,EE,EE,nVL),2)[1]
>    =0,t,0..0.5/freq);

                                                   -6
                              T1 := .2823239921*10
> xT1:=f1(T1,xT0,EE,EE,nVL);

                                        [  27.63986854 ]
                                        [              ]
                                 xT1 := [ -27.63986854 ]
                                        [              ]
                                        [  193.9919675 ]
> T2:=(1/freq)*(180-phase)/360;

                                                   -6
                              T2 := .6502525253*10
> xT2:=f1(T2-T1,xT1,EE,EE,-nVL);
```

```
                                    [  52.65900880 ]
                                    [              ]
                           xT2 :=   [ -60.10239478 ]
                                    [              ]
                                    [  108.4017109 ]
> T3:=fsolve(row(f1(t-T2,xT2,0,EE,-nVL),3)[1]=0,t,T2..0.5/freq);

                                                  -5
                           T3 := .1007077400*10
> xT3:=f1(T3-T2,xT2,0,EE,-nVL);

                                    [ -12.28311840 ]
                                    [              ]
                                    [ -54.89952189 ]
                           xT3 :=   [              ]
                                    [           -6 ]
                                    [   -.10*10    ]
> phi3:=t->matrix([[cos(w*t),0,-sqrt(C/L)*sin(w*t)],[0,1,0],
>    [sqrt(L/C)*sin(w*t),0,cos(w*t)]]):
> psi3:=(t,e1,e2,vp)->matrix(3,1,[sqrt(C/L)*sin(w*t)*(e1-EE-vp),
>    (-e2+EE-vp)*t/L,(1-cos(w*t))*(e1-EE-vp)]):
> f3:=(t,xinit,e1,e2,vp)->add(multiply(phi3(t),xinit),psi3(t,e1,e2,vp)):
> f3(t,xT3,0,EE,-nVL):
> T4:=fsolve(row(f3(t-T3,xT3,0,EE,-nVL),1)[1]
>    =row(f3(t-T3,xT3,0,EE,-nVL),2)[1],t,T3..0.5/freq);

                                                  -5
                           T4 := .1184745130*10
> xT4:=f3(T4-T3,xT3,0,EE,-nVL);

                                    [ -42.81811628 ]
                                    [              ]
                           xT4 :=   [ -42.81811625 ]
                                    [              ]
                                    [ -24.80133736 ]
> w0:=1/sqrt(2*L*C):
> phi0:=t->matrix([[0,cos(w0*t),-sqrt(C/(2*L))*sin(w0*t)],[0,cos(w0*t),
>    -sqrt(C/(2*L))*sin(w0*t)],[0, sqrt(2*L/C)*sin(w0*t),cos(w0*t)]]):
> psi0:=(t,e1,e2,vp)->matrix(3,1,[sqrt(C/(2*L))*sin(w0*t)*(e1-e2-2*vp),
>    sqrt(C/(2*L))*sin(w0*t)*(e1-e2-2*vp), (1-cos(w0*t))*(e1-e2-2*vp)]):
> f0:=(t,xinit,e1,e2,vp)->add(multiply(phi0(t),xinit),psi0(t,e1,e2,vp)):
> T5:=.5/freq;

                                                  -5
                           T5 := .1818181818*10
> xT5:=f0(T5-T4,xT4,0,EE,-nVL);

                                    [ -46.83786937 ]
                                    [              ]
                           xT5 :=   [ -46.83786937 ]
                                    [              ]
                                    [ -179.9731167 ]
> phi2:=t->matrix([[1, 0, 0],[0, cos(w*t), -sqrt(C/L)*sin(w*t)],
>    [0, sqrt(L/C)*sin(w*t), cos(w*t)]]):
> psi2:=(t,e1,e2,vp) ->matrix(3,1,[(e1-vp)*t/L,
>    sqrt(C/L)*sin(w*t)*(-e2-vp), (1-cos(w*t))*(-e2-vp)]):
> f2:=(t,xinit,e1,e2,vp)->add(multiply(phi2(t),xinit),psi2(t,e1,e2,vp)):
> f2(t,xT5,0,0,-nVL):
> T6:=fsolve(row(f2(t-T5,xT5,0,0,-nVL),1)[1]+row(f2(t-T5,xT5,0,0,-nVL),2)[1]
>    =0,t,T5..(1/freq));

                                                  -5
                           T6 := .2100505629*10
> xT6:=f2(T1,xT5,0,0,-nVL);

                                    [ -27.63983791 ]
                                    [              ]
```

```
                              xT6 := [  27.63989768 ]
                                     [              ]
                                     [ -193.9919406 ]
> T7:=(1/freq)*(180-phase)/360+(.5/freq);

                                                   -5
                              T7 := .2468434343*10
> xT7:=f2(T7-T6,xT6,0,0,nVL);

                                     [ -52.65899046 ]
                                     [              ]
                              xT7 := [  60.10241309 ]
                                     [              ]
                                     [ -108.4015905 ]
-------------------------------------------------------------------------------
> ph1:=matrix([[1, 0, 0],[0, cos(w*(t1-t0)), -sqrt(C/L)*sin(w*(t1-t0))],
>    [0, sqrt(L/C)*sin(w*(t1-t0)), cos(w*(t1-t0))]]):
> ps1:=matrix(3,1,[(-nVL)*(t1-t0)/L, sqrt(C/L)*sin(w*(t1-t0))*(-nVL),
>    (1-cos(w*(t1-t0)))*(-nVL)]):
> x1:=add(multiply(ph1,[i10,i20,vc0]),ps1):
> ph2:=matrix([[1, 0, 0],[0, cos(w*(t2-t1)), -sqrt(C/L)*sin(w*(t2-t1))],
>    [0, sqrt(L/C)*sin(w*(t2-t1)), cos(w*(t2-t1))]]):
> ps2:=matrix(3,1,[(nVL)*(t2-t1)/L, sqrt(C/L)*sin(w*(t2-t1))*(nVL),
>    (1-cos(w*(t2-t1)))*(nVL)]):
> x2:=add(multiply(ph2,x1),ps2):
> ph3:=matrix([[1, 0, 0],[0, cos(w*(t3-t2)), -sqrt(C/L)*sin(w*(t3-t2))],
>    [0, sqrt(L/C)*sin(w*(t3-t2)), cos(w*(t3-t2))]]):
> ps3:=matrix(3,1,[(-EE+nVL)*(t3-t2)/L, sqrt(C/L)*sin(w*(t3-t2))*(nVL),
>    (1-cos(w*(t3-t2)))*(nVL)]):
> #x3:=add(multiply(ph3,x2),ps3):  # use this line for the 1st point
> x3:=add(multiply(ph3,[i10,i20,vc0]),ps3):  # use this line for the 2nd pt
> ph4:=matrix([[cos(w*(t4-t3)),0,-sqrt(C/L)*sin(w*(t4-t3))],[0,1,0],
>    [sqrt(L/C)*sin(w*(t4-t3)),0,cos(w*(t4-t3))]]):
> ps4:=matrix(3,1,[sqrt(C/L)*sin(w*(t4-t3))*(-EE+nVL),(nVL)*(t4-t3)/L,
>    (1-cos(w*(t4-t3)))*(-EE+nVL)]):
> x4:=add(multiply(ph4,x3),ps4):
> ph5:=matrix([[0,cos(w0*(t5-t4)),-sqrt(C/(2*L))*sin(w0*(t5-t4))],
>    [0,cos(w0*(t5-t4)),-sqrt(C/(2*L))*sin(w0*(t5-t4))],
>    [0, sqrt(2*L/C)*sin(w0*(t5-t4)),cos(w0*(t5-t4))]]):
> ps5:=matrix(3,1,[sqrt(C/(2*L))*sin(w0*(t5-t4))*(-EE+2*nVL),sqrt(C/(2*L))
>    *sin(w0*(t5-t4))*(-EE+2*nVL), (1-cos(w0*(t5-t4)))*(-EE+2*nVL)]):
> x5:=add(multiply(ph5,x4),ps5):
> ph6:=matrix([[1, 0, 0],[0, cos(w*(t6-t5)), -sqrt(C/L)*sin(w*(t6-t5))],
>    [0, sqrt(L/C)*sin(w*(t6-t5)), cos(w*(t6-t5))]]):
> ps6:=matrix(3,1,[(nVL)*(t6-t5)/L, sqrt(C/L)*sin(w*(t6-t5))*(nVL),
>    (1-cos(w*(t6-t5)))*(nVL)]):
> x6:=add(multiply(ph6,x5),ps6):
> ph7:=matrix([[1, 0, 0],[0, cos(w*(t7-t6)), -sqrt(C/L)*sin(w*(t7-t6))],
>    [0, sqrt(L/C)*sin(w*(t7-t6)), cos(w*(t7-t6))]]):
> ps7:=matrix(3,1,[(-nVL)*(t7-t6)/L, sqrt(C/L)*sin(w*(t7-t6))*(-nVL),
>    (1-cos(w*(t7-t6)))*(-nVL)]):
> x7:=add(multiply(ph7,x6),ps7):
>
> # Run the following for the 1st pt
> Jfx:=jacobian(col(x5,1),[i10,i20,vc0]):
> Jft:=jacobian(col(x5,1),[t0,t1,t2,t3,t4,t5]):
> Jfp:=jacobian(col(x5,1),[phas]):
> c:=[t0, x1[1,1]+x1[2,1], t2-(1/freq)*(180-phas)/360, x3[3,1],
>    x4[1,1]-x4[2,1], t5-.5/freq]:
> Jcx:=jacobian(c,[i10,i20,vc0]):
> Jct:=jacobian(c,[t0,t1,t2,t3,t4,t5]):
> Jcp:=jacobian(c,[phas]):
>
> Jfxeval:=evalf(subs(t0=T0,t1=T1,t2=T2,t3=T3,t4=T4,t5=T5,i10=xT0[1,1],
>    i20=xT0[2,1],vc0=xT0[3,1],phas=phase,evalm(Jfx))):
> Jfteval:=evalf(subs(t0=T0,t1=T1,t2=T2,t3=T3,t4=T4,t5=T5,i10=xT0[1,1],
>    i20=xT0[2,1],vc0=xT0[3,1],phas=phase,evalm(Jft))):
> Jfpeval:=evalf(subs(t0=T0,t1=T1,t2=T2,t3=T3,t4=T4,t5=T5,i10=xT0[1,1],
>    i20=xT0[2,1],vc0=xT0[3,1],phas=phase,evalm(Jfp))):
> Jcxeval:=evalf(subs(t0=T0,t1=T1,t2=T2,t3=T3,t4=T4,t5=T5,i10=xT0[1,1],
>    i20=xT0[2,1],vc0=xT0[3,1],phas=phase,evalm(Jcx))):
> Jcteval:=evalf(subs(t0=T0,t1=T1,t2=T2,t3=T3,t4=T4,t5=T5,i10=xT0[1,1],
>    i20=xT0[2,1],vc0=xT0[3,1],phas=phase,evalm(Jct))):
> Jcpeval:=evalf(subs(t0=T0,t1=T1,t2=T2,t3=T3,t4=T4,t5=T5,i10=xT0[1,1],
>    i20=xT0[2,1],vc0=xT0[3,1],phas=phase,evalm(Jcp))):
>
> Fhalf:=evalm(Jfxeval-Jfteval&*inverse(Jcteval)&*Jcxeval);

                       [ .0763320988  -.2362395912  -.06784070852 ]
                       [                                          ]
              Fhalf := [ .0763320988  -.2362395912  -.06784070852 ]
                       [                                          ]
                       [ .9685111164  -1.320938402   -.3086748679 ]
> Ghalf:=evalm(Jfpeval-Jfteval&*inverse(Jcteval)&*Jcpeval);

                                    [ -.04561961303 ]
                                    [               ]
                           Ghalf := [ -.04561961303 ]
                                    [               ]
                                    [  -4.278781491 ]
> Fone:=multiply(Fhalf,Fhalf);

                      [ -.07791055486  .1273898777  .03178895933 ]
                      [                                          ]
              Fone := [ -.07791055486  .1273898777  .03178895933 ]
                      [                                          ]
                      [  -.3258565553  .4909977646   .1191890908 ]
> Gone:=add(multiply(Fhalf,Ghalf),-Ghalf);

                                    [ .3431900989 ]
                                    [             ]
                            Gone := [ .3431900989 ]
                                    [             ]
                                    [ 5.615611399 ]
> eigenvals(Fone);

                                                            -10
                 .1647365088, .003931904912, .819045*10
> dx0:=matrix(3,1,[1,1,0]):
> dxhalf:=multiply(Fhalf,dx0);
> dxone:=multiply(Fone,dx0);
-------------------------------------------------------------------------------
> # Run the following for the 2nd pt
> Jfx:=jacobian(col(x7,1),[i10,i20,vc0]):
> Jft:=jacobian(col(x7,1),[t2,t3,t4,t5,t6,t7]):
> Jfp:=jacobian(col(x7,1),[phas]):
> c:=[t2-(1/freq)*(180-phas)/360, x3[3,1], x4[1,1]-x4[2,1],
>    t5-.5/freq, x6[1,1]+x6[2,1], t7-((1/freq)*(180-phas)/360+.5/freq)]:
> Jcx:=jacobian(c,[i10,i20,vc0]):
> Jct:=jacobian(c,[t2,t3,t4,t5,t6,t7]):
> Jcp:=jacobian(c,[phas]):
>
> Jfxeval:=evalf(subs(t2=T2,t3=T3,t4=T4,t5=T5,t6=T6,t7=T7,i10=xT2[1,1],
>    i20=xT2[2,1],vc0=xT2[3,1],phas=phase,evalm(Jfx))):
> Jfteval:=evalf(subs(t2=T2,t3=T3,t4=T4,t5=T5,t6=T6,t7=T7,i10=xT2[1,1],
>    i20=xT2[2,1],vc0=xT2[3,1],phas=phase,evalm(Jft))):
> Jfpeval:=evalf(subs(t2=T2,t3=T3,t4=T4,t5=T5,t6=T6,t7=T7,i10=xT2[1,1],
>    i20=xT2[2,1],vc0=xT2[3,1],phas=phase,evalm(Jfp))):
> Jcxeval:=evalf(subs(t2=T2,t3=T3,t4=T4,t5=T5,t6=T6,t7=T7,i10=xT2[1,1],
>    i20=xT2[2,1],vc0=xT2[3,1],phas=phase,evalm(Jcx))):
> Jcteval:=evalf(subs(t2=T2,t3=T3,t4=T4,t5=T5,t6=T6,t7=T7,i10=xT2[1,1],
>    i20=xT2[2,1],vc0=xT2[3,1],phas=phase,evalm(Jct))):
> Jcpeval:=evalf(subs(t2=T2,t3=T3,t4=T4,t5=T5,t6=T6,t7=T7,i10=xT2[1,1],
>    i20=xT2[2,1],vc0=xT2[3,1],phas=phase,evalm(Jcp))):
>
> Fhalf:=evalm(Jfxeval-Jfteval&*inverse(Jcteval)&*Jcxeval);
```

```
                    [  .1889237341    .1328452276  -.0755168445 ]
                    [                                           ]
           Fhalf := [ -.6342670110   -.4121407126   .1938912609 ]
                    [                                           ]
                    [  .5175962941    .3857968536  -.2453654538 ]
> Ghalf:=evalm(Jfpeval-Jfteval&*inverse(Jcteval)&*Jcpeval);

                                    [  .5145699196 ]
                                    [              ]
                           Ghalf := [  .3375702549 ]
                                    [              ]
                                    [ -3.399657235 ]
> Fone:=multiply(Fhalf,Fhalf);

                  [ -.08765440699  -.05878747131   .03001982925 ]
                  [                                             ]
          Fone := [  .2419365639    .1604032600   -.07958685641 ]
                  [                                             ]
                  [  -.2739122422  -.1849036128    .09591960546 ]
> Gone:=add(multiply(Fhalf,Ghalf),-Ghalf);

                                   [ -.1157794648 ]
                                   [              ]
                           Gone := [ -1.462235253 ]
                                   [              ]
                                   [  4.630388701 ]
> eigenvals(Fone);

                                                          -9
                  .1647365509, .003931907972, -.316*10
>
```

B.2 Simulation Results from *simsyq.m*

```
Simulation from E=>0: States at transition points at time
   1.0e-05 *

    0.0282
    0.0650
    0.1007
    0.1185
    0.1818
    0.2100
    0.2468
    0.2825
    0.3003
    0.3636
    0.3919

from initial state
   46.8208   46.8208  180.0880

are
   27.6354  -27.6354  194.0877
   52.6356  -60.1382  108.4185
  -12.2764  -54.9412         0
  -42.8453  -42.8453  -24.8411
  -46.8321  -46.8321 -180.0560
  -27.6407   27.6407 -194.0647
  -52.6122   60.1308 -108.4204
   12.3085   54.9333         0
   42.8487   42.8487   24.8332
   46.8208   46.8208  180.0886
   27.6354  -27.6354  194.0882
```

```
Transition matrix and eigenvalues at E=>0:
    0.0500    0.0319
    0.1660    0.1195

    0.0041
    0.1654

Transition matrix and eigenvalues at 0=>-E:
   -0.0872   -0.0577    0.0299
    0.2427    0.1592   -0.0793
   -0.2715   -0.1804    0.0954

    0.1626    0.0000    0.0048
```

Appendix C

Representative Trajectories in Various Operating Modes

For more discussion on steady-state operating modes, see Chapter 6.

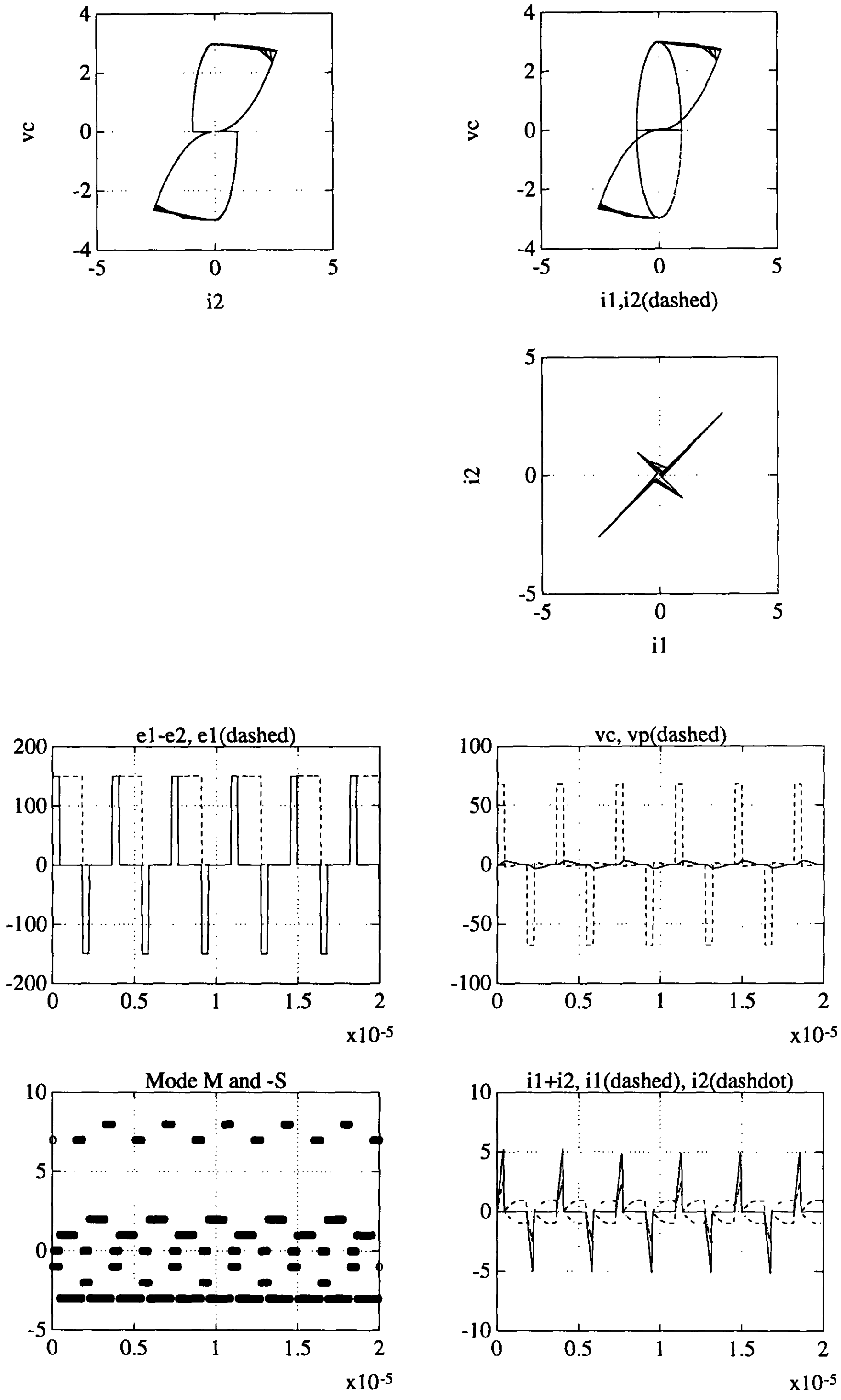


Figure C-1: Trajectory in operating mode 0 from zero initial state at $\phi = 40°$, $E = 150\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

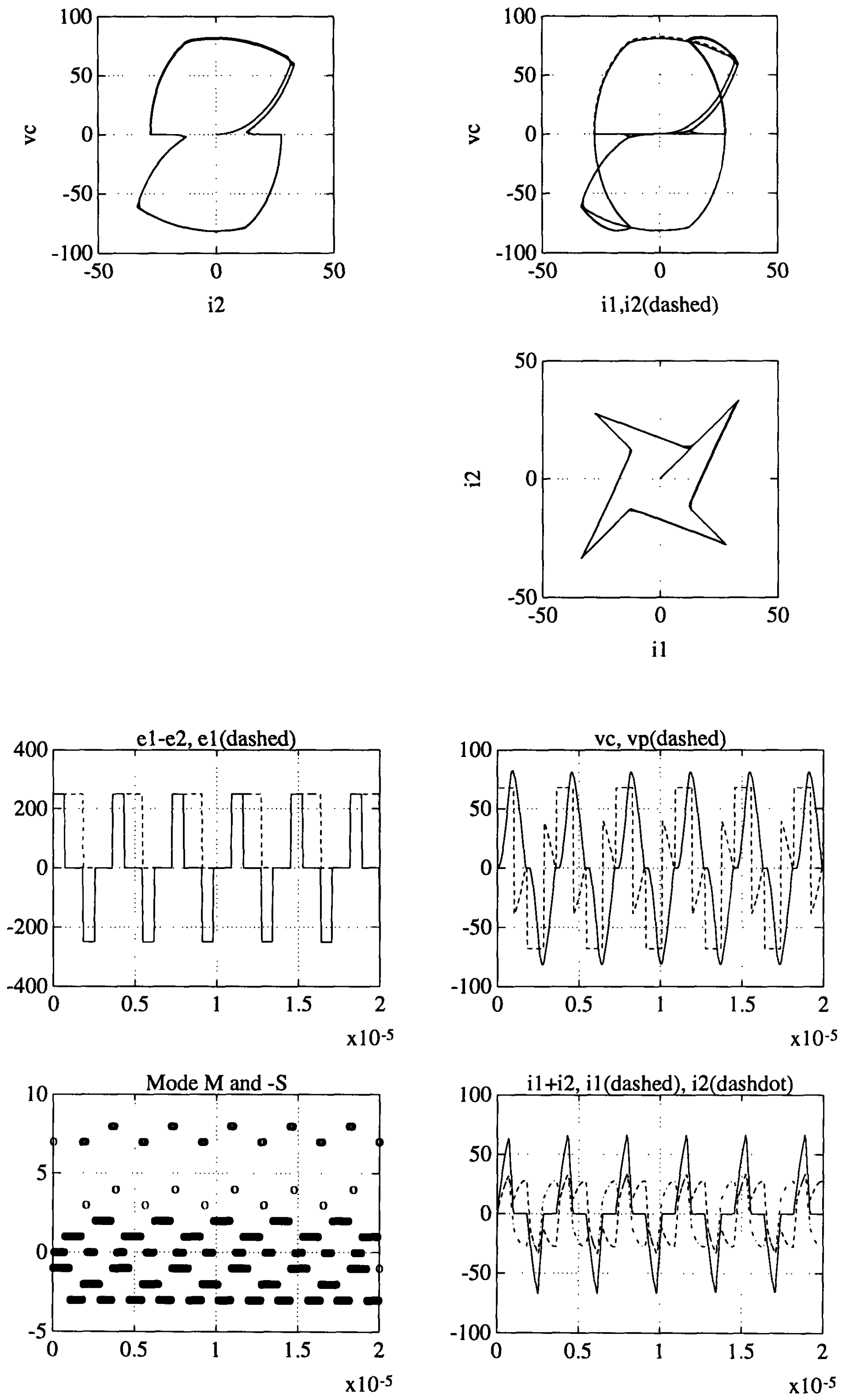


Figure C-2: Trajectory in operating mode 1 from zero initial state at $\phi = 70°$, $E = 250\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

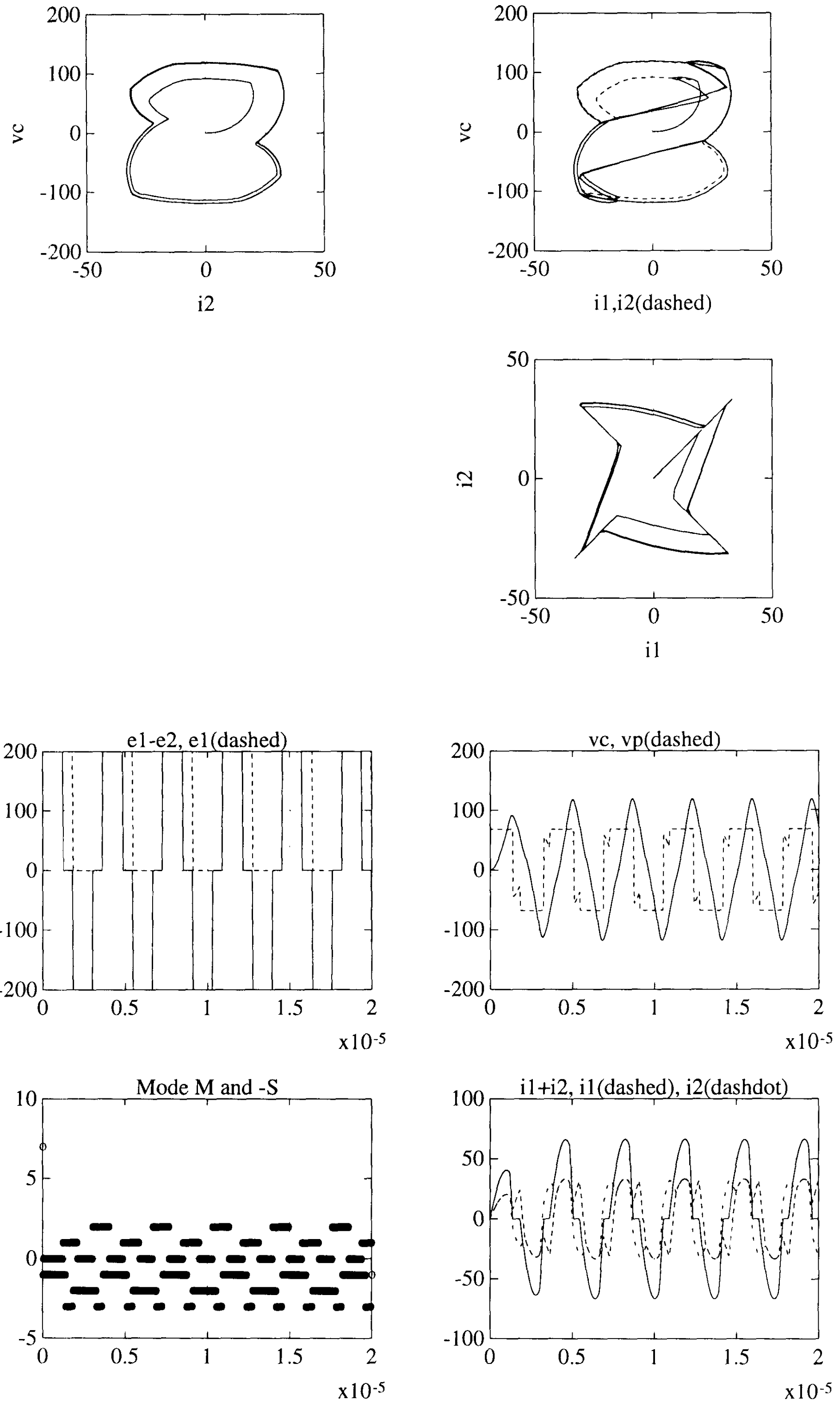


Figure C-3: Trajectory in operating mode 2 from zero initial state at $\phi = 120°$, $E = 200\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

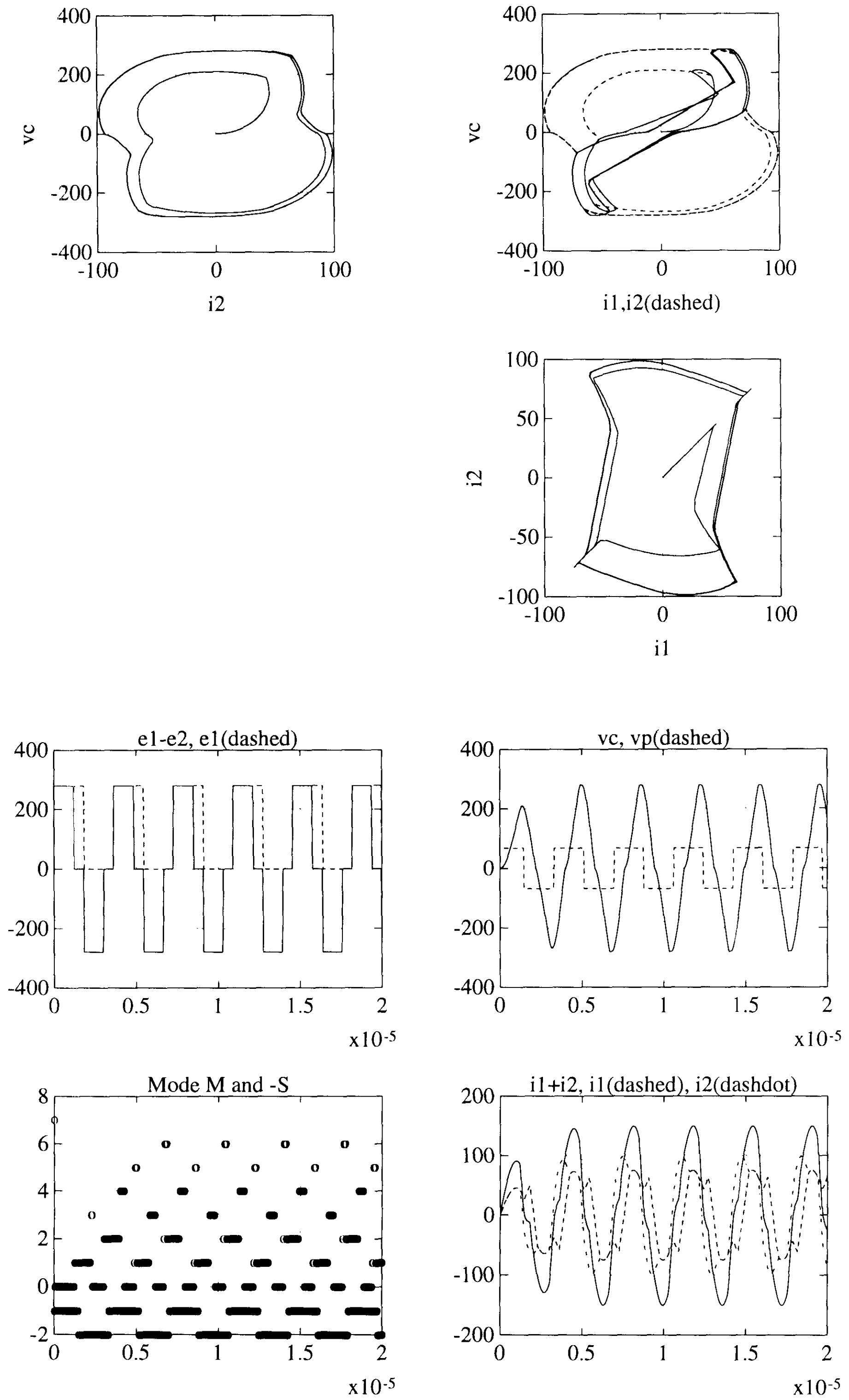


Figure C-4: Trajectory in operating mode 4 from zero initial state at $\phi = 120°$, $E = 280\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

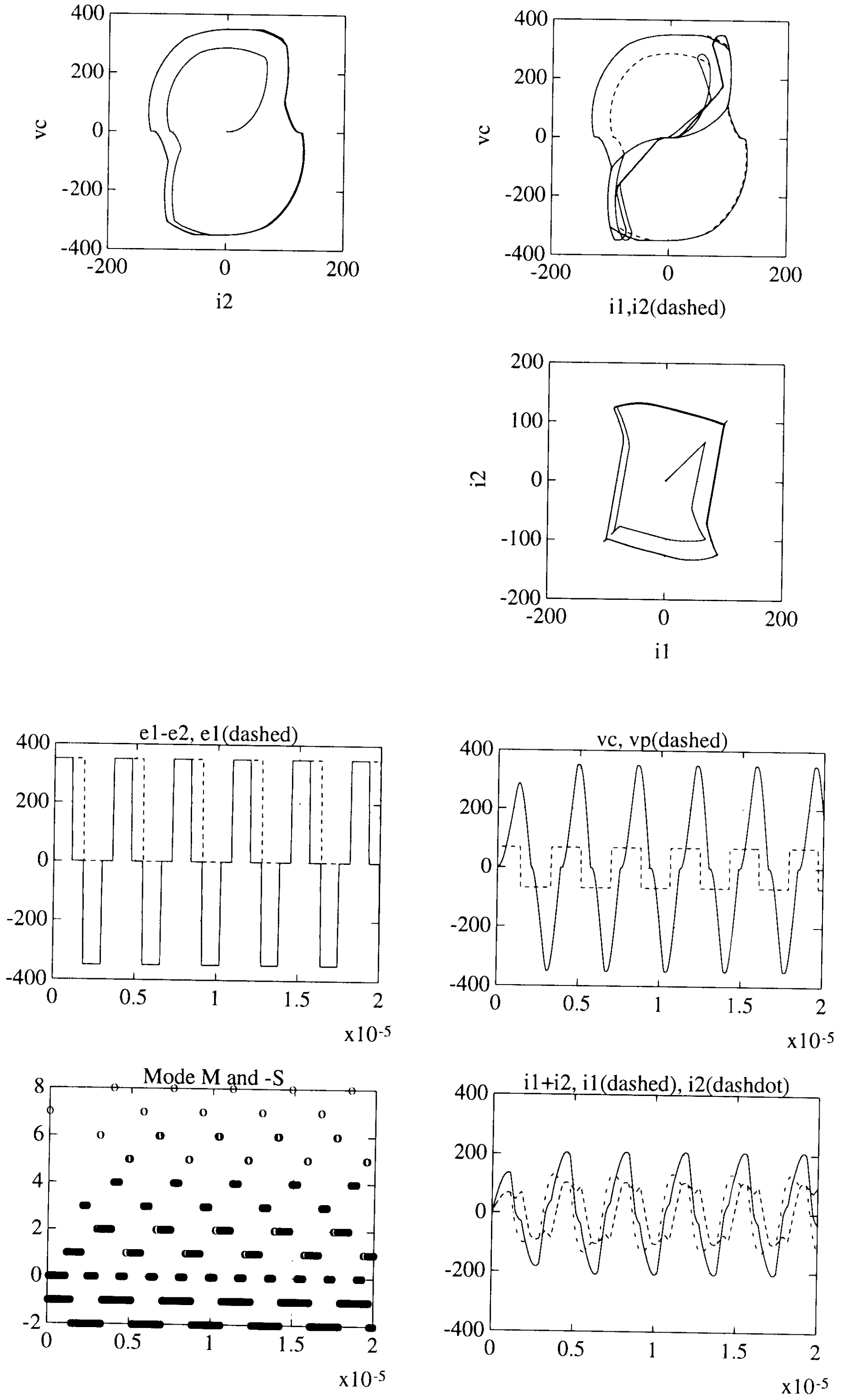


Figure C-5: Trajectory in operating mode 5 from zero initial state at $\phi = 110°$, $E = 350\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

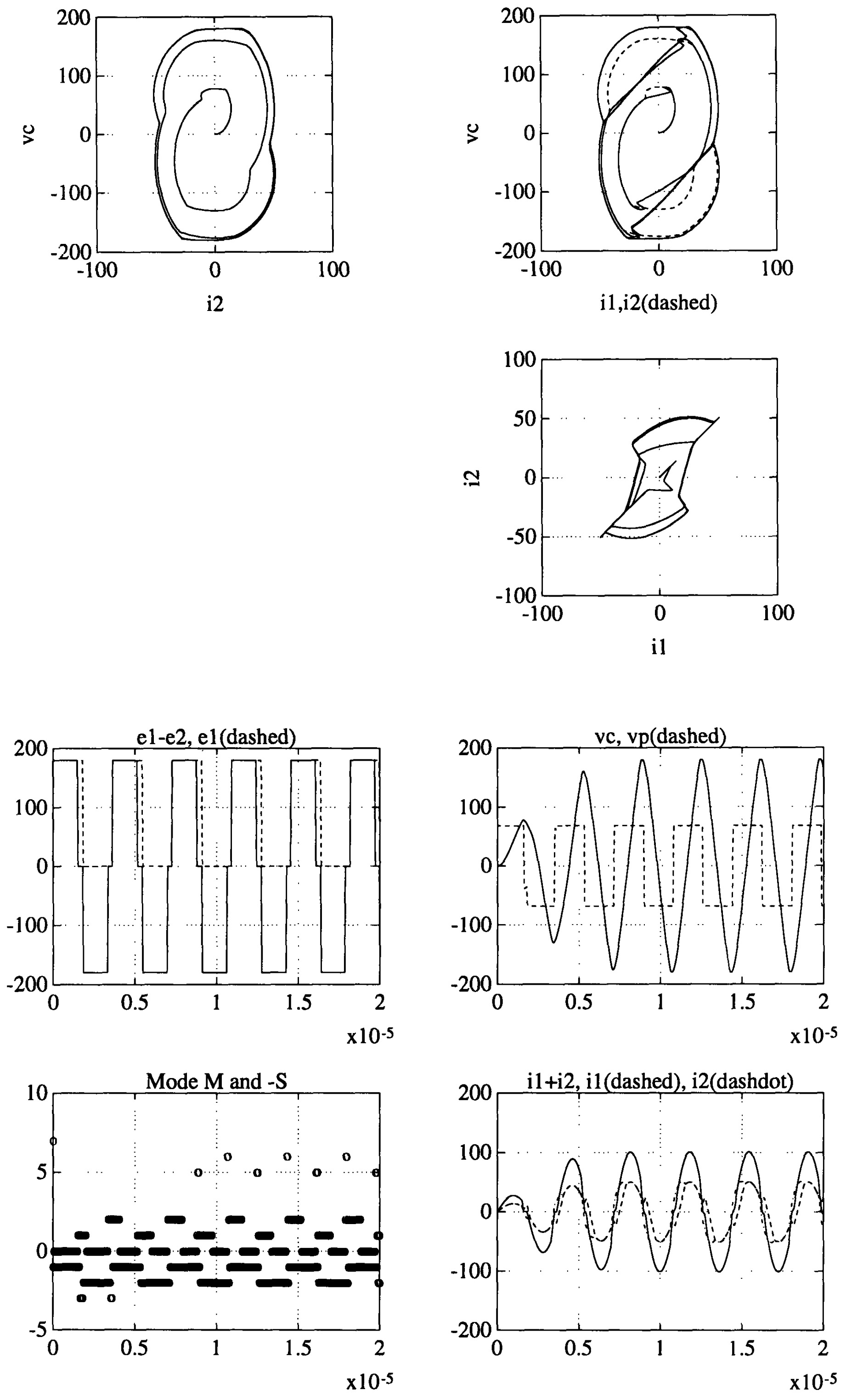


Figure C-6: Trajectory in operating mode 6 from zero initial state at $\phi = 150°$, $E = 180\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

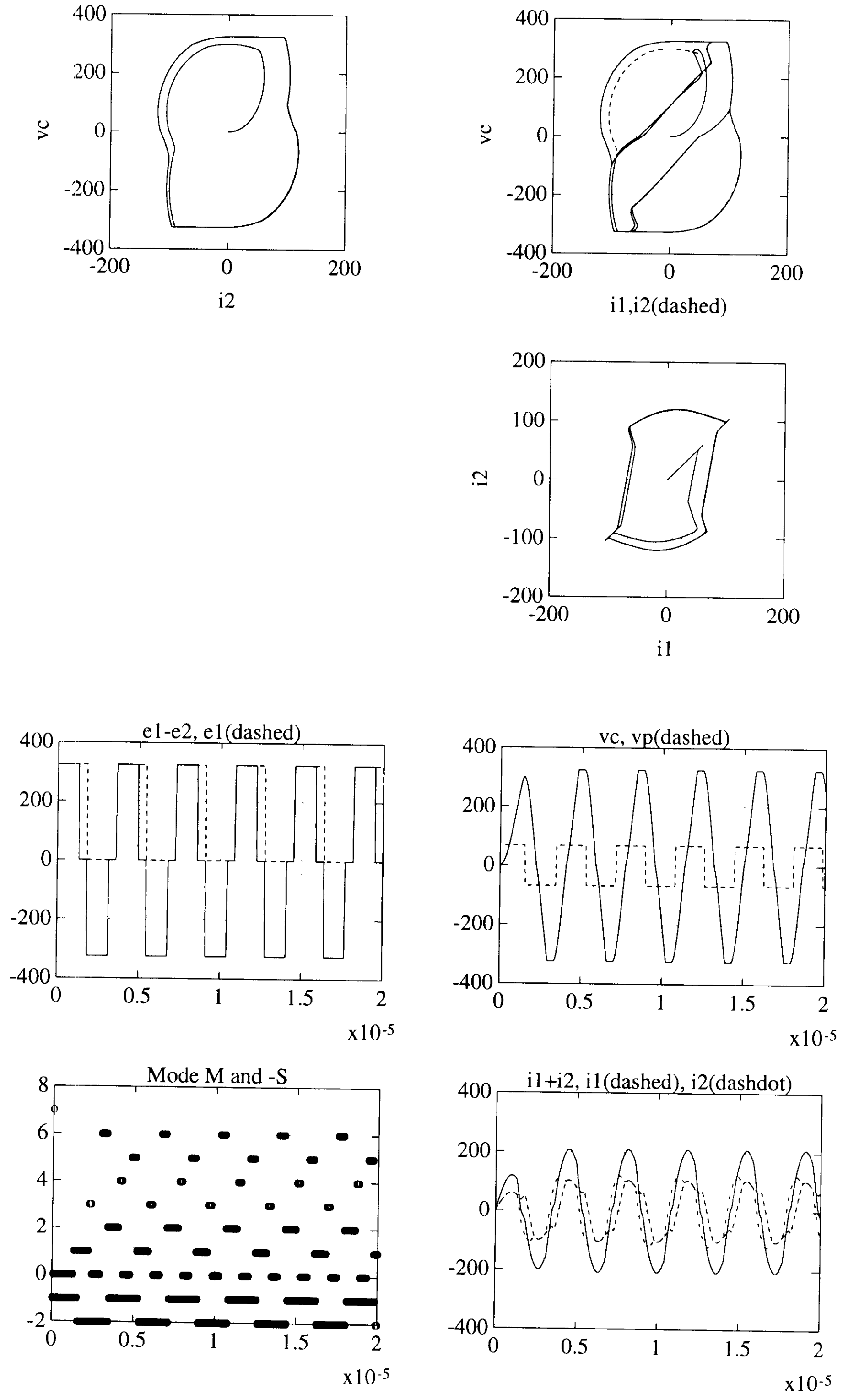


Figure C-7: Trajectory in operating mode 7 from zero initial state at $\phi = 130^\circ$, $E = 325\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

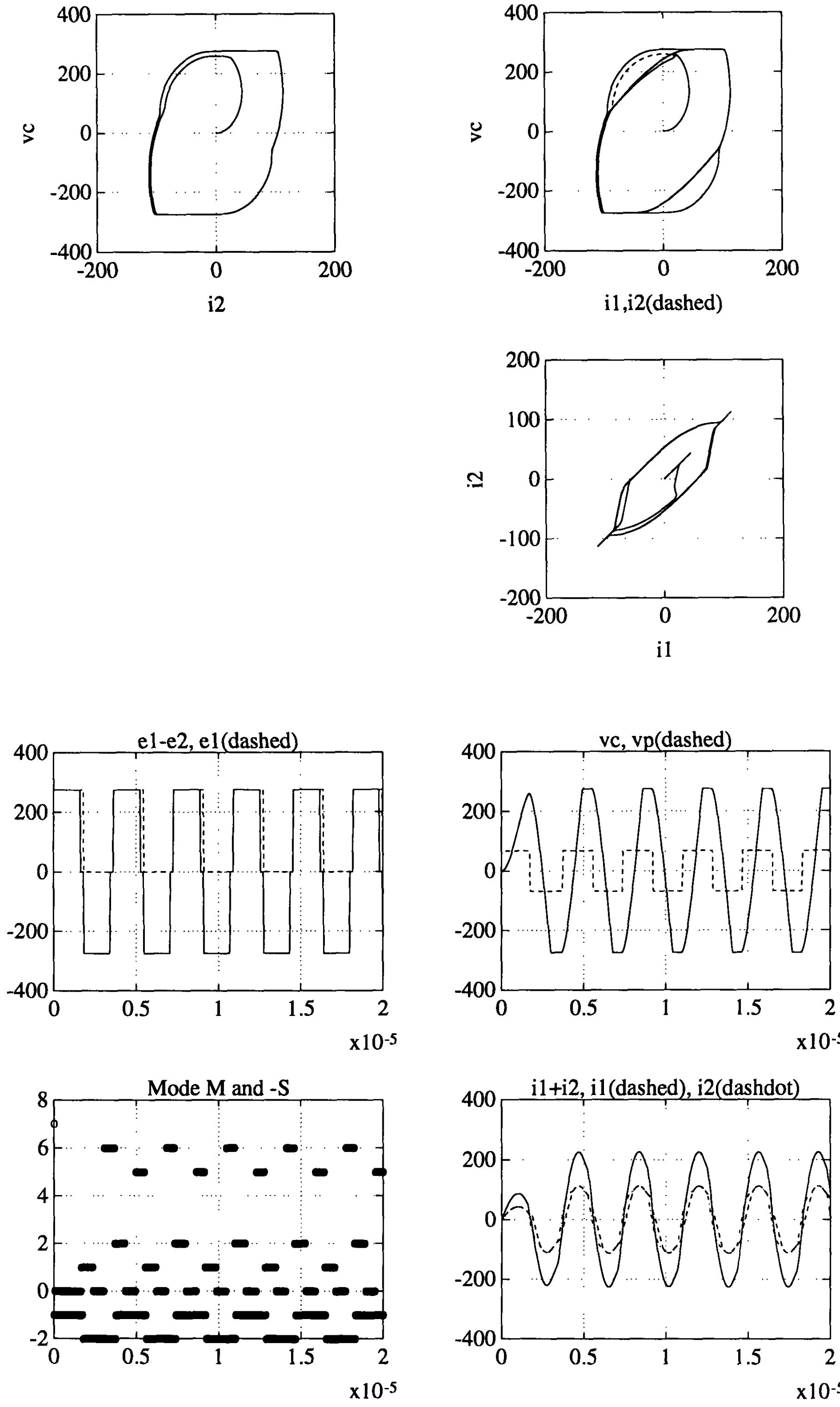


Figure C-8: Trajectory in operating mode 8 from zero initial state at $\phi = 160°$, $E = 275\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

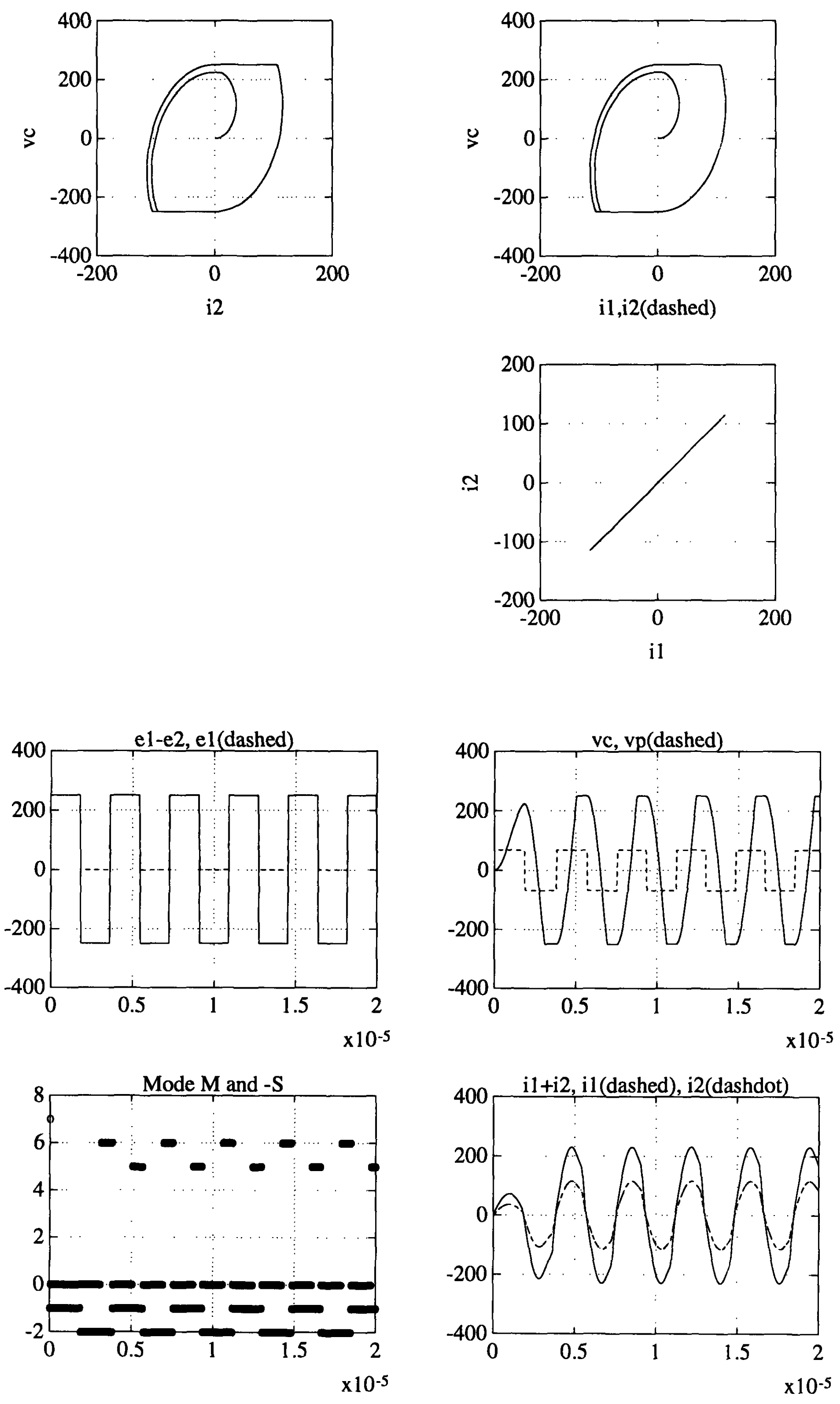


Figure C-9: Trajectory in operating mode 9 from zero initial state at $\phi = 180°$, $E = 250\text{V}$, and $nV_L = 8 \cdot 8.5\text{V}$.

Bibliography

[1] M. E. Elbuluk, "Resonant Converters: Topologies, Dynamic Modeling and Control," Ph.D. Dissertation, EECS Dept., MIT, 1986.

[2] B. S. Jacobson and R. A. DiPerna, "Series Resonant Converter with Clamped Tank Capacitor Voltage," *IEEE APEC*, pp. 137-146, 1990.

[3] J. G. Kassakian, M. F. Schlecht, and G. C. Verghese, *Principles of Power Electronics*, Addison-Wesley, 1991.

[4] T. Kato and G. C. Verghese, "Efficient Numerical Determination of Boundaries Between Operating Modes of a Power Converter," *Third IEEE Workshop on Computers in Power Electronics*, Berkeley, August 1992.

[5] T. Kato and G. C. Verghese, "Operation Analysis of Series Resonant dc/dc Converter with Clamped Tank Capacitor Voltage," internal LEES document.

[6] R. Oruganti, J. Yang, and F. Lee, "Implementation of Optimal Tajectory Control of Series Resonant Converter," *IEEE Transactions on Power Electronics*, Vol. 3, No. 3, pp. 318-327, July 1988.

[7] C. Osawa, "Dynamic Modeling and Control of Switched and Resonant DC-DC Converters," SM Thesis, EECS Dept., MIT, 1993.

[8] S. R. Sanders, J. M. Noworolski, X. Z. Liu, and G. C. Verghese, "Generalized Averaging Method for Power Conversion Circuits," *IEEE Transactions on Power Electronics*, Vol. 6, No. 2, pp. 251-259, April 1991.

[9] R. L. Steigerwald, "A Comparison of Half-Bridge Resonant Converter Topologies," *IEEE Transactions on Power Electronics*, Vol. 3, No. 2, pp. 174-182, April 1988.

[10] F. Tsai, P. Materu, and F. Lee, "Constant-Frequency Clamped-Mode Resonant Converters," *IEEE Transactions on Power Electronics*, Vol. 3, No. 4, pp. 460-473, October 1988.

[11] G. C. Verghese, M. E. Elbuluk, 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, April 1986.