University of Alberta

Switch Mode Power Supply (SMPS) Circuit Simulation by Generalized Transmission Line Modeling (TLM) Method

by

Ying Chen ©

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta
Fall 2006

Your file Votre référence
ISBN: 978-0-494-22242-3
Our file Notre référence
ISBN: 978-0-494-22242-3

Abstract

The advantages of SMPS make the simulation of it very important. There are three main challenges in the SMPS circuit simulation. The first one is how to continue the simulation after forced commutations. The second one is how to depress the numerical oscillation. The last one is how to minimize the size of the system matrices.

In this thesis, a series of new techniques are proposed. Two-branch switch models and modified Femia's technique can reinitialize the simulation. At the same time, the risk of island circuits due to the off-states of switches is vanished. Generalized TLM stub models depress the numerical oscillation effectively. The TLM stub model decoupling technique minimizes the size of the system matrices and the computation burden.

At the end, the simulation results of new proposed algorithm, PSCAD/EMTDC and MULTISIM are compared. The curves are perfectly matched to each other. Thus, the new proposed algorithm works effectively.

Acknowledgements

I would like to extend my deepest gratitude and appreciation to Drs. Venkata Dinavahi and Don Koval for their invaluable guidance, support and encouragement throughout my graduate years.

I am also grateful to my parents, my wife and my son for their patience, love and support throughtout my studies

Ying Chen

Edmonton, Alberta

May 2006

Dedication

This thesis is dedicated to my wife, to my son. and to my parents

Table of Contents

List of Tables

List of Figures

List of Symbols

TL:	Transmission line
TLM:	TL modeling method.
Z_C :	The Characteristic impedance of TL.
τ_S :	The round trip time of a section of TL.
τ_L :	The single trip time of a section of TL
u:	The velocity of traveling wave in TL.
inf:	An infinite number
v^+ :	Positive direction voltage wave
v^- :	Negative direction voltage wave
v^r :	Reflected voltage wave
v^i :	Incident voltage wave
Δt :	Time step.
Δt_L :	Time step of left subnetwork.
Δt_R :	Time step of right subnetwork. Δ tL=n Δ tR.
N_{SW} :	A whole circuit.
N_{RLC} :	A circuit without branch one of the switch model.
μs:	microsecond
ms:	millisecond

Chapter 1 Introduction

1.1 Basic Concepts of Power Supply

A power supply is a buffer circuit between the input power source and the load, which converts an incompatible input power source to the compatible output power source. In order to let the specified circuit work properly and stable, characteristics of the power supply should be compatible with the characteristics of the load. For example, the power source in a residential house in Canada is 60Hz, single phase, 120V / 240 V ac power and the load might be logic circuits in a PC that require regulated +3.3V, +/- 5V and +/- 12V. The circuit that can make the 120V ac power source and logic circuits compatible is called the power supply. A power supply can also be called as power converter or power conditioner. The Power Sources Manufacturers Association (PSMA) gives a formal definition of a power supply:

> Power Supply – A device for the conversion of available power of one set of characteristics to another set of characteristics to meet specified requirements. Typical applications of power supplies include to convert input power to a controlled or stabilized voltage and /or current for the operation of electronic equipment. [1]

To design power supplies is to choose the proper topology and control scheme, which is normally realized by power electronics. The IEEE Power Electronics Society provides a formal definition of power electronics in their constitution.

> Power Electronics – This technology encompasses the effective use of electronic components, the application of circuit theory and design techniques, and the development of analytical tools toward efficient electronic conversion, control, and conditioning of electric power.

According to the control loop, power supplies can be classified as two categories, open loop and close loop control. The open loop control cannot adjust the output characteristics of the power supply to desired requirement if the input does not comply with the ideal design conditions. But the topology of that kind of the power supply system is the simplest one. The close loop control power supply is a power supply that has a feed back control loop. The feed back circuit can give out the different feedback signals according to the different output signals, and the comparison signals between feedback signals and the reference signals can adjust the circuit to let the output reach the desired output characteristics. The block diagram of this type of power supply is showed as Fig. 1.1.

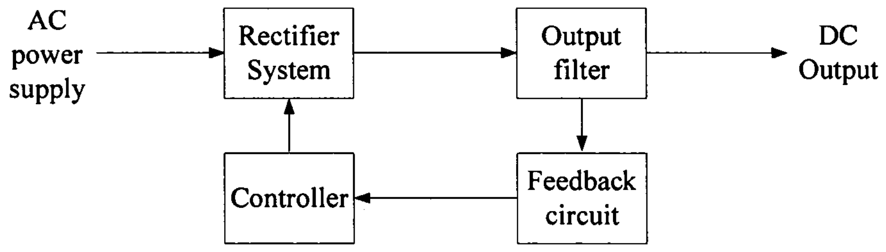


Fig. 1.1 The block diagram of close loop control power supply

The controller is also called regulator, and its function is to guarantee the output characteristics of the power supply constant under varying input conditions. The whole power supply regulation technology can be divided into two distinct forms, linear electronics regulator and the power electronics regulator (switching mode regulator). These technologies have developed with the development of the semiconductor since early 1950's, and especially with the development of integrated circuits form the early 1960's. During 1950's and the early 1960's, the linear regulated power supply was very common because of the availability of the power electronic components. Since late 1960's, the switching mode conversion technique has become more and more popular because of the development of the fast switching power transistors. Because the loads are becoming more and more demanding, power supplies are also becoming more and more complicated and efficient to meet the increments of demand. In the next two sections, I will discuss these two different power supplies.

1.2 Linear Regulation Power Supply

The core circuit that controls the output voltage is called the regulator. It attempts to make the power supply to produce a desired output voltage from a varying input voltage. The linear regulator is so named because it contains a transistor operating in its linear region, with the transistor acting like a variable resistor. Linear regulators have two basic forms: series regulators and shunt regulators. Here are the equivalent circuits for linear regulators in Fig. 1.2. The rectangle in which resistor, capacitor, inductor and source are included indicates a circuit composed by these components.

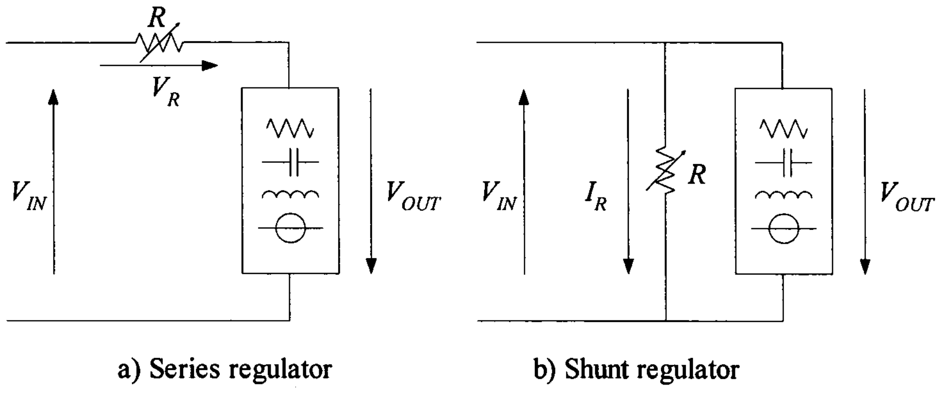


Fig. 1.2 Equivalent circuits for linear regulators

The linear regulator achieves a regulated output voltage independent of input and load variation by controlling the energy dissipation on a variable resistant (*R*). In a linear regulator, the transistor is used as a voltage divider to control the output voltage, and a feedback circuit compares the output voltage to a reference voltage in order to adjust the input to the transistor, thus keeping the characteristics of output voltage reasonably constant. In shunt regulator, the transistor or some other semiconductor components is used as a current divider to control the output voltage, and flow the useless current to ground.

As aforementioned description, the linear regulated power supply must be low efficiency, because the variable resistor must consume a large amount of power during its operation, especially under a low load condition. That is the main disadvantage and also the cause of other shortcomings of the linear regulated power supply. The volume is the important parameter for the power supply. It is determined by the thermal density – the power dissipated per unit volume. Due to the high-energy dissipation, the power supply has to be adequately cooled by mounting heatsink on regulator and giving enough space for air circulation. The heatsink and provision for cooling system makes the power supply bulky and large. In applications where volume and efficiency are emphasized, linear regulated power supply cannot be used. The advantage of the linear regulator is that its structure and control scheme is simple. In application where consumption of the circuit is very low and the power loss can be ignore, that kind of power supply can be used. At the same time, the linear regulator do not have high frequency changed components, so it does not produce electromagnetic interference, which is a big problem for another type of regulator – switching mode regulator. In the next section, I will introduce the theory of the switching mode power supply (SMPS).

1.3 Switching Mode Power Supply

A switching mode power supply is a power supply that provides the power supply function through low loss components such as capacitors, inductors, transformers and switches that are in two states, on or off. The SMPSs are so named because they contain some IGBTs or MOSFETs operating in on/off state like switches, not like in linear regulator operating in linear region. When the switches are on, the turn-on resistance is very small; when the switches are off, the turn-off resistance is very large (at least $10^6\,\Omega$). No matter under what conditions, the power losses will be very low. Plus using those low power loss components, a switching mode regulator overcomes the drawbacks of linear regulators. It becomes more efficient and tends to have efficiency of 80% or up. The size of a switching mode regulator is also reduced due to the improving efficiency.

1.3.1 The Basic Concepts and Usages of the SMPSs

The SMPS is not only with high efficiency, but also with high flexibility. It can step down and/or step up the input voltage. According to the input and output conditions, the switching mode power supplies may be designs to:

(1) step down/up an unregulated dc input voltage to produce a regulated dc output voltage using a circuit knows as Buck/Boost Converter,

(2) step up or step down an unregulated dc input voltage to produce a regulated dc output voltage using a circuit knows as Buck-Boost Converter,

(3) invert the input dc voltage using usually a circuit such as the Cuk converter, and

(4) produce multiple dc outputs using a circuit such as the fly-back converter.

Combining with the rectifier circuit, the SMPSs also can be used to make AC-DC,AC-AC conversion. Due to these four conversion capabilities, the SMPS's technology is mainly used in three areas.

Static Var compensator – There are two traditional Var compensation methods, one is using inductor or capacitor banks to consume or generate reactive power, which compensates the reactive power slowly and imprecisely; the other is using reactive power generator, which is too expensive and complicated. The GTO-controlled static Var compensator, which uses SMPS's technology, can change the effective reactance connected to the power system by controlling the switching action of the GTO. This device can compensate the reactive power precisely and respond to the changes quickly.

PWM drive – The most efficient way to control the ac motor speed is to control the fundamental frequency of the input voltage or current. The PWM drive first uses rectifier and proper dc link to get dc voltage, and then uses the PWM technology to form a voltage or current source with desired frequency. For a dc motor, the most efficient way to control the speed is to control the value of input dc voltage. Through changing the duty cycle of the pulse, the output dc voltage will change accordingly.

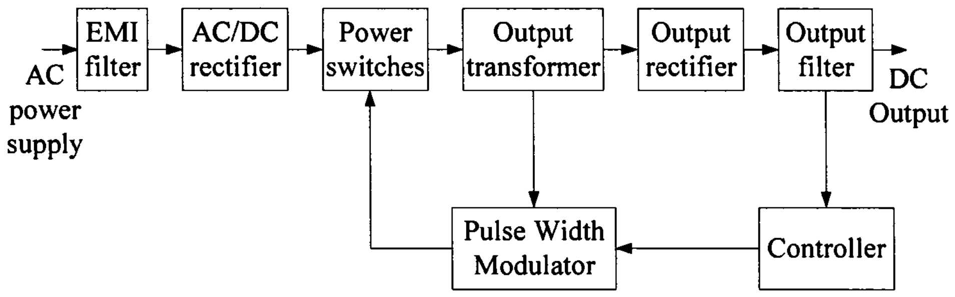


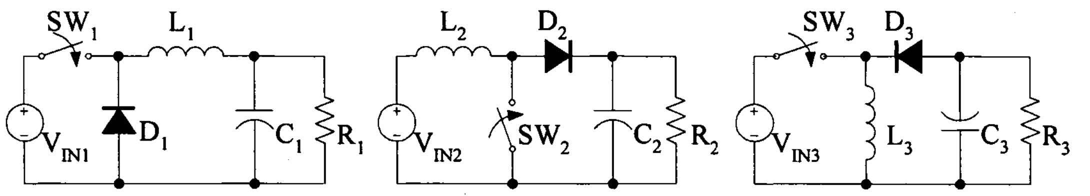
Fig. 1.3 The block diagram of SMPS

Power supply of the electronic circuit – In this area, SMPS converts the ac power to dc voltage source, normally from high ac voltage to a low dc voltage. There are three steps in the basic procedure of conversion. First is to obtain high rectified dc voltage; second is using switching actions to get high frequency (alterative) current; last is using isolation transformer to step down/up the voltage and using the secondary rectifier to obtain the final output voltage. The switching signal is generated by the feedback loop. The block diagram of that SMPS is shown in Fig. 1.3. The function of the EMI filter is to filter the common/differential mode noise and high frequency disturbance from the line and also can block the high frequency disordered component signal generated inside the power supply system. The primary rectifier is used to do the AC-DC transform and generate smooth DC voltage with ripple. The energy storage and waveform filter component is normally a capacitor. The power switches are used to generate the high frequency alterative current, which is controlled by PWM. The output transformer is used to isolate the output circuit form the input power source. Normally, it is a step down transformer. The secondary rectifier and output filter is used to generate required voltage with very small ripple.

The controller and PWM parts are used to generate PWM signals to control the switches.

1.3.2 Topologies of the SMPSs

In this subsection, three basic SMPS topologies – buck converter, boost converter, and buck-boost converter will be briefly discussed. Unlike the classical dynamical system, SMPSs never can reach the steady state due to the constant changes in system structure and parameters. It utilizes the transient response characteristics of the reactive components to reach the cyclic mode[7]. The SMPSs can be controlled in two ways – one is constant-frequency operation or PWM control, and the switching-on time is changed with the changing of duty cycle; the other is variable-frequency operation or control by frequency modulation, in which the switching-on time is constant and duty cycle is a variable. Choosing the PWM control, it is easy to design LC filter and control the ripple content in output voltage. But if the load is below a certain level, the ideal switching-on time will be too small to realize by the practical component. Under this circumstance, the voltage will be easier to control by using variable-frequency control. The follow figures are the basic topologies of the SMPSs.



a) Buck converter   b) Boost converter   c) Buck-boost converter

Fig. 1.4 The three basic topologies of SMPSs

Both converters can operate in three states. When the switch is on, the inductor is energized; when the switch is off and D_1 is on, the inductor is discharged and the capacitor may be charged or discharged according to the load current; when the inductor current falls to zero, the circuit enters the third state, only the capacitor is discharged at that time. The output voltage of buck converter is less than the input voltage. The output voltage of the boost converter is larger than the input voltage. The output voltage of buck-boost converter can be either larger or less than the input voltage, according to the value of the duty cycle.

1.4 Literature Review

In the literature review section, there are two different parts. The first one is the simulation method proposed by the former researcher and the corresponding commercial softwares; the second part is the control schemes used in the SMPS.

1.4.1 Simulation Method

During the design procedure of power electronics equipment, one must test the scheme again and again. To do the simulation one can choose the hardware simulation by setting up the real circuit, or choose the software simulation by setting up the artificial circuit in computer. Nowadays more and more designers use software simulation for the first step scheme test because of its low cost and quick speed. There are different kinds of simulation methods. From the modeling point of view, there are two simulation modes: detailed mode and behavior mode[4]. From the theory base, there are also two approaches: one is state variable approach; the other is nodal analysis approach.

In detailed-mode simulation, detailed device models are used. It allows us the do an accurate analysis of switching characteristics and power losses. The more detailed, the more accurate the results are. The drawback of this method is that we need to obtain the exact model of the device being used, including the equivalent value of stray/parasitic components. So the detailed-mode simulation still cannot replace final hardware test. In behavior-mode simulation, we use ideal or simplified device models to do the system level studies. Due to the ignorance of the detailed characteristics, the simulation cannot give us the detailed response of the system, but still can give out the approximate results quickly. According to the simplified level, this simulation method can classify into two basic types, one is to use ideal device models; the other is to use average models. When we use ideal devise models, we will ignore all stray/parasitic components, linearize all non-linear components as much as possible, and turn any analogue switch into an ideal switch, etc.. If the on-state resistance is zero and off-state resistance is infinity, one problem for this method is that if there is any switch in the circuit, the topology of the circuit will change with the change of switch states. If the on-state and off-state resistance are all finitary number, the system matrix will be changed when the switches state is changed. In order to overcome this drawback, some people invent the switching function for the behavior-mode simulation. In this approach, the equivalent device for a switch is a controlled voltage or current source. The control function determines the state of the switch. By choosing this approach, the topology of the circuit will not change with the change of the switch states. The only change is the instant value of the control function. In average-mode approach, the high frequency switches are modeled as the average function, which is determined by the duty cycle of switches, and only the lower frequency switches are left. Therefore, the simulation speed is very quick, but the high frequency switches information is lost.

The state variable approach is based on state space theory and generated for general circuit analysis. The first step of this method is to set up state equations, and then choose appropriate numerical method to calculate them. Normally this approach chooses voltages across capacitors and currents through inductors as the state variables, and forms the minimum number of equations using graph theory. It can easily track the dynamic behavior of a circuit by observing the trajectory of state variables, and justify its stability by calculating matrices of state equations. The drawback of this approach is that the setup procedure is complicated. The software CIRCUIT, SCRIPT, ATOSEC, and other similar simulators use this approach as a base.

The nodal analysis is based on KCL. Normally it discretizes the basic electrical elements, and all electrical components are decomposed as the combination of the resistors, sources, and switches. Therefore, the system equations are very easy to set up, but it also has some limitations, including: (1) it is difficult to treat current-dependent element; (2) it is difficult to perform a fast steady-state analysis; (3) it is difficult to adopt variable time-step analysis. The most popular programs based on that approach are EMTP, ATP.

In order to overcome the drawbacks of these two basic approaches, a lot of modified approaches are proposed. The modified nodal analysis includes circuit components such as voltage sources and various current dependent elements into its basic component library. The SPICE based softwares are mostly based on that approach. A power electronic circuit is a nonlinear, time-varying, switching circuit, but it normally is a periodic circuit. The state-space averaging technique is generated to solve that kind of circuits. It averages state variables during one cycle. Because the system has been converted as LTI system, the simulation is very fast. But there is still a big limitation that the system must have a constant duty cycle.

1.4.2 Control Scheme

In order to get the more qualified output, like any other system the SMPS also needs a close loop control system. The basic control schemes can be divided into two categories: one is the PWM control of the switches; the other is resonant circuit control of the switches. The PWM control scheme is very simple. For normal power consumption, the operation frequency is constant, and the duty cycle of the switches is changed according to the change of the power load. For a load below the minimum level, the constant on-time should be kept, and let the frequency variable. Normally the operation frequency for PWM is in the range of 20k-250kHz. The advantage of this scheme is simple and flexible for different load; the disadvantages are switching noise and power losses, which are all the results of the hard switching. The switching signal for resonance control is generated by a resonant circuit, which is composed by inductor and/or capacitor at a predetermined frequency. In this scheme some stray/parasitic inductors/capacitors, which are harmful in other conditions, will become useful in forming resonant circuit. This kind of converter includes purely resonant converter, quansi-resonant converter, and multi-resonant converter. The main advantage for that kind of control is that it let the zero voltage switching (ZVS) and/or zero current switching (ZCS) become possible, which kind of switching technique is called soft switching. That technique allows circuit work at higher frequency between 500kHz-10MHz. Although the high operation frequency soft switching technique weakens the switching noise and switching power loss, it still has two main drawbacks. First, using frequency to control circuit make the circuit more complex. Second, the high frequency resonant circuit increases the wire conduction losses.

The most recent developed technique is a hybrid of resonant soft-switching and PWM converters. It combines advantages of two control schemes. During switching transmission the circuit works in resonant mode; otherwise, the circuit works in PWM mode. In this way, the switching stress is reduced due to the soft switching technique; the conduction losses keeps low during non-switching time due to the low frequency operation of PWM. That kind of the control scheme is the future trend of the development.

1.5 Schematic of a Real Computer's SMPS

The computer SMPS's have three parts, the AC-DC full-bridge rectifier, the switch mode DC-DC converter, and the control circuits. AC-DC rectifier is a diode rectifier, so its operating frequencies are very low. For 110V/60Hz mains supply, its operating frequency is 120 Hz; for 220/50Hz mains supply, its operating frequency is 100Hz. DC-DC converter adopts PWM control converter and operates at high frequencies (10s of kHz typical)[51]. The control and protection circuits consist of some IC circuits. Their functions include output voltage stabilizing, overvoltage protection, and PWM signal generation.

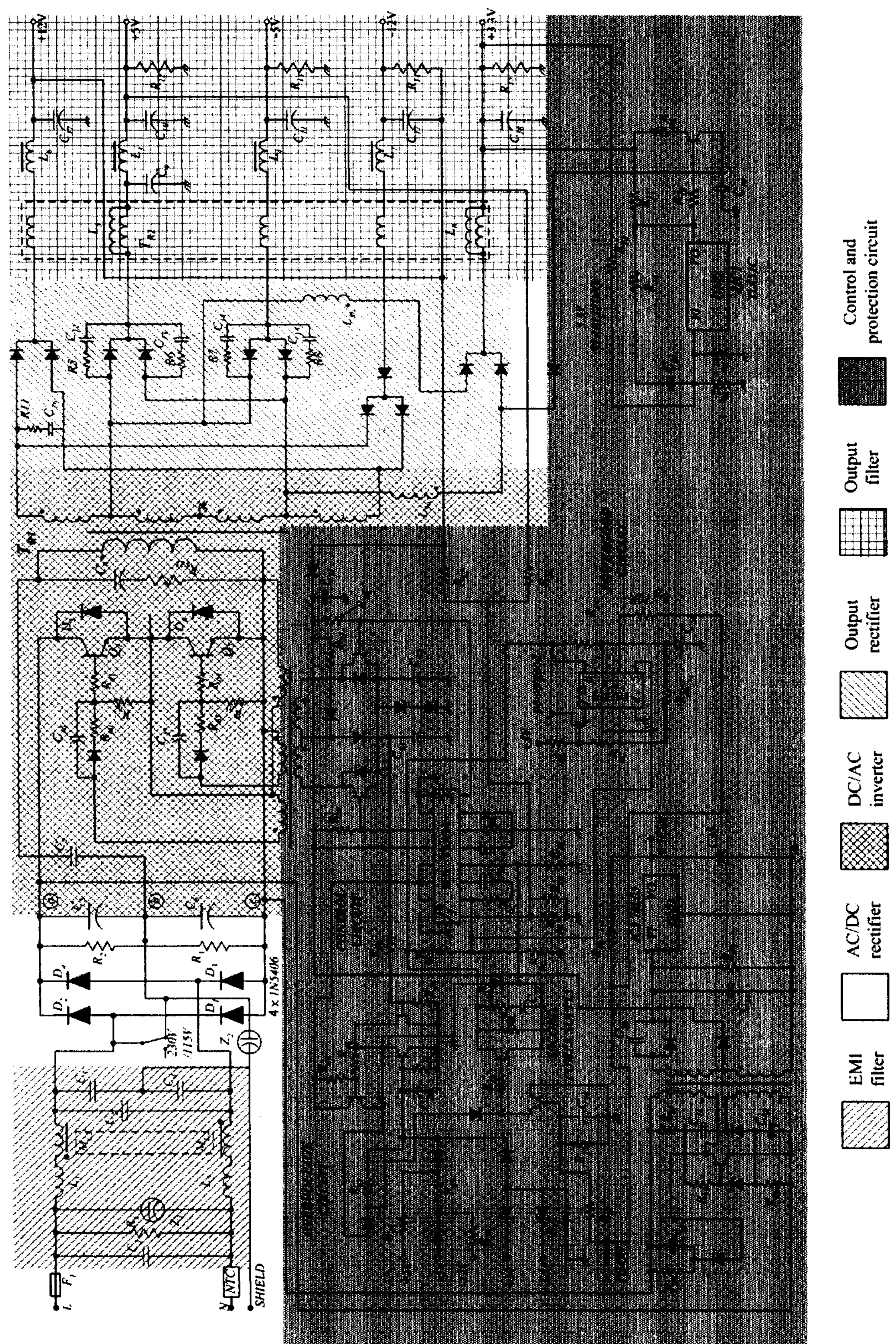


Fig. 1.5 The schematic of SMPS

The components (L_1, L_2, M_{L1}, C_1, C_2, C_3, C_4) form the EMI filter to limit the SMPS's noices. When the voltage is 115V, the rectifier works like a voltage doubler; when the voltage is 230V, the rectrifier works as a normal full-bridge rectifier.Varistors Z_1 and Z_2 have overvoltage protect function on the line input. Thermistor NTC limits input current. Behind the voltage regulator IC3 will be voltage 5V, which goes into the motherboard and it is necessary for turn-on logic. The input signals of IC1 are the feedback signals of the output voltages, the output signals of it are the PWM signals used to control Q_1 and Q_2. The overvoltage circuit guards all output voltage. When some limits are exceed, the power supply is stopped. The +3.3V Voltage stabilisation is used to compensate the cable volatage drop. POWERGOOD signal represents the output volatage are stable.

1.6 The Problems of the SMPSs

Switching power supplies offer many advantages over linear regulators such as high efficiency and relatively small size and weight, so SMPSs are widely adopted by computers, television receivers, digital equipments, etc. The switching frequency is usually above 20kHz, which is much higher than the frequency of the mains. Because of the significantly high frequency input, the step-down output transformer can be make smaller using ferrite cores than the step-down input transformer that the linear regulated power supply uses. Since the circuit between output transformer and primary rectifier is a high frequency chopper, the energy storage can be accomplished on the primary side of the step-down transformer by a relatively smaller capacitors than linear counterpart can be used.

Although the benefits of SMPS techniques are great, there are also the penalties paid due to the high frequency switching. The electromagnetic interference (EMI) is the most serious problem produced by the SMPSs, which includes radiated EMI and conducted EMI[54]. According to the Faraday's Laws, any current flowing through a conductor will generate a magnetic field that surrounds this conductor. If the current is ac current, it will generate an electromagnetic field and radiate the energy to the space. Due to the high frequency switching, the frequency of input current of output transformer is significantly high. Therefore, these conductors through which the high frequency current flows will behave like an antenna. The energy that radiate by the conductors will affect the surrounding space, and let the circuits malfunction[54]. That is radiated EMI. According to the Fourier transformation, the square pulse waveform is full of high frequency components. The switches of SMPS produce a lot of high frequency square pulse current, which, of cause, contains high frequency component. That is the conducted EMI. It is well know that in the circuit that contains real or parasite reactive component any abrupt change in current or voltage will generate high frequency transient oscillation. In the SMPS, the state of switches changes quickly and periodically, so the transient oscillation never can be annihilated and become a periodic event. That is also the conducted EMI. Although the digital circuits of the control section of SMPSs are another source of radiated emission in the radio frequency range, it does not have enough power to pose any serious threat to the proper functioning of the neighboring converter section. On the other hand, the electromagnetic energy emanating from the converter section in the form of radiated emission has enough energy to affect the proper functioning of the control section. In order to obtain the effect of the EMI, the most important job is to get the nodal voltage and circuit current.

In this thesis, I will focus on the circuit simulation. The first chapter will be the introduction of SMPS; the second chapter will introduce tranditional TLM models of the basic branches and components; the third chapter will give out the new generalized TLM models of the basic branches and components; the fourth chapter will give out a two-branch switch model and its modification; the fifth chapter will present a new proposed system decoupling technique by using TLM stub model; the sixth chapter will present the system modeling procedure; the seventh chapter will compare the simulation results to results from other methods; the last chapter will give out the conclusion and the future jobs.

Chapter 2 The Conventional TLM Method

There are three steps to do the system simulation. First step is to model the basic component into the proper form; the second step is to set up system matrices by adopting certain type of technique; the last step is to do the computation. Each step will affect the computation efficiency. These three steps will be discussed in the following chapters. In this chapter, the conventional TLM model of the basic component will be introduced.

2.1 Transmission-line Modeling

From the point of the field theory, all electric networks in the physical world are distributed parameter networks, and real lumped parameter networks are never exist in the physical world. We must use Maxwell's equations to solve the electric networks to obtain accurate results. In fact, even when we use Maxwell's equations to deal with all problems, we still cannot get the exact results of the real networks, because the results from solving Maxwell's equations just give us the exact results of the mathematical abstractions of physical electrical networks. The mathematical abstraction of a physical electrical network is just the approximation of the real one. The more complex of the abstraction, the more accurate of the model, but it will never be the exact model of the real network. There are three ranges in the frequency spectrum for which theories for field problems in general have been developed. In terms of the wavelength (λ) and the dimension of the physical electric network (l), these ranges are $\lambda >> l$, $\lambda \approx l$, $\lambda << l$ [43],[44]. In the first range, the analysis techniques are known as circuit theory. It was developed from experimentally obtained laws. All elements include the connect lines are treated as lumped parameter elements, which are the points in the networks and has no physical dimension, so there is no traveling waves in the networks. In the second range, the analysis techniques are known as microwave theory; and in the third range, the techniques are known as geometric optics. They are all based on Maxwell's theory. They normally treat the electric elements as distributed parameter elements if the dimensions of the elements are comparable or far larger than λ. Due to the existence of the distributed parameter elements, there are traveling waves in the networks.

2.1.1 Single-phase Two-wire Transmission Line[43,44]

Although we can use the Maxwell's equations to solve the distributed electric networks, we seldom really solve them by this method. The most popular way is to replace a complicated and/or distributed network by a simple or a series of simple equivalent circuits, and then resorts to Kirchhoff Current/Voltage laws (KCL/KVL). Transmission-line modeling (TLM), also known as the transmission-line-matrix method, is a numerical technique for solving field problems using circuit equivalent. It is based on the equivalence between Maxwell's equations and the equations for voltages and currents on a mesh of continuous two-wire TLs. The key technique of this method is to divided the TL into small sections (dimension= Δx), which can be equivalent to a PI or T circuit, and then taking the limit as the $\Delta x \rightarrow 0$ to get the final system equations. Comparing with the lumped network model, the transmission-line model is more general and performs better at high frequencies where the transmission and reflection properties of geometrical discontinuities cannot be regarded as lumped.

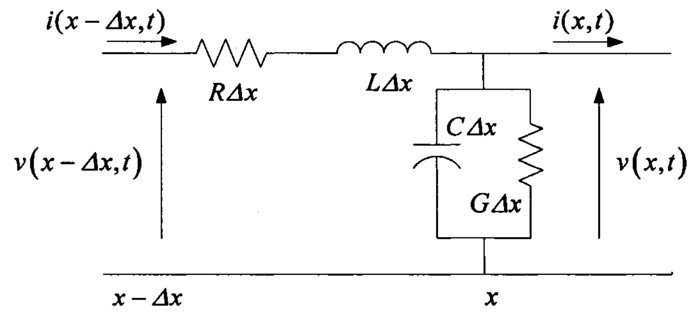


Fig. 2.1 Single-phase two-wire line section of length of Δx

We consider a single-phase two-wire TL. Fig. 2.1 shows a line section of length Δx meters. We assume the line has a loop inductance L Henry/m, a line-to-line capacitance C Farad/m, series resistance R Ohms/m, and shunt conductance G Mhos/m. Writing a KVL and KCL equation for the circuit in Fig. 2.1.

$$v(x-\Delta x,t) = i(x-\Delta x,t)R\Delta x + L\Delta x\frac{\partial i(x-\Delta x,t)}{\partial t} + v(x,t)$$
$$i(x-\Delta x,t) = i(x,t) + v(x,t)G\Delta x + C\Delta x\frac{\partial v(x,t)}{\partial t}$$

Rearrange the equations and get the following equations,

$$v(x,t) - v(x-\Delta x,t) = -L\Delta x\frac{\partial i(x-\Delta x,t)}{\partial t} - i(x,t)R\Delta x \quad (2.1)$$
$$i(x,t) - i(x-\Delta x,t) = -C\Delta x\frac{\partial V(x,t)}{\partial t} - GV(x,t) \quad (2.2)$$

We use partial derivatives here because *v*(*x*,*t*) and *i*(*x*,*t*) are differentiated with respect to both position *x* and time *t*. After solving these two equations, we can get the solutions in the Laplace transform. The details deduction procedure is shown in Appendix A.

$$V(x,s) = V^{+}(s)\exp(-\gamma(s)x) + V^{-}(s)\exp(\gamma(s)x) \quad (2.3)$$

$$I(x,s) = I^{+}(s)\exp(-\gamma(s)x) + I^{-}(s)\exp(\gamma(s)x) \quad (2.4)$$

In general, the inverse Laplace transforms of (2.3) and (2.4) are not closed form expressions. However, for the special cases of a distortionless line, which has the property *R*/*L*=*G*/*C*, or lossless line, which has the property *R*=*G*=0, the inverse Laplace transform can be express in general forms as follows:

$$v(x,t) = \exp(-\alpha x)v^{+}\left(t - x/u\right) + \exp(-\alpha x)v^{-}\left(t + x/u\right) \quad (2.5)$$

$$i(x,t) = \exp(-\alpha x)i^{+}\left(t - x/u\right) + \exp(-\alpha x)i^{-}\left(t + x/u\right) \quad (2.6)$$

$$i^{+} = v^{+}/Z_c \text{ and } i^{-} = -v^{-}/Z_c \quad (2.7)$$

where $u = 1/\sqrt{LC}$, $Z_c = \sqrt{L/C}$ and $\alpha = \sqrt{RG}$ u: velocity of the wave.

From the aforementioned procedure, we can find that the TLM is a physical discretization approach not a mathematical discretization approach[45].

2.1.2 Model of the Single-phase Two-wire Lossless Line[44]

In this section, I will focus on the lossless line. From (2.3) and (2.4), we know at any point on the TL the instantaneous voltage and current value are equal to the sum of the forward wave and backward wave. Letting R=G=0 and combing (2.7), (2.5) and (2.6) become

$$V(x,s)=V^{+}(s)\exp(-sx/u)+V^{-}(s)\exp(sx/u) \tag{2.8}$$

$$I(x,s)=\frac{V^{+}(s)}{Z_c}\exp(-sx/u)-\frac{V^{-}(s)}{Z_c}\exp(sx/u) \tag{2.9}$$

Rearrange (2.8) and (2.9) and take inverse Laplace transform, (2.10) and (2.11) are shown

$$v(x,t)+Z_c i(x,t)=2v^{+}\left(t-{x}/{u}\right) \tag{2.10}$$

$$v(x,t)-Z_c i(x,t)=2v^{-}\left(t+{x}/{u}\right) \tag{2.11}$$

In (2.10), the values of the left side are determined by the arguments x and t. But if we can keep (t - x/u) constant, the left side will also keep constant. That means if τ is the transit time from terminal k to terminal m of the TL, the following equation will be satisfied.

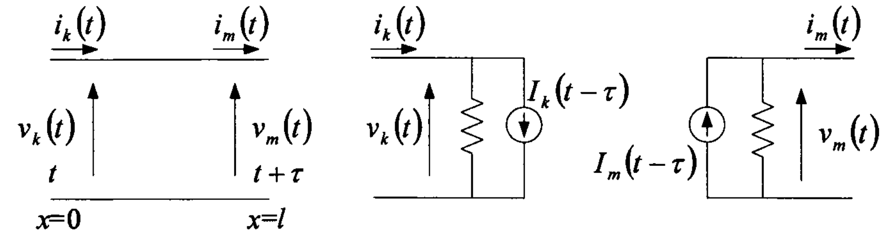
$$v_k(t-\tau)+Z_c i_k(t-\tau)=v_m(t)+Z_c i_m(t) \tag{2.12}$$

$$v_m(t-\tau)-Z_c i_m(t-\tau)=v_k(t)-Z_c i_k(t) \tag{2.13}$$

Eqns (2.12) and (2.13) can be rewritten as

$$i_m(t)=I_m(t-\tau)-v_m(t)/Z_c \quad \text{where} \quad I_m(t-\tau)=i_k(t-\tau)+v_k(t-\tau)/Z_c \tag{2.14}$$

$$i_k(t) = I_k(t-\tau) + v_k(t)/Z_c \quad \text{where} \quad I_k(t-\tau) = i_m(t-\tau) - v_m(t-\tau)/Z_c \tag{2.15}$$



a) Terminal variables b) Discrete time equivalent circuit

Fig. 2.2 Lossless line equivalent circuit

From (2.12) - (2.15), we find the voltage and current values of one terminal can be determined by the voltage and current values of the other terminal τ time ago. Using this property, the network can be decoupled from here. So one complicated system is divided into two simpler systems, it makes the system equations are easier to be solved.

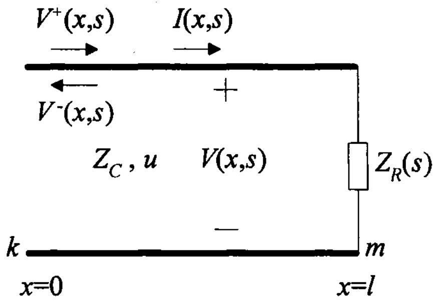


Fig. 2.3 Single-phase two-wire lossless line with terminations

Fig. 2.3 shows a TL, which length is l. terminated by an impedance Z_R at the terminal m. The boundary condition at the terminal m is $V_m(s)=Z_R(s)I_R(s)$. Using (2.8), (2.9), and boundary condition, the relationship between V^+ and V^- at terminal m is shown in (2.16)

$$V^-(s)\exp\left(\frac{sl}{u}\right) = F_R(s)V^+(s)\exp\left(-\frac{sl}{u}\right)$$
$$\text{where} \quad F_R(s) = \left(\frac{Z_R(s)}{Z_C} - 1\right) \Big/ \left(\frac{Z_R(s)}{Z_C} + 1\right) \tag{2.16}$$

By observing (2.16) and (2.8), we can find the left side of (2.16) is the negative x direction traveling voltage wave at terminal m, right side except $F_R(s)$ is the positive x direction traveling voltage wave at terminal m. So the function $F_R(s)$ has a physical meaning, the reflection coefficient.

$$V^{-}(l,s) = F_R(s)V^{+}(l,s)$$

When the terminal m is short-circuited, the reflection coefficient F_R is -1; when the terminal m is open-circuited, the reflection coefficient F_R is 1; when the terminal m is terminated by an impedance, which value is Z_C, the reflection coefficient F_R is 0.

2.2 Basic Models of the TLM Technique

In the last section, we physically discretize the distributed parameter TL and obtain the discrete-time equivalent circuit. Through the equivalent circuit, the whole system is decoupled from here. Normally, in order to make the network easier to solve, we try our best to make it equivalent to simple lumped parameter network. But in TLM technique, we inversely use the ordinary technique, and models lumped reactive components as distributed TL sections. To do this, we can model the system by TLM models of the circuit devices.

There are two kinds of models in the TLM technique, stubs and links. Both of them can represent the inductor and/or capacitor. The only difference is that the stub is a one-port device, in which the far terminal is ended by impedance with different value according to the characteristic of the original electrical device, but the link is a two-port device. These models are used by Hui and Christopoulos [10,11,15,16] for transient analysis. After modeling a lumped reactive component to be a section of distributed TL, the modeling error is unavoidably occurred, due to the capacitance and inductance effects of TL. The details of stub and link models of passive components will be introduced in the following sections.

1. TLM Stubs

In the last paragraph of the last section, I give out the reflection coefficients under three different conditions. Now I will discuss these three cases in details. According to (2.10) , if the argument $(t\text{-}x/u)$ is constant, the $v^{+}(x,t)$ is constant. Similarly to the $v^{-}(x,t)$, if $(t+x/u)$ is constant. If we assuming τ_S, time step, is equal to the round trip time, which allows the wave traveling from one terminal to another terminal and come back, we have the following relationship

$$v^{+}(0,t)=v^{+}(l,t+{}^{\tau_S}\!/_2)=\frac{1}{F_R}v^{-}(l,t+{}^{\tau_S}\!/_2)=\frac{1}{F_R}v^{-}(0,t+\tau_S)$$

The V-I relationship at the terminal k in Fig. 2.3, is shown as following

$$i(0,t)=\frac{1}{Z_C}\left(v^{+}(0,t)-v^{-}(0,t)\right) \tag{2.17}$$

$$i(0,t-\tau_S)=\frac{1}{Z_C}\left(v^{+}(0,t-\tau_S)-v^{-}(0,t-\tau_S)\right) \tag{2.18}$$

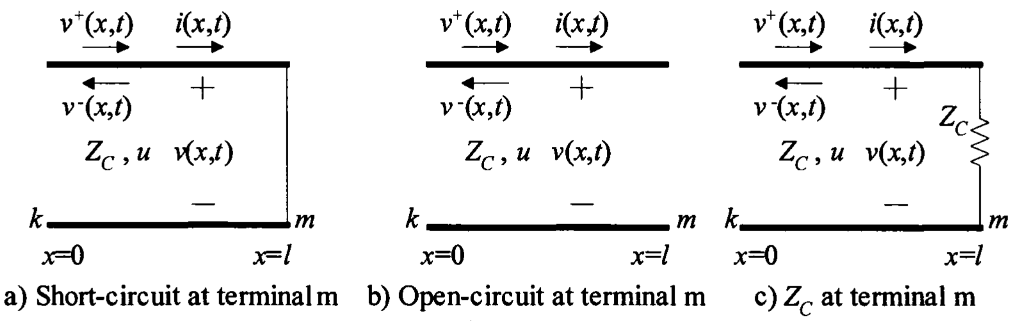


Fig. 2.4 Three kinds of terminations for single-phase two-wire transmission line

In case one, a section of TL is short-circuited at the terminal m, as shown in Fig. 2.4a. The reflection coefficient at terminal m is –1. We use (2.17) minus (2.18) and adopt relationship of $v=v^{+}+v^{-}$, (2.19) is introduced.

$$v(0,t)+v(0,t-\tau_S)=Z_C\left(i(0,t)-i(0,t-\tau_S)\right) \tag{2.19}$$

If we think about an inductance L and use trapezoidal rule to get discrete-time V-I relationship of it, we can obtain a similar characteristic equation shown as (2.20). That means we can use short-circuited TLM stub model to represent an inductance in the circuit.

$$v(0,t)+v(0,t-\tau_S)=\left(i(0,t)-i(0,t-\tau_S)\right)2L/\tau_S \tag{2.20}$$

The characteristic impedance ($Z_{C(sb)}$) of this TL is $2L/\tau_S$. The modeling error is $\tau_S^2/4L$, which is treated as the capacitance of this TL.

Similarly in the case two, a section of TL is open-circuited at the terminal m, as shown in Fig. 2.4b. The reflection coefficient at terminal m is 1. We use (2.17) plus (2.18) and adopt relationship of $v=v^{+}+v^{-}$, (2.21) is introduced.

$$v(0,t)-v(0,t-\tau_S)=Z_C\left(i(0,t)+i(0,t-\tau_S)\right) \tag{2.21}$$

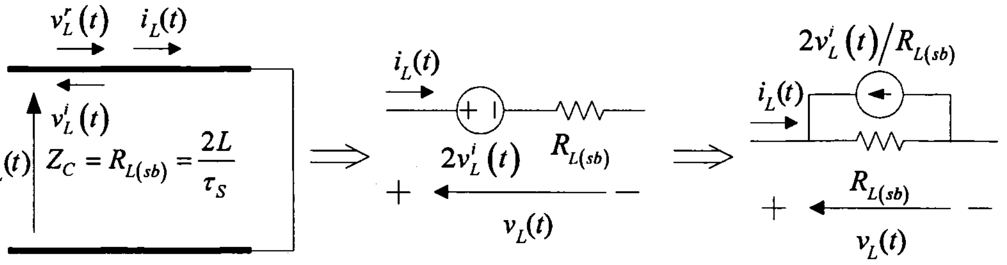
That equation is equivalent to the discrete-time V-I relationship of capacitance by using trapezoidal rule. So we can use open-circuited TLM stub model to represent a capacitance in the circuit. In that model, the characteristic impedance of TL is $\tau_S/2$, the modeling error is $\tau_S^2/4C$.

In the case three, a section of TL is terminated by a resistor whose value is equal to the characteristic impedance of the TL at the terminal m, as shown in Fig. 2.4c. The reflection coefficient at terminal m is 0. Eqn. (2.17) becomes the following equation, which is completely same as the characteristic of a pure resistance.

$$i(0,t)=v(0,t)/Z_C \tag{2.22}$$

From these procedures, we can find the pure inductance, capacitance, and resistance can be modeled as TL with different terminating impedance. Next, I will give out the TLM stub model, which will be used in the system model.

During the system modeling, any wave entered the system is call incident wave, denoted as v^i, any wave left the system is call reflected wave, denoted as v^r. That notation is more clear that v^+ and v^-. The relationships of these two notations are $v^+=v^r$, $v^-=v^i$. The TLM stub model of an inductance is shown in Fig. 2.5 and (2.23), (2.24). i_L is the inductor current, which reference together with the voltage reference across the inductor forms a standard reference.



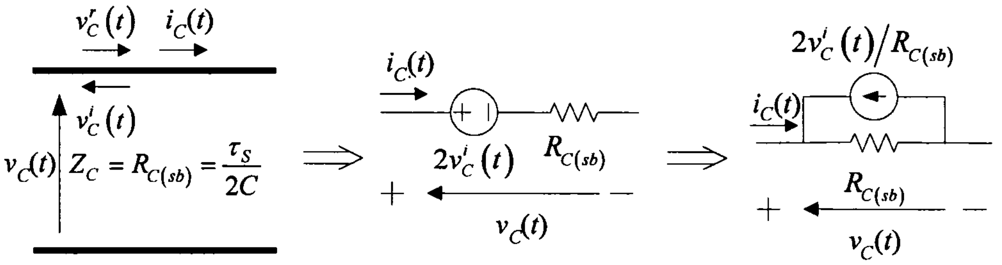
a) Short-circuit at terminal b) Thevenin equivalent c) Norton equivalent

Fig. 2.5 TLM stub model of inductor

Eqn. (2.23) determines the V-I relationship. The (2.24) determines the next time step incident voltage wave.[11]

$$V_L(t) = V_L^i(t) + V_L^r(t) = V_L^i(t) + \left(i_L(t) - i_L^i(t)\right)R_{L(sb)} = 2V_L^i(t) + i_L(t)R_{L(sb)} \tag{2.23}$$

$$V_L^i(t+\tau_S) = -{}_nV_L^r(t) = {}_nV_L^i(t) - {}_nV_L(t) \tag{2.24}$$



a) Open-circuit at terminal b) Thevenin equivalent c) Norton equivalent

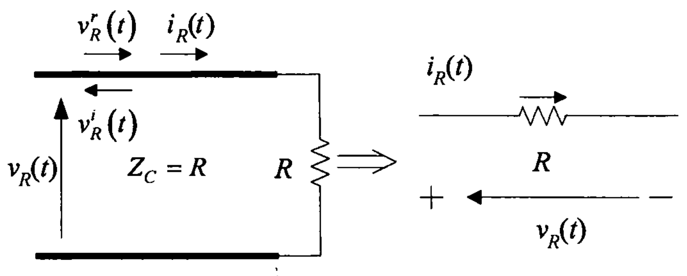
Fig. 2.6 TLM stub model of capacitor

The TLM stub model of a capacitance is shown in Fig. 2.6 and (2.25), (2.26). Like the inductance model, i_C is the capacitance current, which reference together with the voltage reference across the inductor forms a standard reference.

$$V_C(t)=V_C^i(t)+V_C^r(t)=V_C^i(t)+\left(i_C(t)-i_C^i(t)\right)R_{C(sb)}=2V_C^i(t)+i_C(t)R_{C(sb)} \tag{2.25}$$

$$V_C^i(t+\tau_S)={}_nV_C^r(t)={}_nV_C(t)-{}_nV_C^i(t) \tag{2.26}$$

From (2.22), it shows the TLM stub model of a resistance is just a resistance connected in the system without equivalent voltage/current source.



a) Short-circuit at terminal b) Equivalent circuit

Fig. 2.7 TLM stub model of resistance

The equivalent circuits obtained from (2.23)-(2.26) are shown in Fig. 2.5, Fig. 2.6, and Fig. 2.7. The direction of v is the direction of the potential, not the voltage. Using these three equivalent circuits, any electrical network, which contains these three kinds of components, can be represented as a network of transmission sections.

2. TLM Links

Like TLM stubs, the link models of reactive components are also obtained by approximating the lumped reactive components to be distributed TLs. But there are two main different properties. First, the link model is a two-port model, no matter it is a capacitor or an inductor. Second, whether a link model represents an inductor or a capacitor is determined by the relationship between the terminal variable and component variable not by the termination type.

Fig. 2.8 shows the link model of reactive component. Let's assume time τ_L is equal to the single-trip time of that TL. For an inductor L, the characteristic impedance ($Z_{L(lk)}$) is L/τ_L, the modeling error is $\tau_L^2/2L$. For an capacitor C, the characteristic impedance ($Z_{C(lk)}$) is τ_L/C, the modeling error is $2C/\tau_L^2$[15]. The difference between the characteristic impedances of the link and stub is due to the different single-trip time[14]. Similar to the procedure described in the last section, the link model is shown in Fig. 2.8. L and R mean the left ending and right ending of the reactive components. We can arbitrarily choose one end of link model as left ending and leave another ending as right ending. Its variables have the following relationship.

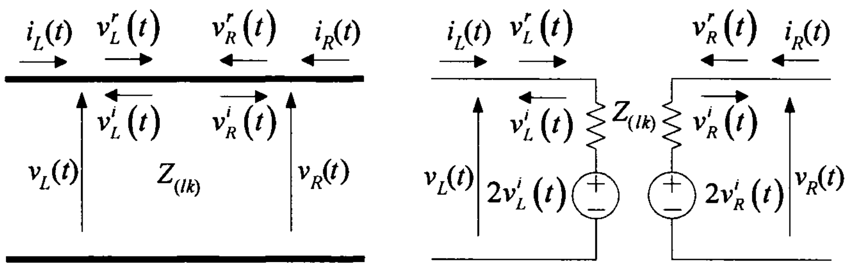


Fig. 2.8 TLM link model of reactive components

$$_Lv(t) = {}_Li(t)Z_{(lk)} + 2\,{}_Lv^i(t) \tag{2.27}$$

$$_Rv(t) = {}_Ri(t)Z_{(lk)} + 2\,{}_Rv^i(t) \tag{2.28}$$

$$_Lv(t-\tau_L) = {}_Li(t-\tau_L)Z_{(lk)} + 2\,{}_Lv^i(t-\tau_L) \tag{2.29}$$

$$_Rv(t-\tau_L) = {}_Ri(t-\tau_L)Z_{(lk)} + 2\,{}_Rv^i(t-\tau_L) \tag{2.30}$$

$$_Lv^i(t) = {}_Rv(t-\tau_L) - {}_Rv^i(t-\tau_L) \tag{2.31}$$

$$_Rv^i(t) = {}_Lv(t-\tau_L) - {}_Lv^i(t-\tau_L) \tag{2.32}$$

Let the summation of (2.27) and (2.30) subtract the summation of (2.28) and (2.29), and use (2.31) and (2.32) in the result. We have the equation

$$\begin{aligned}&\left({}_{L}v(t)-{}_{R}v(t)\right)+\left({}_{L}v(t-\tau_L)-{}_{R}v(t-\tau_L)\right)\\&=\left(\left({}_{L}i(t)-{}_{R}i(t)\right)-\left({}_{L}i(t-\tau_L)-{}_{R}i(t-\tau_L)\right)\right)Z_{(lk)}\\&=\left(\frac{{}_{L}i(t)-{}_{R}i(t)}{2}-\frac{{}_{L}i(t-\tau_L)-{}_{R}i(t-\tau_L)}{2}\right)\frac{2L}{\tau_L}\end{aligned} \tag{2.33}$$

If we let $Z_{(l\,k)}=L\,/\,\tau_L$, and assume the current (i) flow through the TL is equal to the average value of the current (${}_{L}i$ and $-_{R}i$) flow from the left terminal to the right terminal. The (2.33) will have the same form as an inductor model discretized by trapezoidal rule.

Using (2.31) and (2.32) in the sum of (2.27)-(2.30), we have the following equation

$$\begin{aligned}&{}_{L}v(t)+{}_{R}v(t)-\left({}_{L}v(t-\tau_L)+{}_{R}v(t-\tau_L)\right)\\&=\left({}_{L}i(t)+{}_{R}i(t)+{}_{L}i(t-\tau_L)+{}_{R}i(t-\tau_L)\right)Z_{(lk)}\quad\text{or}\\&\frac{{}_{L}v(t)+{}_{R}v(t)}{2}-\frac{{}_{L}v(t-\tau_L)+{}_{R}v(t-\tau_L)}{2}\\&=\left(\left({}_{L}i(t)+{}_{R}i(t)\right)+\left({}_{L}i(t-\tau_L)+{}_{R}i(t-\tau_L)\right)\right)\frac{\tau_L}{2C}\end{aligned} \tag{2.34}$$

If we let $Z_{(l\,k)}=\tau_L\,/\,C$, and assume the voltage (v) over the TL is equal to the average value of the voltage (${}_{L}v$ and ${}_{R}v$) over the left terminal and the right terminal. The (2.34) will have the same form as a capacitor model discretised by trapezoidal rule.

The relationship between ${}_{L}v$, ${}_{R}v$, v, ${}_{L}i$, ${}_{R}i$, and i determine which element is represented by TLM link model.

2.3 TLM Stub Models of Electric Branches

From last section, we can find that the TLM stub models of one-port reactive components are also one-port models. So any electrical circuit can be represented as a network of transmission sections without topology changes by simply replacing the reactive components with the corresponding TLM stubs. If we can define some first order branches as basic branches to form arbitrary circuit, the number of the circuit node will be reduced and computation will be accelerated. Because I will use modified nodal analysis method to set up the system equations, the Norton equivalent circuits are preferred. The following section will discuss some basic braches. The time step for discrete time modeling since then will be noted as Δt.

Here some assumptions used in this thesis are listed:

(1) All electrical components are lumped components, because the traveling waves are not issues in my research.

(2) The magnetic branch of transformer is omitted. There are two main reasons to do this assumption. First, to avoid the possibility of ill-condition system matrices; second, under the regular operating condition, the magnetic current is very small comparing with the load current.

(3) The line stray inductance and capacitance are ignored. In this thesis, high frequency property of the circuit is not an issue.

(4) The stray capacitance of high frequency transformer are also ignored. In the tesed circuit, the input and output coils of the only high frequency transformer are always parallel connected with clamp capacitor.

2.3.1 Equivalent Branch Reduction

Adopting the TLM models of the inductor and capacitor, which is mention in the previous section, we can obtain the equivalent TLM model of any RLC circuit. The size of the circuit keeps invariant after modeling. If we make the basic branch contain more than one series element, the branch will be reduced into an equivalent resistance and current source, effectively removing unnecessary nodes, which will decrease the size of the circuit conductance matrix, and improve the simulation speed. The following part of this section will discuss two type of equivalent branch: type one is that a resistor connects in series with parallel-connected resistor and capacitor, the type two is that a resistor connects in series with parallel-connected resistor and inductor. The TLM stub models for these two equivalent branches are given out in the next part of this section. The detail calculation procedure is shown in Appendix C.

1. Type One

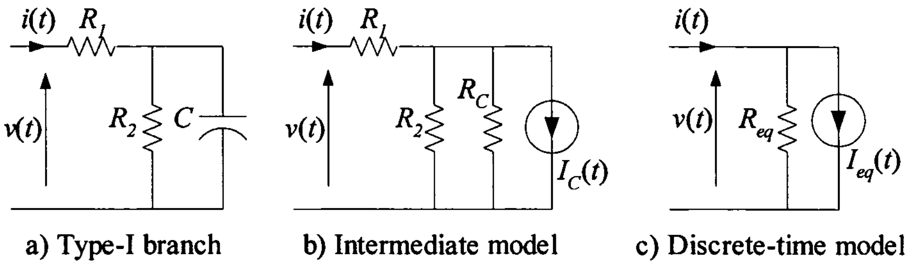


Fig. 2.9 TLM model of type one branch

Recently, some simulation methods that use the companion models have been proposed[9,22,24]. In these modeling methods, every reactive element is connected with a resistor in series. This resistor can be a physical element in the circuit, or parasitic resistance with the reactive element. In order to generalize them further, I add another resistor in the companion models, which are shown in Fig. 2.9 and Fig. 2.10.

$$v_C(t) = R_C i_C(t) + 2v_C^i(t)$$

$$i(t) = v_C(t)/R_2 + i_C(t)$$

$$v(t) = v_C(t) + R_1 i_C(t)$$

$$v_C^i(t) = v_C(t-\Delta t) - v_C^i(t-\Delta t)$$

Rearranging these equations, we can get the branch expression as follow.

$$i(t)=\frac{v(t)}{R_{eq}}+I_{eq}(t)=\frac{v(t)}{R_{eq}}-2v_C^i(t)\frac{R_2}{R_2R_C+R_1R_C+R_1R_2} \tag{2.35}$$

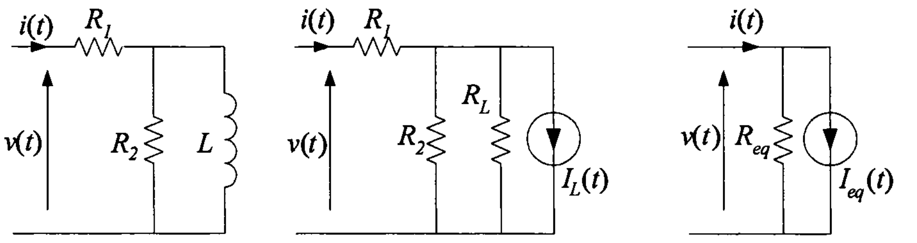
where $v_C^i(t)=\frac{R_2R_C}{R_2R_C+R_1R_C+R_1R_2}v(t-\Delta t)+\frac{R_1R_2-R_2R_C-R_1R_C}{R_2R_C+R_1R_C+R_1R_2}v_C^i(t-\Delta t)$

$$R_{eq}=R_1+\frac{R_2R_C}{R_2+R_C} \qquad R_C=\frac{\Delta t}{2C}$$

By using the Kirchhoff law and TLM stub model of inductors, the original branch is transformed into a resistance R_{eq} in parallel with a current source $I_{eq}(t)$. This model is similar to the standard lumped capacitance model, but with some modification on the formula.

2. Type Two

Similar to type one branch, type two also contains two resistor, one is connected in series and the other is connected in parallel. Type two equivalent branch can also be modeled as a purely resistance R_{eq} parallel connected with a current source $I_{eq}(t-\Delta t)$



a) Type-II branch b) Intermediate model c) Discrete-time model

Fig. 2.10 TLM model of type two branch

$$v_L(t)=R_Li_L(t)+2v_L^i(t)$$

$$i(t)=v_L(t)/R_2+i_L(t)$$

$$v(t)=v_L(t)+i(t)R_1$$

$$v_L^i(t)=-v_L(t-\Delta t)+v_L^i(t-\Delta t)$$

Rearranging these equations, we can get the equivalent resistance and voltage source expression as follow.

$$i(t)=\frac{v(t)}{R_{eq}}+I_{eq}(t)=\frac{v(t)}{R_{eq}}-2v_L^i(t)\frac{R_2}{R_2R_L+R_1R_L+R_1R_2} \tag{2.36}$$

$$\text{where } v_L^i(t)=-\frac{R_2R_L}{R_2R_L+R_1R_L+R_1R_2}v(t-\Delta t)+\frac{R_2R_L+R_1R_L-R_1R_2}{R_2R_L+R_1R_L+R_1R_2}v_L^i(t-\Delta t)$$

$$R_{eq}=R_1+\frac{R_2R_L}{R_2+R_L} \qquad R_L=\frac{2L}{\Delta t}$$

Eqns. (2.35) and (2.36) are the discrete time equations for two basic branches. The detailed derivation process are shown in Appendix C.

2.3.2 TLM Model of Mutual Coupled Inductance

In the SMPS system, there are a lot of mutual coupled inductances, such as the EMI circuits, high frequency transformers, and some unexpected coupled inductances. No matter what is the function of the mutual coupled inductance, from the view of the physics, they are the same thing. The only difference is the values of mutual inductances. By using TLM technique, we can obtain the TLM models of mutual inductance.

As we know, the voltage change over the capacitor in a certain time period is proportional to the integration of the current flowing through the capacitor in the same time period; the voltage over the inductor at certain time point is proportional to the current derivative at that time point. From the section 2.2, generalizing the capacitor and inductor TLM stub model, we have the following transform formula for any integral term and derivative terms[15,16].

$$y(t)=a\frac{dx(t)}{dt} \Leftrightarrow \begin{array}{l} y(t)=\frac{2a}{\Delta t}x(t)+2y^i(t) \\ y^i(t)=-\left(y(t-\Delta t)-y^i(t-\Delta t)\right) \end{array} \tag{2.37}$$

$$y(t)=\frac{1}{a}\int dx(t)dt \Leftrightarrow \begin{array}{l} y(t)=\frac{\Delta t}{2a}x(t)+2y^{i}(t) \\ y^{i}(t)=y(t-\Delta t)-y^{i}(t-\Delta t) \end{array} \tag{2.38}$$

For the two-winding mutual coupled inductance, the system equations are shown

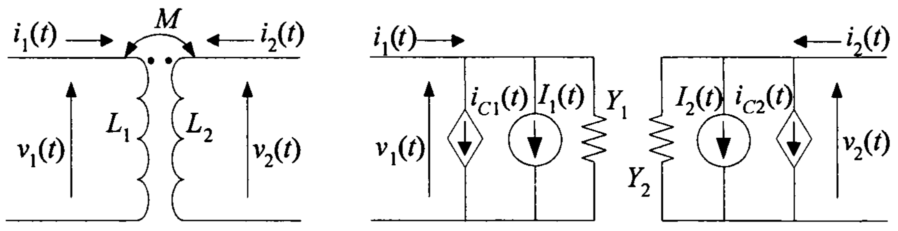
$$v_1(t)=L_1\frac{di_1(t)}{dt}+M\frac{di_2(t)}{dt}+i_1(t)R_1$$
$$v_2(t)=M\frac{di_1(t)}{dt}+L_2\frac{di_2(t)}{dt}+i_2(t)R_2$$

After using the TLM technique and rearrange the equations, we have

$$\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}=\begin{bmatrix} R_{L1}+R_1 & R_M \\ R_M & R_{L2}+R_2 \end{bmatrix}\begin{bmatrix} i_1(t) \\ i_2(t) \end{bmatrix}+2\begin{bmatrix} \left(v_{L1}^{i}(t)+v_{M12}^{i}(t)\right) \\ v_{L2}^{i}(t)+v_{M21}^{i}(t) \end{bmatrix}$$

$$\begin{bmatrix} i_1(t) \\ i_2(t) \end{bmatrix}=Z_T^{-1}\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}-2Z_T^{-1}\begin{bmatrix} v_{L1}^{i}(t)+v_{M12}^{i}(t) \\ v_{L2}^{i}(t)+v_{M21}^{i}(t) \end{bmatrix} \tag{2.39}$$

where $Z_T=\frac{2}{T}\begin{bmatrix} L_1 & M \\ M & L_2 \end{bmatrix}+\begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$ $\quad Z_T^{-1}=\begin{bmatrix} Y_1 & Y_{12} \\ Y_{21} & Y_2 \end{bmatrix}$



a) Two-winding transformer b) TLM model

Fig. 2.11 TLM model of two winding mutual coupled inductance

The TLM model of mutual inductance obtained from (2.39) is shown in Fig. 2.11. I_1 and I_2 represent independent current sources, which are determined by the history value. i_{C1} and i_{C2} represent dependent current sources, which exist due to the mutual inductances.

$$i_{C1}(t)=Y_{12}v_2(t) \qquad i_{C2}(t)=Y_{21}v_1(t)$$

$$\begin{bmatrix} I_1(t) \\ I_2(t) \end{bmatrix} = -2Z_T^{-1}\begin{bmatrix} v_{L1}^i(t)+v_{M12}^i(t) \\ v_{L2}^i(t)+v_{M21}^i(t) \end{bmatrix}$$

For a three-winding mutual coupled inductance, its model has the similar form as two-winding ones. The TLM model and system equation are shown in Fig. 2.12 and (2.40).

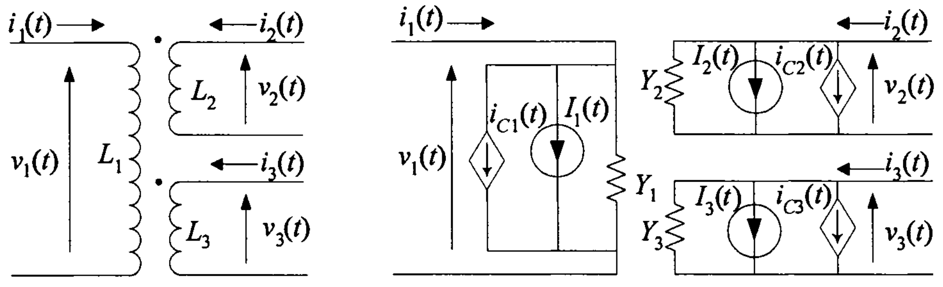
$$\begin{bmatrix} i_1(t) \\ i_2(t) \\ i_3(t) \end{bmatrix} = Z_T^{-1}\begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} - 2Z_T^{-1}\begin{bmatrix} v_1^i(t)+v_{12}^i(t)+v_{13}^i(t) \\ v_{21}^i(t)+v_2^i(t)+v_{23}^i(t) \\ v_{31}^i(t)+v_{32}^i(t)+v_3^i(t) \end{bmatrix} = Z_T^{-1}\begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} + \begin{bmatrix} I_1(t) \\ I_2(t) \\ I_3(t) \end{bmatrix} \tag{2.40}$$

$$\begin{bmatrix} i_{C1}(t) \\ i_{C2}(t) \\ i_{C3}(t) \end{bmatrix} = \begin{bmatrix} Y_{12}v_2(t)+Y_{13}v_3(t) \\ Y_{21}v_1(t)+Y_{23}v_3(t) \\ Y_{31}v_1(t)+Y_{32}v_2(t) \end{bmatrix}$$

$$\begin{bmatrix} v_1^i(t) \\ v_{12}^i(t) \\ v_{13}^i(t) \\ v_{21}^i(t) \\ v_2^i(t) \\ v_{23}^i(t) \\ v_{31}^i(t) \\ v_{32}^i(t) \\ v_3^i(t) \end{bmatrix} = \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_{12} & 0 \\ 0 & 0 & R_{13} \\ R_{12} & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_{23} \\ R_{13} & 0 & 0 \\ 0 & R_{23} & 0 \\ 0 & 0 & R_3 \end{bmatrix} Z_T^{-1}\begin{bmatrix} v_1(t-\Delta t) \\ v_2(t-\Delta t) \\ v_3(t-\Delta t) \end{bmatrix} - \begin{bmatrix} v_1^i(t) \\ v_{12}^i(t-\Delta t) \\ v_{13}^i(t-\Delta t) \\ v_{21}^i(t-\Delta t) \\ v_2^i(t-\Delta t) \\ v_{23}^i(t-\Delta t) \\ v_{31}^i(t-\Delta t) \\ v_{32}^i(t-\Delta t) \\ v_3^i(t-\Delta t) \end{bmatrix}$$

$$-2\begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_{12} & 0 \\ 0 & 0 & R_{13} \\ R_{12} & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_{23} \\ R_{13} & 0 & 0 \\ 0 & R_{23} & 0 \\ 0 & 0 & R_3 \end{bmatrix} Z_T^{-1}\begin{bmatrix} v_1^i(t-\Delta t)+v_{12}^i(t-\Delta t)+v_{13}^i(t-\Delta t) \\ v_{21}^i(t-\Delta t)+v_2^i(t-\Delta t)+v_{23}^i(t-\Delta t) \\ v_{31}^i(t-\Delta t)+v_{32}^i(t-\Delta t)+v_3^i(t-\Delta t) \end{bmatrix}$$

$$Z_T = \frac{2}{\Delta t}\begin{bmatrix} L_1 & L_{12} & L_{13} \\ L_{12} & L_2 & L_{23} \\ L_{13} & L_{23} & L_3 \end{bmatrix} + \begin{bmatrix} R_1 & 0 & 0 \\ 0 & R_2 & 0 \\ 0 & 0 & R_3 \end{bmatrix} \qquad Z_T^{-1} = \begin{bmatrix} Y_1 & Y_{12} & Y_{13} \\ Y_{12} & Y_2 & Y_{23} \\ Y_{13} & Y_{23} & Y_3 \end{bmatrix}$$



a) Three-winding transformer b) TLM Model

Fig. 2.12 TLM model of three-winding mutual coupled inductance

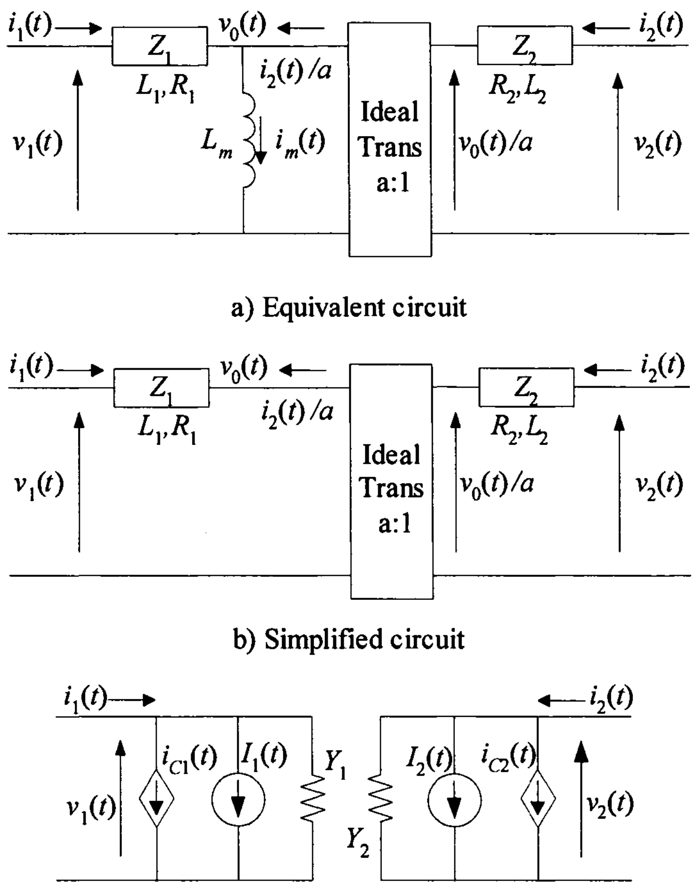
The relationship between the self-inductance and mutual inductance is

$$M = k\sqrt{L_1 L_2} \le \sqrt{L_1 L_2}$$

k is called the coupling coefficient, which does never exceed 1. When the inductances are perfect coupled to each other, the *k* will be equal to 1. Let's see the Z_T for a two-winding transformer, it becomes a singular matrix, so the inverse of Z_T does not exist. But there is no perfect transformer in the world, because of the power losses of the transformer. Therefore, we always can obtain the model in theory, which is shown in Fig. 2.11 and Fig. 2.12

2.3.3 TLM Model of Transformer

When the coefficient of coupling *k* approaches unity, which means the coils are coupled tightly, the determinant of Z_T is near zero, and the elements of the inverse inductance matrix become large and approach infinity. This makes it impossible to solve the transformer state by using (2.35) and (2.36) in computer due to the ill conditioned device characteristic equations. In the EMTP theory book, it gives out an advice that only leakage reactance and no magnetizing branches (set its value to infinity) are modeled. Let us thinking about a two-winding transformer first, which is shown in Fig. 2.13. By observing the simplified equivalent circuit in Fig. 2.13, the V-I relationship of the transformer is shown in (2.41),



a) Equivalent circuit

b) Simplified circuit

c) TLM model of simplified circuit

Fig. 2.13 TLM model of two-winding transformer

$$v_1(t) - v_0(t) = L_1 \frac{di_1(t)}{dt} + i_1(t) R_1 \qquad \text{(a)}$$

$$v_2(t) - \frac{v_0(t)}{a} = L_2 \frac{di_2(t)}{dt} + i_2(t) R_2 \qquad \text{(b)} \qquad (2.41)$$

$$i_m(t) = i_1(t) + i_2(t)/a = 0 \qquad \text{(c)}$$

Adopting TLM technique, the equation transform into the equation

$$v_1(t)-v_0(t)=(R_{L1}+R_1)i_1(t)+2v_{L1}^i(t)=Z_1i_1(t)+2v_{L1}^i(t) \quad \text{(a)}$$
$$v_2(t)-v_0(t)/a=(R_{L2}+R_2)i_2(t)+2v_{L2}^i(t)=Z_2i_2(t)+2v_{L2}^i(t) \quad \text{(b)} \qquad (2.42)$$

Using (2.42) into (2.41) the following equation can be deducted.

$$v_0(t)=Y_{T2}^3\left[v_1(t)-2v_{L1}^i(t);\quad v_2(t)-2v_{L2}^i(t)\right] \quad \text{(a)}$$

$$\begin{bmatrix} i_1(t) \\ i_2(t) \end{bmatrix}=Y_{T2}^1\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}-Y_{T2}^1\begin{bmatrix} 2v_{L1}^i(t) \\ 2v_{L2}^i(t) \end{bmatrix}=Y_{T2}^1\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}+\begin{bmatrix} I_1(t) \\ I_2(t) \end{bmatrix}$$
$$\begin{bmatrix} i_{C1}(t) \\ i_{C2}(t) \end{bmatrix}=\begin{bmatrix} Y_{T2}^1(1,2)v_2(t) \\ Y_{T2}^1(2,1)v_1(t) \end{bmatrix} \quad \text{(b)} \qquad (2.43)$$

$$\begin{bmatrix} v_{L1}^i(t+T) \\ v_{L2}^i(t+T) \end{bmatrix}=-Y_{T2}^2\begin{bmatrix} v_1(t)-2v_{L1}^i(t) \\ v_2(t)-2v_{L2}^i(t) \end{bmatrix}-\begin{bmatrix} v_{L1}^i(t) \\ v_{L2}^i(t) \end{bmatrix} \quad \text{(c)}$$

where $Y_{T2}^3=\begin{bmatrix} a^2Z_2 & aZ_1 \end{bmatrix}/(a^2Z_2+Z_1)$

$$Y_{T2}^1=\begin{bmatrix} 1 & -a \\ -a & a^2 \end{bmatrix}/(a^2Z_2+Z_1)$$

$$Y_{T2}^2=\begin{bmatrix} Z_{L1} & 0 \\ 0 & Z_{L2} \end{bmatrix}Y_{T2}^1$$

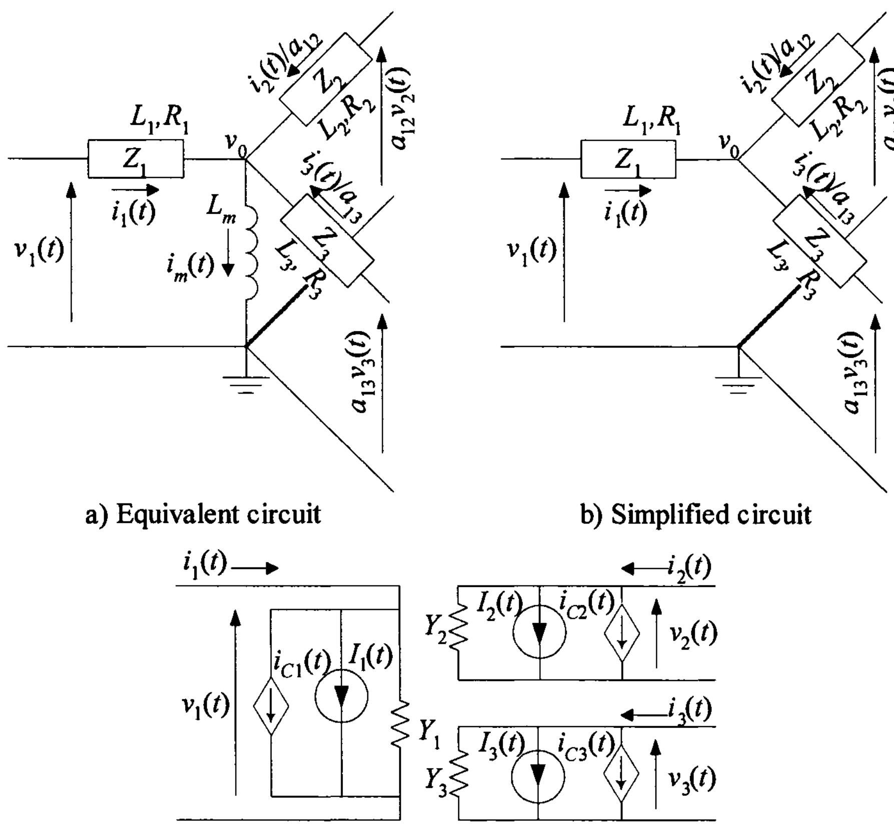
Fig. 2.14 is a model of three-winding transformer, which omits the magnetic branch. Observing the simplified circuit of the three-winding transformer, we can have the following system equations. a_{12} is the turn ratio of the first winding and the second winding; a_{13} is the turn ratio of the first winding and the third winding.

$$v_1(t)-v_0(t)=L_1\frac{di_1(t)}{dt}+i_1(t)R_1 \quad \text{(a)} \qquad (2.44)$$

$$v_2(t)-\frac{v_0(t)}{a_{12}}=L_2\frac{di_2(t)}{dt}+i_2(t)R_2 \quad \text{(b)}$$

$$v_3(t) - \frac{v_0(t)}{a_{13}} = L_3 \frac{di_3(t)}{dt} + i_3(t) R_3 \qquad \text{(c)}$$

$$i_m(t) = i_1(t) + \frac{i_2(t)}{a_{12}} + \frac{i_3(t)}{a_{13}} = 0 \qquad \text{(d)}$$



a) Equivalent circuit b) Simplified circuit

c) TLM model

Fig. 2.14 TLM model of three-winding transformer

By using the TLM technique to (2.44), we have the following characteristic equations.

$$v_0(t) = Y_{T3}^3 \left[v_1(t) - 2v_{L1}^i(t);\quad v_2(t) - 2v_{L2}^i(t);\quad v_3(t) - 2v_{L3}^i(t) \right] \qquad \text{(a)} \qquad (2.45)$$

$$\begin{bmatrix} i_1(t) \\ i_2(t) \\ i_3(t) \end{bmatrix} = Y_{T3}^1 \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} - Y_{T3}^1 \begin{bmatrix} 2v_{L1}^i(t) \\ 2v_{L2}^i(t) \\ 2v_{L3}^i(t) \end{bmatrix} = Y_{T3}^1 \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} + \begin{bmatrix} I_1(t) \\ I_2(t) \\ I_3(t) \end{bmatrix} \qquad \text{(b)}$$

$$\begin{bmatrix} v^i_{L1}(t+T) \\ v^i_{L2}(t+T) \\ v^i_{L2}(t+T) \end{bmatrix} = -Y^2_{T3} \begin{bmatrix} v_1(t)-2v^i_{L1}(t) \\ v_2(t)-2v^i_{L2}(t) \\ v_3(t)-2v^i_{L3}(t) \end{bmatrix} - \begin{bmatrix} v^i_{L1}(t) \\ v^i_{L2}(t) \\ v^i_{L3}(t) \end{bmatrix} \qquad \text{(c)}$$

where $\begin{bmatrix} i_{C1}(t) \\ i_{C2}(t) \\ i_{C3}(t) \end{bmatrix} = \begin{bmatrix} Y^1_{T3}(1,2)v_2(t)+Y^1_{T3}(1,3)v_3(t) \\ Y^1_{T3}(2,1)v_1(t)+Y^1_{T3}(2,3)v_3(t) \\ Y^1_{T3}(3,1)v_1(t)+Y^1_{T3}(3,2)v_2(t) \end{bmatrix}$

$$Y^1_{T3} = \frac{\begin{bmatrix} a_{12}^2Z_2+a_{13}^2Z_3 & -a_{12}a_{13}^2Z_3 & -a_{12}^2a_{13}Z_2 \\ -a_{12}a_{13}^2Z_3 & a_{12}^2a_{13}^2Z_3+a_{12}^2Z_1 & -a_{12}a_{13}Z_1 \\ -a_{12}^2a_{13}Z_2 & -a_{12}a_{13}Z_1 & a_{12}^2a_{13}^2Z_2+a_{13}^2Z_1 \end{bmatrix}}{a_{12}^2a_{13}^2Z_2Z_3+a_{12}a_{13}^2Z_1Z_3+a_{12}^2a_{13}Z_2Z_1}$$

$$Y^2_{T3} = \begin{bmatrix} R_{L1} & 0 & 0 \\ 0 & R_{L2} & 0 \\ 0 & 0 & R_{L3} \end{bmatrix} Y^1_{T3}$$

$$Y^3_{T3} = \frac{\begin{bmatrix} a_{12}^2a_{13}^2Z_2Z_3 & a_{12}a_{13}^2Z_1Z_3 & a_{12}^2a_{13}Z_2Z_1 \end{bmatrix}}{a_{12}^2a_{13}^2Z_2Z_3+a_{12}a_{13}^2Z_1Z_3+a_{12}^2a_{13}Z_2Z_1}$$

2.4 Conclusion

The TLM stub models of two basic branch, mutual inductors and transformers are all given out now. According to the different functions and physical properties, we can choose different models respectively. When the reactive components used to decouple the network, link models may be needed. Otherwise the stub models are the only models we need[14,16-18]. When the mutual inductors are tightly coupled, we should treat them as transformers. If a transformer is not coupled tightly, we can treat it as a mutual coupled inductors. Eqns. (2.19-21) shows that the TLM stub models equal to the discrete time model associated with trapezoidal rule. They give out the physical means of the modeling error. The TLM link models of capacitor and inductors are also given out in this chapter. From their characteristic equations, we know that the network can be decoupled into subnetworks by using TLM link model. But the choice of the simulation time step is still limited by the traveling time. The solution of this limitation will be given out in chapter 5. In the next chapter, the generalized TLM transform technique will be given out.

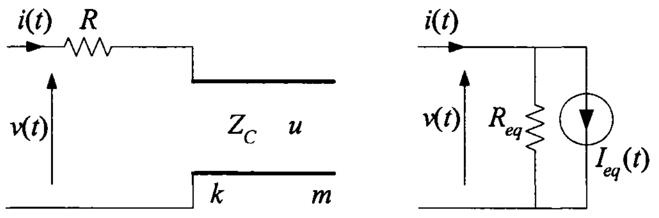
Chapter 3 The Generalized TLM Method

In TLM transform technique, the integral or derivative terms is transferred into two discrete equations, which are described by four coefficients. In this chapter, a new TLM transform formula will be described. In section 3.1, the new TLM transform technique is introduced. In section 3.2, the new TLM models of devices which were given out in the last chapter are given out by adopting the new TLM transform formula.

3.1 Generalized TLM Transform Technique

From the section 2.2, we know that TLM models are equivalent to ADC models associated with trapezoidal rule. As the paper[56] said that the trapezoidal rule has a numerical oscillation problem when current value flowing through an inductor has a high derivative value or the voltage over a capacitor has a high derivative value. In the SMPS, that is always the case because that is the fundamental theory behind the operation of SMPS. Therefore, this drawback of TLM method is really a pain when the SMPS is simulated by this method. As mentioned in this paper, the back Euler rule can depress the numerical oscillation. If TLM models of reactive components can be equivalent to the ADC model associated with back Euler rule, the numerical oscillation can also be depressed.

3.1.1 Integral Term



a) Open-circuit at terminal m b) Discrete-time model

Fig. 3.1 Open-circuit at terminal m

If we connect an open-circuit TL with a resistor (R) in series, the oscillation of the terminal voltage or current over the TL, due to the high derivative voltage value, can be depressed. Let discuss the circuit shown in Fig. 3.1, we can have the following equations.

$$i(t)=\frac{v(t)}{R_{eq}}-I_{eq}(t)=\frac{v(t)}{R_{eq}}-2v_C^i(t)\frac{1}{R_{eq}} \tag{3.1}$$

where $v_C^i(t)=\frac{Z_C}{R_{eq}}v(t-\tau_S)+\frac{R-Z_C}{R_{eq}}v_C^i(t-\tau_S)$

$R_{eq}=R+Z_C$

Combining the (3.1) at time t and t - τ_S, we have the following equation

$$\frac{1}{Z_C}\left(v(t)-v(t-\tau_S)\right)=\frac{R+Z_C}{Z_C}i(t)+\frac{Z_C-R}{Z_C}i(t-\tau_S) \tag{3.2}$$

As the description in section 2.2, a section of an open-circuit TL is the stub model of the capacitor (C), with characteristic impedance (Z_C = $\tau_S/2C$). The use of weight-averaged integration rule to discretize the V-I characteristic equation of a capacitor, we have

$$\frac{xi(t)+(2-x)i(t-\tau_S)}{2}\tau_S=C\left(v(t)-v(t-\tau_S)\right) \tag{3.3}$$

$$\frac{1}{Z_C}\left(v(t)-v(t-\tau_S)\right)=xi(t)+(2-x)i(t-\tau_S) \tag{3.4}$$

Table 3.1 The choice of R corresponding to the integration rule

R	**Weight Coefficient (x)**	**Integration rule**
$R=0$	1	Trapezoidal rule
$R=-Z_C$	0	Forward Euler rule
$R=Z_C$	2	Backward Euler rule

Comparing (3.2) - (3.4), with the changing of R, the weights of the history value and present value are changed accordingly. The value of R is located in the range [-Z_C, Z_C] Three typical cases are listed in Table 3.1.

So a capacitor's TLM model corresponding to the Backward Euler rule is shown as:

$$i(t) = \frac{v(t)}{R_{eq}} + I_{eq}(t) = \frac{v(t)}{R_{eq}} - 2v_C^i(t)\frac{1}{R_{eq}} \tag{3.5}$$

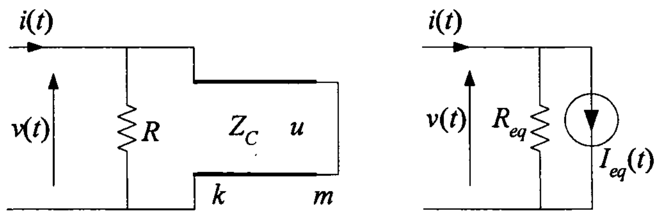
where $v_C^i(t) = \frac{1}{2}v(t - \tau_S)$

$$Z_C = \frac{\tau_S}{2C} \qquad R_{eq} = 2Z_C = \frac{\tau_S}{C}$$

Similar as the description in the section 2.3.2, the characteristic equation of a capacitor can be generalized into an integral term. Therefore, we have the following transform formula associated with Backward Euler rule for any integral term.

Integral term	**Generalized TLM transform**
$y(t) = \frac{1}{a}\int x(t)dt$	$x(t) = \frac{a}{\Delta t}y(t) - 2\frac{a}{\Delta t}y^i(t)$ $y^i(t) = \frac{1}{2}y(t - \Delta t)$

3.1.2 Derivative Term



a) Short-circuit at terminal *m* b) Discrete-time model

Fig. 3.2 Short-circuit at terminal m

Observing the type two branch, the resistor R_2 can limit the voltage over the inductor and then depress the numerical oscillation. Similarly as the discussion of integral term, by proper choosing the parallel connected resistor, the simulation accuracy and oscillation depression can be both achieved. Let discuss the type two branch with zero serial connected resistor.

$$i(t) = \frac{v(t)}{R_{eq}} - I_{eq}(t) = \frac{v(t)}{R_{eq}} - 2v_L^i(t)\frac{1}{Z_C} \tag{3.6}$$

where $v_L^i(t) = -v(t-\tau_S) + v_L^i(t-\tau_S)$

$$R_{eq} = \frac{RZ_C}{R+Z_C}$$

Combining the (3.6) at time t and t - τ_S, we have the following equation

$$Z_C\left(i(t) - i(t-\tau_S)\right) = \frac{R+Z_C}{R}v(t) + \frac{R-Z_C}{R}v(t-\tau_S) \tag{3.7}$$

As the description in section 2.2, a section of an short-circuit TL is the stub model of the inductor (L), with characteristic impedance ($Z_C = 2L/\ \tau_S$). The use of weight-averaged integration rule to discretize the V-I characteristic equation of a capacitor, we have

$$\frac{xv(t) + (2-x)v(t-\tau_S)}{2}\Delta t = L\left(i(t) - i(t-\tau_S)\right) \tag{3.8}$$

$$Z_C\left(i(t) - i(t-\tau_S)\right) = xv(t) + (2-x)v(t-\tau_S) \tag{3.9}$$

Table 3.2 The choice of R corresponding to the integration rule

R	Weight Coefficient (x)	Integration rule
R = inf	1	Trapezoidal rule
$R = -Z_C$	0	Forward Euler rule
$R = Z_C$	2	Backward Euler rule

Comparing (3.7) - (3.9), with the changing of R, the weights of the history value and present value are changed accordingly. The value of R falls in the range $(-\infty, -Z_C] \cup [Z_C, \infty)$.

So an inductor's TLM model corresponding to the Backward Euler rule is shown as:

$$i(t)=\frac{v(t)}{R_{eq}}+I_{eq}(t)=\frac{v(t)}{R_{eq}}-v_L^i(t)\frac{1}{R_{eq}} \tag{3.10}$$

where $v_L^i(t)=-v(t-\tau_S)+v_L^i(t-\tau_S)$

$$Z_C=\frac{2L}{\tau_S} \qquad R_{eq}=\frac{Z_C}{2}=\frac{L}{\tau_S}$$

Similar as the description in the section 2.3.2, the characteristic equation of an inductor can be generalized into an derivative term. Therefore, we have the following transform formula associated with Backward Euler rule for any derivative term.

Derivative term	**Generalized TLM transform**
$y(t)=a\frac{dx(t)}{dt}$	$y(t)=\frac{a}{\Delta t}x(t)+y^i(t)$ $y^i(t)=-(y(t-\Delta t)-y^i(t-\Delta t))$

In this section, the generalized TLM transforms of integral and derivative terms are given out. Like the conventional TLM transforms of integral and derivative terms, this new transforms can also used to model the two basic branch, mutual inductors, transformer, and so on.

3.2 Generalized TLM Models of Electric Components

In section 2.3, the old TLM models of some electric components are given out. In this section, the new TLM models of these components will be given out.

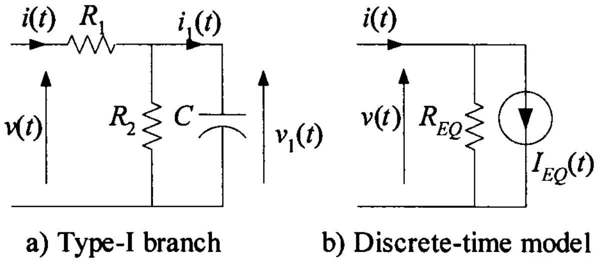


Fig. 3.3 The generalized TLM model of type one branch

3.2.1 Equivalent Branch Reduction

Applying (3.5) into the type one branch, which is shown in Fig. 3.3a, we have

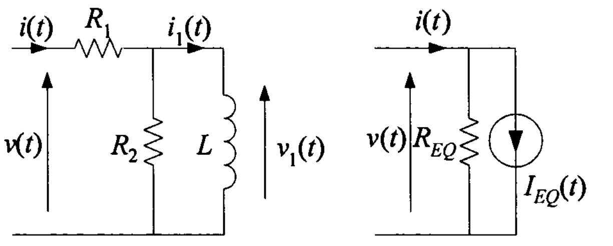
$$
\begin{aligned}
&v_1(t) = i_1(t)R_{eq} + 2v_C^i(t) && \text{(a)}\\
&v_C^i(t) = v_1(t-\Delta t)/2 && \text{(b)}\\
&v(t) = i(t)R_1 + v_1(t) && \text{(c)}\\
&i(t) = i_1(t) + v_1(t)/R_2 && \text{(d)}
\end{aligned}
\qquad (3.11)
$$

Rearranging (3.11), we have (3.12).

$$i(t) = \frac{v(t)}{R_{EQ}} + I_{EQ}(t) = \frac{v(t)}{R_{EQ}} - 2v_C^i(t)\frac{R_2}{R_2R_{eq} + R_1R_{eq} + R_1R_2} \qquad (3.12)$$

where $v_C^i(t) = \frac{1}{2}\frac{R_2R_{eq}}{R_2R_{eq} + R_1R_{eq} + R_1R_2}v(t-\Delta t) + \frac{R_1R_2}{R_2R_{eq} + R_1R_{eq} + R_1R_2}v_C^i(t-\Delta t)$

$$R_C = \frac{\Delta t}{2C} \qquad R_{eq} = 2R_C \qquad R_{EQ} = R_1 + \frac{R_2R_{eq}}{R_2 + R_{eq}}$$



a) Type-II branch b) Discrete-time model

Fig. 3.4 New TLM model of type two branch

Applying (3.10) into the type two branch, which is shown in Fig. 3.4a, we have (3.13).

$$v_1(t) = i_1(t)R_{eq} + v_L^i(t) \qquad \text{(a)} \qquad (3.13)$$

$$v_L^i(t) = -v_1(t-\Delta t) + v_L^i(t-\Delta t) \tag{b}$$

$$v(t) = i(t)R_1 + v_1(t) \tag{c}$$

$$i(t) = i_1(t) + v_1(t)/R_2 \tag{d}$$

Rearranging (3.13), we have (3.14).

$$i(t) = \frac{v(t)}{R_{EQ}} + I_{EQ}(t) = \frac{v(t)}{R_{EQ}} - v_L^i(t)\frac{R_2}{R_2R_{eq} + R_1R_{eq} + R_1R_2} \tag{3.14}$$

where $v_L^i(t) = -\frac{R_2R_{eq}}{R_2R_{eq} + R_1R_{eq} + R_1R_2}v(t-\Delta t) + \frac{R_1R_{eq} + R_2R_{eq}}{R_2R_{eq} + R_1R_{eq} + R_1R_2}v_L^i(t-\Delta t)$

$R_L = 2L/\Delta t \qquad R_{eq} = R_L/2 \qquad R_{EQ} = R_1 + R_2R_{eq}/(R_2 + R_{eq})$

Eqns. (3.12) and (3.14) are the new TLM models of two basic branches. The detailed derivation process are shown in Appendix D.

3.2.2 Generalized TLM Model of Mutual Coupled Inductors

Using the generalized TLM transform formula, we can obtain the generalized TLM models of mutual inductance. For the two-winding mutual coupled inductance, the system equations are shown as (3.15).

$$\begin{aligned} v_1(t) &= L_1\frac{di_1(t)}{dt} + M\frac{di_2(t)}{dt} + i_1(t)R_1 \\ v_2(t) &= M\frac{di_1(t)}{dt} + L_2\frac{di_2(t)}{dt} + i_2(t)R_2 \end{aligned} \tag{3.15}$$

After using the generalized TLM technique and rearrange the equations, we have

$$\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} = \begin{bmatrix} R_{L1} + R_1 & R_M \\ R_M & R_{L2} + R_2 \end{bmatrix} \begin{bmatrix} i_1(t) \\ i_2(t) \end{bmatrix} + \begin{bmatrix} (v_{L1}^i(t) + v_{M12}^i(t)) \\ v_{L2}^i(t) + v_{M21}^i(t) \end{bmatrix}$$

$$\begin{bmatrix} i_1(t) \\ i_2(t) \end{bmatrix} = Z_T^{-1} \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} - Z_T^{-1} \begin{bmatrix} v_{L1}^i(t) + v_{M12}^i(t) \\ v_{L2}^i(t) + v_{M21}^i(t) \end{bmatrix} \tag{3.16}$$

where $Z_T = \frac{1}{\Delta t}\begin{bmatrix} L_1 & M \\ M & L_2 \end{bmatrix} + \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} = \begin{bmatrix} R_{L1}+R_1 & R_M \\ R_M & R_{L2}+R_2 \end{bmatrix}$ $\quad Z_T^{-1} = \begin{bmatrix} Y_1 & Y_{12} \\ Y_{21} & Y_2 \end{bmatrix}$

$$\begin{bmatrix} v_{L1}^i(t) \\ v_{M12}^i(t) \\ v_{M21}^i(t) \\ v_{L2}^i(t) \end{bmatrix} = - \begin{bmatrix} R_{L1} & 0 \\ 0 & R_M \\ R_M & 0 \\ 0 & R_{L2} \end{bmatrix} Z_T^{-1} \begin{bmatrix} v_1(t-\Delta t) - v_{L1}^i(t-\Delta t) - v_{M12}^i(t-\Delta t) \\ v_2(t-\Delta t) - v_{L2}^i(t-\Delta t) - v_{M21}^i(t-\Delta t) \end{bmatrix}$$

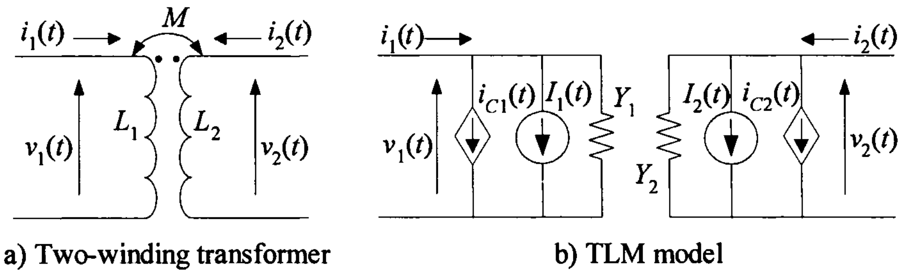


Fig. 3.5 TLM model of two winding mutual coupled inductance

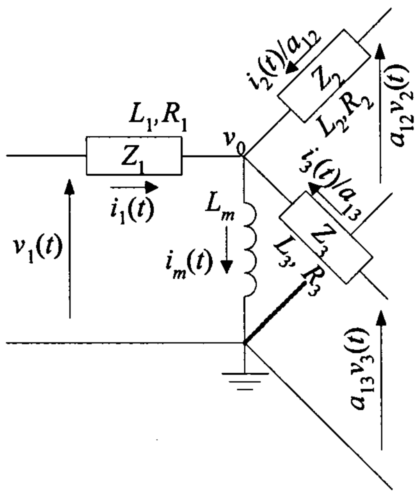
The TLM model of mutual inductance that is obtained from (3.16) is shown in Fig. 3.5 which is the same as Fig. 2.11. I_1, I_2 represent independent current sources, which are determined by the history value. i_{C1}, i_{C2} represent dependent current sources, which exist due to the mutual inductances. The expressions of these variables are shown as follow.

$$\begin{bmatrix} I_1(t) \\ I_2(t) \end{bmatrix} = -Z_T^{-1} \begin{bmatrix} v_{L1}^i(t) + v_{M12}^i(t) \\ v_{L2}^i(t) + v_{M21}^i(t) \end{bmatrix}$$

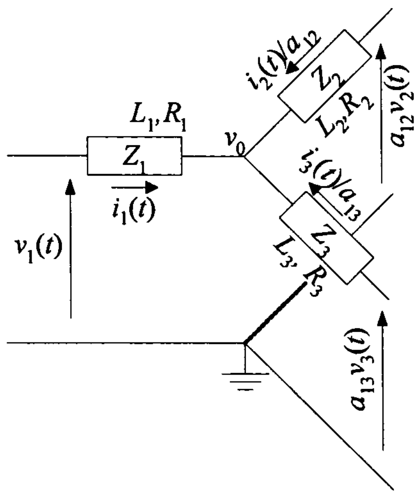
$$i_{C1}(t) = Y_{12} v_2(t) \qquad i_{C2}(t) = Y_{21} v_1(t)$$

3.2.3 Generalized TLM Model of A Three Winding Transformer

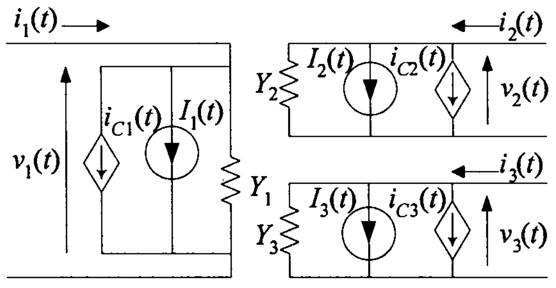
Fig. 3.6 is a model of three-winding transformer. Its characteristic equations are shown as (3.17). a_{12} is the turn ratio of the first winding and the second winding; a_{13} is the turn ratio of the first winding and the third winding.



a) Equivalent circuit



b) Simplified circuit



c) TLM model

Fig. 3.6 TLM model of three-winding transformer

$$v_1(t)-v_0(t)=L_1\frac{di_1(t)}{dt}+i_1(t)R_1 \qquad \text{(a)}$$

$$v_2(t)-\frac{v_0(t)}{a_{12}}=L_2\frac{di_2(t)}{dt}+i_2(t)R_2 \qquad \text{(b)}$$

$$v_3(t)-\frac{v_0(t)}{a_{13}}=L_3\frac{di_3(t)}{dt}+i_3(t)R_3 \qquad \text{(c)}$$

$$i_m(t)=i_1(t)+i_2(t)/a_{12}+i_3(t)/a_{13}=0 \qquad \text{(d)}$$

(3.17)

Adopting the generalized TLM technique to (3.17),

$$v_0(t)=Y^3_{T3}\left[v_1(t)-v^i_{L1}(t);\quad v_2(t)-v^i_{L2}(t);\quad v_3(t)-v^i_{L3}(t)\right] \qquad \text{(a)}$$

$$\begin{bmatrix} i_1(t)\\ i_2(t)\\ i_3(t)\end{bmatrix}=Y^1_{T3}\begin{bmatrix} v_1(t)\\ v_2(t)\\ v_3(t)\end{bmatrix}-Y^1_{T3}\begin{bmatrix} v^i_{L1}(t)\\ v^i_{L2}(t)\\ v^i_{L3}(t)\end{bmatrix}=Y^1_{T3}\begin{bmatrix} v_1(t)\\ v_2(t)\\ v_3(t)\end{bmatrix}+\begin{bmatrix} I_1(t)\\ I_2(t)\\ I_3(t)\end{bmatrix}$$

$$\begin{bmatrix} i_{C1}(t)\\ i_{C2}(t)\\ i_{C3}(t)\end{bmatrix}=\begin{bmatrix} Y^1_{T3}(1,2)v_2(t)+Y^1_{T3}(1,3)v_3(t)\\ Y^1_{T3}(2,1)v_1(t)+Y^1_{T3}(2,3)v_3(t)\\ Y^1_{T3}(3,1)v_1(t)+Y^1_{T3}(3,2)v_2(t)\end{bmatrix} \qquad \text{(b)} \qquad (3.18)$$

$$\begin{bmatrix} v^i_{L1}(t+T)\\ v^i_{L2}(t+T)\\ v^i_{L2}(t+T)\end{bmatrix}=-Y^2_{T3}\begin{bmatrix} v_1(t)-v^i_{L1}(t)\\ v_2(t)-v^i_{L2}(t)\\ v_3(t)-v^i_{L3}(t)\end{bmatrix}-\begin{bmatrix} v^i_{L1}(t)\\ v^i_{L2}(t)\\ v^i_{L3}(t)\end{bmatrix} \qquad \text{(c)}$$

where $$Y^3_{T3}=\frac{\begin{bmatrix} a_{12}^2a_{13}^2Z_2Z_3 & a_{12}a_{13}^2Z_1Z_3 & a_{12}^2a_{13}Z_2Z_1\end{bmatrix}}{a_{12}^2a_{13}^2Z_2Z_3+a_{12}a_{13}^2Z_1Z_3+a_{12}^2a_{13}Z_2Z_1}$$

$$Y^1_{T3}=\frac{\begin{bmatrix} a_{12}^2Z_2+a_{13}^2Z_3 & -a_{12}a_{13}^2Z_3 & -a_{12}^2a_{13}Z_2\\ -a_{12}a_{13}^2Z_3 & a_{12}^2a_{13}^2Z_3+a_{12}^2Z_1 & -a_{12}a_{13}Z_1\\ -a_{12}^2a_{13}Z_2 & -a_{12}a_{13}Z_1 & a_{12}^2a_{13}^2Z_2+a_{13}^2Z_1\end{bmatrix}}{a_{12}^2a_{13}^2Z_2Z_3+a_{12}a_{13}^2Z_1Z_3+a_{12}^2a_{13}Z_2Z_1}$$

$$Y^2_{T3}=\begin{bmatrix} R_{L1} & 0 & 0\\ 0 & R_{L2} & 0\\ 0 & 0 & R_{L3}\end{bmatrix}Y^1_{T3}$$

$$\begin{bmatrix} Z_1\\ Z_2\\ Z_3\end{bmatrix}=\begin{bmatrix} R_{L1}+R_1\\ R_{L2}+R_2\\ R_{L3}+R_3\end{bmatrix}$$

3.3 Conclusion

The generalized TLM stub models of two basic branch, mutual inductors and transformers are all given out in this chapter. According to simulation condition, we can choose the models that were given out in chapter 2 or the model that are given out in this chapter. When there is a high voltage/current derivative in the circuit, the new models give out more satisfied simulation results. Otherwise, the models that were given out in chapter 2 are preferred. In the next chapter, the switch model will be given out.

Chapter 4 Switch Modeling and Its Problem

A challenging problem of the SMPS system simulation is the change of the circuit topologies, which is caused by high frequency switching. Various kinds of switch models can be chosen to model the switching circuits. In some general-purpose circuit analysis software packages, such as some SPICE-based software, they adopt detailed micromodeling of semiconductor devices to model the power electronic circuit. It is unnecessary to do that for studying SMPS circuits at system level, in which the semiconductor device always operates in the switch modes. The inclusion of detailed models for semiconductor devises together with other devices in the circuit just increase the complexity of the system, and computation time. Recently some different kinds of switch models are developed for the simulation of SMPS. Some models can keep the system matrix constant to reduce the computation time[9,11, 20, 21, 23]. Some models try to minimize the effects of the switch changes[27]. Those efforts really can improve the simulation efficiency. Except the modeling problem, there are still two main problem related to the switch circuit simulation, one is the switching time detection, the other is the system reinitialization technique. I will discuss these problems one by one in this chapter..

4.1 New Switch Models

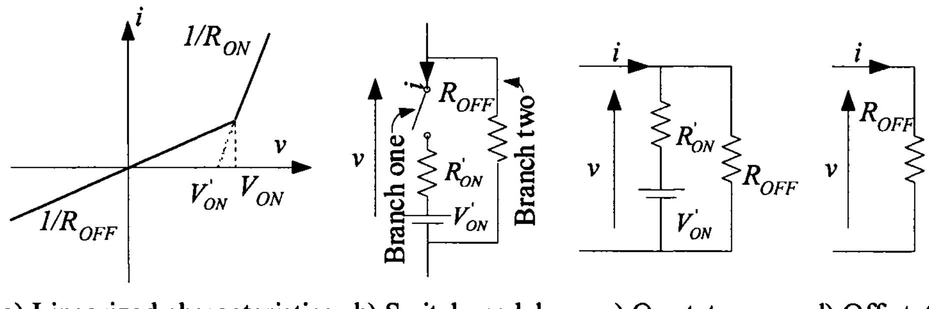
The TLM switch model proposed by Hui and Christopoulos is a TLM stub model. When the switch is on, the switch is modeled as an inductor (L); when the switch is off, the switch is modeled as a capacitor (C). In order to have a constant system matrix, the L and C satisfied the equation $2L/\Delta t = \Delta t/2C$. This models looks beautiful, and solve the difficulty of the SMPS circuit simulation. When we investigate the model deeply, we can find this model cannot give out the satisfied results when there is one or more state variables with high derivative values[21], even cannot give out satisfied off-state current value of the switch. In order to overcome this limitation, the paper **[21]** just uses it to model the switch with snubber branch to avoid the high derivative values.

In paper [27], semiconductor switches are modeled as binary inductors, as a very low value of inductance (determined by the rating of the semiconductor) during on state, and infinite value of inductance otherwise. Then the system is partitioned into two parts, one is the switch part, which includes all switches, and the other is the circuit left. So the changes in topology are taken into consideration by the least amount of computation. If the backward Euler method of integration is used for the solution of system equations, the limitation of high derivative value of switch current at switching time can be overcome. If the switch is just modeled as a resistance with different values under different states, the derivative value of switch current will not show up in system equations. Combined with aforementioned switch models, the two-branch model of the semiconductor switch is shown as follow:

Branch one: $i = S\dfrac{v - V'_{ON}}{R'_{ON}}$ $\quad S = \begin{cases} 1 & switch\ \ on \\ 0 & switch\ \ off \end{cases}$

Branch two: $i = \dfrac{v}{R_{OFF}}$

where $R'_{ON} = \dfrac{R_{ON}R_{OFF}}{R_{OFF} - R_{ON}}$ $\quad V'_{ON} = \dfrac{R_{OFF}}{R_{OFF} + R_{ON}}V_{ON}$



a) Linearized characteristics b) Switch model c) On state d) Off state

Fig. 4.1 New switch model

In the Fig. 4.1, R_{OFF} represents the off-state resistance of the switch; the combination of R_{ON} and R_{OFF} represents the on-state resistance of the switch; the DC voltage source represent the voltage source $V_{ON}^{'}$ in the Fig. 4.1a). From the Fig. 4.1 c) and d), we can find no matter what the switch state is, there is always an off-state resistance connected in the circuit. So the structure of the circuit will not change due to switch state changing, and the system matrix will not be ill conditioned due to the floating subcircuit.

4.2 Switch Problems

Basically the new switch model is an idealized switch model, which equals to a two-valued resistor together with a two-valued current source. The transition between these two modes of operation is assumed instantaneous and is triggered by appropriate switching control signals, which can be an internal control or external control signals. Thus, the idealized switch model is discontinuous by nature. Its effect on the simulation has three very distinct aspects, if fixed time-step simulation is adopted [31].

1). Switching Delay

Switching event rarely coincides with the simulation time points. Thus, the discontinuous behaviour of the switch is asynchronous with regards to the simulation time step. When the switch is allowed to change state only at the fixed calculation time step, it introduces a lag between the actual switching time and the simulated switching time, and a false forced commutation in the simulation results. Furthermore, if the circuit variables are not renewed with updated switched states as soon as the switching conditions have been detected, the time delay may be as large as two time steps.

2). Initial Condition

If there is a true forced commutation in the system, at least one current or voltage discontinuity is involved since the operating point slides form one branch to the other branch of the V-I characteristic passing through the break point P_{th}. A voltage and/or current value is discontinuous, which means the value at time $t+$ is not equal to the value at time t^- at switching time. If the initial condition after the switching action cannot be calculated correctly, the error will show up in the simulation. It cannot just cause a long lasting error, but also can causes an error system state.

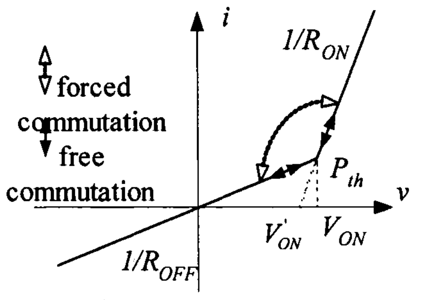


Fig. 4.2 Resistive model for switches

3) Simultaneous Switching

Because the transition between two modes of switch is assumed instantaneous, the opening or closing of a switch may cause a series of other switches to open or close, in a chain reaction, especially a forced commutation happened. During the procedure of determining new switch state, the system passes through a series of "mythical modes" until, hopefully, it arrives at a new mode that permits "divergence of time," meaning that the simulation can continue and does not get stuck.[31]

The aforementioned three aspects are all related to the switches' commutation. In the case of free commutation, we must guarantee zero current while the switching event happens. In the case of forced commutation, we must find the correct system state before and after one or more switches have been commanded to change state while non-zero current are sustained. The efficiency of simulation of SMPS networks seriously depends on the capability of solving these three problems. In the following section of this chapter, I will discuss some solutions.

4.3 Switching Time Detection [38,39,50]

One way to reduce switching delay and obtain more accurate switching time is to reduce the time step. However, this will also increase the computation time proportionately, and still may not give good enough results. The more precise switching time detection can be achieved by using a variable time step solution, where if a switching event is detected, the program will sub-divide the time step into smaller intervals, but it will increase the computation burden. However, this does not circumvent the problem of spurious voltage and current spikes, due to current and voltage differentials when switching inductive and capacitive circuits.[39]

Table 4.1 Operating conditions for various switching elements

Switching Element	Conditions for 'ON' state
Diode	$v_{sw} > V_{ON}$
Transistor (BJT,MOSFET,IGBT)	$v_{sw} > V_{ON}$, and $v_{gate} > 0$
Thyristor (SCR)	$v_{sw} \geq 0$, if previously 'ON'; $v_{sw} \geq 0$, and $v_{gate} > 0$

We all know that the computation speed of fixed time-step simulation is very fast. If we can obtain the acceptably accurate switching time by using fixed time-step algorithm, the usage of variable time-step for switching time detection will become unnecessary. When we choose the simulation time-step, we have a criterion that the chosen time-step should be far smaller than the system dynamic time constant that we are concerning. That means the system variable trajectory can be fitted by a certain type of curve between two adjacent time points. So the interpolation/extrapolation techniques can be used to determine the switching time. When we use interpolation/extrapolation technique to estimate the switching time, we must remember one thing. No matter what kind of commutation happens, the signal that is interpolated/extrapolated is always a control signal. For internal control switch, the control signal is its own current or voltage signal; for external control switch, the control signal is the input data of its control logic.

The operating conditions for different semiconductors are listed in Table 4.1.

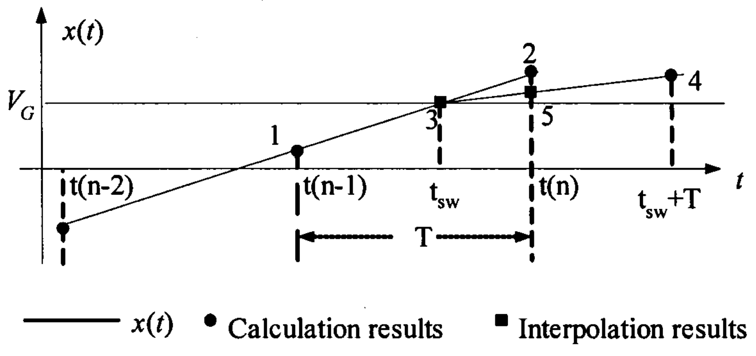


Fig. 4.3 Switching time detection

The Fig. 4.3 shows the algorithm of switching time detection, which is called double interpolation technique that has been used by PSCAD/EMTDC[50]. Assume $x(t(k)), k=1,2,3,4,5$ represent control signal of the switch at time point k; V_G represents the threshold value of the control signal. With the assumption in mind, when the switching occurs at time t_{sw} between $t(n\text{-}1)$ and $t(n)$, the switching time can be obtained by (4.1), and the state value at point 3 can be accurately obtained by using linear interpolation by (4.2). By using the estimated state variables and system equations for time-step (Δt), we can obtain state variables at point 4. Through another linear interpolation, we obtain the state variables at point 5. Then the system state can be updated at the regular simulation time point, and resume the simulation after discontinuity.

$$t_{sw} = \Delta t \frac{V_G - x(t(n-1))}{x(t(n)) - x(t(n-1))} \tag{4.1}$$

$$x(t_{sw}) = x(t(n-1)) + \frac{x(t(n)) - x(t(n-1))}{\Delta t} t_{sw} \tag{4.2}$$

Eqns. (4.1) and (4.2) are the formula for linear interpolation. The equations of extrapolation are (4.3) and (4.4)

$$t_{sw} = \frac{V_G - x(t(n-1))}{x(t(n-1)) - x(t(n-2))} \Delta t \tag{4.3}$$

$$x(t_{sw}) = x(t(n-1)) + \frac{x(t(n-1)) - x(t(n-2))}{\Delta t} t_{sw} \tag{4.4}$$

If multiple switching actions happen in one time step, the only thing we need to do is to repeat those step that adopts interpolation technique until reach the normal simulation time point[39]. Because the extrapolation technique requires two history values, it cannot be used to solve the multiple-switching case. Under free commutation condition, there are no peak variations of voltage/current or Dirac impulse in the circuit. Therefore, we do not need try to get the initial conditions after the switching, and the value that is obtained by (4.2) or (4.4) can be used to restart the simulation procedure[42]. In the case of forced commutation, the switching time still can be obtained by this method. But we cannot use the interpolated system variable generated by (4.2) or (4.4) to do the simulation continuously due to the existence of the voltage/current discontinuities[36]. In the next section, I will deal with the case, in which forced commutation happened.

4.4 Initial Condition After Forced Commutation

Through a forced switching, one or more system variables are discontinuous. In order to restart trapezoidal integration, two different kind of technique can be adopted. First kind of technique tries to keep the structure of the circuit constant and use different discrete-time integration technique to restart the simulation. This kind of technique has a common step that is to use backward Euler technique to start the first step of calculation, because the backward Euler technique does not need history values of discontinuous state variables. It is assumed from theoretical considerations that the inductor currents and capacitor voltages cannot jump between t^+ and t^-. After that some methods continue to simulate forward and go back to the normal time mesh; some methods use negative time-step to return to the last switching time t^+ and then restart the simulation[41]. Second kind of technique is to transfer the system into a DC networks, and use Compensation Theorem to find the correct switch state after any forced commutation [36]. This method is also base on the same theoretical considerations as the first kind of technique. The following section will introduce the second technique.

4.4.1 The Compensation Theorem[52]

The compensation Theorem is direct consequence of superposition and permits easier determination of some voltage or current variable. In this theorem, a subnetwork is replaced by an ideal voltage source whose generated emf at any instant is equal to the instantaneous potential difference across the terminal of this subnetwork or an ideal current source whose generated current at any instant is equal to the instantaneous current flow through the terminal of this subnetwork. A general situation is depicted in Fig. 4.4, where a two-terminal subnetwork B is shown separated from the rest of the network. Assuming that we have already known the current i and the voltage v of subnetwork B, it would be possible to place a voltage source $v_g = v$ in parallel with subnetwork B without affecting any other system variable, since the voltage across this terminal is not changed. But now subnetwork B can be removed entirely, since subnetwork A is not affected by this remove. The result, shown in Fig. 4.4b, is to replace a two-terminal subnetwork by a voltage source whose voltage is equal to the already calculated voltage of the subnetwork.

A similar discussion applies that involves the already determined current in subnetwork B. The only difference is that the current source $i_g = i$ is in series with the subnetwork B. So the subnetwork A just can see the current source, and cannot see the subnetwork B. So the subnetwork B can be removed. The result, shown in Fig. 4.4c, is to replace a one-terminal subnetwork by a current source whose current is the previously calculated value of the current in the subnetwork. From the above description, we can notice that, in order to apply the theorem, it is first necessary to know the voltage or current of a one-terminal subnetwork.

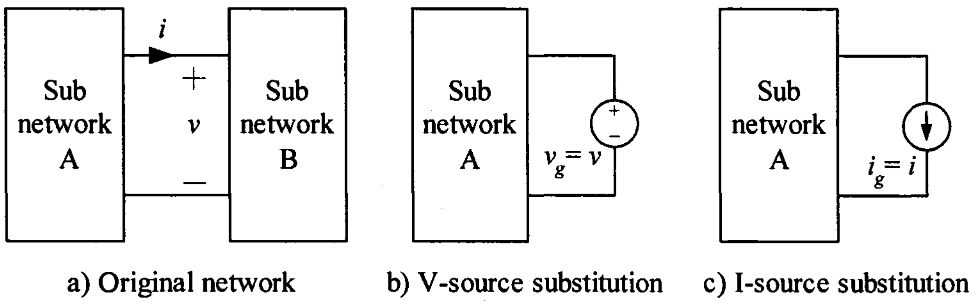
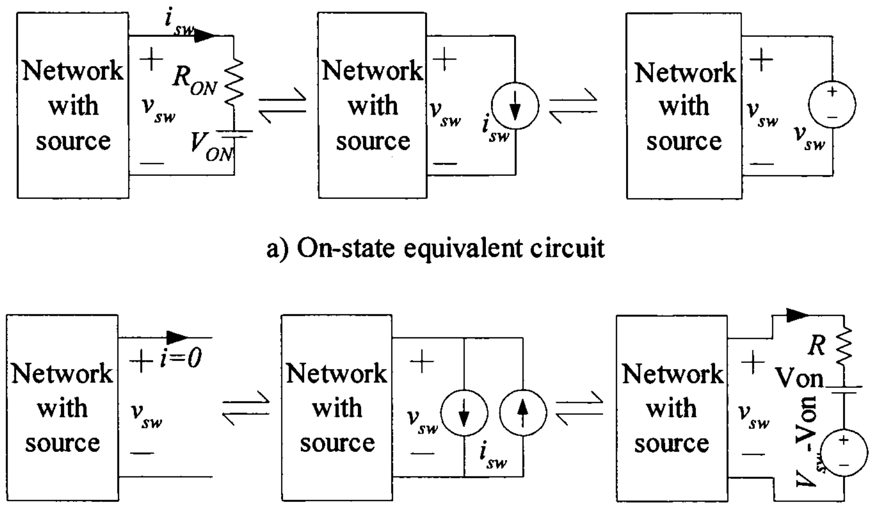


Fig. 4.4 The compensation theorem

4.4.2 Modification of Femia's Technique

To Use superposition and compensation theorem, we can easily evaluate the effects of a branch resistance change upon all system variable of any LTI network. Considering the simplified V-I characteristic of a switch, the state change means resistance change. By using this technique, the initial condition after switching can be easily obtained. Thinking about the new switch model I proposed, the resistance goes to infinite, when the switch is off. That means the technique used by N.Femia[36] failed when dealing with new model. But if we use another type of compensation source (current source), the compensation theorem becomes useful again.



a) On-state equivalent circuit

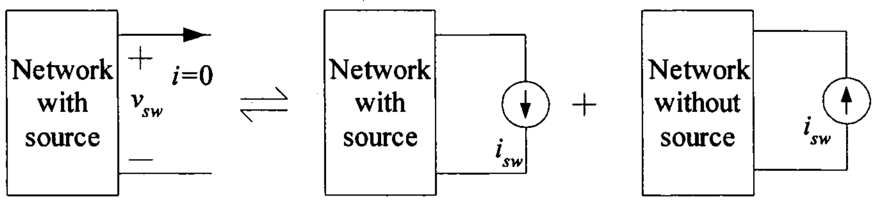
b) Off-state equivalent circuit

Fig. 4.5 Equivalent circuit

The basic assumption for compensation source technique is that the inductor currents and capacitor voltages cannot jump between t^+ and t^-[36]. At the time of switching, the inductor can be treated as a DC current source and the capacitor can be treated as a DC voltage source[31]. Then the whole system turns into a DC circuit at the switching time. That brings an added advantage – the switching time decoupling, to the system simulation, which will be shown in chapter 6. According to compensation theorem, the equivalent networks when the switch is on or off is shown in Fig. 4.5.

Let us discuss on-off forced commutation of the switch at time t first. The changing procedure is depicted in Fig. 4.6. The network comprises LTI resistors and ideal independent current and voltage dc sources. As shown in Fig. 4.6a, the current output of network is zero, when the switch is off. A zero current output could be represented as two identical current sources with opposite directions. Using superposition, the Fig. 4.6a become the sum of Fig. 4.6b and Fig. 4.6c. If we let i_{sw} equal to the switch current at time t^-, Fig. 4.6b is equal to the system when switch is on at time t^-. All system variables ($x(t^-)$) have already known. The only thing we need to do is to solve the auxiliary circuit, which is excited only by one current source. When the auxiliary circuit is solved (x_a), the initial condition at time t^+ is

$$x\left(t^+\right)=x\left(t^-\right)+x\left(a\right) \tag{4.5}$$



a) Off-state b) On-state c) Auxiliary circuit

Fig. 4.6 Switch state transition from on to off

Now let us discuss state transition from off to on. The changing procedure is depicted in Fig. 4.7. First look at Fig. 4.7b, the voltage over two voltage source is equal to the terminal voltage, so the terminal current is zero. It represents off state of switch. Fig. 4.7a represents the system when the switch turns on. According to superposition, the circuit in Fig. 4.7a equals to the sum of Fig. 4.7b and Fig. 4.7c. If we let V_{sw} equal to the switch voltage at time t^- (switching happens at time t), Fig. 4.7b is equal to the system when switch is off at time t. All system variables have already known. The only thing we need to do is to solve the auxiliary circuit, which is excited only by one voltage source. If we use nodal analysis method to solve the auxiliary circuit, we need to change the Norton equivalent branch into Thevenin equivalent branch. If we use mesh analysis method, we just leave Norton equivalent branch alone.

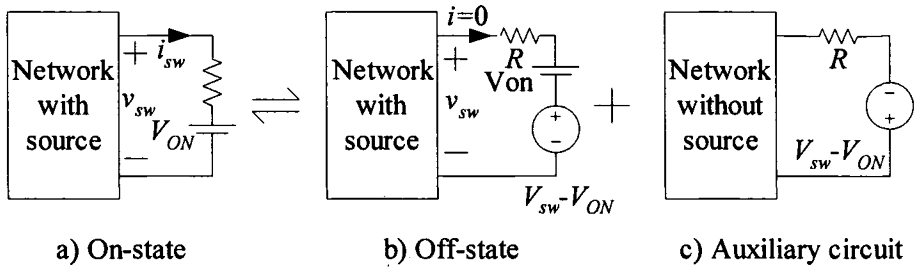


Fig. 4.7 Switch state transition from off to on

There is an advantage of transferring the system into a DC network. Due to the forced commutation, normally simultaneous switching will happen in the system. During the procedure of finding the correct system switch state, the problem for simultaneous multiple switching is also solved.

4.5 The Search of the Consistent Switches State

From the last section, we know if we obtain the system state of auxiliary circuit, we can easily obtain the initial condition after forced commutation, and get the consistent switch state(CSS) at the same time. For solving auxiliary circuit, I choose modified nodal analysis method. The system equation express as (4.6).

$$V_a = Y_{ini}^{-1} I_a \tag{4.6}$$

I_a is vector of currents flowing into the switch caused by compensation source, which is determined by the transition direction of individual switches;

V_a is vector of the voltages of auxiliary circuit;

Y_{ini} is DC net system admittance matrix changing with the state of switching devices.

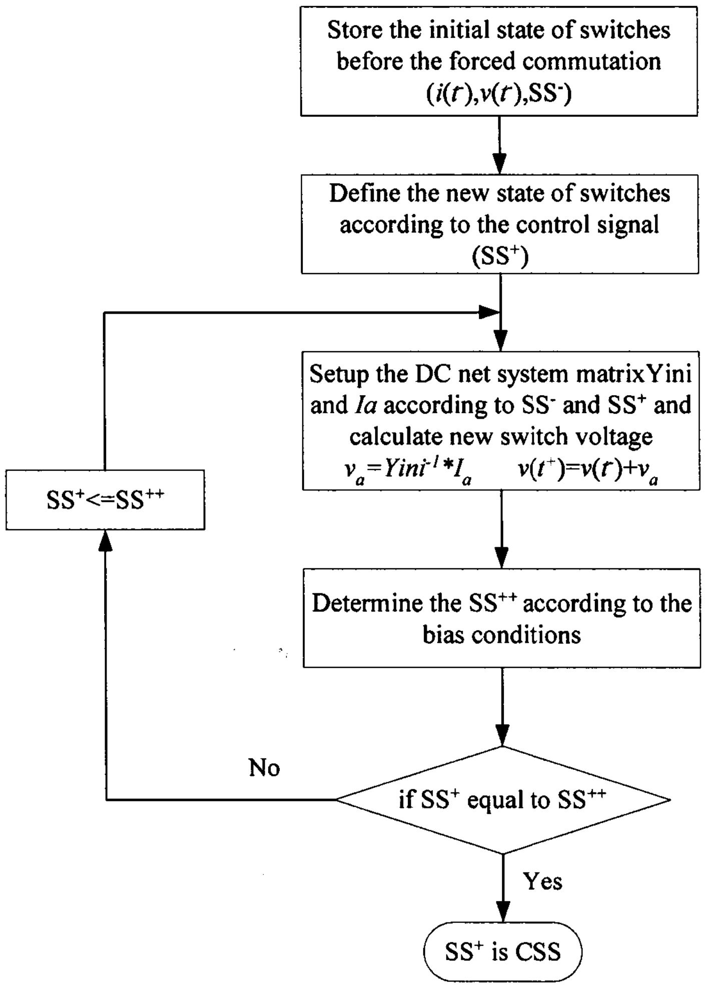

Store the initial state of switches before the forced commutation ($i(t^-)$,$v(t^-)$,SS$^-$)
Define the new state of switches according to the control signal (SS$^+$)
Setup the DC net system matrixYini and *Ia* according to SS$^-$ and SS$^+$ and calculate new switch voltage v_a=$Yini^{-1}$*I_a $v(t^+)$=$v(t^-)$+v_a
SS$^+$<=SS^{++}
Determine the SS^{++} according to the bias conditions
No
if SS$^+$ equal to SS^{++}
Yes
SS$^+$ is CSS

Fig. 4.8 Flow chart of CSS search algorithm

Fig. 4.8 is the CSS search algorithm. At each step the Y_{ini} is just determined by the trial switch state. The value of I_a is determined by both switch state at time t^- and the trial switch state. For example, when a switch transits from on to off, the compensation current is the switch current at time t^-; when a switch transits from off to on, the compensation current is decided by the switch voltage at time t^- and switch thresh hold voltage. All discontinuities of voltages and currents caused by the synchronous change of switch state should be involved in this current source vector. According to (4.6), the updated bias conditions can be check again. If all bias conditions are fulfilled. The CSS has been found. Otherwise, this step should repeat again according to the new trial switch state. The main advantage of this algorithm is that no prior knowledge of the circuit is required.

4.6 Conclusion

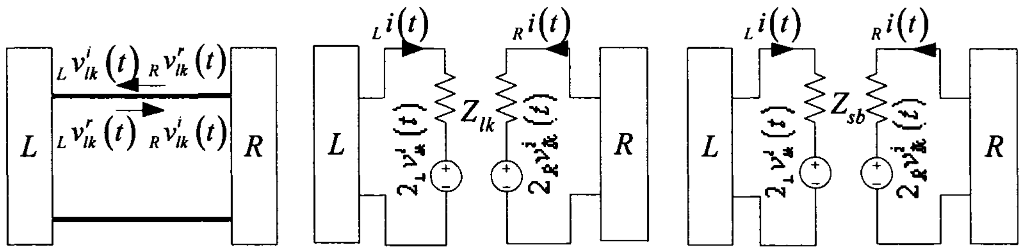
This chapter gives out the switch model and shows that problems of free commutations and forced commutations of switching can be solved by the same type of switch model. The only difference between forced commutations and free commutations is whether we need to get the initial condition after switching. In the next chapter, I will discuss how to use TLM stub model to decouple the system.

Chapter 5 System Decoupling Technique

In the last chapter, the switch model being used in this thesis has been given out. Accompany with the CT technique, the initial condition after the forced commutation can be easily obtained by solving a series of DC networks. In this chapter, the system decoupling technique is proposed based on TLM stub model.

5.1 TLM Link Model Decoupling Technique

Thinking about a network, which consists of two subnetworks – ***L*** and ***R***, which are connected by a capacitor. By using the capacitor's TLM link model, the whole network is decoupled into two subnetworks, which is shown as Fig. 5.1.



a) TL Linked network b) Decoupled network c) Decoupled network

Fig. 5.1 Link-stub model decoupled network

In the procedure of proving (2.27) and (2.28), we use (2.31) and (2.32), which are rewritten in (5.1)

$$\begin{aligned} {}_L v^i(t) &= {}_R v(t-\Delta t) - {}_R v^i(t-\Delta t) \\ {}_R v^i(t) &= {}_L v(t-\Delta t) - {}_L v^i(t-\Delta t) \end{aligned} \tag{5.1}$$

Combining equivalent circuits of link models and these equations, we find that the equivalent circuit of left ending can be determined independently from the L-subnetwork; the equivalent circuit of right ending can be determined independently from the R-subnetwork. Till now, the whole network has decoupled as two subnetworks and can be simulated at the same time, which increases the simulation efficiency. As we know, the switching frequency of diode rectifier part of the SMPS is very low and the voltage across the DC link capacitors will not change much within a few microseconds, so the time step for simulating that part of networks does not need to be nanoseconds; the switching frequency of DC-DC converter is very high, in order to let simulation results show the switching waveforms correctly, the time step for simulating that part of networks need to be hundreds nanoseconds. During the simulation procedure, two subnetworks need exchange energy every time step[18]. So the final simulation time step is the smaller time step, which is unnecessary for the low frequency subnetworks. In order to improve efficiency of the simulation, we need to make some change to allow us choose time step separately.

The advantage of link model over the stub model is that the link model is a two-port device, which makes the network decoupling possible. But it becomes a disadvantage of it when we try to improve efficiency further, due to the need of energy exchange. The disadvantage of stub model is a single-port device, which cannot decouple the networks. But after decoupling, this disadvantage becomes its advantage over link model, because it does not need energy exchange and makes the subnetworks independent networks. By combining these two models, we can choose the time step separately according to the different dynamic property of different subnetworks. Δt_L and Δt_R represent the time step for L-subnetwork and R-subnetwork respectively. Δt_L is integer times of Δt_R. The basic simulation procedure is shown as follow:

1) Determine the incident pulse value for link models w.r.t. Δt_L according to the calculation results of the last step, and calculate the terminal voltages and currents of both endings.

2) Determine the reflected pulse of link models w.r.t. Δt_L. The reflected pulse is equal to the difference between terminal voltage and the incident pulse. According to (5.1), the reflected

pulse take one time step (Δt_L) to reach another terminal to become the incident pulse, so the simulation can do continuously. But there are two shortcomings in this stub-link decoupling technique. First, it introduces an error into the simulation procedure -- the voltage over the reactive component has two different values at two terminals when we use link models. This generates fluctuations at the two terminals of the TLM link model [11]. In order to depressing this side effect, an improved TLM link model was proposed in **[18]**. It uses the average voltage value of the two ending instead of the individual voltage value in (5.1). The adjusted equations are shown in (5.2). Second, The voltages over the decoupling capacitors are assumed to be constant between two simulation time points. It will limit the choice of the decoupling components. I will introduce voltage evaluation algorithm, which allows us to adjust the voltages over the decoupling capacitor according to the system simulation results.

$$\begin{aligned}
&v(t-T)=\left({}_{L}v(t-T)+{}_{R}v(t-T)\right)/2\\
&{}_{L}v^{i}(t)=v(t-T)-{}_{R}v^{i}(t-T)\\
&{}_{R}v^{i}(t)=v(t-T)-{}_{L}v^{i}(t-T)
\end{aligned}\tag{5.2}$$

3) Determine the reflected pulse of stub models w.r.t. Δt_R. This step needs the conversion between link model and stub model. We can choose different time step for link model and stub model to increase the simulation efficiency. Providing two models are all equal to a same device, its terminal variables (v and i) should be equal. So the conversion equations are,

$$v^{i}_{(sb)}=i\left(Z_{(lk)}-Z_{(sb)}\right)/2+v^{i}_{(lk)} \qquad v^{i}_{(lk)}=i\left(Z_{(sb)}-Z_{(lk)}\right)/2+v^{i}_{(sb)} \tag{5.3}$$

From the incident pulse for link model, we can obtain the reflected pulse for stub model, which is the incident pulse for the next time step. The detail description is in **[11]**.

4) Use the incident pulse of stub models w.r.t. Δt_R to solve the subnetworks. After obtaining terminal voltage, we can find difference between terminal voltages and incident pulse,

which is the reflected pulse. According to (2.26), this reflected pulse is the incident pulse for the next time step. Continue to solve the network till the time for the next energy exchange. Go to 1)

By doing step 1 to 4, the high frequency part and low frequency part of the network can be decoupled and simulate parallel. Only at the predetermined simulation time point, the adjacent subnetworks need to do the voltage adjustment.

5.2 New Proposed TLM Stub Decoupling Technique

Eqns (5.2) indicate that we need to know ${}_{L}v$ and ${}_{R}v$ before v. Let's assume that ${}_{L}v(t)$, ${}_{R}v(t)$, ${}_{L}i(t)$, ${}_{R}i(t)$, $v(t)$, and $i(t)$ represent the state value of the decoupling capacitor where a TLM stub model is chosen. If we choose different time step for two subnetworks, for any time which is exactly coincident with the L-subnetwork time grid, the average capacitor voltage can be obtain by the exact simulation results; for any other time, not all system states of L-subnetwork and R-subnetwork can be obtained simultaneously, so the estimation values are need for those time points.

5.2.1 Voltage Changes of the Decoupling Capacitor

The V-I relationship at time t_0 and $t_0+\Delta t$ at L-terminal satisfy the following equations.

$$\begin{aligned} &{}_{L}i(t_0)Z_{(sb)} + 2\,{}_{L}v^{i}(t_0) = {}_{L}v(t_0) && \text{(a)} \\ &{}_{L}i(t_0+\Delta t)Z_{(sb)} + 2\,{}_{L}v^{i}(t_0+\Delta t) = {}_{L}v(t_0+\Delta t) && \text{(b)} \end{aligned} \qquad (5.4)$$

Using (2.26) into (5.4), we can find the sum of these two equations is the voltage change of capacitor at the L-terminal between time t and $t_0+\Delta t$.

$$\Delta_{L}v(t_0+\Delta t) = {}_{L}v(t_0+\Delta t) - {}_{L}v(t_0) = \left({}_{L}i(t_0+\Delta t) + {}_{L}i(t_0)\right)Z_{(sb)} \qquad (5.5)$$

Similarly, for the R-terminal we have the following equation.

$$\Delta_R v(t_0+\Delta t) = {}_R v(t_0+\Delta t) - {}_R v(t_0) = \left({}_R i(t_0+\Delta t) + {}_R i(t_0)\right) Z_{(sb)} \tag{5.6}$$

From the basic V-I relationship of the capacitor and adopting trapezoidal rule, we have (5.7). It means the voltage change of the capacitor is the sum of the voltage change at the both terminals of the stub model.

$$\begin{aligned} \Delta v(t_0+\Delta t) &= \frac{\left(i(t_0+\Delta t)+i(t_0)\right)\Delta t}{2C} \\ &= \left({}_L i(t_0+\Delta t) + {}_L i(t_0) + {}_R i(t_0+\Delta t) + {}_R i(t_0)\right) Z_{(sb)} \\ &= \Delta_L v(t_0+\Delta t) + \Delta_R v(t_0+\Delta t) \end{aligned} \tag{5.7}$$

$$\begin{aligned} v(t_0+\Delta t) &= \Delta_L v(t_0+\Delta t) + \Delta_R v(t_0+\Delta t) + v(t_0+\Delta t) \\ &= \Delta_L v(t_0+\Delta t) + {}_R v(t_0+\Delta t) = \Delta_R v(t_0+\Delta t) + {}_L v(t_0+\Delta t) \end{aligned} \tag{5.8}$$

From (5.7) we know the voltage adjustment at each energy exchanging time can be done by just using TLM stub models, and the new voltage can be obtained according to (5.8). Therefore, the TLM sub-link conversion becomes unnecessary and only the TLM stub model is needed to do the system decoupling.

5.2.2 Voltage Evaluation Method

Like it was said before, when the time steps, Δt_L and Δt_R, are different, the evaluation of the decoupling capacitor voltages in the time interval $[t_0, t_0+\Delta t_L]$ should be done. All evaluation method are based on an assumption that the capacitor is large enough and the time step is small enough that the voltage curve can be fitted in one time interval by low order polynomial [22]. There are three algorithms can be used to do the capacitor voltage estimation.

Algorithm I: Zero-order holder

This algorithm looks like connecting a zero-order holder at the stub terminal. In the time interval $[t_0, t_0+\Delta t_L)$, the capacitor voltage will keep constant that is equal to $v(t_0)$. Therefore, we have the following equations to get the incident wave for the system simulation.

$$ {}_R v^i\left(t_0+k\Delta t_R\right)=v\left(t_0\right)-{}_R v^i\left(t_0+\left(k-1\right)\Delta t_R\right) \tag{5.9}$$

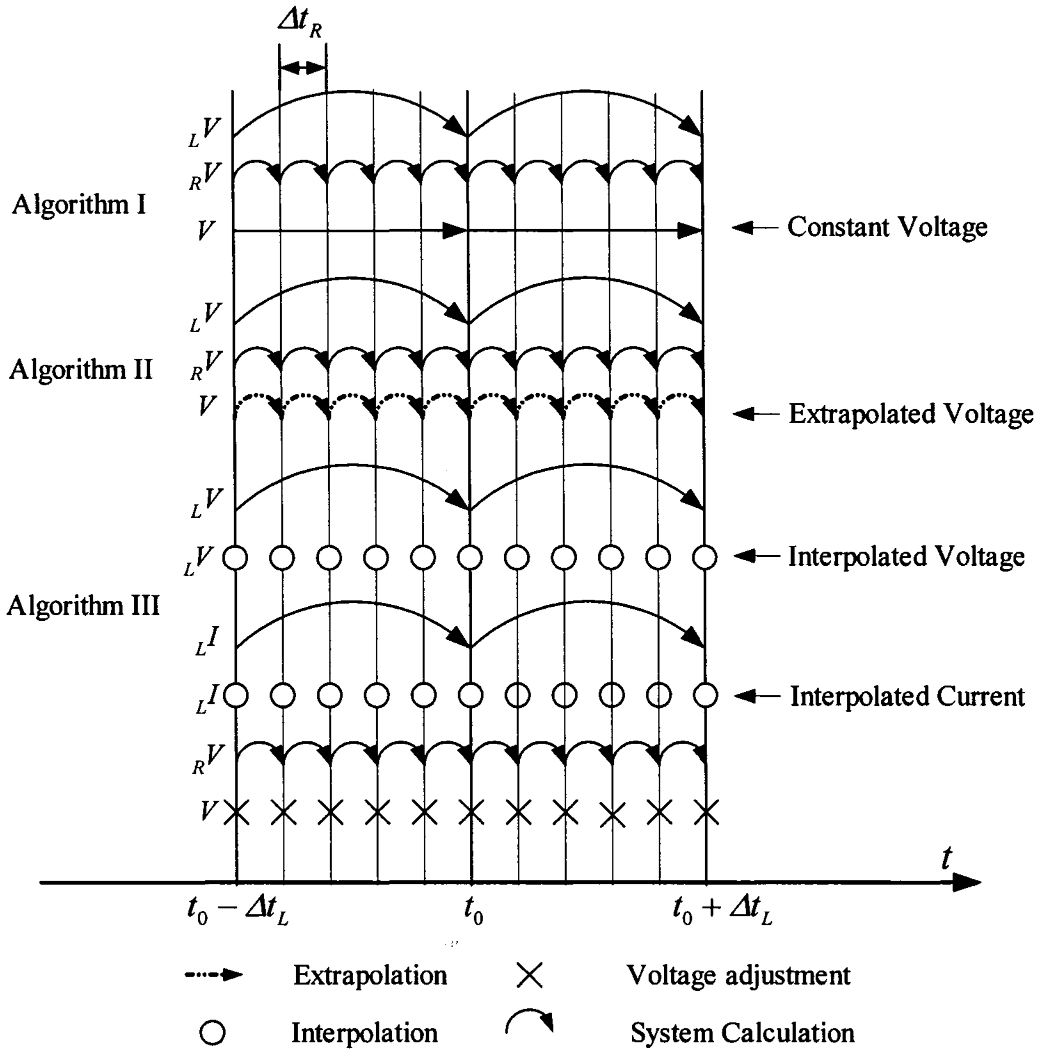


Fig. 5.2 Evaluation algorithm for obtaining capacitor's link model voltage

In the time period $[t_0, t_0+\Delta t_L]$, the voltage increment of R-subnetwork can be obtained by

$$\begin{aligned}&\Delta_R v(t_0+\Delta t_L)\\&=\left({}_R i(t_0+k\Delta t_R)+{}_R i(t_0+(k-1)\Delta t_R)\right)\frac{\Delta t_R}{2C}+\Delta_R v(t_0+\Delta t_L) \qquad k\geq 1\\&\Delta_R v(t_0+\Delta t_L)=0 \qquad k=0\end{aligned} \tag{5.10}$$

After the simulation of time point $t_0+\Delta t_L$, the voltages over decoupling capacitors should be updated by (5.11)

$$\begin{aligned}&\Delta_L v(t_0+\Delta t_L)={}_L v(t_0+\Delta t_L)-v(t_0)=\left({}_L i(t_0+\Delta t_L)+{}_L i(t_0)\right)\frac{\Delta t_L}{2C}\\&\Delta_R v(t_0+k\Delta t_R)={}_R v(t_0+k\Delta t_R)-{}_R v(t_0+(k-1)\Delta t_R)\\&\qquad=\left({}_R i(t_0+k\Delta t_R)+{}_R i(t_0+(k-1)\Delta t_R)\right)\frac{\Delta t_R}{2C}\\&v(t_0+\Delta t_L)=v(t_0)+\Delta_L v(t_0+\Delta t_L)+\sum\Delta_R v(t_0+k\Delta t_R)\\&\qquad={}_L v(t_0+\Delta t_L)+\Delta_R v(t_0+\Delta t_L)\end{aligned} \tag{5.11}$$

Algorithm II: Voltage Extrapolation

Because we know all history values, so we can perform extrapolation to obtain the capacitor voltage of any time point (t_{in}) in time interval $[t_0, t_0+\Delta t_L)$. According to the actual voltage curve shape, we can choose different orders polynomial to perform curve fitting. Normally the linear polynomial can give out a satisfied result. With the increasing of the order, more history data are required to perform extrapolation. The linear extrapolation and second order extrapolation are shown as:

$$v(t_0+t_{in})=v(t_0)+\frac{v(t_0)-v(t_0-\Delta t_L)}{\Delta t_L}t_{in} \tag{5.12}$$

$$\begin{aligned}&\begin{bmatrix}a\\b\\c\end{bmatrix}=\begin{bmatrix}0&0&1\\\Delta t_L^2&-\Delta t_L&1\\4\Delta t_L^2&-2\Delta t_L&1\end{bmatrix}^{-1}\begin{bmatrix}v(t_0)\\v(t_0-\Delta t_L)\\v(t_0-2\Delta t_L)\end{bmatrix}\\&v(t_0+t_{in})=a\,t_{in}^2+b\,t_{in}+c\end{aligned} \tag{5.13}$$

Eqn. (5.12) is for linear extrapolation, and (5.13) is for second order extrapolation. Anyway, for any kind of extrapolation at the start point, a zero order polynomial is used, which keep the capacitor voltage constant in the following time interval. In fact, the algorithm I is that kind of extrapolation.

Algorithm III: Voltage or Current Interpolation

The advantage of algorithm II is its flexibility. We can choose different type of extrapolation curve. The disadvantage of it is its stability. If we can use interpolation to do the estimation, the stability of the simulation will be improved. There I will introduce the method using interpolation technique.

Eqn. (5.7) means the voltage change of the capacitor is the sum of the voltage changes at the both terminals of the stub model. So we can interpolate the voltage change from 0 to ${}_Lv(t_0+\Delta t_L)$ - ${}_Lv(t_0)$ or current from ${}_Li(t_0)$ to ${}_Li(t_0+\Delta t_L)$ to obtain the voltage change contributed by the L-subnetwork during any time interval. The capacitor voltage of R-subnetwork is already includes the voltage change caused by its own subnetwork. Therefore, the capacitor voltage at time t_0+t_{in} is

$${}_Lv(t_0+t_{in})=\frac{{}_Lv(t_0+\Delta t_L)-{}_Lv(t_0)}{\Delta t_L}t_{in} \tag{5.14}$$

or

$$\begin{aligned}{}_Li(t_0+t_{in})&={}_Li(t_0)+\frac{{}_Li(t_0+\Delta t_L)-{}_Li(t_0)}{\Delta t_L}t_{in}\\ \Delta_L v(t_0+t_{in})&=\left({}_Li(t_0)+{}_Li(t_0+t_{in})\right)Z_{(sb)}\frac{t_{in}}{\Delta t_L}\end{aligned} \tag{5.15}$$

$$v(t_0+t_{in})=\Delta_L v(t_0+t_{in})+{}_Rv(t_0+t_{in}) \tag{5.16}$$

The above algorithms are indeed capable of doing good estimation. The algorithm I is the simplest algorithm, which does not adjust the capacitor voltage between two adjacent

L-subnetwork time points. Both algorithm II and III adjust the capacitor voltage according to the simulation results, but they also have difference. Algorithm II uses history value to do the adjustment by using extrapolation; algorithm III uses current value to do the adjustment by using interpolation. From theory, the interpolation is more reliable than the extrapolation.

5.3 Conclusion

In this chapter, a new system decoupling technique by using TLM stub models is introduced. The old decoupling technique introduced by Hui and Christopoulos needs the TLM link model at each energy exchanging time point, and then convert the incident wave of link model to incident wave of stub model to do the system simulation between two adjacent energy exchanging time points. This requires two different system matrices according to the usage of the different TLM models. In the new decoupling technique, there is no usage of the TLM link model. Therefore, the conversion between the link and stub model is avoided and the system matrix that is adopted to do the energy exchanging is the same as the system matrix that is adopted to do the system simulation.

Chapter 6 System Modeling

In the chapter 2, 3, 4 and 5, the TLM device models are all given out. In order to do the system simulation, not only the device models but also a suitable and high efficient simulation algorithm are needed. It determines whether the device models are practical and reasonable. Nowadays, there are two main popular approaches: one is the state variable technique; the other is the modified nodal analysis method. By using the state variables technique, we can get the minimized system integro-differential equations and easily track the dynamic behavior and test the stability of the circuit. But the setup procedure of this technique is much more complicated than nodal analysis method. Some researches [47,48,49] have been done to simplify the setup procedure of this technique. The modified nodal analysis method is based on KCL and device's V-I relationship. It has a set of direct procedure for us to set up the system algebra equations, even though the size of the system equations is not minimized. Due to the adoption of the discrete device models, all electrical devices are transferred into resistors, controlled or uncontrolled sources and their combinations. These changes make the system easier to deal with by modified nodal analysis method.

6.1 System Setting-up Procedure

As mention in chapter 4, the new switch model is a nonlinear device, which contains a two-valued resistor and a two-valued current source. If we do not separate that model from other devices, all entries in the system matrix Y that are related to the nodes connected with this switch will change with the switch state changing. In order to limit that side effect of the switch state changing, we had better isolated switch models from other circuit. By the way, during the circuit designing stage, monitoring the voltage/current information of the switches is always a key issue. So if we add the switch current as the system variables, it will not add extra burden to the simulation computation.

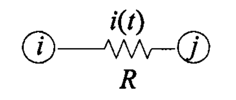
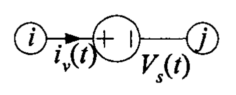
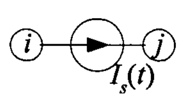
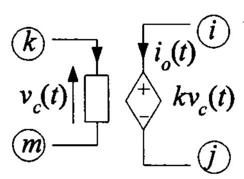
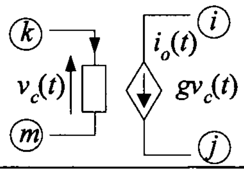
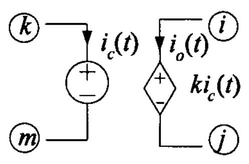
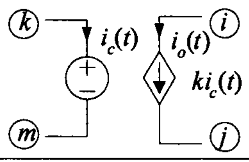
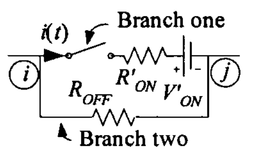
6.1.1 Device Models for Modified Nodal Analysis Method

Modified nodal analysis method is based on the node voltage and uses the admittance description of the device where available, such as resistors, capacitors, inductors, and voltage-controlled current sources. For the circuit consisted by those devices, the node voltages are chosen to be the unknown variables. For the device whose admittance description is not available, such as pure voltage sources, and voltage-controlled voltage sources, the current of it is chosen to be the unknown variables. Due to the adoption of discrete–time model, the capacitor and inductor are transferred to be a parallel-connected branch that consists of a resistor and current source; the transformer is transferred to a combination of resistors, controlled sources, and pure sources. Table 6.1 gives out the modified nodal formulations of resistor and sources.

In the table6.1, the formulations of two kinds of current-controlled sources are not given out, because these two devices can be transfer as the pure sources or voltage-controlled sources. Such as a current-controlled source, if the control signal is generated by a pure current source, that source is transferred into a pure source; if the control signal is generated by a resistor, that source is transferred into a voltage-controlled source; if the control signal is generated by a combination of resistor and current source; that source is transferred into a combination of voltage-controlled source and a pure source. Other devices such as mutual inductors, operational amplifiers, ideal transformers etc. are easily to be modeled by combining these basic devices. Therefore, the unknown variables for modified nodal analysis method not only include the node voltages but also include the currents of voltage sources.

The switch model is a combination of two resistors and a two-valued voltage source. The treatment of this model is a little different from normal resistor. Unlike the normal resistor, the current flowing through branch one that has an ideal switch in it is also set as an unknown variable. So an extra equation corresponding to the switch is shown in the set of simultaneous equations.

Table 6.1 Modified nodal formulation of some components

Device	Symbol	Device equations
Resistor		For node *i*: $i(t)=\left(v_i(t)-v_j(t)\right)/R$ For node *j*: $i(t)=\left(v_j(t)-v_i(t)\right)/R$
Voltage source		$i_v(t)$ is unknown, For current $i_v(t)$: $v_i(t)-v_j(t)=v_s(t)$
Current source		For node *i*: $I_S(t)$ is negative; For node *j*: $I_S(t)$ is positive. Locate at right side
Voltage-controlled voltage source		$i_o(t)$ is unknown, For node *i*: $i_o(t)$ is positive; $v_i(t)-v_j(t)-k\left(v_k(t)-v_m(t)\right)=0$
Voltage-controlled current source		For node *i*: $i_o(t)=g\left(v_k(t)-v_m(t)\right)$ For node *j*: $i_o(t)=-g\left(v_k(t)-v_m(t)\right)$
Current-controlled voltage source		$i_o(t)$ is unknown, For node *i*: $i_o(t)$ is positive; $v_i(t)-v_j(t)-ki_o(t)=0$
Current-controlled current source		$i_c(t)$ is unknown, For node *i*: $i_o(t)$ is positive;
Switch	 i(t) Branch one i j R_{OFF} R'_{ON} V'_{ON} Branch two	Branch one: $i(t)$ is unknown: $i(t)=S*\dfrac{v_i(t)-v_j(t)-V'_{ON}}{R'_{ON}}\quad S=\begin{cases}1 & switch\ on\\0 & switch\ off\end{cases}$

6.1.2 System Matrix Setting-up Procedure[33,34,35]

Till now, the model descriptions of devices used in the SMPS are all listed out. The next step is to give out the system equations building procedure. Suppose the network has *n* nodes, *m* voltage sources, and p switches. Select a reference node (usually ground) and name the remaining n-1 nodes, name the currents flowing through the m voltage sources and branch one currents of p switches. Also label currents through each current source. So the system has n-1+m+p unknown variables. The equations of the system are denoted as

$$AX = Z \tag{6.1}$$

A is the system matrix, which is a $(n\text{-}1+m+p)\times(n\text{-}1+m+p)$ matrix; X represents the unknown vector and includes all node voltage, currents in the voltage sources, branch one of all switches and controlled voltage sources, which is a $(n\text{-}1+m+p)\times 1$ vector; Z represents the system input, which is also a $(n\text{-}1+m+p)\times 1$ vector.

Omit the branch one of all switches. Due to the existence of the R_{OFF}, the system topology keeps unchanged. Write an equation for each node and each voltage sources according to the KCL, KVL and device characteristics for N_{RLC}. Only current term of the current source and voltage term of the voltage source are left at the right side of the equation. Other terms are all located at the left side of the equation. For network N_{RLC}, the system equations are

$$YX_1 = Z_1$$
$$Y = \begin{bmatrix} Y_1 & B_Y \\ B_Y^t & 0 \end{bmatrix} \quad X_1 = \begin{bmatrix} v_{node} \\ i_{source} \end{bmatrix} \quad Z_1 = \begin{bmatrix} I_{source} \\ V_{source} \end{bmatrix} \tag{6.2}$$

Y is a $(n\text{-}1+m)\times$ $(n\text{-}1+m)$ matrix, which is the system connection matrix and contains the information about the structure of system N_{RLC} and device admittance; Y_1 is a $(n\text{-}1)\times$ $(n\text{-}1)$ matrix, which is the node conductivity matrix; B_Y is a $(n\text{-}1)\times$ m matrix, which is the voltage source connection matrix; B_Y^t is the transpose matrix of B_Y; v_{node} is a $(n\text{-}1)\times 1$ vector, which contains the unknown node voltages; i_{source} is a $m\times 1$ vector, which contains the unknown currents in the idea voltage source; I_{source} is a $(n\text{-}1)\times 1$ vector, in which each entry is the current flowing into the corresponding node from current source; V_{source} is a $m\times 1$vector, in which each entry is the voltage of the corresponding voltage source.

And then write voltage equations for branch one of all switches. So system matrix Y will extend to be A. More sub matrixes will be shown in the system matrix A.

$$A = \begin{bmatrix} Y & B \\ B' & D \end{bmatrix} \quad X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \quad X_2 = [i_{sw}]$$
$$Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \quad Z_2 = [S] \times [V'_{ON}] \tag{6.3}$$

All matrixes ever shown in the last step are kept constant. The new matrixes are added due to the existence of the switches. B is a $(n\text{-}1+m)\times p$ matrix, which is the switch connection matrix; B^t is the transpose matrix of B; D is a $p\times p$ diagonal matrix, which diagonal entries are the equivalent resistances of the branch one of switch model changed with the switch state changing; i_{sw} is a $p\times 1$ vector, which contains the unknown currents in the branch one of switch model; S is a $p\times p$ matrix, which diagonal element represent the state of the corresponding switch; V'_{ON} is a $p\times 1$ vector, in which each entry means the threshold voltage of the corresponding switch.

The main advantage of modified nodal analysis method is that it is very suitable for computer analysis. Just according to some very simple rules, the system equations can be easily obtained. These rules are classified into six categories. The subscript O represents the original system; the subscript N represents the new system where an extra device is connected into the original system.

1. Resistor *R*

Assume a resistor R is connected between node i and j.

$$A_N(i,i) = A_O(i,i) + 1/R \qquad A_N(j,j) = A_O(j,j) + 1/R$$
$$A_N(i,j) = A_O(i,j) - 1/R \qquad A_N(j,i) = A_O(j,i) - 1/R \tag{6.4}$$

2. Voltage Source *V*

Assume voltage source *V* is connected between node *i* and *j*. Because the voltage and current of a voltage source are independent to each other, the current in the voltage source should become an extra unknown variable of the new system in order to get the KCL equation for node *i* and *j*. This current is denoted as i_v that is the *k*-th variable of the new system and flow from node *i* to *j*. At the same time, an extra constraint equations for voltages at node *i* and *j* are introduced.

$$\begin{array}{llll} v_i - v_j = V \quad \text{constraint equation} & & & \\ A_N(i,k)=1 & A_N(j,k)=-1 & A_N(k,i)=1 & A_N(k,j)=-1 \\ Z_N(k)=V & & & \end{array} \tag{6.5}$$

3. Current Source *I*

Assume current source *I* is connected between node *i* and *j* whose current flows from *i* to *j*:

$$Z_N(i)=Z_O(i)-I \qquad Z_N(j)=Z_O(j)+I \tag{6.6}$$

4. Voltage-controlled Voltage Source

Assume a voltage-controlled voltage source is connected between node *i* and *j* whose control signal come from the voltage signal between node *k* and *m*. As the Table 6.1 listed, a new variable should be added. It is the current of this voltage source, which is the *p*-th variable, flows from *i* to *j* and denoted as i_c. And a new constraint equation is given out.

$$\begin{array}{llll} v_i - v_j = K(v_k - v_m) \quad \text{constraint equation} & & & \\ A_N(p,k)=-K & A_N(p,m)=K & A_N(p,i)=1 & A_N(p,j)=-1 \\ A_N(i,p)=1 & A_N(j,p)=-1 & Z_N(p)=0 & \end{array} \tag{6.7}$$

These equations show that the existence of that device makes the system matrix unsymmetrical.

5. Voltage-controlled Current Source

Assume a voltage-controlled current source is connected between node *i* and *j* whose control signal comes from the voltage signal between node *k* and *m*. Because the constraint equation of this device is a linear equation and has gain with conductance unit, this equation can be inserted the original system matrix without extending the system matrix. According to the table 6.1, we can have the following equation. From them we find the existence of that device makes the system matrix unsymmetrical.

$$\begin{aligned} &i_C = g(v_k - v_m) \quad \text{constraint equation} \\ &A_N(i,k) = A_O(i,k) + g \qquad A_N(i,m) = A_O(i,m) - g \\ &A_N(j,k) = A_O(j,k) - g \qquad A_N(j,m) = A_O(j,m) + g \end{aligned} \tag{6.8}$$

6. Switch

Assume switch is connected between node *i* and *j*. Because extra unknown current variable i_{SW} is added, which is the k-th variable, the extra constraint equation is

$$\begin{aligned} &v_i - v_j - V_{ON}' - i_{SW} R = 0 \ \text{ or } \ i_{SW} = G(v_i - v_j) - GV_{ON}' \\ &\quad \text{where} \quad G = 1/R \end{aligned} \tag{6.9}$$

When the switch is on, the *R* is equal to R_{ON}'; when the switch is off the *R* goes to infinite. Therefore in the practical computation, the term of *G* is more prefer than the term of *R* for switch model. The changes of the system matrix are

$$\begin{aligned} &A_N(i,k) = 1 \qquad A_N(j,k) = -1 \qquad A_N(k,i) = 1 \qquad A_N(k,j) = -1 \\ &A_N(i,k) = -R \qquad Z_N(k) = V_{ON}' \end{aligned} \tag{6.10}$$

By following the aforementioned four steps of setup procedure and system matrix formation rules, the matrix form of system equations is easily formed. In the next section, the practical computation used in computer will be discussed.

6.2 Practical Computation Technique [27]

From the last section, the matrix form of system equations has the following form. All matrixes are partitioned into some sub matrixes according to classification of the system unknown variables. X_2 and Z_2 corresponding to the switches' variables in the system.

$$\begin{aligned}&AX = Z\\&A=\begin{bmatrix}Y & B\\ B' & D\end{bmatrix}\quad X=\begin{bmatrix}X_1\\ X_2\end{bmatrix}\quad Z=\begin{bmatrix}Z_1\\ Z_2\end{bmatrix}\end{aligned}\tag{6.11}$$

During the simulation procedure, if we can easily obtain the inverse matrix of A after the switch state changes, the simulation speed will not be affected very much. By using the formula of partitioned matrix inversion, the inverse matrix of A has the following form.

$$\begin{aligned}&A^{-1}=\begin{bmatrix}E & F\\ G & H\end{bmatrix}=\begin{bmatrix}Y & B\\ B' & D\end{bmatrix}^{-1}\\&=\begin{bmatrix}\left(Y+BD^{-1}B^T\right)^{-1} & -Y^{-1}B\left(D+B^TY^{-1}B\right)^{-1}\\ D^{-1}B^T\left(Y+BD^{-1}B^T\right)^{-1} & \left(D+B^TY^{-1}B\right)^{-1}\end{bmatrix}\\&\text{where}\qquad B^T=-B'\end{aligned}\tag{6.12}$$

Further manipulating the equations as shown in appendix E, two different results can be obtained. In certain cases, one result will be more efficient than the other result.

6.2.1 Calculation of the Inverse Matrix

Comparing with the matrix multiplication, to invert a matrix is much more time consuming.

Decreasing the order of the matrix that needs to be inverted always can improve the computation efficiency. According to the relationship between the number of the switches and the number of other system variables except the switch current, the computation procedure can be divided into two categories.

1. Case One: $p \le n\text{-}1+m$

$$X_1 = EZ_1 + FZ_2 = \left(Y^{-1} - P_1\left(I+P_3\right)^{-1}P_2\right)Z_1 - P_1\left(I+P_3\right)^{-1}D^{-1}Z_2 \qquad \text{(a)}$$

$$X_2 = GZ_1 + HZ_2 = \left(P_2 - P_3\left(I+P_3\right)^{-1}P_2\right)Z_1 + \left(I+P_3\right)^{-1}D^{-1}Z_2 \qquad \text{(b)} \qquad (6.13)$$

where $P_1 = Y^{-1}B,\ P_2 = D^{-1}B^TY^{-1},\ P_3 = D^{-1}B^TY^{-1}B$

Here P_1 is constant at all times, and P_2 and P_3 are constant matrices between semiconductor switching and changed with the switching actions. I is the unit matrix of appropriate dimensions. When switch is on, the corresponding entry in matrix D is equal to $R_{ON}^{/}$, so the corresponding entry in matrix D^{-1} is equal to $1/R_{ON}^{/}$; when switch is off, the corresponding entry in matrix D is infinite, so the corresponding line in matrix D^{-1} is equal to 0. So whenever switching occurs P_2 and P_3 can be determined algebraically without resorting to matrix multiplication because of the fact that the P_{2ON} and P_{3ON} that correspond to the system state that all switches are on have been stored at the beginning of the computation for any circuit. The premultiplication of D^{-1} under certain switch states will correspond to setting the corresponding rows of P_2 and P_3 to zeros. Also A^{-1}, and P_1 need to be found only once at the beginning of the computation for any circuit. The efficiency of the method lies in the fact that any time a switching occurs, the changes in P_2 and P_3 are affected algebraically by setting some rows to zero, and the maximum dimension of the matrix inverted at any switching instant as seen from (6.13) is only equal to the number of semiconductors in the circuit.

2. Case Two: $p > n\text{-}1+m$

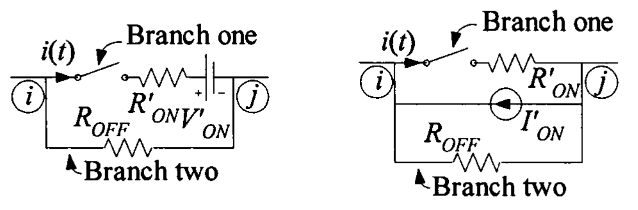
Like in case one P_1 is constant at all times, and P_2 and P_3 are constant matrices between semiconductor switching and changed with the switching actions. **I** is the unit matrix of appropriate dimensions. P_2 is obtained through the same algorithm in case one. Also Y^{-1}, and P_1 need to be found only once at the beginning of the computation. Unlike in case one, P_3 is only able to obtain through matrix multiplication. That decreases the computation efficiency. However, if $p > n\text{-}1+m$, the size of P_3 in case two is smaller than the P_3 in case one. This change will improve the efficiency of inverse matrix computation, and then improve the simulation efficiency.

$$\begin{aligned} X_1 &= EZ_1 + FZ_2 = Y^{-1}(I+P_3)^{-1} Z_1 - P_1 HZ_2 && \text{(a)} \\ X_2 &= GZ_1 + HZ_2 = P_2(I+P_3)^{-1} Z_1 + \left(I - P_2(I+P_3)^{-1} B\right) D^{-1} Z_2 && \text{(b)} \end{aligned} \qquad (6.14)$$

$$P_1 = Y^{-1}B,\ \ P_2 = D^{-1}B^T Y^{-1},\ \ P_3 = BD^{-1}B^T Y^{-1}$$

Now two computation algorithms are given out. Each algorithm is suitable for certain type of circuit. For normal circuit, p should be smaller than n-1+m, so there is more chance for using algorithm one. If a SMPS has lots of switches and p is larger than n-1+m, the algorithm two should be more suitable.

6.2.2 Further Improvement to Enhance the Computation Efficiency



a) Original switch model b) Modified switch model

Fig. 6.1 Modified switch model

The switch model in Fig. 6.1a) is a Thevenin equivalent circuit. The auxiliary equation for branch one is shown in Table 6.1. The model in Fig. 6.1b) is a Norton equivalent circuit. The auxiliary equation for branch one is

Branch one: $i(t)=S(v_i(t)-v_j(t))/R_{ON}'$ (a)

Branch two:

For node *i*: (b)

$$i(t)=(v_i(t)-v_j(t))/R_{OFF}+SV_{ON}'/R_{ON}'=(v_i(t)-v_j(t))/R_{OFF}+I_{Dpre}$$

(6.15)

For node *j*:

$$i(t)=(v_j(t)-v_i(t))/R_{OFF}-SV_{ON}'/R_{ON}'=(v_j(t)-v_i(t))/R_{OFF}-I_{Dpre}$$ (c)

$$S=\begin{cases}1 & \text{switch ON}\\ 0 & \text{switch OFF}\end{cases}$$

The difference between two models is which branch the effect of the threshold voltage is classified into. The model in Fig. 6.1a) classify the threshold voltage into branch one; The model in Fig. 6.1b) classify the threshold voltage into branch two.

Table 6.2 The different affects of two models

	Original Switch model	Modified switch model
Auxiliary equation	$i(t)=S(v_i(t)-v_j(t)-V_{ON}')/R_{ON}'$	$i(t)=S(v_i(t)-v_j(t))/R_{ON}'$
Z vector	$\pm SV_{ON}'$	0

By adopting modified switch model in Fig. 6.1b), the Z_2 vector is always zero vector. The calculation of the *F* and *H* matrix becomes unnecessary. At the same time, the current source vector I_{source} need to be modified. For the *i*-th node, $I_{source}(i)$ has an extra positive current input; for the *j*-th node, $I_{source}(j)$ has an extra negative current input.

The new approach to the formulation and solution of switch circuit equations have been described. The major advantages of the new approach are list as the following:

The well-known and highly efficient modified nodal formulation of circuit equations by inspection forms the basis of the approach.

The switch equations are partly separated from other equations in such a manner that switch monitoring becomes very simple.

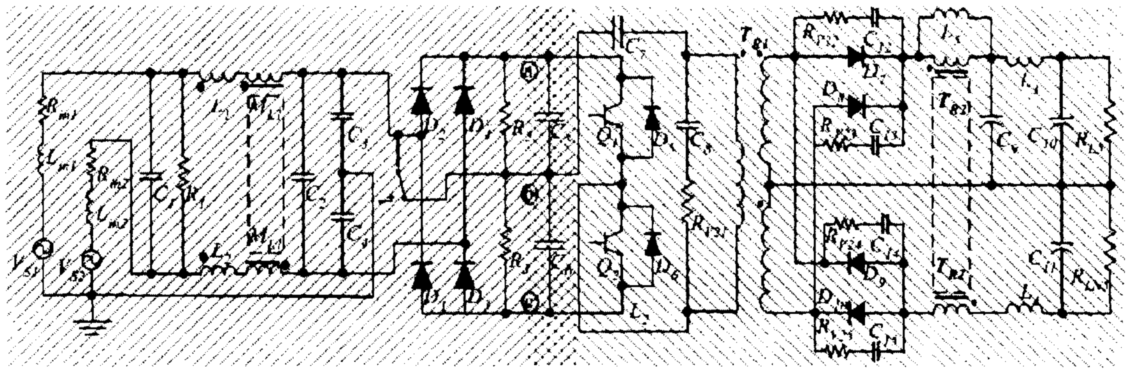
The topology structure of the circuit keeps connected, even all switches are off.

The number of matrices is reduced with adoption of modified switch model.

The coefficient matrix of the circuit equations that needs to be inverted remains highly sparse. Application of sparse matrix techniques is thus adoptable.

In the next section, the system equations of the studied SMPS circuit will be presented by using this new proposed method, and the simulation algorithm will be discussed further.

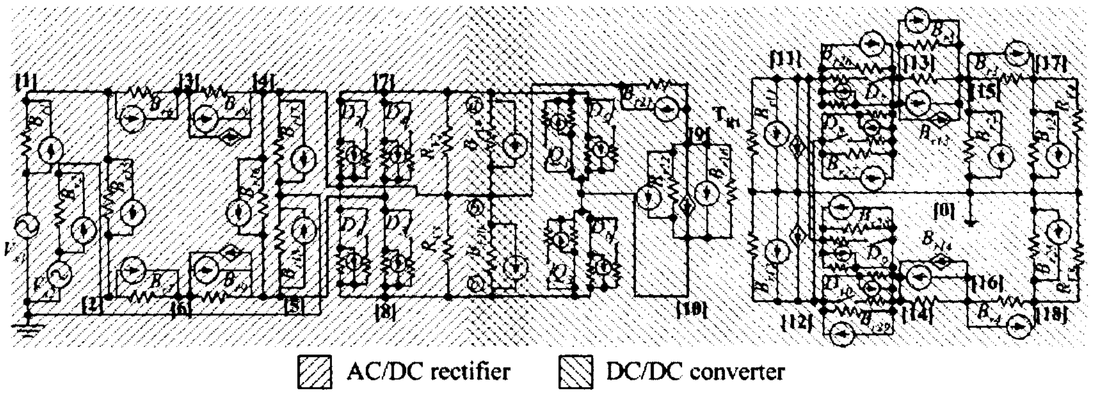
6.3 System Matrix and System's Initial Condition



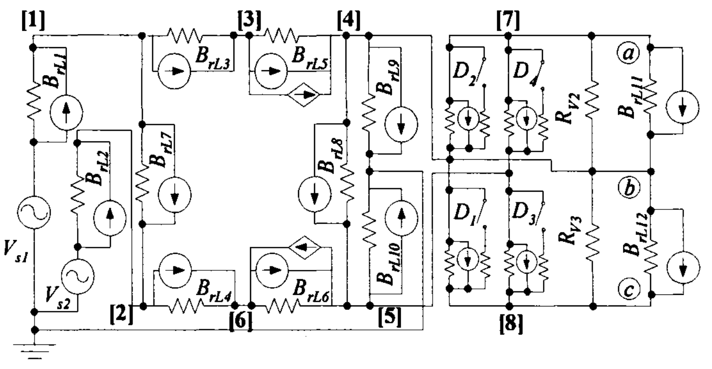
▨ AC/DC rectifier ▧ DC/DC converter

Fig. 6.2 Simplified computer power supply circuit

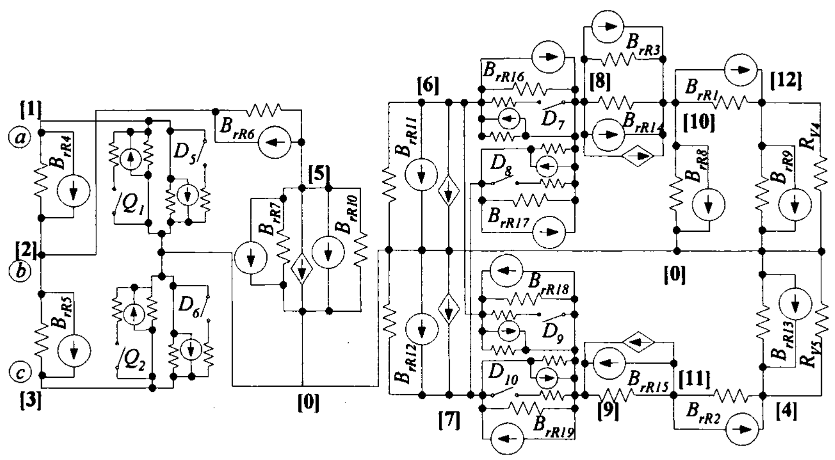
Fig. 6.2 is a simplified computer power supply system. The capacitors C_5, C_6, C_9, C_{10}, C_{11} are the bulky DC link capacitors. The voltages of them can almost keep constant in a few microseconds and separate the network into two total independent subnetworks, so all of them can be chosen to decouple the system. But in practice the C_9, C_{10}, and C_{11} are rejected due to the small size of the subnetwork behind them. And then the system is decoupled into subnetworks.



a) TLM model of simplified circuit



b) TLM model of AC/DC rectifier



c) TLM model of DC/DC converter

Fig. 6.3 TLM model of SMPS

The discrete models of the two subnetworks are shown in Fig. 6.3. The low frequency subnetwork contains 9 nodes and 4 switches; the high frequency subnetwork contains 13 nodes and 8 switches. So (6.13) is chosen. In the two subnetworks, the node numbers and branch numbers should be reordered starting from 1.

Due to the existence of the transformer in the DC/DC converter, we should ground the primary side of the circuit to avoid the ill-conditioned system matrix. There we connect node 10 to the ground.

6.3.1 Setting Up the System Matrices and Vectors

To simulate the system, one matrix and one vector need to be set up. One is the system matrix A and the other is the system input *Z*. In order to easily describe *A* and *Z*, the general forms of V-I characteristic equations, which are used in this circuit simulation, are listed next. For *Br*(*i*, *j*), *i* means the *i*-th branches; *j* means the *j*-th coefficient of *i*-th branches. Let express the V-I characteristic of any branch in general form:

$$\begin{aligned} i(n) &= Gv(n) + I_{eq}(n) = Br(1,1)v(n) + Br(1,2)V^i(n) \\ V^i(n+1) &= Br(1,3)v(n) + Br(1,4)V^i(n) \end{aligned}$$

The general form of V-I characteristics of two-coupled inductance is:

$$\begin{bmatrix} i_1(n) \\ i_2(n) \end{bmatrix} = G\begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} + \begin{bmatrix} I_{eq1}(n) \\ I_{eq2}(n) \end{bmatrix} = Br(1:2,1:2)\left(\begin{bmatrix} v_1(n) \\ v_2(n) \end{bmatrix} - 2\begin{bmatrix} v_{L1}^i(n) + v_{M12}^i(n) \\ v_{L2}^i(n) + v_{M21}^i(n) \end{bmatrix}\right)$$

$$\begin{bmatrix} v_{L1}^i(n) \\ v_{M12}^i(n) \\ v_{M21}^i(n) \\ v_{L2}^i(n) \end{bmatrix} = \begin{bmatrix} Br(1:2,3:4) \\ Br(1:2,5:6) \end{bmatrix}\left(\begin{bmatrix} v_1(n-1) \\ v_2(n-1) \end{bmatrix} - 2\begin{bmatrix} v_{L1}^i(n-1) + v_{M12}^i(n-1) \\ v_{L2}^i(n-1) + v_{M21}^i(n-1) \end{bmatrix}\right) - \begin{bmatrix} v_{L1}^i(n-1) \\ v_{M12}^i(n-1) \\ v_{M21}^i(n-1) \\ v_{L2}^i(n-1) \end{bmatrix}$$

Table 6.3 Coefficients of branch V-I characteristic equations (TLM)

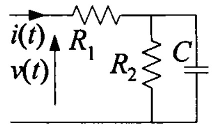
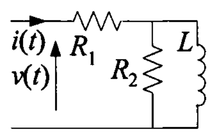
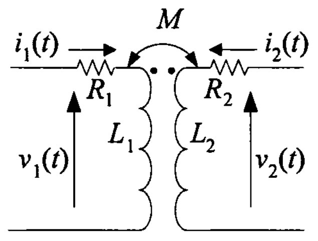
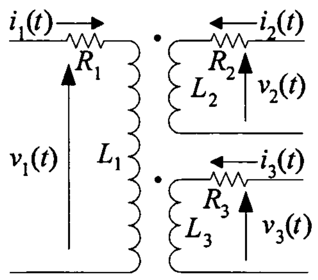
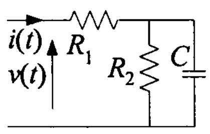
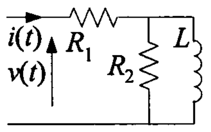
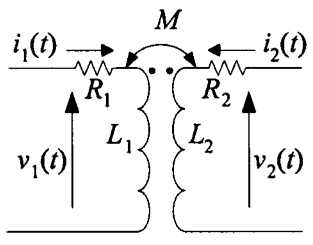
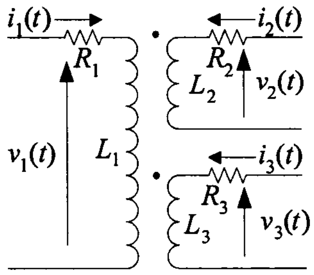
Branch Type	**Equations**
	$Br=\dfrac{\left[R_2+R_C \quad -2R_2 \quad R_2R_C \quad R_2R_1-R_2R_C-R_1R_C\right]}{R_2R_C+R_2R_1+R_1R_C}$
R_2=*inf*	$Br=\left[1 \quad -2 \quad R_C \quad R_1-R_C\right]/\left(R_1+R_C\right)$
R_1=0	$Br=\left[\left(R_2+R_C\right)/\left(R_2R_C\right) \quad -2/R_C \quad 1 \quad -1\right]$
R_1=0 R_2=*inf*	$Br=\left[1/R_C \quad -2/R_C \quad 1 \quad -1\right]$
	$Br=\dfrac{\left[R_2+R_L \quad -2R_2 \quad -R_2R_L \quad R_2R_L+R_1R_L-R_2R_1\right]}{R_2R_L+R_2R_1+R_1R_L}$
R_2=*inf*	$Br=\left[1 \quad -2 \quad -R_L \quad R_L-R_1\right]/\left(R_1+R_L\right)$
R_1=0	$Br=\left[\left(R_2+R_L\right)/\left(R_2R_L\right) \quad -2/R_L \quad -1 \quad 1\right]$
R_1=0 R_2=*inf*	$Br=\left[1/R_L \quad -2/R_L \quad -1 \quad 1\right]$
 M $i_1(t)$ $i_2(t)$ R_1 R_2 L_1 L_2 $v_1(t)$ $v_2(t)$	$Br(1:2,1:2)=\left(\begin{bmatrix}R_{L1} & R_M \\ R_M & R_{L2}\end{bmatrix}+\begin{bmatrix}R_1 & 0 \\ 0 & R_2\end{bmatrix}\right)^{-1}$ $Br(1:2,3:4)=\begin{bmatrix}R_{L1} & 0 \\ 0 & R_M\end{bmatrix}Br\left(1:2,1:2\right)$ $Br(1:2,5:6)=\begin{bmatrix}R_M & 0 \\ 0 & R_{L2}\end{bmatrix}Br\left(1:2,1:2\right)$
 $i_1(t)$ $i_2(t)$ R_1 R_2 L_2 $v_2(t)$ $v_1(t)$ L_1 $i_3(t)$ R_3 L_3 $v_3(t)$	$Br\left(1:3,1:3\right)=Y_T^1$

Table 6.4 Coefficients of branch V-I characteristic equations (Generalized TLM)

Branch Type	Equations
	$Br=\frac{\begin{bmatrix}R_2+R_{eq} & -2R_2 & R_2R_{eq}/2 & R_2R_1\end{bmatrix}}{R_2R_{eq}+R_2R_1+R_1R_{eq}}$
R_2=*inf*	$Br=\begin{bmatrix}1 & -2 & R_{eq}/2 & R_1\end{bmatrix}/(R_1+R_{eq})$
R_1=0	$Br=\begin{bmatrix}(R_2+R_{eq})/(R_2R_{eq}) & -2/R_{eq} & 1/2 & 0\end{bmatrix}$
R_1=0 R_2=*inf*	$Br=\begin{bmatrix}1/R_{eq} & -2/R_{eq} & 1/2 & 0\end{bmatrix}$
	$Br=\frac{\begin{bmatrix}R_2+R_{eq} & -R_2 & -R_2R_{eq} & R_2R_{eq}+R_1R_{eq}\end{bmatrix}}{R_2R_{eq}+R_2R_1+R_1R_{eq}}$
R_2=*inf*	$Br=\begin{bmatrix}1 & -1 & -R_{eq} & R_{eq}\end{bmatrix}/(R_1+R_{eq})$
R_1=0	$Br=\begin{bmatrix}(R_2+R_{eq})/(R_2R_{eq}) & -1/R_{eq} & -1 & 1\end{bmatrix}$
R_1=0 R_2=*inf*	$Br=\begin{bmatrix}1/R_{eq} & -1/R_{eq} & -1 & 1\end{bmatrix}$
	$Br(1:2,1:2)=\left(\begin{bmatrix}Z_{L1} & Z_M\\ Z_M & Z_{L2}\end{bmatrix}+\begin{bmatrix}R_1 & 0\\ 0 & R_2\end{bmatrix}\right)^{-1}$ $Br(1:2,3:4)=\begin{bmatrix}Z_{L1} & 0\\ 0 & Z_M\end{bmatrix}Br(1:2,1:2)$ $Br(1:2,5:6)=\begin{bmatrix}Z_M & 0\\ 0 & Z_{L2}\end{bmatrix}Br(1:2,1:2)$
	$Br(1:3,1:3)=Y_T^1$

The general form of V-I characteristics of three-winding transformer is:

$$\begin{bmatrix} i_1(n) \\ i_2(n) \\ i_3(n) \end{bmatrix} = G \begin{bmatrix} v_1(n) \\ v_2(n) \\ v_3(n) \end{bmatrix} + \begin{bmatrix} I_1(n) \\ I_2(n) \\ I_3(n) \end{bmatrix} = Br(1:3,1:3) \left(\begin{bmatrix} v_1(n) \\ v_2(n) \\ v_3(n) \end{bmatrix} - 2 \begin{bmatrix} v_1^i(n) \\ v_{21}^i(n) \\ v_{31}^i(n) \end{bmatrix} \right)$$

$$\begin{bmatrix} v_1^i(n+1) \\ v_{21}^i(n+1) \\ v_{31}^i(n+1) \end{bmatrix} = Br(1:3,4:6) \left(\begin{bmatrix} v_1(n) \\ v_2(n) \\ v_3(n) \end{bmatrix} - 2 \begin{bmatrix} v_1^i(n) \\ v_{21}^i(n) \\ v_{31}^i(n) \end{bmatrix} \right) - \begin{bmatrix} v_1^i(n) \\ v_{21}^i(n) \\ v_{31}^i(n) \end{bmatrix}$$

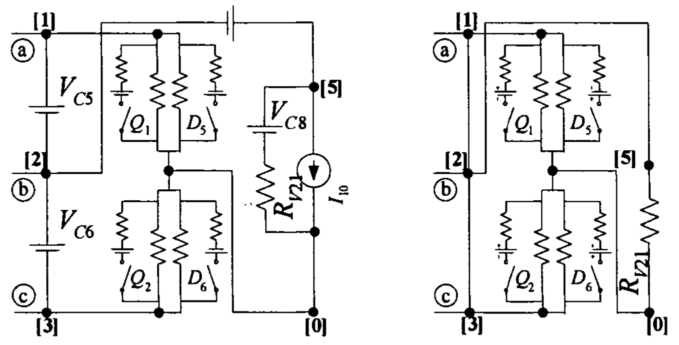
Table 6.3 lists the tranditional TLM stub model of some selected adopted electrical components, Table 6.4 lists the generalized TLM stub model of those components.

Observing the branch one and two in the low frequency part of the circuit, they have extra voltage sources comparing with the normal branch. The only modification is to substitute *v*(*n*) with *v*(*n*)-*Vs*(*n*). The others keep unchanged.

In order to avoid the effects of the control algorithm when we do the simulation results comparison, the control logic is ignored in this thesis and set the duty cycle of two semiconductors as a constant.

6.3.2 Initial Condition After Forced Commutation

As mentioned before, the transition time of the switching action is assumed to be zero. So the system states can change in no time [t^-, t^+], and the values of some system variables are discontinuous at the switching time. But the capacitor voltage and inductor current keep constant during that time period, that means a capacitor can be treated as a voltage source and an inductor can be treated as a current source. Due to the existence of linkage inductances of the three-winding transformer (T_{R1}), the current flowing into the primary winding of the T_{R1} keep constant when the forced commutation happens. The high frequency part of circuit is decoupled further for CSS identification after the forced commutation. The Fig. 6.4 shows the circuit to do this job.



a) Circuit for getting initial condition b) Auxiliary circuit

Fig. 6.4 Circuits for solving CSS after forced commutation

The technique used to identify the CSS is described in section 4 of chapter 4. The focus work is to solve the circuit in Fig. 6.4b). In this circuit, node 1,2,3, and 5 are one node in effect, which is denoted as node a, and all resistors and equivalent sources are parallel connected between node a and the ground. When the final voltage v_a is determined, the node voltages of R-subnetwork after forced commutation are determined. If v_a is not equal to zero, the voltage across the primary side of the T_{RI} is discontinuous. So the equivalent incident waves of T_{RI}'s leakage inductance need to be updated.

According to the system state at time t^- and updated system state at time t^+, the terminal voltages, currents of three winding are known values. So the value of v_0 can be obtain by the following equations according to the Fig. 2.14. The detail procedure is shown in Appendix F.

$$v_0(n) = Y_{T3}^4\left[v_1(n)-i_1(n)R_1;\;\; v_2(n)-i_2(n)R_2;\;\; v_3(n)-i_3(n)R_3\right]$$
$$\text{where}\quad Y_{T3}^4 = \frac{\left[a_{12}^2a_{13}^2L_2L_3 \quad a_{12}a_{13}^2L_1L_3 \quad a_{12}^2a_{13}L_1L_2\right]}{a_{12}^2a_{13}^2L_2L_3 + a_{13}^2L_1L_3 + a_{12}^2L_2L_1} \tag{6.16}$$

And the new incident waves after forced commutation can be obtained from the equations.

$$\begin{bmatrix} v_{L1}^i(n) \\ v_{L2}^i(n) \\ v_{L3}^i(n) \end{bmatrix} = \left(\begin{bmatrix} v_1(n)-v_0(n) \\ v_2(n)-v_0(n)/a_{12} \\ v_2(n)-v_0(n)/a_{13} \end{bmatrix} - \begin{bmatrix} Z_1 & 0 & 0 \\ 0 & Z_2 & 0 \\ 0 & 0 & Z_3 \end{bmatrix} \begin{bmatrix} i_1(n) \\ i_2(n) \\ i_3(n) \end{bmatrix} \right) \Big/ 2 \tag{6.17}$$

Till now, the CSS and the initial conditions after forced commutations, which are needed to restart the simulation, are all figured out.

6.4 Conclusion

This Chapter gives out the procedure to set up the system equations and simulate the circuit. It includes three parts. One is how to solve the system equation. According to the relationship between p and n-1+m, circuits are divided into two categories, which adopt different formula to calculate the system equations. The second part describes how to set up the system equations. The last part describes how to obtain initial conditions after the forced commutations by modified N.Femia technique. According to these procedures, the simulation program can be made. In the next chapter, the simulation results will be given out. The results will be compared with the simulation results of Multisim and PSCAD.

Chapter 7 The Results Comparison and Analysis

In this chapter, two works will be done, one is the simulation results comparison between the proposed algorithm, PSCAD and Multisim, and the other is the EMI filter performance analysis. Before doing these two works, the introduction of two simulation softwares, PSCAD/EMTDC and Multisim, will be given out first.

7.1 Introduction of PSCAD

PSCAD is a general-purpose system-level time domain simulation tool for studying transient behavior of electrical networks. Its solution engine is Electromagnetic Transients including DC (EMTDC), which is based on the principles outlined in the classic 1969 paper by Hermann Dommel [3]. The codes of PSCAD were begun to write by Dennis Woodford at Manitoba Hydro in 1975. It is used extensively for many types of AC and DC power simulation studies, including: Power electronics (FACTS), contingency of the power system, fault analysis, filter system design, control system design, and equipment testing.

7.1.1 The Setting-up Procedure of the Project

PSCAD V4 has a graphical user interface. We can easily set up a graphically constructed circuit by using imbedded component. The following Fig. 7.1 is the graphical user interface of the PSCAD. In the work space, all loaded project are shown in there. The library projects contain the component models; the case projects are where most work is performed in PSCAD. In addition to performing the functions of a library project, cases may be compiled, built and run. Simulated results can be viewed directly within the project through on-line meters and/or plots. Case projects are saved with the file extension '*.psc'. Only the active project can be run in PSCAD, and only one project can be set as active at one time.

To do a simulation, one should add a group of specified components and connected them together in one project. There are four methods to add a desired component to a project[51].

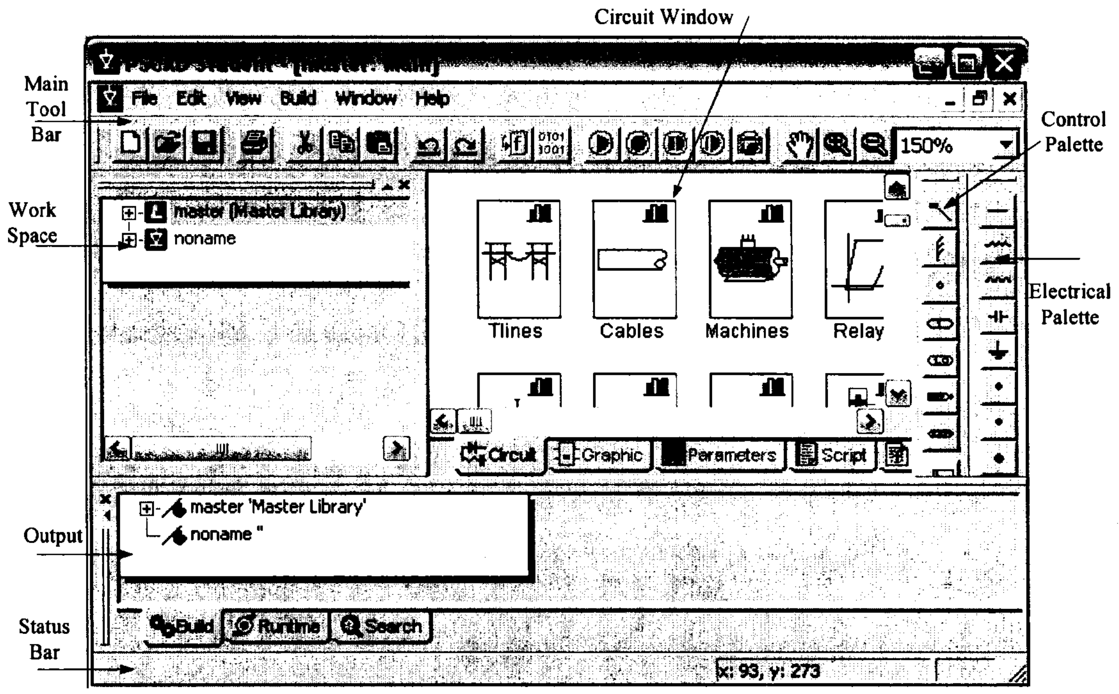


Fig. 7.1 The user interface of PSCAD

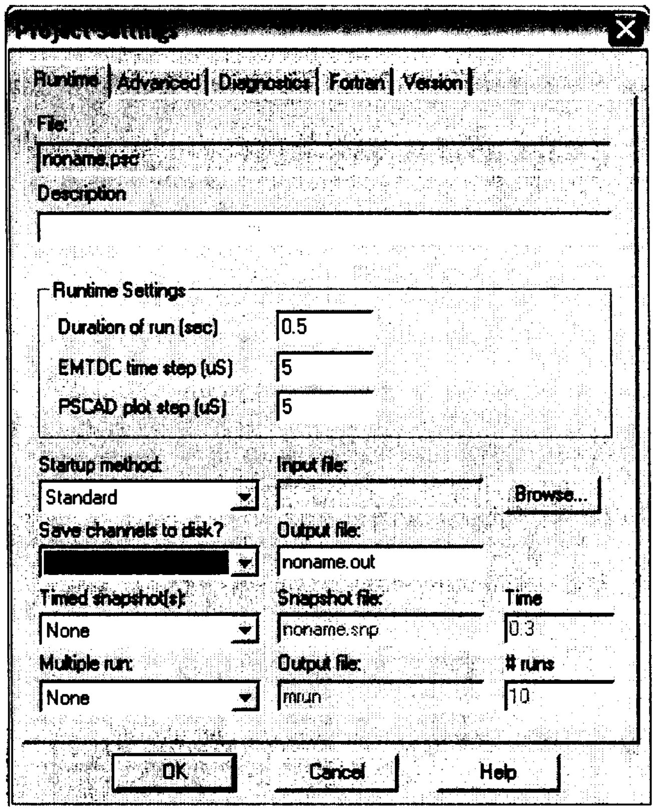
Manual Copy/Paste: Open the Master library, navigate to the area containing the desired component, and copy that component. Open the project page where you wish to add the component, and past the component.

Right-Click Menu: Right-click over a blank area of the page and select **Add Component**. A sub-menu will appear containing the most commonly used components from the Master Library. Select a component and it will be automatically added.

Library Pop-Up Menu: Press **Ctrl + right mouse button** over a blank area of the page to invoke the library pop-up menu system. Select a component and it will be automatically added.

Control and Electrical Palettes: Left-click on any of the palette buttons and then drag your mouse pointer over the Circuit window - you should see the object attached to your pointer. Continue to move the object to where you want it placed, and then left-click again.

After drawing the circuit, we should edit project setting. In the project setting dialog, duration of run, simulation time step, and plot time step can be adjusted. If we want to save output data, we also can save them in an output file by choosing "Yes" for question "Saving channels to disk?". The project setting dialog is shown in Fig. 7.2 When the project setting has done, we can begin to run the project.


Project Settings

Runtime | Advanced | Diagnostics | Fortran | Version

File: noname.psc

Description

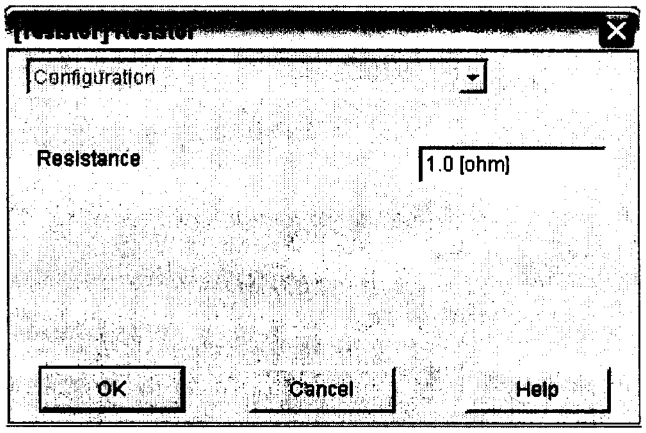
Runtime Settings
Duration of run (sec) 0.5
EMTDC time step (uS) 5
PSCAD plot step (uS) 5

Startup method: Standard
Input file:
Save channels to disk?
Output file: noname.out
Timed snapshot(s): None
Snapshot file: noname.snp
Time 0.3
Multiple run: None
Output file: mrun
# runs 10

Fig. 7.2 Project setting dialog

7.1.2 Editing Component Properties in PSCAD

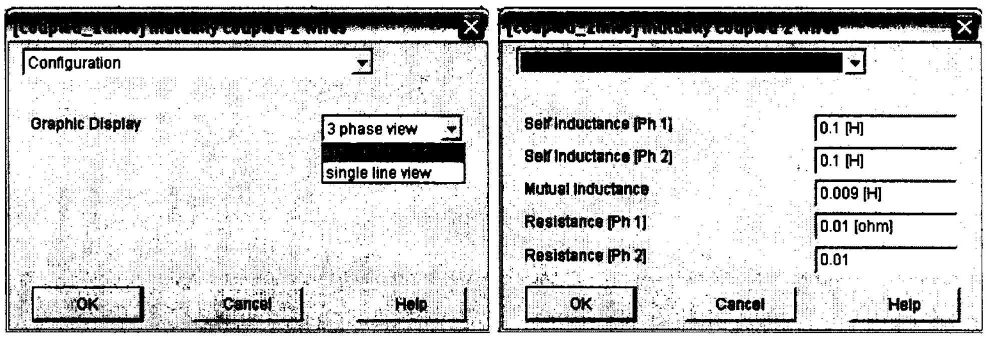
The SMPS circuit to be simulated includes resistors, inductors, capacitors, power electronic switches, transformers, mutual inductors, and voltage sources. When the circuit has been connected, the properties of those components should be edited as follow. The resistor, inductor, and capacitor only have one input parameter; the unit of the input parameter is ohm for resistor, μF for capacitance, and H for inductance. The property editing dialog is shown in Fig. 7.3. The dialogs of resistor, capacitor, and inductor are similar to each other.



Resistance 1.0 [ohm]

Fig. 7.3 Property editing dialog of resistor.

The Fig. 7.4 is the property editing dialog of mutual inductors. The figure a) is used to determined the graphic mode of the mutual inductors, and the figure b) is the parameters adjustment dialog. The parasitic resistance of the mutual inductors also can be adjusted.



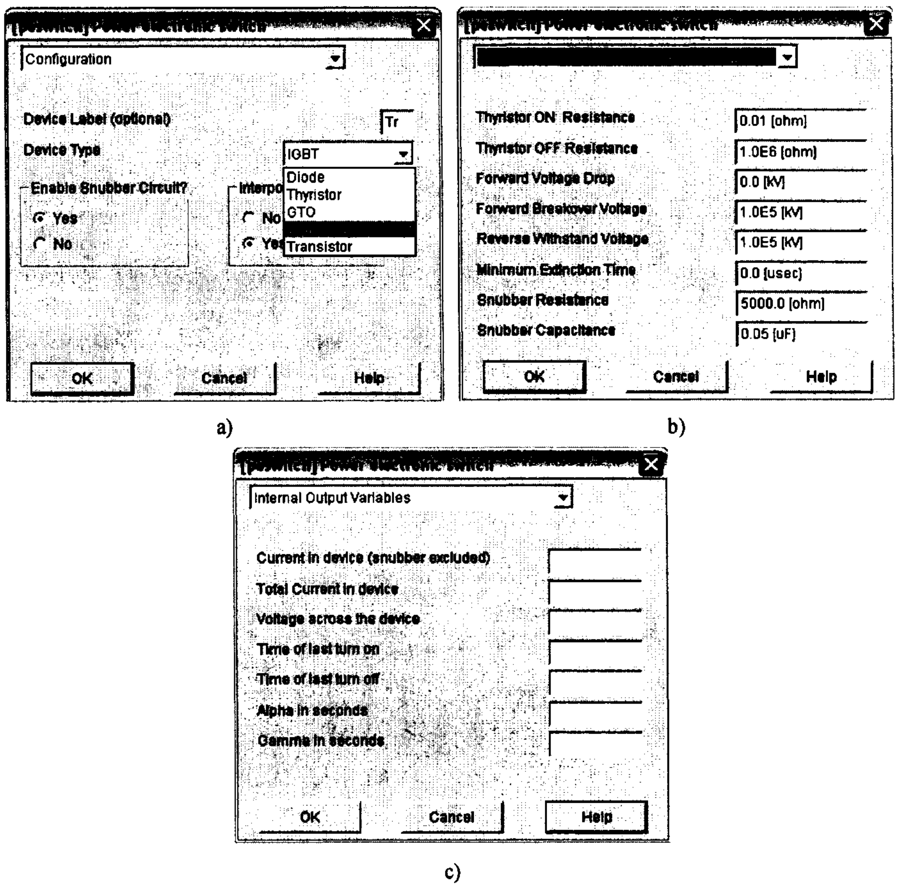
Graphic Display 3 phase view

Self Inductance [Ph 1] 0.1 [H]
Self Inductance [Ph 2] 0.1 [H]
Mutual Inductance 0.009 [H]
Resistance [Ph 1] 0.01 [ohm]
Resistance [Ph 2] 0.01

a) b)

Fig. 7.4 Property editing dialog of mutual inductor.

All semiconductor switch models in PSCAD are simplified models, which are dual value resistors that satisfied the linearized characteristic shown in Fig. 4.1a), and similar to the switch model I am using in this thesis. The difference between different electronic switches is the control signal, such as for a diode, the control signal is the voltage over the diode, for a BJT, the voltage is the voltage over the base and emitter.


a)

Configuration

Device Label (optional)	Tr
Device Type	IGBT

Diode
Thyristor
GTO
Transistor

Enable Snubber Circuit? [✓ Yes] No

Interpo... No [✓ Yes]

OK Cancel Help

b)

Thyristor ON Resistance	0.01 [ohm]
Thyristor OFF Resistance	1.0E6 [ohm]
Forward Voltage Drop	0.0 [kV]
Forward Breakover Voltage	1.0E5 [kV]
Reverse Withstand Voltage	1.0E5 [kV]
Minimum Extinction Time	0.0 [usec]
Snubber Resistance	5000.0 [ohm]
Snubber Capacitance	0.05 [uF]

OK Cancel Help

c)

Internal Output Variables

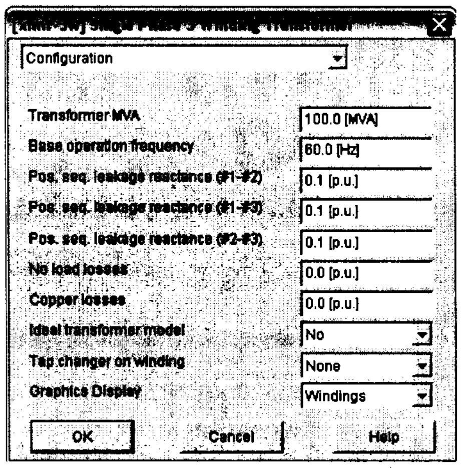
Current in device (snubber excluded)	
Total Current in device	
Voltage across the device	
Time of last turn on	
Time of last turn off	
Alpha in seconds	
Gamma in seconds	

OK Cancel Help

Fig. 7.5 Property editing dialog of electronic switch.

In PSCAD all power electronic switch share the same property window. From the Device Type in Fig. 7.5a), the type of the component can be determined. The second dialog, Main data, is used to determine the characteristics of the switch. The last dialog is used to monitor the state of the switch. From the first dialog, we also know that the snubber circuit is combined in the switch model.

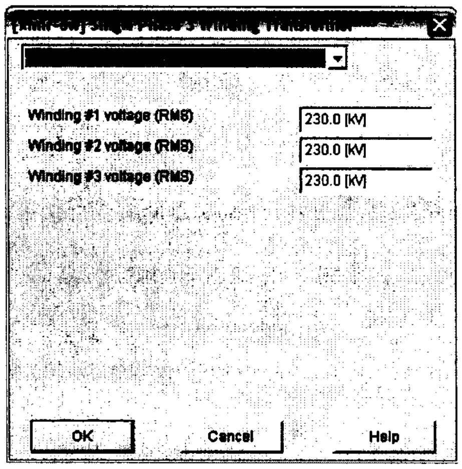
The transformer model in PSCAD includes the nonlinear property of the magnetic core, but in this thesis the nonlinear property of the transformer is not dealt. The equivalent circuit of a three winding transformer is same as the circuit in Figure 2.14. The Fig. 7.6 is property editing dialogs of 3 winding transformer in PSCAD. There are four different dialogs for 3 winding transformer property editing window, which are configuration, winding voltage rating, saturation, and monitoring of current and flux. The first dialog is used to obtain parameters of passive components in the equivalent circuit; second is used to obtain parameters of control source in the equivalent circuit; the third one is used to set up the nonlinear property of the transformer, which is no treated in this thesis; the last one is used to monitor the transformer status.



Configuration

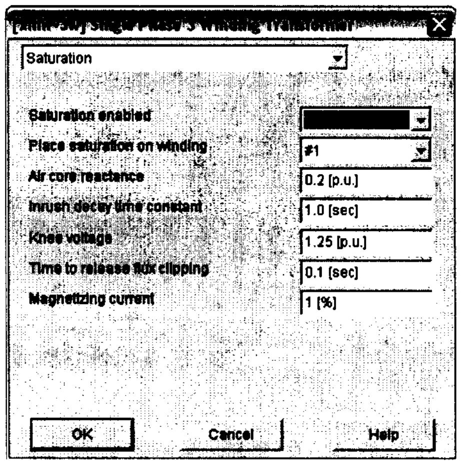
Transformer MVA	100.0 [MVA]
Base operation frequency	60.0 [Hz]
Pos. seq. leakage reactance (#1-#2)	0.1 [p.u.]
Pos. seq. leakage reactance (#1-#3)	0.1 [p.u.]
Pos. seq. leakage reactance (#2-#3)	0.1 [p.u.]
No load losses	0.0 [p.u.]
Copper losses	0.0 [p.u.]
Ideal transformer model	No
Tap changer on winding	None
Graphics Display	Windings

a)



Winding #1 voltage (RMS)	230.0 [kV]
Winding #2 voltage (RMS)	230.0 [kV]
Winding #3 voltage (RMS)	230.0 [kV]

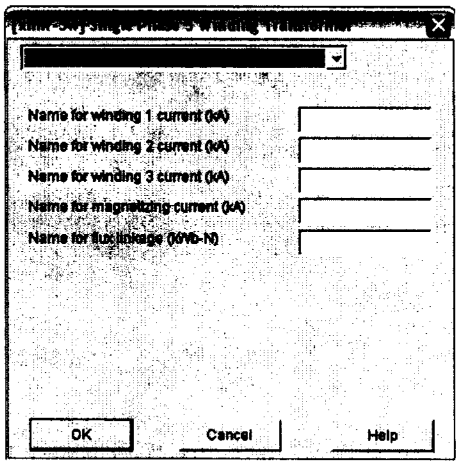
b)



Saturation

Saturation enabled	
Place saturation on winding	#1
Air core reactance	0.2 [p.u.]
Inrush decay time constant	1.0 [sec]
Knee voltage	1.25 [p.u.]
Time to release flux clipping	0.1 [sec]
Magnetizing current	1 [%]

c)

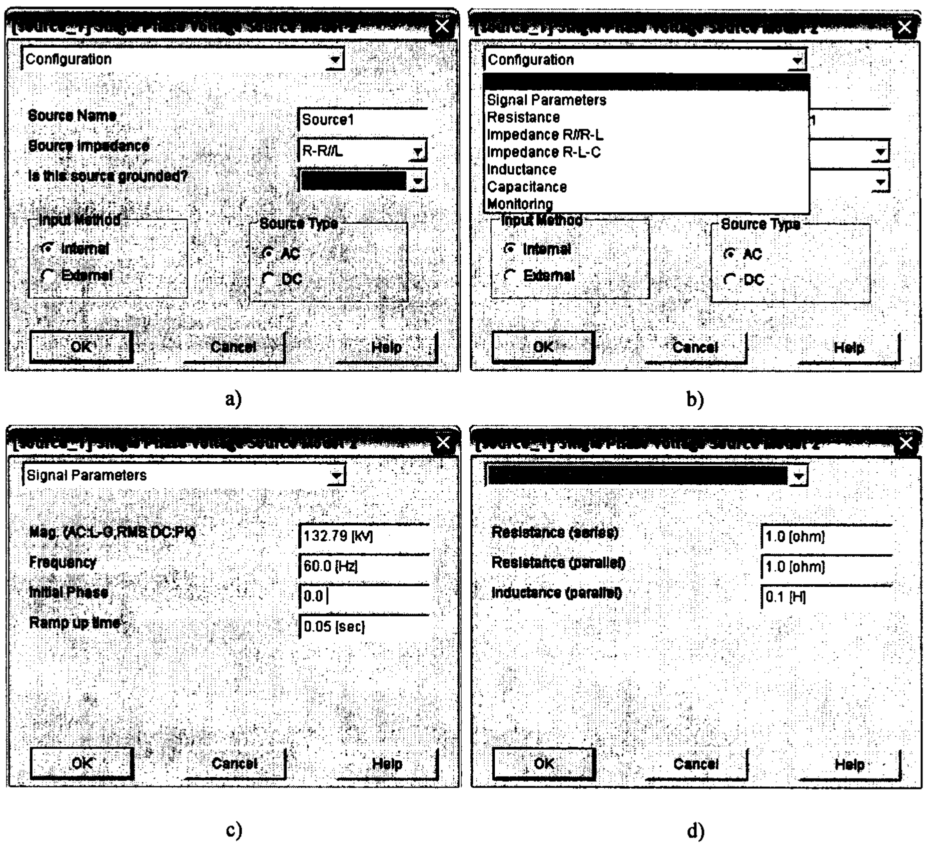


Name for winding 1 current (kA)	
Name for winding 2 current (kA)	
Name for winding 3 current (kA)	
Name for magnetizing current (kA)	
Name for flux linkage (kWb-N)	

d)

Fig. 7.6 Property editing dialog of 3 winding transforemr.

In PSCAD, a voltage source can be set as a DC/AC, grounded/ungrounded, ideal/non-ideal, internal/external control source from the configuration dialog, which is shown in Fig. 7.7a). the signal parameters dialog is used to set up the magnitude, frequency, and initial condition of the voltage source. If the voltage source is not an ideal source, the corresponding internal impedance editing dialog will be active for editing, such as the impedance R/R-L dialog shown in Fig. 7.7d).



a) Configuration

Source Name: Source1
Source impedance: R-R//L
Is this source grounded?
Input Method: [✓ Internal] External
Source Type: [✓ AC] DC

b) Configuration

Signal Parameters
Resistance
Impedance R//R-L
Impedance R-L-C
Inductance
Capacitance
Monitoring

c) Signal Parameters

Mag. (AC:L-G,RMS DC:PK): 132.79 [kV]
Frequency: 60.0 [Hz]
Initial Phase: 0.0
Ramp up time: 0.05 [sec]

d)

Resistance (series): 1.0 [ohm]
Resistance (parallel): 1.0 [ohm]
Inductance (parallel): 0.1 [H]

Fig. 7.7 Property editing dialog of voltage source.

In this section, the usage of the PSCAD is briefly described. In the next section, another software, Multisim, will be introduced

7.2 Introduction of Multisim

Multism is a general-purpose board-level time domain simulation tool for studying electronic networks. It is based on SPICE program, which stands for Simulation Program Integrated Circuits Especially. Multisim 2001 is a complete system design tool that offers a large component database, schematic entry, full analog/digital SPICE simulation, VHDL/Verilog HDL design entry/simulation, FPGA/CPLD synthesis, RF capabilities, Postprocessing features and seamless transfer to PCB layout packages such as Ultiboard, also from Electronics Workbench. Here only the analog SPICE simulation is used to verify the new proposed simulation algorithm. Fig. 7.8 is the user interface of Multisim 2001.

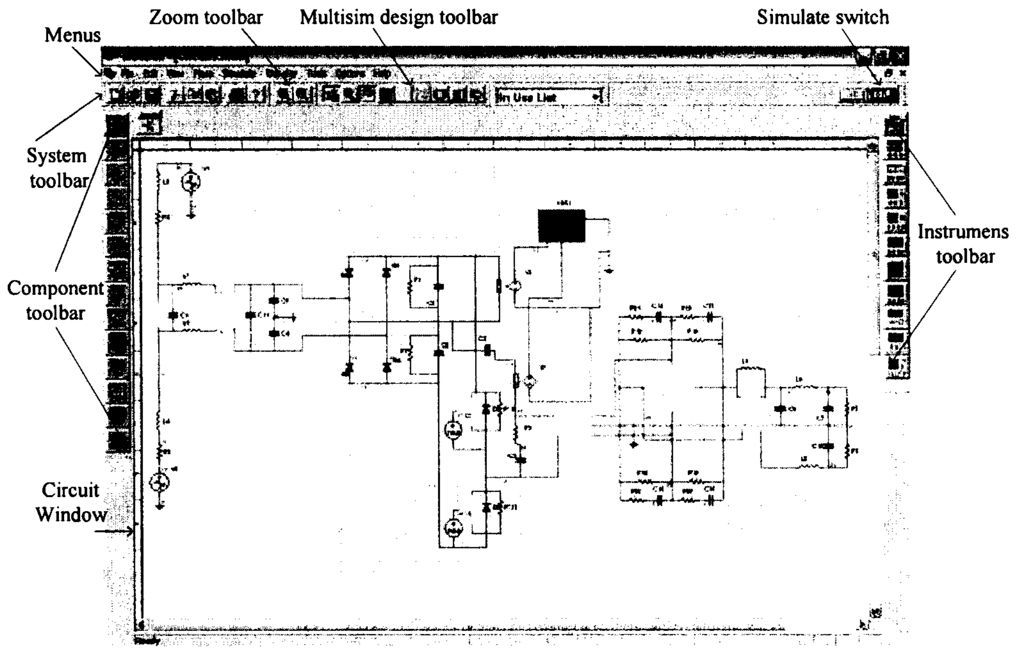


Fig. 7.8 The user interface of Multisim 2001.

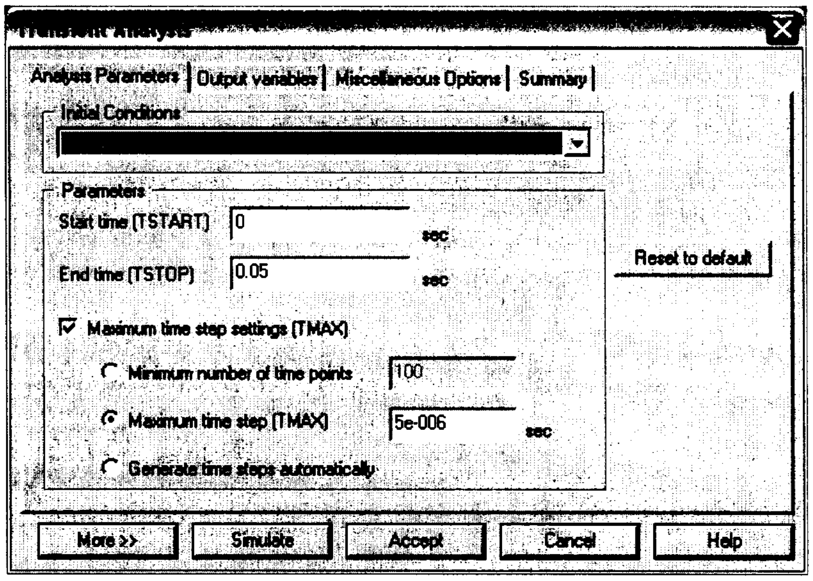
7.2.1 The Setting-up Procedure of the Circuit Schematic

Like all simulation softwares that have graphical user interface, we can obtain the component models from the library, place them on the circuit window in the desired position and orientation, wire them together, modify component properties and then to do the circuit simulation. There are two ways to choose and place a component in Multisim.

On the desired Component toolbar, place the cursor on the desired Parts Bin button and click. The associated Parts Bin appears. From the Parts Bin, click the button for the desired component family. If the selected component is a virtual component, you can simply place the component. For other components, a simplified Browser screen appears by default. To view the full Browser screen click Advanced

You can display the full Browser screen by choosing Place/Place Component. From the Browser screen, select the desired component from the Component List. Information about that component appears.

If you selected the wrong component family from the toolbar, choose the correct component family from the Component family field of the Browser screen. The information in the Select Component area will change accordingly.



Start time (TSTART) 0 sec
End time (TSTOP) 0.05 sec
[✓] Maximum time step settings (TMAX)
( ) Minimum number of time points 100
(●) Maximum time step (TMAX) 5e-006 sec
( ) Generate time steps automatically

Fig. 7.9 Transient analysis dialog

After the setting up the schematic, we can do the multisim simulation by clicking the analysis button on the design bar, and then choose transient analysis button. A transient analysis window will be pop up. Form this window, we can adjust time step, start/end time, output variables and simulation/calculation options. The window is shown in Fig. 7.9.

The Multisim can also set up circuit through forming netlist file (.cir file), which is a free style text file. It describe the circuit to be analyzed by a set of element cards, which define the circuit topology and element values, and a set of control cards, which define the model parameters and the run controls. The first card in the input deck must be a title card, and the last card must be a .END card. The order of the remaining cards is arbitrary. Each element in the circuit is described by one element card that contains the element name, the circuit nodes to which the element is connected, and the values of the parameters that determine the electrical characteristics of the element. The first letter of the element name specifies the element type. For example, a resistor name must begin with the letter R and can contain from one to eight characters; the inductor name must begin with L; the capacitor name must begin with R. Data fields that are enclosed in signs '< >' are optional. All other are required. With respect to branch voltages and currents, Multisim uniformly uses the associated reference convention (current flows in the direction of voltage drop).

7.2.2 Editing Component Properties in Multisim

Like in the PSCAD, the component properties can also be edited in a graphical dialog window in Multisim. For virtual resistor, inductor, and capacitor, the property windows are similar to each other that contain four label tabs, which is shown in the following figure. Only label tab and value tab are useful to do the system transient simulation. Reference ID likes the name of the component that can be name by your will. The value of the component is filled in according to the component in the real circuit.

The element cards of the resistor, inductor, and capacitor have the following general form.

```
RXXXXXXX N1 N2 VALUE <TC=TC1<,TC2>>
CXXXXXXX N+ N- VALUE <IC=INCOND>
LYYYYYYY N+ N- VALUE <IC=INCOND>
```

N1 and N2 are the two element nodes. VALUE is the resistance (in ohms), the capacitance in Farads, or the inductance in Henries. TC1 and TC2 are the temperature coefficients, which are optional; if not specified, zero is assumed for both. IC is the initial capacitor voltage in Volts or the initial inductor current in Amps that flows from N+, through the inductor, to N-. The value of the resistor as a function of temperature is given by:

value(TEMP) = value(TNOM)×(1+TC1×(TEMP-TNOM)+TC2×(TEMP-TNOM)×2))

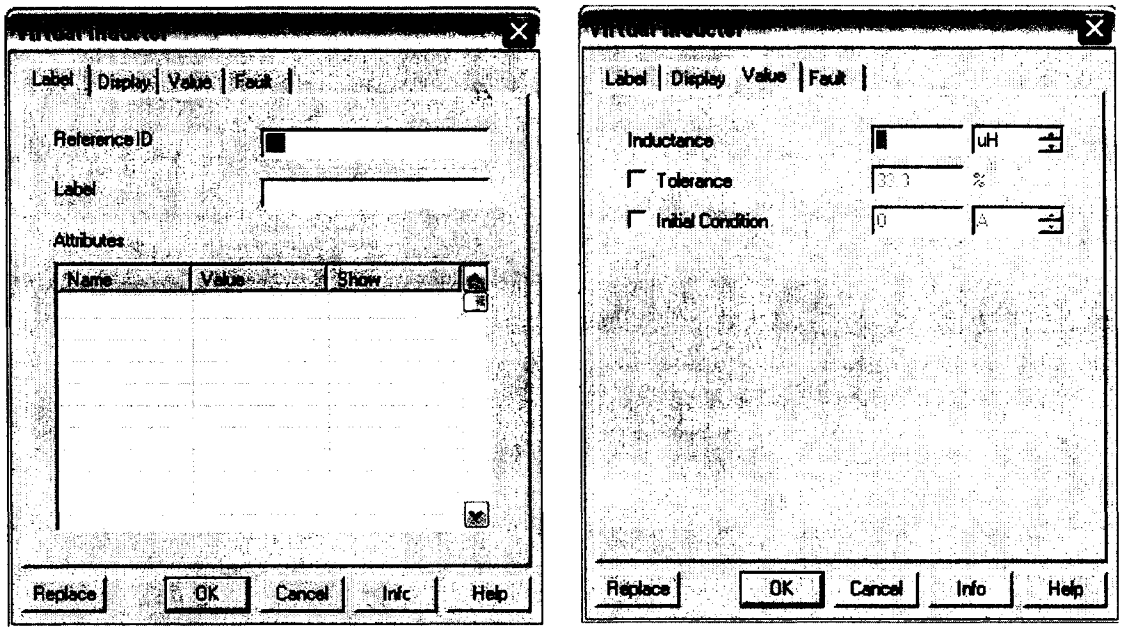


Fig. 7.10 Property screens of virtual inductor

For a diode, the property screen is similar with the resistor, inductor, and capacitor. Because the characteristics of the diode is determined by a bunch of parameter, Multisim will open a new screen to change the diode parameter by click button "Edit Model". In the new screen, the editable parameters of the diode are shown, which are listed in the Table 7.1.

The element card of diode has the following general form

DXXXXXXX N+ N- MNAME <AREA> <OFF> <IC=VD>

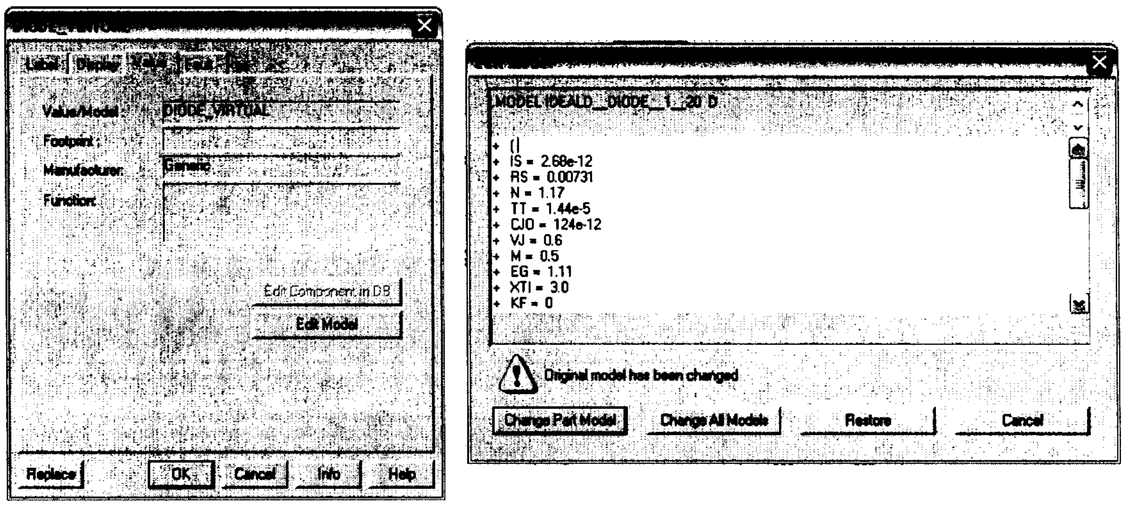
N+ and N- are the positive and negative nodes, respectively. MNAME is the model name, AREA is the area factor, and off indicates a (optional) starting condition on the device for dc analysis. The characteristic of the specified diode model is determined by its .Model card.

Table 7.1 Diode parameters and typical value

Symbol	**Parameter Name**	**Typical Value**	**Unit**
IS	Saturation current (IS)	1e-9 - 1e-18	A
RS	Ohmic resistance(rS)	10	W
CJO	Zero-bias junction capacitance (Cj(0))	0.01-10e-12	F
VJ	Junction potential (Φ 0)	0.05-0.7	V
TT	Transit time (τ D)	1.0e-10	s
M	Grading coefficient (m)	0.33-0.5	-
BV	Reverse bias breakdown voltage	-	V
N	Emission coefficient (n)	1	-
EG	Activation energy	1.11	eV
XTI	Temperature exponent for effect on IS	3.0	-
KF	Flicker noise coefficient	0	-
AF	Flicker noise exponent	1	-
FC	Coefficient for forward-bias depletion capacitance formula	0.5	-
IBV	Current at reverse breakdown voltage	1.0e-03	A
TNOM	Parameter measurement temperature	27-50	°C

MODEL MNAME TYPE(PNAME1=PVAL1 PNAME2=PVAL2 ...)

MNAME is the model name, and type is one of the following seven types: NPN (NPN BJT model), PNP (PNP BJT model), D (Diode model), NJF (N-channel JFET model), PJF (P-channel JFET model), NMOS (N-channel MOSFET model), and PMOS (P-channel MOSFET model).



Value/Model: DIODE_VIRTUAL

Footprint:

Manufacturer: Generic

Function:

```
.MODEL IDEALD__DIODE__1__20 D
+ (
+ IS = 2.68e-12
+ RS = 0.00731
+ N = 1.17
+ TT = 1.44e-5
+ CJO = 124e-12
+ VJ = 0.6
+ M = 0.5
+ EG = 1.11
+ XTI = 3.0
+ KF = 0
```

Original model has been changed

Fig. 7.11 Property screens of virtual diode

Parameter values are defined by appending the parameter name, as given in Table 7.1 for each diode type, followed by an equal sign and the parameter value. Model parameters that are not given a value are assigned the default values given below for each model type.

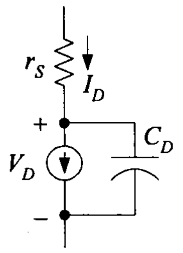


Fig. 7.12 Diode model used in multisim

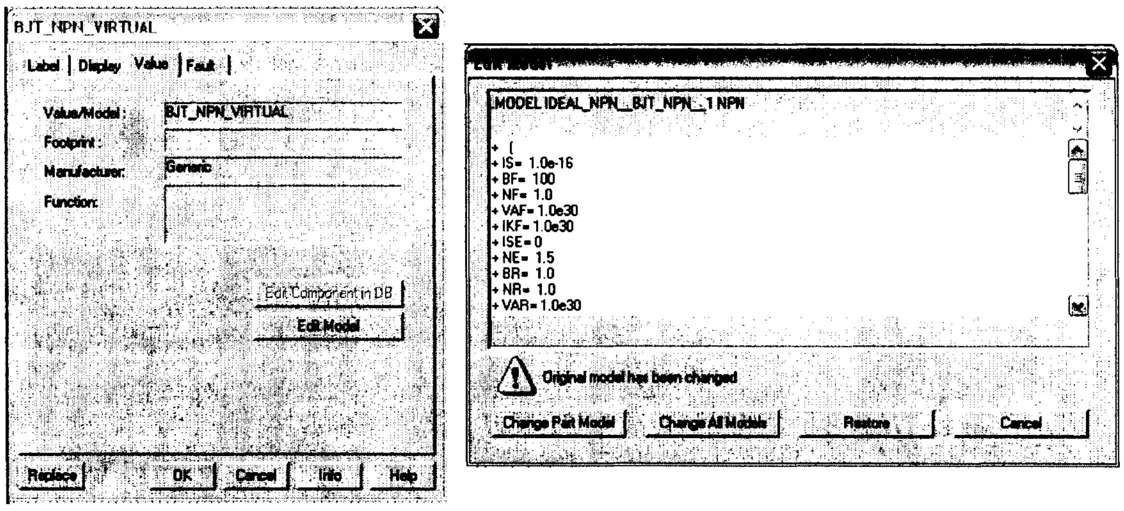
This model defines the operation of the diode, taking into account its charge-storage effects of capacitance. There are two types of capacitances: diffusion or storage capacitance, and depletion or junction capacitance. The mathematic description of this model is shown as the following equaitons

$$I_D = \begin{cases} I_S\left(e^{V_D/nV_T} - 1\right) + V_D G_{\min} & for\ V_D \geq -5nV_T \\ I_S + V_D G_{\min} & for\ -BV \leq V_D \leq -5nV_T \\ -IBV & for\ V_D = -BV \\ -I_S\left(e^{-(BV+V_D)/V_T} - 1 + \dfrac{BV}{V_T}\right) & for\ V_D \leq -BV \end{cases}$$

$$C_D = \begin{cases} \tau_D \dfrac{dI_D}{dV_D} + C_j(0)\left(1 - \dfrac{V_D}{\phi_0}\right)^{-m} & for\ V_D \leq FC \times \phi_0 \\ \tau_D \dfrac{dI_D}{dV_D} + \dfrac{C_j(0)}{F_2}\left(F_3 - \dfrac{mV_D}{\phi_0}\right) & for\ V_D \geq FC \times \phi_0 \end{cases}$$

$$\text{where} \begin{cases} F_2 = (1 - FC)^{1+m} \\ F_3 = 1 - FC(1+m) \end{cases}$$

The property screen of BJT is complete same as the diode one. In the screen for edit model, all editable parameters are shown up, which are listed in Table 7.2



Value/Model: BJT_NPN_VIRTUAL
Footprint:
Manufacturer: Generic
Function:

```
.MODEL IDEAL_NPN__BJT_NPN__1 NPN

+ (
+ IS= 1.0e-16
+ BF= 100
+ NF= 1.0
+ VAF= 1.0e30
+ IKF= 1.0e30
+ ISE= 0
+ NE= 1.5
+ BR= 1.0
+ NR= 1.0
+ VAR= 1.0e30
```

Fig. 7.13 Property screens of virtual BJT

Fig. 7.14 shows the *Gummel-Poon* model of an *npn* BJT. This model is represented by two groups of diodes connected end to end and a dependent non-linear current source, in parallel with these diodes. The resistances(r_C, r_B, r_E) in the figure represent the transistor's ohmic resistance from its active region to its collector, base and emitter terminals, respectively. These three ohmic resistances improve the dc characterization of the model. Charge storage effects in a BJT can be modeled with the help of three types of capacitors: two nonlinear junction capacitors (C_{JE}, C_{JC}), two nonlinear diffusion capacitors(C_{TE}, C_{TC}) and a constant substrate capacitor(C_{CS}). The diode D1 and D2 are used to represent the base-collector and base-emitter junctions, which are satisfied the ideal diode equations.the diode D3 and D4 are used to represent the beta variation with current, which are not satisfied the ideal diode equations[56].

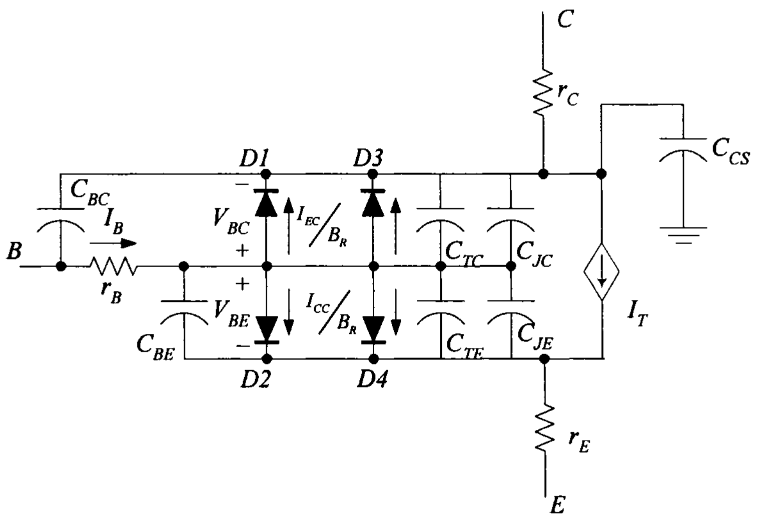


Fig. 7.14 *Gummel-Poon* model of BJT

The element card of the BJT is similar to that of the diode, and the characteristics of the BJT also defined by a specified .Model card. The parameter names that are used in Model. Card are listed in Table 7.2.

NC, NB, and NE are the collector, base, and emitter nodes, respectively. NS is the (optional) substrate node. If unspecified, ground is used.

General form:

QXXXXXXX NC NB NE <NS> MNAME <AREA> <OFF> <IC=VBE,VCE>

Table 7.2 BJT parameters and typical value

Symbol	Parameter Name	Example	Unit
IS	Saturation current	1e-15	A
bF	Forward current gain coefficient	100	-
bR	Reverse current gain coefficient	1	-
rb	Base ohmic resistance	100	W
re	Emitter ohmic resistance	10	W
rc	Collector ohmic resistance	1	W
Cs	Substrate capacitance	1	F
Ce, Cc	Zero-bias junction capacitances	2e-09	F
fe,fc	Junction potentials	0.75	V
tF	Forward transit time	1e-13	s
tR	Reverse transit time	10e-09	s
me, mc	Junction grading coefficients	0.5	-
VA	Early voltage	200	V
Ise	Base emitter leakage saturation current	1e-13	A
Ikf	Forward beta high-current knee-point	0.01	A
Ne	Base-emitter leakage emission coefficient	2	-
NF	Forward current emission coefficient	1	-
NR	Reverse current emission coefficient	1	-
VAR	Reverse early voltage	200	V
IKR	Reverse beta roll-off corner current	0.01	A
ISC	B-C leakage saturation current	0.01	A
NC	B-C leakage emission coefficient	1.5	-
IRB	Current for base resistance equal to (rb+RBM)/2	0.1	A
RBM	Minimum base resistance at high currents	10	W
XTF	Coefficient for bias dependence of tF	0	-
VTF	Voltage describing VBC dependence of tF	-	V

Continued Table 7.2

Symbol	Parameter Name	Example	Unit
ITF	High current dependence of tF	-	A
PTF	Excess phase at frequency equal to 1/(tF×2PI) Hz	-	Deg
XCJC	Fraction of B-C depletion capacitance connected to internal node	-	-
VJS	Substrate junction build-in potential	-	V
MJS	Substrate junction exponential factor	0.5	-
XTB	Forward and reverse beta temperature exponent	-	-
EG	Energy gap for temperature effect on IS	-	eV
XTI	Temperature exponent for effect on IS	-	-
KF	Flicker noise coefficient	-	-
AF	Flicker noise exponent	-	-
FC	Coefficient for forward-bias depletion capacitance formula	-	-
TNOM	Parameter measurement temperature	50	°C

In the Multisim 2001 personal edition, there is no separate mutual inductor component in the master library. In order to represent the two mutual coupled inductors, the user-defined component should be created, which is a subcircuit block. The .SUBCKT card of mutual coupled inductors is shown as follow

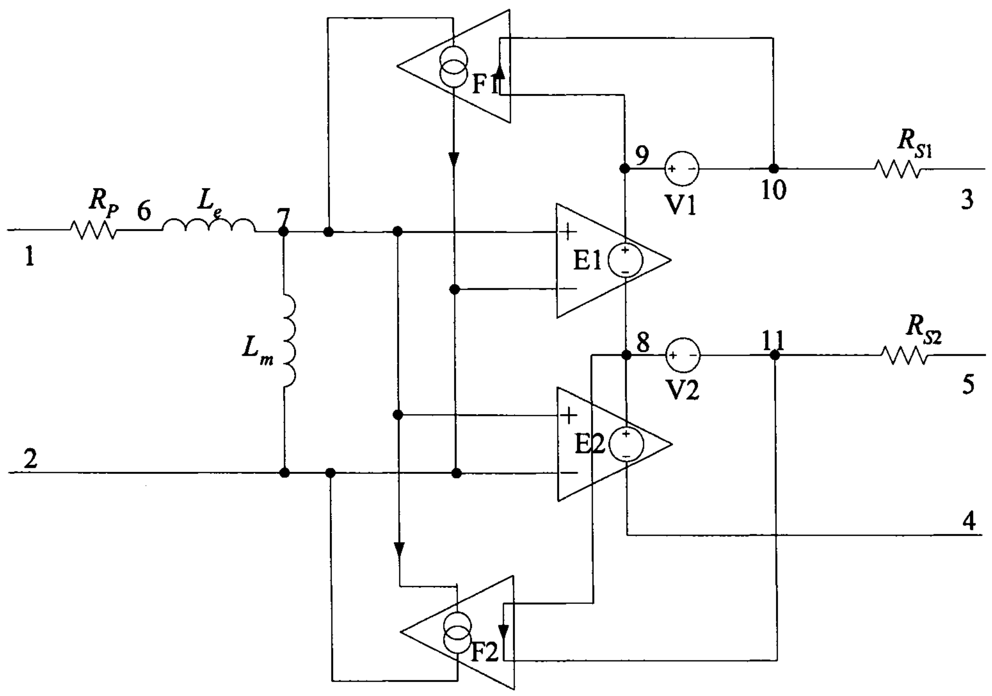
```
.SUBCKT subname 1 2 3 4
*   *1, 2-- primary winding,   *3,4-- secondary terminal
Rs1 1 11 1e-003
Rl2 31 3 1e-003
L1   11 2 3e-003
L2   31 4 3e-003
K12  L1 L2 9.99e-001
.ENDS
```

A subcircuit definition is begun with a .SUBCKT card. subname is the subcircuit name, and 1,2,3,4 ... are the external nodes, which cannot be zero. Second line is an explain line. The following group of element cards define the subcircuit. The last card in a subcircuit definition is the .ENDS card, which means the definition of the subcircuit is ended here.

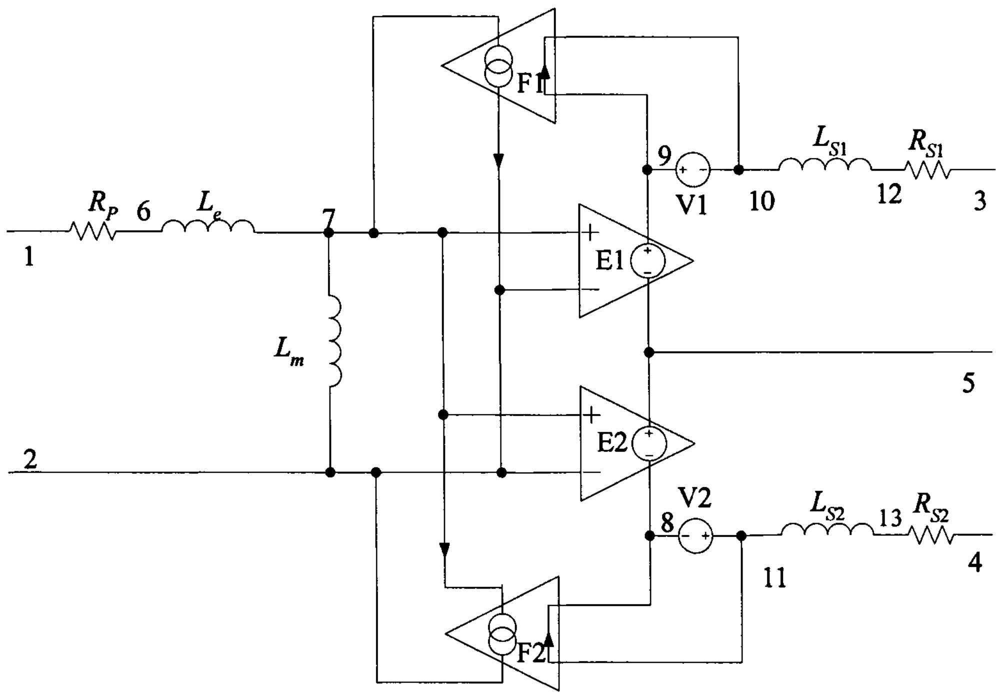
In the Multisim 2001 personal edition, there are two types of central tapped transformer models. One model treats the central tapped transformer as three mutual coupled inductors, such as the transformer model TS_PQ4_10; the other includes an ideal transformer model in it, such as TS_VIRTUAL. Fig. 7.15 a) is the equivalent circuit of TS_VIRTUAL. When the load of the pin 5 is ground and pin 3 and 4 have different load, the current flowing into the pin 1 is not correct. In order to get the correct value, the equivalent should be changed as Fig. 7.15 b). The description of this circuit is

```
.SUBCKT TRbase8 1 2 3 4 5
* EWB Version 4 - TRbase8ransformer Model
* n= 10 Le= 3-006 Ls1=3e-9 Ls1=3e-9 Lm= 6e5 Rp= 10e-007 Rs= 5e-007
    Rp    1   6 10e-007ohm
    Rs1 12    3 5e-007ohm
    Rs2 13    4 5e-007ohm
    Le    6   7 3e-006H
    Lm     7   2 6e5H
    Ls1     10    12 3e-009H
    Ls2     11    13 3e-009H
    E1    9   5   7   2 0.05
    E2    5   8   7   2 0.05
    V1    9 10 DC 0V
    V2    11 8 DC 0V
    F1    7   2 V1 0.05
    F2    7   2 V2 0.05
```

.ENDS



a)



b)

Fig. 7.15 The model of center taped transformer

Now, The components that will be used for SMPS circuit simulation have been given out. In the next section, the simulation results will be compared.

7.3 Simulation Results Comparison

In this section, the efficiency of the proposed simulation algorithm is verified by comparing its results with the simulation results of two commercial softwares, PSCAD and Multisim. All device models in PSCAD used for results comparison are linear models; all semiconductor devices in Multisim used for results comparison are nonlinear models and other device models are linear models. In the PSCAD, the trapezoidal rule are used, and in the Multisim, the gear rule are used. The following table gives out the component .

Table 7.3 The components used in the circuit

Resistor			
Name	**Resistance (ohm)**	**Name**	**Resistance(ohm)**
R_{in1}, R_{in2}	0.01	R_1	510k
R_2, R_3	330k	R_{V21}	47
R_{V22}, R_{V23}, R_{V24}, R_{V25}	4.7	R_{L5}	0.3
R_{LN5}	1		
Capacitor			
Name	**Capacitance (F)**	**Name**	**Capacitance (F)**
C_1, C_2	220p	C_3, C_4	2p
C_5, C_6	680μ	C_7	2 μ
C_8	2p	C_9, C_{10}	3.3m
C_{11}	470 μ	C_{12}, C_{13},C_{14}, C_{15}	10p
Inductor			
Name	Inductance (H)	Name	Inductance (H)
L_{in1}, L_{in2}, L_4, L_5	1 μ	L_1, L_2, L_3	10 μ

Continued Table 7.3

Mutual Inductor			
Name	**Self Inductance (H)**	**Coupled coefficient**	**Resistance(ohm)**
M_{L1}	3m, 3m	0.999	1m,1m
T_{R2}	6 μ, 6 μ	0.983	1 μ, 1 μ
Three-winding transformer			
Name (Ratio)	**Leakage Inductance (H)**	**Magnetic Inductance (H)**	**Resistance(ohm)**
T_{R1} (1:20)	3 μ, 3p, 3p (1,2,3)	30m	1μ, 0.5μ, 0.5μ (1,2,3)
Diode/BJT			
Name	**Model**	**Name**	**Model**
D_1-D_4	In5406	D_5-D_6	Diode-Virtual
D_7-D_{10}	MBRB2545	Q_1,Q_2	2SC4242

Before starting the simulation, the linearized characteristics of the power electronic switch should be obtained first for the PSCAD and new proposed simulation algorithm.

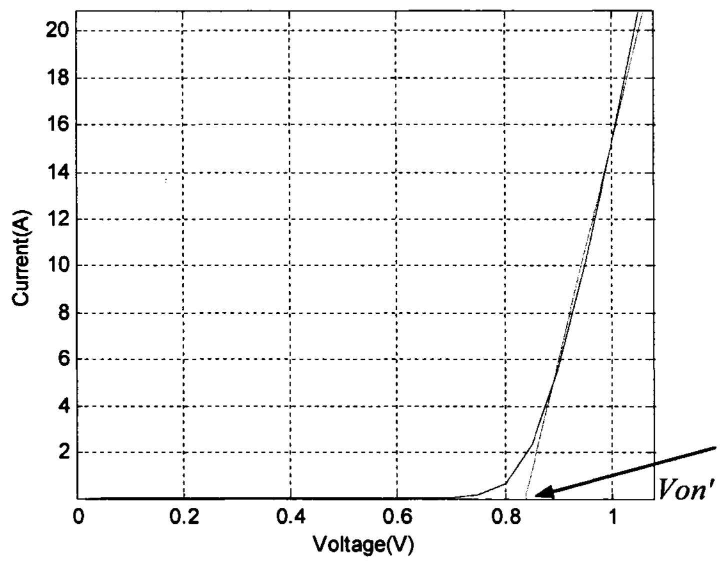


Fig. 7.16 The characteristics curve of diode and its linearized curve

The blue curve in the Fig. 7.16 is the V-I curve obtained from Multisim DC swipe. If the current flows through this device is in the range of [2A, 20A] at the most of the on time period, the R_{ON} and V_{ON} are obtained from the curve between 2A and 20A. The V-I characteristics for switches used are shown in Fig. 7.17. According to the linearization method described before, the equivalent R_{ON}, and V_{ON} for each semiconductor switch is listed in Table 7.4.

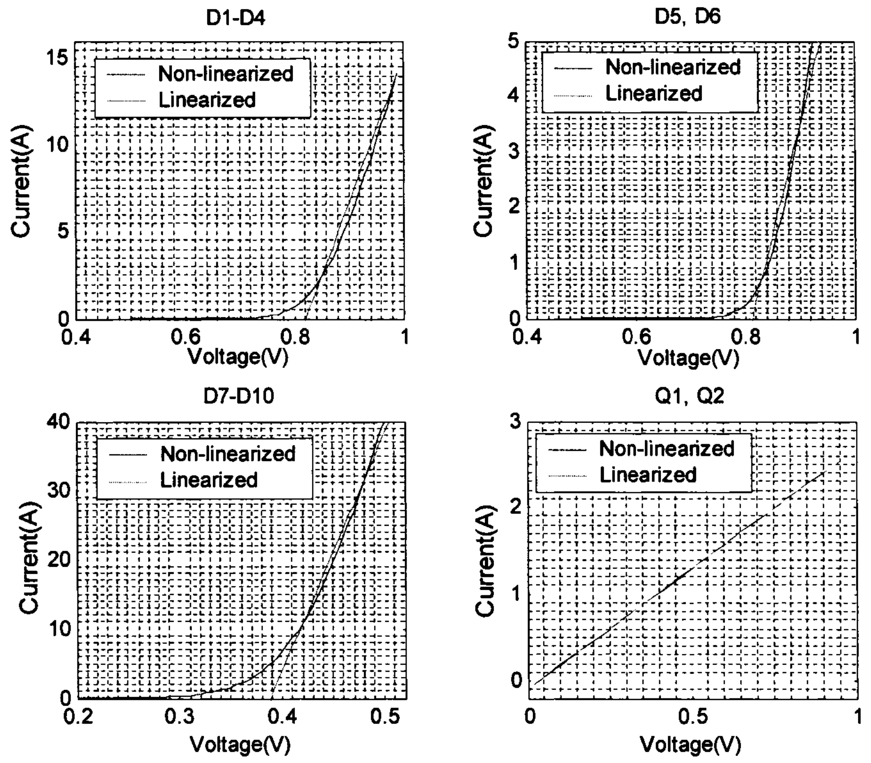


Fig. 7.17 Linearized characteristics of semiconductor switches

Table 7.4 Linearized switch parameters

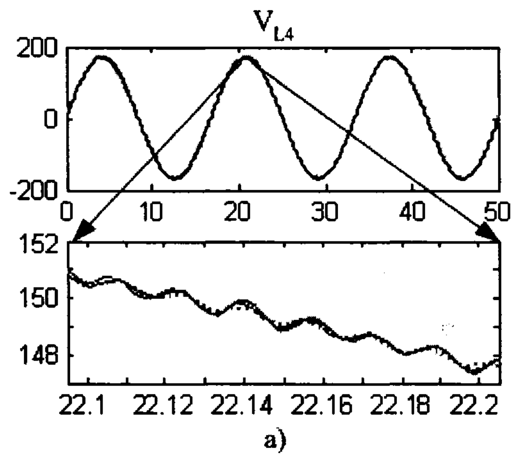
Name	**Linearized Condition**	**Von(V)**	**R_{ON}(Ohm)**
D1-D4	Current between (2A,14A)	0.8217	0.0119
D5, D6	Current between (0.6A, 3A)	0.8131	0.0253
D7-D10	Current between (10A, 30A)	0.3883	0.00293
Q1, Q2	Current between (0.6A, 3A) Vgate=5V	0.0349	0.3601

In order to make comparison more completely, the simulation results between time interval T_{sim}=[0, 50ms], which include both transient and steady state responses, are compared. In the first part of time interval T_{sim}, the circuit is in transient state; but in the end part of T_{sim}, the circuit is in steady state. There the steady state does not mean the system states do not change, it means the system response is a periodic response. The duty cycle the switches (Q1 and Q2) is set to be 0.45. So the combined effect of these two switches has the duty cycle of 0.9. The PWM frequency is 25kHz. The simulation parameters used to do the SMPS circuit are listed.

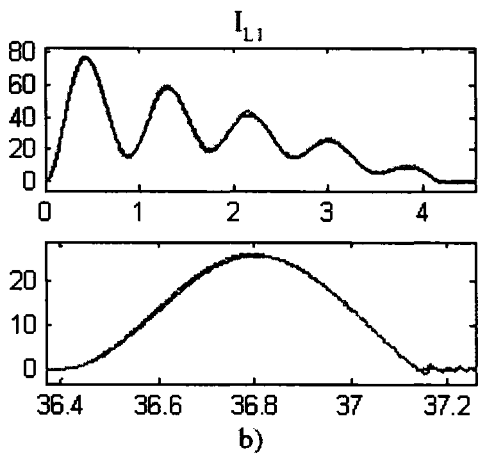
Table 7.5 Simulation parameter

Program	**Time step**		**Discrete time rule**
	L-sub	**R-sub**	
PSCAD	0.2μs		Trapzoidal
Multisim	flexible		Gear
New Method	2μs	0.2μs	Generalized TLM model

When there is a forced commutation in the system, generalized TLM stub model associated with Back Euler rule is used for one time step to depress the numerical oscillation, and then return to regular TLM stub model. The new method also adopts voltage evaluation algorithm III to evaluate the decoupling capacitor voltage.



a)



b)

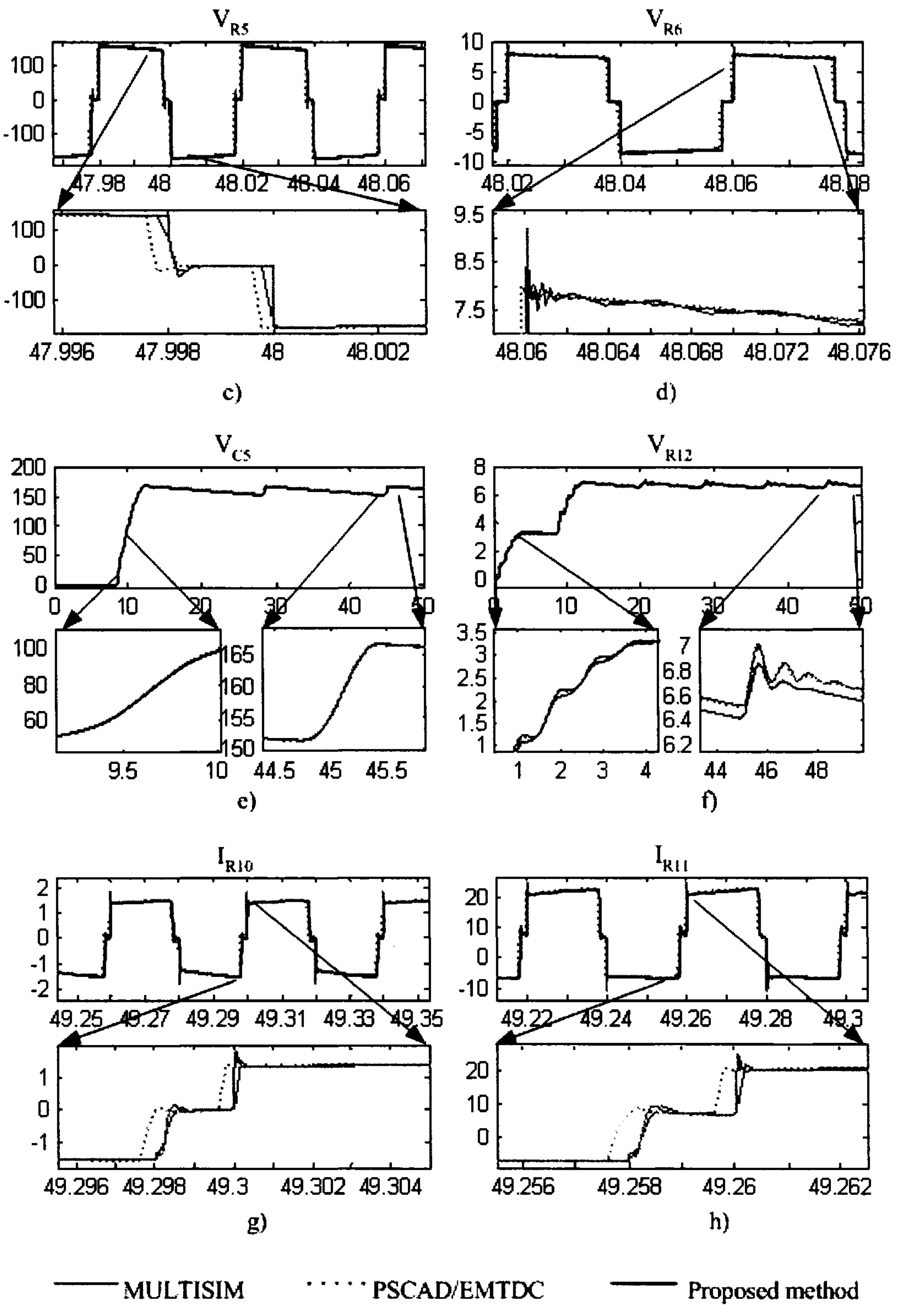


Fig. 7.18 Verification of proposed method.

From now on, if no speciasified, the units in figures are *V* for voltage curve, *A* for current curve, and *ms* for time. In Fig. 7.18 a) and b), there are waveforms of input voltage and current. The waveforms are covered with each other, even the high frequency components are very similar to each other. The PWM frequency is 25kHz, so the period of each voltage and current pulse is 40 μs, which is clearly shown in the Fig. 7.18c), d), g), and h). Due to the effect of the generalized TLM stub model, the spike in the simulation results of the New algorithm is not so high as the results of MULTISIM, but similar to the simulation results of the PSCAD/EMTDC, which adopts the ADC models associated with Back Euler rule under some circumstances. Due to the linearized method I used, in the working region, the voltage drop of the real switch always larger or equal to the voltage drop of the linearized switch model. So the output voltage generated by new proposed method and PSCAD/EMTDC are a little bit higher than that of MULTISIM. The comparison shows that the new algorithm gives out accurate results, no matter at input part, high frequency transformer part, and DC output part.

All figures show the three simulation results are perfectly matched to each other, and the validity of the proposed simulation algorithms, which include TLM device model, new switch model, and decoupling algorithm III, is proved.

7.4 The Comparison Between Two Decoupling Algorithms.

In this section, the decoupling algorithm I and III are compared. As mentioned in chapter 3, the voltages over the decoupling capacitors keep constant during simulation time interval of L-subnetwork in algorithm I; the voltages over the decoupling capacitors keep changing according to the current flowing into capacitors from the both subnetwork or the voltage changes at both sides of the capacitors during simulation time interval of L-subnetwork in algorithm III. In order to increase voltage changes over decoupling capacitors, the capacitance of C5 and C6 is decreased to 100μF, and all other parameters in the circuit keep unchanged.

Table 7.6 The simulation conditions of three comparing simulation results

Program	Time Step		Decoupling Algorithm
	L-sub	R-sub	
Multisim	Flexible (Trapzodial)		No
Case 1	50μs	0.2μs	Algorithm III
Case 2	50μs	0.2μs	Algorithm I

The Table 7.6 gives out the simulation conditions of four comparing simulation results. Eqns. (7.1) are used to adjust the voltage value; (7.2) are used to get the incident wave for the adjusted voltage value; and (7.3) are used to obtain the incident wave for the next time step.

$$V_{new}(t_0) = V_{old}(t_0) + \Delta v(t_0) = V_{old}(t_0) + \Delta_L v(t_0) + \Delta_R v(t_0) \tag{7.1}$$

$$\begin{aligned} {}_R V_{new}^i(t) &= \left(V_{new}(t_0) - i_R(t_0)_R Z_{(sb)}\right)/2 \\ {}_L V_{new}^i(t) &= \left(V_{new}(t_0) - i_L(t_0)_R Z_{(sb)}\right)/2 \end{aligned} \tag{7.2}$$

$$\begin{aligned} {}_L V^i(t_0 + T) &= V_{new}(t_0) - {}_L V_{new}^i(t_0) \\ {}_R V^i(t_0 + T) &= V_{new}(t_0) - {}_R V_{new}^i(t_0) \end{aligned} \tag{7.3}$$

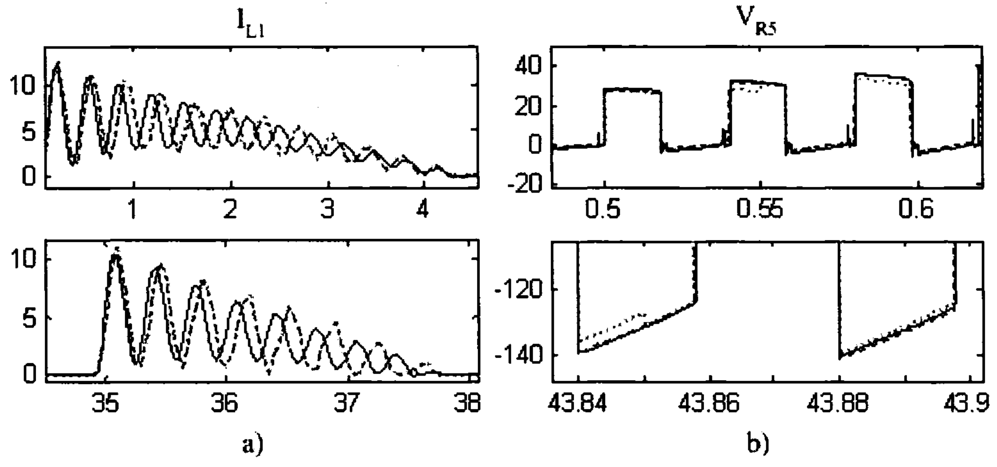


Fig. 7.19 Comparison of two decoupling algorithm

From the Fig. 7.19, we can easily find that both decoupling algorithm introduce current phase error in L-sub, and the decoupling algorithm I introduces a voltage jump at the node 5 of R-sub at the time of the energy exchanging. When the system is in steady state due to the stable voltage over the decoupling capacitors, there is no significant difference between two algorithm,

7.5 The Comparison Between Two TLM Model

In this section, two types of TLM models are compared. As mentioned before, the conventional TLM models of electrical components are equivalent to the ADC models associated with trapezoidal rule; the new TL models are equivalent to the ADC models associated with weight-averaged integration rule. The efficiency of the new TL models are proved by the comparisons between 4 cases.

Table 7.7 The simulation conditions of four comparing simulation results

Program	**Time Step**		**Modeling algorithm**
	L-sub	**R-sub**	
Case 1	2μs	0.2μs	Generalized TLM model
Case 2	20μs	0.4μs	Generalized TLM model
Case 3	2μs	0.2μs	Conventional TL model
Case 4	20μs	0.4μs	Conventional TL model

From Fig. 7.20, we can easily find that the for the variable with a smooth trajectory, no matter which modeling algorithm is adopted, the simulation results are all satisfied. Fig. 7.20 a) and b) are the input voltage and current waveforms. In the AC/DC part where there is no forced commutation, . The oscillation in the zoomed figure is really existed high frequency components. When the time step is too large, the simulation results just cannot accurately show the real high frequency component of the waveform. So no matter what time step is used, there is no numerical oscillation in the waveform.

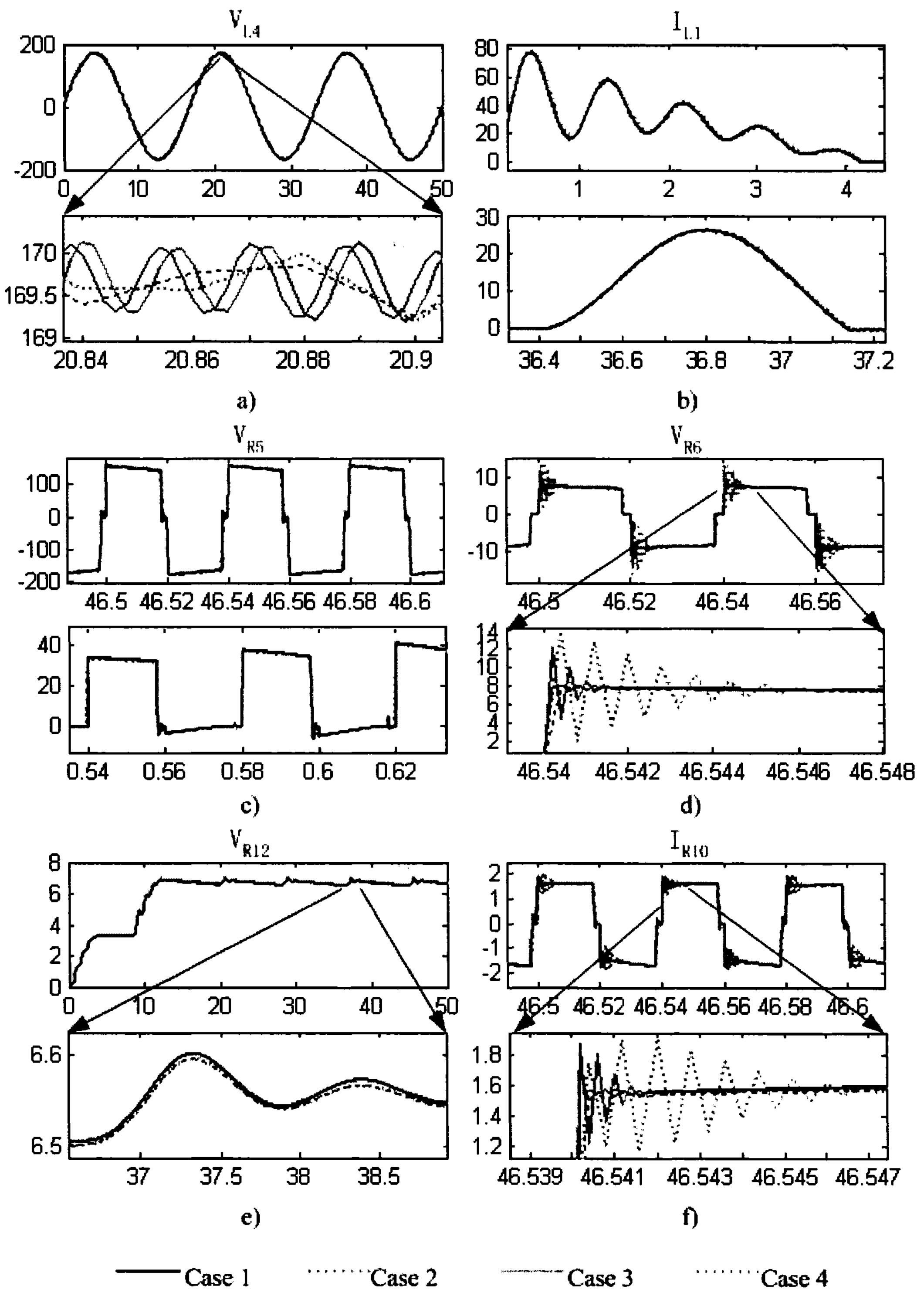


Fig. 7.20 Comparison of two TL models

Fig. 7.20 d) and f) are the output voltage and current waveforms of high frequency transformer. They are affected by the forced commutation of the Q_1 and Q_2. There are numerical oscillation in the waveform of case 3 and 4, which are corresponding to the tranditional TLM model. The oscillation time increases with the increasing of the time step. But in the case 1 and 2, which are corresponding to the generalized TLM model, there is no numerical oscillation. That proves that the generalized TLM stub models can really depress the numerical oscillation.

7.6 EMI Filter Performance

The simplified SMPS circuit contains three parts, EMI filter, AC-DC rectifier, and DC-DC converter. The electromagnetic interference (EMI) filter locates between the main power supply and AC-DC rectifier. It contains series inductive (load-bearing) and parallel capacitive (nonload-bearing) components, which provides a low impedance path around the protected circuit for high frequency noise. The main purpose of the EMI filter is to limit the interference, which is conducted or radiated from the power circuit. Filters also attenuate impulses since a Fourier's Analysis of a spike will reveal it is composed of high frequency waveforms. Excessive conducted or radiated interference can cause erratic behavior in other systems which are in close proximity, or which share an input source with the power circuit. If this interference affects the power circuit, it can cause erratic operation, excessive ripple, or degraded regulation, which can lead to system level problems. Input EMI filters may also be used to limit inrush current, reduce conducted susceptibility, and suppress spikes. The main source of the CM noise is the capacitance of the switch Q_1 and Q_2 in Figure 6.2. The main source of the DM noise current comes from the high frequency transformer T_{R1}, which produces many switching harmonics currents. Since the noise sources are more like current sources, the reference impedance used to measure their action takes on importance. International standards have defined reference impedance on which the measurements will be made. This impedance is guaranteed by a Line Impedance Stabilization Network (LISN) and precisely defined by CISPR16 document. Fig. 7.21 portrays how this device is made. It offers a 50 Ω impedance over the frequency of interest (e.g.

150 kHz–30 MHz for CISPR22) and shields the measurement against unwanted incoming noises.

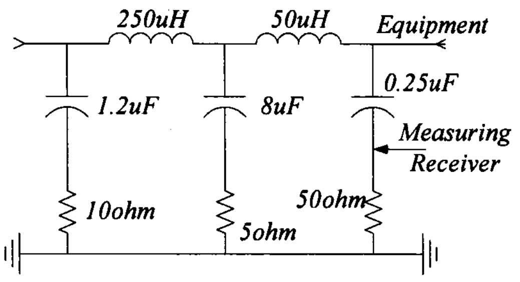
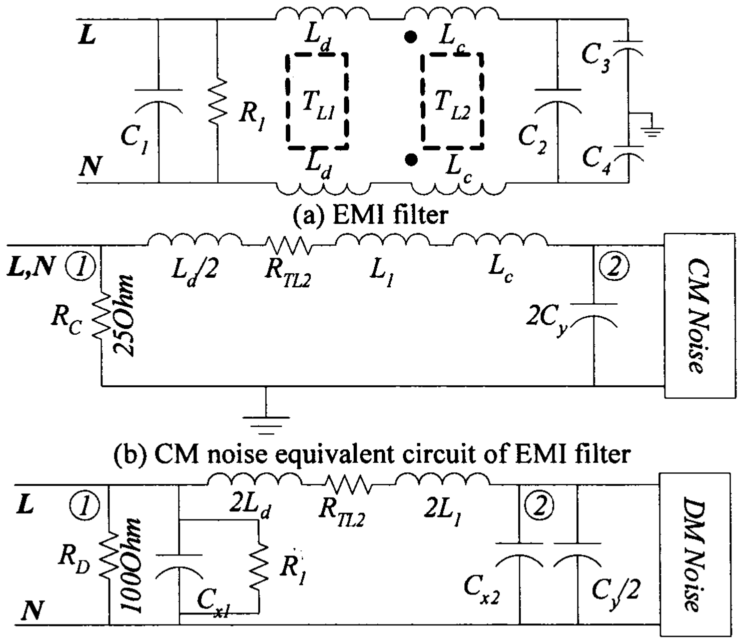


Fig. 7.21 LISN circuit.

In the simulated SMPS circuit, the pair inductors L_1, L_2, the mutual coupled inductance M_{L1}, capacitor C_1, C_2, C_3 and C_4 , and resistor R_1 form the EMI filter. It is shown in Fig. 6.2. L_1 and L_2 is a differential-mode choke and M_{L1} is a common-mode choke, C_1 and C_2 are DM capacitors (called "X" capacitors), C_3 and C_4 is a common-mode capacitor (called a "Y" capacitor). Different types of conducted noise should be dealt with by different parts of a conducted EMI filter as indicated by Fig. 7.22 b and c, which take into account the effect of the 50 ohms input impedance of the LISN.

Due to the small leakage current limit on the grounded capacitor, the Y capacitor is supposed to be very small compared with the X capacitor. Therefore, the inductance of common-mode choke L_1 and L_2 should be much larger than the differential-mode choke M_{L1}. It is found that the leakage inductance due to the coupling imperfection of a practical CM choke also has a filtering effect on the DM noise by observing the equivalent filter circuit in Fig. 6.7. In practice, the magnitude of leakage inductance in a CM choke is usually about 0.1% to 1% of the CM inductance. In Fig. 7.22 b, the CM noise is mainly affected by the parallel effects of both "Y" capacitors and the self-inductances of two CM inductors, though the leakage inductance of CM inductors and DM inductors also take negligible effects. In Fig. 7.22 c, both the inductance L_d of the DM choke and the leakage inductance L_l of the CM choke can attenuate the DM noise.



(c) DM noise equivalent circuit of EMI filter

Fig. 7.22 Conducted EMI filter adopted in SMPS here

Although the two "Y" capacitors also affect the DM noise, their effect on DM noise attenuation is negligible in comparison with that of the two Y capacitors with large capacitance. In some cases, if the leakage inductance L_l of CM choke (M_{L1}) is large enough to be DM chokes, the L_d shown in Fig. 7.22 c can be omitted.

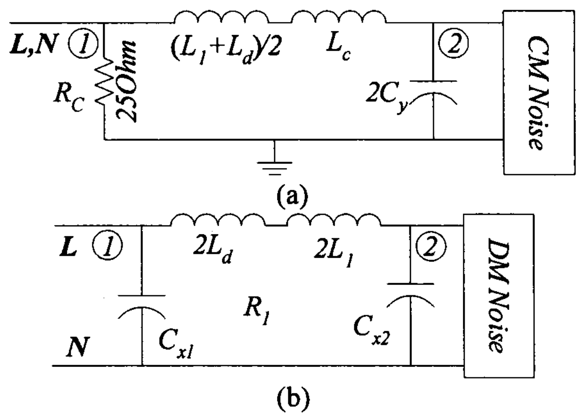
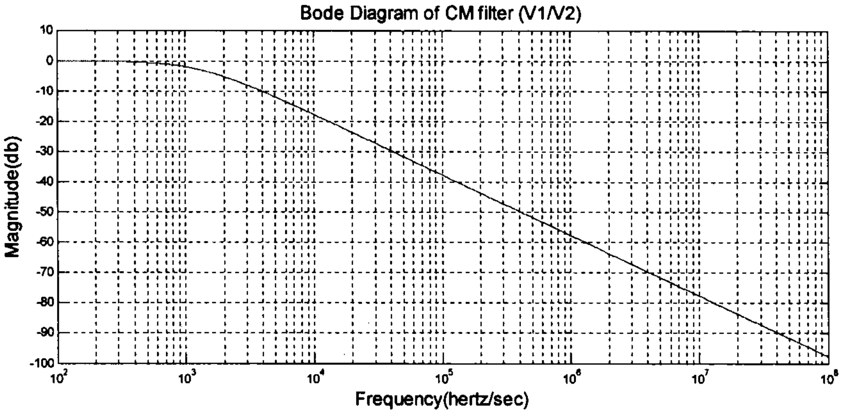


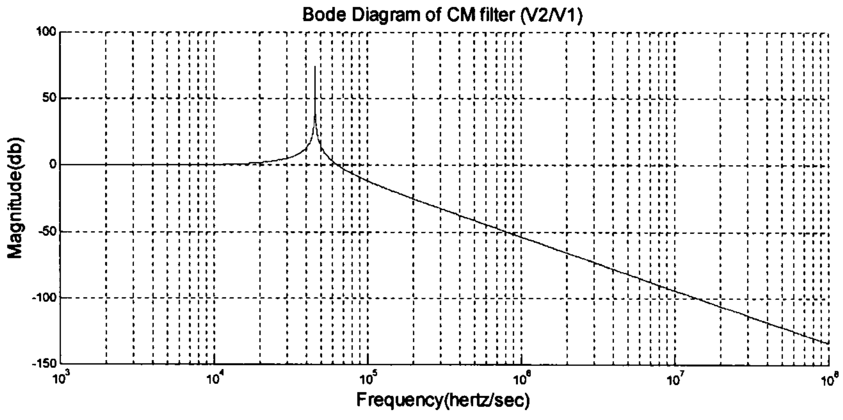
Fig. 7.23 Simplified EMI filter

Over the frequencies of interest, the impedance of Cx is much smaller than the impedance of LISN, the impedances of two chokes are much larger than the resistance of the two chokes. So the two filters can be simplified as two circuits in Fig. 7.23.

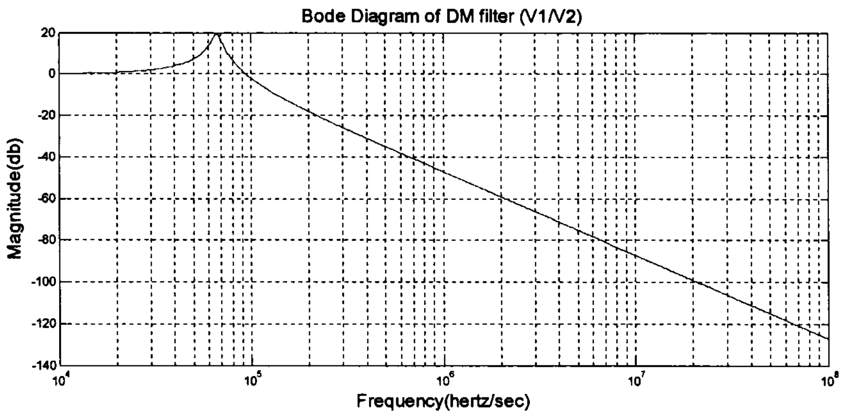
Fig. 7.24 gives out the Bode plots of the two filters. Two tests have been done to each type of filters. One is to apply noise source at node 2 of each equivalent filter circuit, then the capability of EMI filter to attenuate the noise from SMPS is given out. The other is to apply noise source at node 1 of each equivalent filter circuit, then the capability of EMI filter to attenuate the noise from the power source is given out.



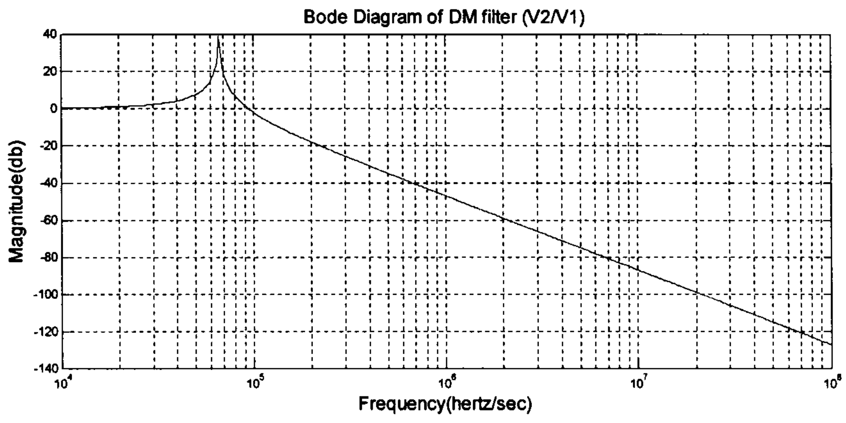
(a)



(b)



(c)



(d)

Fig. 7.24 Bode diagram of EMI filter

When the CM noise applied at SMPS side, it means the CM noise applied at node 2 of CM noise filter. From the circuits we know the filter is just a first order filter, the corner frequency is about 1.32 kHz. The attenuation for the interested frequencies (150 kHz-30 MHz) is from –41 db to –87 db. When the CM noise applied at the main power supply side, it means the CM noise applied at node 1 of CM noise filter. The corner frequency of this second order filter is 45.89 kHz, and the gain is 74.42 db, which is very large due to the lack of the damping factor. The attenuation for the interested frequencies is from –19.72 db to –112.6 db. When the DM noise applied at SMPS side, it means the DM noise applied at node 2 of DM noise filter. The corner frequency is about 66.5 kHz, and the gain is 19.28 db. The attenuation for the interested frequencies is from –12.23 db to –106 db. When the DM noise applied at main power supply side, it means the DM noise applied at node 2 of DM noise filter. Due to the symmetric characteristics, the corner frequency is also about 66.5 kHz, and the gain is 98.16 db, which is very high due to the lack of the damping resistor. The attenuation for the interested frequencies is from –12.26 db to –106 db. Those results show that the filter has effects for noise come from both sides.

7.7 Conclusion

The comparisons that have been done in this chapter show the efficiency the new proposed simulation algorithm. The TLM stub model really can decouple the total network and give out the satisfied results; The combing TL modeling algorithm can really improve the simulation efficiency and accuracy; and the CT technique can give out the correct system initial condition after forced commutation. The EMI filter analysis gives out the performance of the filter used in this SMPS. According to the ICES-003, the maxim limits of DM and CM noise generated by this SMPS will be easily obtained.

Chapter 8 Conclusion and Future Work

This chapter summarizes the fundamental theories adopted and improvement made in this research.

8.1 Main Fundamental Theories and Contribution

The fundamental theories used in this thesis are generalized TLM method, MNA method, matrix computation formula (Sherman-Morrison-Woodbury matrix identity), Compensation Theorem (Superposition).

In generalized TLM method, there are two types of models—stub model and link model. The stub models of two-pin devices, inductor and capacitor, are one-port device models, which have same structure as the physical device and do not change the topology of the electrical network; and the link models of them are two-port device models, which have one extra port than physical device and change the topology of the electrical network. By this extra port and the traveling time of the TL link models, it is possible to decouple the electrical network. The conventional link model can decouple network, but all subnetworks should adopt the same simulation time step, which is determined by the smallest time step, and the voltages over the decoupling capacitors have fluctuations. By adopting the TLM stub and link conversion technique and improved TLM link algorithms in **[14]**, the simulation time step of each sub network can be selected independently, and the fluctuation is minimized. The stub and link conversion technique introduces two extra steps, one is the stub and link conversion every bigger simulation time step; the other is that the double computation efforts to get system matrices are needed due to the usage of both link and stub model of decoupling devices. According to the new introduced (5.7), the voltage modification of decoupling capacitors can be obtained just by using stub models. The two extra steps introduced by stub and link conversion technique are wiped off, and the advantages of this technique are kept. By using the decoupling technique, One big system matrix is divided into two smaller system matrix, which decrease the calculation burden. The simulation time of two cases are compared under the same conditions (AMD Athlon Processor 1.25GHz, 1G Ram, Matlab code, Δt_L=2us, Δt_R=0.2us). The computation time for 50ms' circuit simulation time is 525 seconds for undecoupled system, and 198 seconds for decoupled system.

The conventional TLM models of electrical components are equivalent to the ADC models associated with trapezoidal rule; the new TL models are equivalent to the ADC models associated with weight-averaged integration rule. By using generalized TLM stub model, the numerical oscillation due to the forced commutations can be depressed.

As mentioned in chapter 6, the system variables include all node voltages and currents flow through branch one of each switch models. It means the system matrix is a $(n\text{-}1+p)\times(n\text{-}1+p)$ matrix. By using partitioned matrix inversion formula, the maximum dimension of matrix need to be inverted is dropped from $(n\text{-}1+p)\times(n\text{-}1+p)$ to $(n\text{-}1)\times(n\text{-}1)$; the number of matrix need to be inverted is decreased to one by using the Sherman-Morrison-Woodbury matrix identity.

By using the new proposed switch model introduced in chapter 4, the off-state losses and on-state losses both can be modeled at the same time. By the way, due to the existence of the off-state resistor, there is no island sub network in the network no matter what are the states of the switches. It guarantees that the singular matrix does not occur, even though all switches are off. After adopting the little modification described in chapter 4, the computation burden of obtaining system matrices is decreased further.

The switch model used by N.Femia[36] is a two-valued resistor. According to the switch voltage before the forced commutation, he uses compensation theorem to obtain the initial condition after forced commutation. It can accurately obtain the effects of resistance changed upon all node voltage and branch current in linear network. In the proposed switch model, the resistance of branch one can reach an infinite value. That makes the N.Femia technique unusable. But the algorithm used in this research to reinitialize the network is still based on Compensation theorem (Superposition). When the switch state is transferred form off to on, the switch voltage before forced commutation is used to obtain the initial condition after forced commutation; when the switch state is transferred form on to off, the switch current before forced commutation is used to obtain the initial condition after forced commutation.

8.2 Topics for Future Research

From section 3.1, we know the equivalent resistances of reactive components are changed with two variables, one is the resistance *R*, the other is the time step. For a capacitor, the maximum resistance equivalents to characteristic impedance of the TL, the minimum resistance equivalents to zero. For an inductor, the minimum resistance equivalents to the characteristic impedance of the TL, the maximum resistance is infinity. If the time step changed in the range of $[\Delta t,\ 2\Delta t]$, the equivalent resistance can keep constant if the proper value of resistance (R) is chosen. That means the generalized TLM models can be used in variable time step simulation with constant system matrices. It is possible to resynchronize the simulation after switching without recalculation of present time step. In the future, I will find a way to use generalized TLM method into real time simulation area.

In this research, the semiconductor switch is modeled as a two-valued resistor. It accurately represents the characteristics of the semiconductor switch on system level, but the switching transition process is neglected. In the future, the switching transition process will be considered, which will introduce more complicated characteristics of the semiconductor switch.

In this thesis, the transformer models are simplified model that the magnetic branch of the transformer is neglected. Doing this omission will introduce a huge current error when the transformer input current is small. In the future, TL models of the more complicated physical transformer model will be introduced into the system simulation.

At the last part of the chapter 7, the performance of the EMI input filter is roughly discussed. As we know, the parasitic parameters of components, such as the parasitic inductance of the capacitor, the parasitic capacitance of the inductor, and the PCB trace parasitic parameters, the real high frequency characteristic of the EMI input filter will be much different form the expected characteristic. The detailed discussion will be done in the future.

Appendix

Appendix A Transmission Line Modeling

The following equations are the KCL and KVL equations of the single-phase two-wire line section of length of Δx.

$$v(x+\Delta x,t)-v(x,t)=-L\Delta x\frac{\partial i(x,t)}{\partial t}-i(x,t)R\Delta x \tag{A.1}$$

$$i(x,t)-i(x-\Delta x,t)=-C\Delta x\frac{\partial V(x,t)}{\partial t}-G\Delta xV(x,t) \tag{A.2}$$

We use partial derivatives here because *v*(*x*,*t*) and *i*(*x*,*t*) are differentiated with respect to both position *x* and time *t*. Dividing (A.1) and (A.2) by Δx and taking the limit as $\Delta x \to 0$ and taking the Laplace transform, we obtain

$$\frac{dV(x,s)}{dx}=-sLI(x,s)-I(x,s)R \tag{A.3}$$

$$\frac{dI(x,s)}{dx}=-sCV(x,s)-V(x,s)G \tag{A.4}$$

Differentiating (A.3) w.r.t. *x* and using (A.4) to eliminating *V*(*x*,*s*), we get,

$$\frac{d^2V(x,s)}{dx^2}=s^2LCV(x,s)+s(RC+GL)V(x,s)+GRV(x,s) \tag{A.5}$$

Similarly, we can show that

$$\frac{d^2I(x,s)}{dx^2}=s^2LCI(x,s)+s(RC+GL)I(x,s)+GRI(x,s) \tag{A.6}$$

The solution of (A.5) and (A.6) are shown as follow:

$$V(x,s)=V^{+}(s)\exp(-\gamma(s)x)+V^{-}(s)\exp(\gamma(s)x) \tag{A.7}$$

$$I(x,s) = I^{+}(s)\exp(-\gamma(s)x) + I^{-}(s)\exp(\gamma(s)x) \tag{A.8}$$

where $\gamma(s) = \sqrt{s^2LC + s(RC+GL) + GR}$

In general, the inverse Laplace transforms of (A.7) and (A.8) are not closed form expressions. However, for the special cases of a distortionless line, which has the property *R/L=G/C*, or lossless line, which has the property *R=G=*0, the inverse Laplace transforms have a closed form. Assuming *R/L=G/C=M* , if *M*=0, the expression becomes the solution for lossless case; if *M*≠0, the expression becomes the solution for distortionless case.

$$V(x,s) = V^{+}(s)\exp\left(-\left(s+{}^{R}\!/_{L}\right)x/u\right) + V^{-}(s)\exp\left(\left(s+{}^{R}\!/_{L}\right)x/u\right) \tag{A.9}$$

$$I(x,s) = I^{+}(s)\exp\left(-\left(s+{}^{R}\!/_{L}\right)x/u\right) + I^{-}(s)\exp\left(\left(s+{}^{R}\!/_{L}\right)x/u\right) \tag{A.10}$$

where $u = 1/\sqrt{LC}$ and $\alpha = \sqrt{RG}$ The inverse Laplace transform of these equations is

$$v(x,t) = \exp(-\alpha x)v^{+}\left(t - {}^{x}\!/_{u}\right) + \exp(-\alpha x)v^{-}\left(t + {}^{x}\!/_{u}\right) \tag{A.11}$$

$$i(x,t) = \exp(-\alpha x)i^{+}\left(t - {}^{x}\!/_{u}\right) + \exp(-\alpha x)i^{-}\left(t + {}^{x}\!/_{u}\right) \tag{A.12}$$

The α is the attenuation term, due to the power losses consumed by the *R* and *G* . Then I will prove the relationship between v^{+} , v^{-} , i^{+} , i^{-} . Using (A.10) in (A.4)

$$\begin{aligned} &\frac{(s+(R/L))}{u}\left(-I^{+}(s)\exp\left(-(s+(R/L))x/u\right) + I^{-}(s)\exp\left((s+(R/L))x/u\right)\right) \\ &= -(sC+G)\left(V^{+}(s)\exp\left(-(s+(R/L))x/u\right) + V^{-}(s)\exp\left((s+(R/L))x/u\right)\right) \end{aligned} \tag{A.13}$$

Equating the coefficients of exponential functions on both sides of (A.13),

$$\frac{V^{+}(s)}{I^{+}(s)} = \frac{(s+M)\sqrt{LC}}{sC+CM} = \sqrt{{}^{L}\!/_{C}} \qquad \frac{V^{-}(s)}{I^{-}(s)} = -\frac{(s+M)\sqrt{LC}}{sC+CM} = -\sqrt{{}^{L}\!/_{C}} \tag{A.14}$$

Appendix B TLM Stub and Link Model

There are lumped capacitance C and lumped inductance L; I will give out the impedance calculation procedure in this section. C_l and L_l indicate the capacitance and inductance per unit length. $l \times C_l$ and $l \times L_l$ indicate the total capacitance and inductance of each reactive component.

For TLM stub model, we assume the time step τ_S is the round trip time , which is the time for travelling wave move from one end of the element to another end and back.

Propagation velocity: $u = 1/\sqrt{C_l L_l}$

Travelling time: $\tau_S = 2l/u = 2l\sqrt{C_l L_l}$

Equivalent inductance for capacitance C: $lL_l = \tau_S{}^2/4C = L$

Equivalent inductance for capacitance L: $lC_l = \tau_S{}^2/4L = C$

So $Z_C = \sqrt{L/C} = \tau_S/2C$ and $Z_L = \sqrt{L/C} = 2L/\tau_S$

For TLM stub model, we assume the time step τ_L is the single trip time , which is the time for travelling wave move from one end of the element to another end.

Propagation velocity: $v = 1/\sqrt{C_l L_l}$

Travelling time: $\tau_L = l/v = l\sqrt{C_l L_l}$

Equivalent inductance for capacitance C: $lL_l = \tau_L{}^2/C = L$

Equivalent inductance for capacitance L: $lC_l = \tau_L{}^2/L = C$

So $Z_C = \sqrt{L/C} = \tau_L/C$ and $Z_L = \sqrt{L/C} = L/\tau_L$

Appendix C TLM Stub Model of Electrical Branches

I. TLM Model for Type One Equivalent Branch

According to the TLM model for individual passive element, we can have the following equations

$$v_C(t) = R_C i_C(t) + 2v_C^i(t) \qquad \text{(a)}$$

$$i(t) = v_C(t)/R_2 + i_C(t) \qquad \text{(b)} \qquad \text{(C.1)}$$

$$v(t) = v_C(t) + R_1 i_C(t) \qquad \text{(c)}$$

Using (*a*) and (*b*) into (*c*) and rearrange it,

$$v(t) = R_C\left(i(t) - \left(v(t) - i(t)R_1\right)/R_2\right) + 2v_C^i(t) + i(t)R_1$$

$$v(t)(R_2 + R_C)/R_2 = i(t)(R_2R_C + R_1R_C + R_1R_2)/R_2 + 2v_C^i(t)$$

$$i(t) = \frac{R_2 + R_C}{R_2R_C + R_2R_1 + R_1R_C}v(t) + \frac{-2R_2}{R_2R_C + R_2R_1 + R_1R_C}v_C^i(t) \qquad \text{(C.2)}$$

$$\begin{aligned} v_C^i(t+\Delta t) &= v_C(t) - v_C^i(t) \\ &= \frac{R_2R_C}{R_2R_C + R_2R_1 + R_1R_C}v(t) + \frac{R_2R_1 - R_2R_C - R_1R_C}{R_2R_C + R_2R_1 + R_1R_C}v_C^i(t) \end{aligned} \qquad \text{(C.3)}$$

Let's name a vector *Br*.

$$Br = \frac{\begin{bmatrix} R_2 + R_C & -2R_2 & R_2R_C & R_2R_1 - R_2R_C - R_1R_C \end{bmatrix}}{R_2R_C + R_2R_1 + R_1R_C}$$

So (C.2), (C.3) can be express as

$$i(t) = Br(1)v(t) + Br(2)v_C^i(t) \qquad \text{(C.4)}$$

$$v_C^i(t+\Delta t) = Br(3)v(t) + Br(4)v_C^i(t) \qquad \text{(C.5)}$$

II. TLM Model for Type Two Equivalent Branch

Similar to type one equivalent branches, we can have the following equations

$$v_L(t) = R_L i_L(t) + 2v_L^i(t) \qquad \text{(a)}$$

$$i(t) = v_L(t)/R_2 + i_L(t) \qquad \text{(b)} \qquad \text{(C.6)}$$

$$v(t) = v_L(t) + i(t)R_1 \qquad \text{(c)}$$

Using (a) and (b) into (c) and rearrange it,

$$v(t) = R_L\left(i(t) - \left(v(t) - i(t)R_1\right)/R_2\right) + 2v_L^i(t) + i(t)R_1$$

$$v(t)(R_2 + R_L)/R_2 = i(t)(R_2R_L + R_1R_L + R_1R_2)/R_2 + 2v_L^i(t)$$

$$i(t) = \frac{R_2 + R_L}{R_2R_L + R_2R_1 + R_1R_L} v(t) + \frac{-2R_2}{R_2R_L + R_2R_1 + R_1R_L} v_L^i(t) \qquad \text{(C.7)}$$

$$\begin{aligned} v_L^i(t+\Delta t) &= -v_L(t) + v_L^i(t) \\ &= -\frac{R_2R_L}{R_2R_L + R_2R_1 + R_1R_L} v(t) + \frac{R_2R_1 - R_2R_L - R_1R_L}{R_2R_L + R_2R_1 + R_1R_L} v_L^i(t) \end{aligned} \qquad \text{(C.8)}$$

Let's name a vector *Br*.

$$Br = \frac{\left[R_2 + R_L \quad -2R_2 \quad R_2R_L \quad R_2R_1 - R_2R_L - R_1R_L\right]}{R_2R_L + R_2R_1 + R_1R_L}$$

So (C.7), (C.8) can be expressed as

$$i(t) = Br(1)v(t) + Br(2)v_L^i(t) \qquad \text{(C.9)}$$

$$v_L^i(t+\Delta t) = Br(3)v(t) + Br(4)v_L^i(t) \qquad \text{(C.10)}$$

III. TLM Model for Two Winding Transformer

For the simplified transformer model in Figure 2.15, the time domain characteristic equations can be written as

$$v_1(t)-v_0(t)=L_1\frac{di_1(t)}{dt}+i_1(t)R_1 \quad \text{(a)}$$

$$v_2(t)-\frac{v_0(t)}{a}=L_2\frac{di_2(t)}{dt}+i_2(t)R_2 \quad \text{(b)} \qquad \text{(C.11)}$$

$$i_m(t)=i_1(t)+i_2(t)/a=0 \quad \text{(d)}$$

Adopting TLM technique, the equation transform into the equation

$$v_1(t)-v_0(t)=(R_{L1}+R_1)i_1(t)+2v_{L1}^i(t)=Z_1i_1(t)+2v_{L1}^i(t) \quad \text{(a)}$$

$$v_2(t)-v_0(t)/a=(R_{L2}+R_2)i_2(t)+2v_{L2}^i(t)=Z_2i_2(t)+2v_{L2}^i(t) \quad \text{(b)} \qquad \text{(C.12)}$$

Multiplying (C.12) by *Z2* and (C.12) by Z_1/a, then applying then into (C.11), the formula for calculating v_0 can be obtained.

$$\begin{aligned}&v_0(n)=mid2Tx(:,1)\left[v_1(n)-2v_{L1}^i(n);\quad v_2(n)-2v_{L2}^i(n)\right]\\&\text{where } mid2Tx(1,:)=\left[a^2Z_2 \quad aZ_1\right]\big/\left(a^2Z_2+Z_1\right)\end{aligned} \qquad \text{(C.13)}$$

Using (C.13) into (C.12), we have

$$\begin{aligned}&\begin{bmatrix} i_1(n)\\ i_2(n)\end{bmatrix}=Y2Tx(:,:,1)\begin{bmatrix} v_1(n)\\ v_2(n)\end{bmatrix}-Y2Tx(:,:,1)\begin{bmatrix} 2v_{L1}^i(n)\\ 2v_{L2}^i(n)\end{bmatrix}\\&\text{where } Y2Tx(:,:,1)=\begin{bmatrix} 1 & -a\\ -a & a^2\end{bmatrix}\Big/\left(a^2Z_2+Z_1\right)\end{aligned} \qquad \text{(C.14)}$$

As we know the voltage of any node is equal to the sum of the incident wave and the reflect wave, thus the incident waves for the next time step are

$$\begin{bmatrix} v_{L1}^{i}(t+\Delta t) \\ v_{L2}^{i}(t+\Delta t) \end{bmatrix} = -\begin{bmatrix} v_1(t)-v_0(t)-i_1(t)R_1-v_{L1}^{i}(t) \\ v_2(t)-v_0(t)-i_1(t)R_1-v_{L1}^{i}(t) \end{bmatrix} - \begin{bmatrix} Z_{L1} & 0 \\ 0 & Z_{L2} \end{bmatrix} \begin{bmatrix} i_1(t) \\ i_2(t) \end{bmatrix} - \begin{bmatrix} v_{L1}^{i}(t) \\ v_{L2}^{i}(t) \end{bmatrix}$$

$$= -Y2Tx(:,:,2)\begin{bmatrix} v_1(t)-2v_{L1}^{i}(t) \\ v_2(t)-2v_{L2}^{i}(t) \end{bmatrix} - \begin{bmatrix} v_{L1}^{i}(t) \\ v_{L2}^{i}(t) \end{bmatrix}$$

IV. TLM Model for Three-winding Transformer

The time domain characteristic equations for three-winding transformer, which is shown in Figure 2.16, is listed following.

$$v_1(t)-v_0(t)=L_1\frac{di_1(t)}{dt}+i_1(t)R_1 \quad \text{(a)}$$

$$v_2(t)-\frac{v_0(t)}{a_{12}}=L_2\frac{di_2(t)}{dt}+i_2(t)R_2 \quad \text{(b)} \qquad \text{(C.15)}$$

$$v_3(t)-\frac{v_0(t)}{a_{13}}=L_3\frac{di_3(t)}{dt}+i_3(t)R_3 \quad \text{(c)}$$

$$i_m(t)=i_1(t)+i_2(t)/a_{12}+i_3(t)/a_{13}=0 \quad \text{(d)}$$

Adopting TLM technique, the equation transform into the equation

$$v_1(t)-v_0(t)=(R_{L1}+R_1)i_1(t)+2v_{L1}^{i}(t)=Z_1 i_1(t)+2v_{L1}^{i}(t) \quad \text{(a)}$$

$$v_2(t)-v_0(t)/a_{12}=(R_{L2}+R_2)i_2(t)+2v_{L2}^{i}(t)=Z_2 i_2(t)+2v_{L2}^{i}(t) \quad \text{(b)} \qquad \text{(C.16)}$$

$$v_3(t)-v_0(t)/a_{13}=(R_{L3}+R_3)i_3(t)+2v_{L3}^{i}(t)=Z_3 i_3(t)+2v_{L3}^{i}(t) \quad \text{(c)}$$

Multiplying (C.16) by Z_2Z_3 , (C.16) by Z_1Z_3/a_{12} and (C.16) by Z_1Z_2/a_{13}, then applying then into (C.15), the formula for calculating v_0 can be obtained.

$$v_0(t) = mid3Tx(1,:)\left[v_1(t) - 2v_{L1}^i(t);\quad v_2(t) - 2v_{L2}^i(t);\quad v_3(t) - 2v_{L3}^i(t)\right]$$
$$\text{where } mid3Tx(1,:) = \frac{\left[a_{12}^2 a_{13}^2 Z_2 Z_3 \quad a_{12} a_{13}^2 Z_1 Z_3 \quad a_{12}^2 a_{13} Z_1 Z_2\right]}{a_{12}^2 a_{13}^2 Z_2 Z_3 + a_{13}^2 Z_1 Z_3 + a_{12}^2 Z_2 Z_1} \tag{C.17}$$

Using (C.17) into (C.16), we have

$$\begin{bmatrix} i_1(t) \\ i_2(t) \\ i_3(t) \end{bmatrix} = Y3Tx(:,:,1)\begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} - Y3Tx(:,:,1)\begin{bmatrix} 2v_{L1}^i(t) \\ 2v_{L2}^i(t) \\ 2v_{L3}^i(t) \end{bmatrix}$$
$$\text{where } Y3Tx(:,:,1) = \frac{\begin{bmatrix} a_{12}^2 Z_2 + a_{13}^2 Z_3 & -a_{12} a_{13}^2 Z_3 & -a_{12}^2 a_{13} Z_2 \\ -a_{12} a_{13}^2 Z_3 & a_{12}^2 a_{13}^2 Z_3 + a_{12}^2 Z_1 & -a_{12} a_{13} Z_1 \\ -a_{12}^2 a_{13} Z_2 & -a_{12} a_{13} Z_1 & a_{12}^2 a_{13}^2 Z_2 + a_{13}^2 Z_1 \end{bmatrix}}{a_{12}^2 a_{13}^2 Z_2 Z_3 + a_{12} a_{13}^2 Z_1 Z_3 + a_{12}^2 a_{13} Z_2 Z_1} \tag{C.18}$$

As we know the voltage of any node is equal to the sum of the incident wave and the reflect wave, thus the incident waves for the next time step are

$$\begin{bmatrix} v_{L1}^i(t+\Delta t) \\ v_{L2}^i(t+\Delta t) \\ v_{L3}^i(t+\Delta t) \end{bmatrix} = -\begin{bmatrix} v_1(t) - v_0(t) - i_1(t)R_1 - v_{L1}^i(t) \\ v_2(t) - v_0(t)/a_{12} - i_2(t)R_2 - v_{L2}^i(t) \\ v_3(t) - v_0(t)/a_{13} - i_3(t)R_3 - v_{L3}^i(t) \end{bmatrix}$$
$$= -\begin{bmatrix} R_{L1} & 0 & 0 \\ 0 & R_{L2} & 0 \\ 0 & 0 & R_{L3} \end{bmatrix}\begin{bmatrix} i_1(t) \\ i_2(t) \\ i_3(t) \end{bmatrix} - \begin{bmatrix} v_{L1}^i(t) \\ v_{L2}^i(t) \\ v_{L3}^i(t) \end{bmatrix} \tag{C.19}$$
$$= -Y3Tx(:,:,2)\begin{bmatrix} v_1(t) - 2v_{L1}^i(t) \\ v_2(t) - 2v_{L2}^i(t) \\ v_3(t) - 2v_{L3}^i(t) \end{bmatrix} - \begin{bmatrix} v_{L1}^i(t) \\ v_{L2}^i(t) \\ v_{L3}^i(t) \end{bmatrix}$$
$$\text{where } Y3Tx(:,:,2) = \begin{bmatrix} R_{L1} & 0 & 0 \\ 0 & R_{L2} & 0 \\ 0 & 0 & R_{L3} \end{bmatrix} Y3Tx(:,:,1)$$

Appendix D The Generalized TLM stub models

I. New TLM Model for Type One Equivalent Branch

According to the new TLM model for individual passive element, we can have the following equations

$$
\left.
\begin{aligned}
&v_1(t) = i_1(t)R_{eq} + 2v_C^i(t) && \text{(a)}\\
&v_C^i(t) = v_1(t-\Delta t)/2 && \text{(b)}\\
&v(t) = i(t)R_1 + v_1(t) && \text{(c)}\\
&i(t) = i_1(t) + v_1(t)/R_2 && \text{(d)}
\end{aligned}
\right\} \quad \text{(D.1)}
$$

Using (D.1) (b), (c), and (d) into (a), we have s (D.2).

$$v(t) - i(t)R_1 = \left(i(t) - \frac{v(t) - i(t)R_1}{R_2} \right) R_{eq} + 2v_C^i(t)$$

$$\frac{R_2R_{eq} + R_1R_{eq} + R_1R_2}{R_2} i(t) = \frac{R_2 + R_{eq}}{R_2} v(t) - 2v_C^i(t)$$

$$i(t) = \frac{v(t)}{R_{EQ}} + I_{EQ}(t) = \frac{v(t)}{R_{EQ}} - 2v_C^i(t) \frac{R_2}{R_2R_{eq} + R_1R_{eq} + R_1R_2} \quad \text{(D.2)}$$

where $R_C = \dfrac{\Delta t}{2C}$ $\quad R_{eq} = 2R_C$ $\quad R_{EQ} = R_1 + \dfrac{R_2R_{eq}}{R_2 + R_{eq}}$

Using (D.1) (a), (c), and (d) into (b), we have (D.3)

$$
\begin{aligned}
v_C^i(t+\Delta t) &= \left(v(t) - i(t)R_1\right)/2\\
&= \frac{v(t)}{2} - \frac{1}{2}\left(\frac{v_1(t) - 2v_C^i(t)}{R_{eq}} + \frac{v_1(t)}{R_2} \right) R_1\\
&= \frac{v(t)}{2} - \left(\frac{(R_{eq} + R_2)v_C^i(t+\Delta t) - R_2v_C^i(t)}{R_{eq}R_2} \right) R_1
\end{aligned}
$$

$$v_C^i(t+\Delta t) = \frac{1}{2}\frac{R_2R_{eq}}{R_2R_{eq} + R_1R_{eq} + R_1R_2} v(t) + \frac{R_1R_2}{R_2R_{eq} + R_1R_{eq} + R_1R_2} v_C^i(t) \quad \text{(D.3)}$$

II. New TLM Model for Type Two Equivalent Branch

According to the new TLM model for individual passive element, we can have the following equations

$$
\begin{aligned}
&v_1(t) = i_1(t)R_{eq} + v_L^i(t) && \text{(a)}\\
&v_L^i(t) = -v_1(t-\Delta t) + v_L^i(t-\Delta t) && \text{(b)}\\
&v(t) = i(t)R_1 + v_1(t) && \text{(c)}\\
&i(t) = i_1(t) + v_1(t)/R_2 && \text{(d)}
\end{aligned}
\qquad \text{(D.4)}
$$

Using (D.4) (a), (b), and (d) into (c), we have (D.5).

$$v(t) = i(t)R_1 + \left(i(t) - \frac{v(t) - i(t)R_1}{R_2}\right)R_{eq} + v_L^i(t)$$

$$\frac{R_2 + R_{eq}}{R_2}v(t) = \frac{R_1R_2 + R_2R_{eq} + R_1R_{eq}}{R_2}i(t) + v_L^i(t)$$

$$i(t) = \frac{v(t)}{R_{EQ}} + I_{EQ}(t) = \frac{v(t)}{R_{EQ}} - v_L^i(t)\frac{R_2}{R_2R_{eq} + R_1R_{eq} + R_1R_2} \qquad \text{(D.5)}$$

where $R_L = \dfrac{2L}{\Delta t}$ $\quad R_{eq} = \dfrac{R_L}{2}$ $\quad R_{EQ} = R_1 + \dfrac{R_2R_{eq}}{R_2 + R_{eq}}$

Using (D.4) (a), (c), and (d) into (b), we have (D.6)

$$
\begin{aligned}
v_L^i(t+\Delta t) &= -v_1(t) + i(t)R_1 + v_L^i(t)\\
&= -v(t) + \left(\frac{-v_L^i(t+\Delta t)}{R_{eq}} + \frac{-v_L^i(t+\Delta t) + v_L^i(t)}{R_2}\right)R_1 + v_L^i(t)
\end{aligned}
$$

$$v_L^i(t+\Delta t) = -\frac{R_2R_{eq}}{R_2R_{eq} + R_1R_{eq} + R_1R_2}v(t) + \frac{R_1R_{eq} + R_2R_{eq}}{R_2R_{eq} + R_1R_{eq} + R_1R_2}v_L^i(t) \qquad \text{(D.6)}$$

Appendix E Calculation of Inverse Matrix

The inverse of block matrix A can be expressed as (E.1), by adopting block matrix inversion formula.

$$A^{-1}=\begin{bmatrix} E & F \\ G & H \end{bmatrix}=\begin{bmatrix} Y & B \\ B' & D \end{bmatrix}^{-1}=\begin{bmatrix} \left(Y+BD^{-1}B^T\right)^{-1} & -Y^{-1}B\left(D+B^TY^{-1}B\right)^{-1} \\ D^{-1}B^T\left(Y+BD^{-1}B^T\right)^{-1} & \left(D+B^TY^{-1}B\right)^{-1} \end{bmatrix} \tag{E.1}$$

where $B^T=-B'$

By using matrix inversion lemma, which is shown in (E.2)

$$\left(I+MN\right)^{-1}=I-M\left(I+NM\right)^{-1}N \tag{E.2}$$

I. Case one: $\boldsymbol{p \le n\text{-}1\text{+}m}$

If we let $M=Y^{-1}B$ and $N=D^{-1}B^T$, the matrix E, F, G, H can be expressed as

$$\begin{aligned} E&=\left(Y+BD^{-1}B^T\right)^{-1}=\left[Y\left(I+Y^{-1}BD^{-1}B^T\right)\right]^{-1}=\left(I+Y^{-1}BD^{-1}B^T\right)^{-1}Y^{-1} \\ &=\left[I-Y^{-1}B\left(I+D^{-1}B^TY^{-1}B\right)^{-1}D^{-1}B^T\right]Y^{-1} \\ &=Y^{-1}-Y^{-1}B\left(I+D^{-1}B^TY^{-1}B\right)^{-1}D^{-1}B^TY^{-1} \\ &=Y^{-1}-P_1\left(I+P_3\right)^{-1}P_2 \end{aligned} \tag{E.3}$$

$$\begin{aligned} F&=-Y^{-1}B\left(D+B^TY^{-1}B\right)^{-1}=-Y^{-1}B\left[D\left(I+D^{-1}B^TY^{-1}B\right)\right]^{-1} \\ &=-Y^{-1}B\left(I+D^{-1}B^TY^{-1}B\right)^{-1}D^{-1}=-P_1\left(I+P3\right)^{-1}D^{-1} \end{aligned} \tag{E.4}$$

$$G=D^{-1}B^T\left(Y+BD^{-1}B^T\right)^{-1}=D^{-1}B^TE=P_2-P_3\left(I+P_3\right)^{-1}P_2 \tag{E.5}$$

$$H=\left(D+B^TY^{-1}B\right)^{-1}=\left[D\left(I+D^{-1}B^TY^{-1}B\right)\right]^{-1}=\left(I+P_3\right)^{-1}D^{-1} \tag{E.6}$$

II. Case two: $\boldsymbol{p > n\text{-}1\text{+}m}$

If we let $M= D^{-1}B^{T}Y^{-1}$and $N= B$, the matrix E, F, G, H can be expressed as

$$\begin{aligned}E&=\left(Y+BD^{-1}B^{T}\right)^{-1}=\left[\left(I+BD^{-1}B^{T}Y^{-1}\right)Y\right]^{-1}\\&=Y^{-1}\left(I+BD^{-1}B^{T}Y^{-1}\right)^{-1}=Y^{-1}\left(I+P_3\right)^{-1}\end{aligned}\tag{E.7}$$

$$\begin{aligned}F&=-Y^{-1}B\left(D+B^{T}Y^{-1}B\right)^{-1}=-Y^{-1}B\left[D\left(I+D^{-1}B^{T}Y^{-1}B\right)\right]^{-1}\\&=-Y^{-1}B\left(I+D^{-1}B^{T}Y^{-1}B\right)^{-1}D^{-1}=-P_1\left(I-P_2\left(I+P_3\right)^{-1}B\right)D^{-1}\end{aligned}\tag{E.8}$$

$$G=D^{-1}B^{T}\left(Y+BD^{-1}B^{T}\right)^{-1}=D^{-1}B^{T}Y^{-1}\left(I+BD^{-1}B^{T}Y^{-1}\right)^{-1}=P_2\left(I+P_3\right)^{-1}\tag{E.9}$$

$$\begin{aligned}H&=\left(D+B^{T}Y^{-1}B\right)^{-1}=\left[D\left(I+D^{-1}B^{T}Y^{-1}B\right)\right]^{-1}\\&=\left[I-D^{-1}B^{T}Y^{-1}\left(I+BD^{-1}B^{T}Y^{-1}\right)^{-1}B\right]D^{-1}\\&=\left(I-P_2\left(I+P_3\right)^{-1}B\right)D^{-1}\end{aligned}\tag{E.10}$$

where $P_1=Y^{-1}B,\ P_2=D^{-1}B^{T}Y^{-1},\ P_3=BD^{-1}B^{T}Y^{-1}$

Appendix F Initial Conditions of Transformer

Due to the existence of the leakage inductance, the current flow through any winding of transformer is continuous at any time. Therefore, at the switching time period [t^-, t^+], the current keep constant. According to the transformer current at the time t^- and the voltage at the time t^+, (C.15) can be written as

$$\begin{aligned}&v_1(t)-i_1(t)R_1-v_0(t)=L_1\frac{di_1(t)}{dt} &&\text{(a)}\\&v_2(t)-i_2(t)R_2-\frac{v_0(t)}{a_{12}}=L_2\frac{di_2(t)}{dt} &&\text{(b)}\\&v_3(t)-i_3(t)R_3-\frac{v_0(t)}{a_{13}}=L_3\frac{di_3(t)}{dt} &&\text{(c)}\\&i_m(t)=i_1(t)+i_2(t)/a_{12}+i_3(t)/a_{13}=0 &&\text{(d)}\end{aligned}\tag{F.1}$$

According to the similar procedure in the section IV of Appendix C, the v_0 can be obtained.

$$v_0(n) = mid3Tx(2,:)\left[v_1(n) - i_1(n)R_1;\ \ v_2(n) - i_2(n)R_2;\ \ v_3(n) - i_3(n)R_3\right] \qquad \text{(F.2)}$$

where $mid3Tx(2,:) = \dfrac{\left[a_{12}^2 a_{13}^2 L_2 L_3 \quad a_{12} a_{13}^2 L_1 L_3 \quad a_{12}^2 a_{13} L_1 L_2\right]}{a_{12}^2 a_{13}^2 L_2 L_3 + a_{13}^2 L_1 L_3 + a_{12}^2 L_2 L_1}$

Then the new incident wave after the switching can be obtain

$$\begin{bmatrix} v_{L1}^i(n) \\ v_{L2}^i(n) \\ v_{L3}^i(n) \end{bmatrix} = \left(\begin{bmatrix} v_1(n) - v_0(n) \\ v_2(n) - v_0(n)/a_{12} \\ v_2(n) - v_0(n)/a_{13} \end{bmatrix} - \begin{bmatrix} Z_1 & 0 & 0 \\ 0 & Z_2 & 0 \\ 0 & 0 & Z_3 \end{bmatrix} \begin{bmatrix} i_1(n) \\ i_2(n) \\ i_3(n) \end{bmatrix} \right) \Big/ 2 \qquad \text{(F.3)}$$

Appendix G Model Data of Used Semiconductor Switches

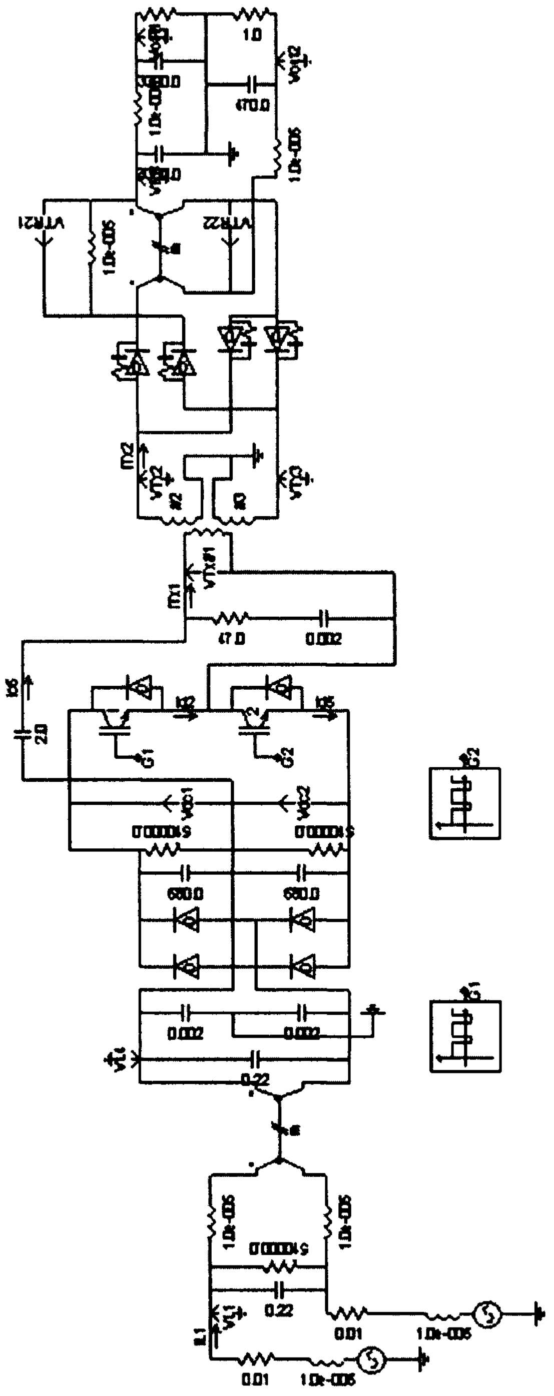
The following content is the semicondutor models used in Multisim for the system simulation.

Model	Model Data
1N5406	IS=2.68e-12 RS=0.00731 N=1.17 TT=1.44e-5 CJO=124e-12 VJ=0.6 M=0.5 EG=1.11 XTI=3.0 KF=0 AF=1 FC=0.5 BV=900 IBV=10e-6 TNOM=27
Diode Virtual	IS=1.0e-14 RS=0.01 N=1 TT=0 CJO=0 VJ=0.1 M=0.5 EG=1.11 XTI=3.0 KF=0 AF=1 FC=0.5 BV=1.0e30 IBV=1.0e-3 TNOM=27
MBRB2545	IS=2e-05 RS= 0.00133491 N= 1.18777 TT= 1e-09 CJO= 1e-011 VJ= 0.7 M=0.5 EG= 0.6 XTI= 4.547 FC= 0.5 BV= 45 IBV= 4e-5 KF= 0 AF= 1
2SC2655	IS=1.925e-014 NF=1.000e+000 ISE=4.331e-016 NE=1.053e+000 BF=1.687e+002 BR=1.000e+000 IKF=2.000e+000 VAF=3.000e+001 VAR=2.000e+001 EG=1.110e+000 XTI=3.000e+000 XTB=0.000e+000 RC=3.500e-001 RB=2.000e+000 RE=0.000e+000 CJE=7.5e-11, MJE=0.33 VJE=0.75 CJC=5.5e-11, MJC=0.33 VJC=0.75

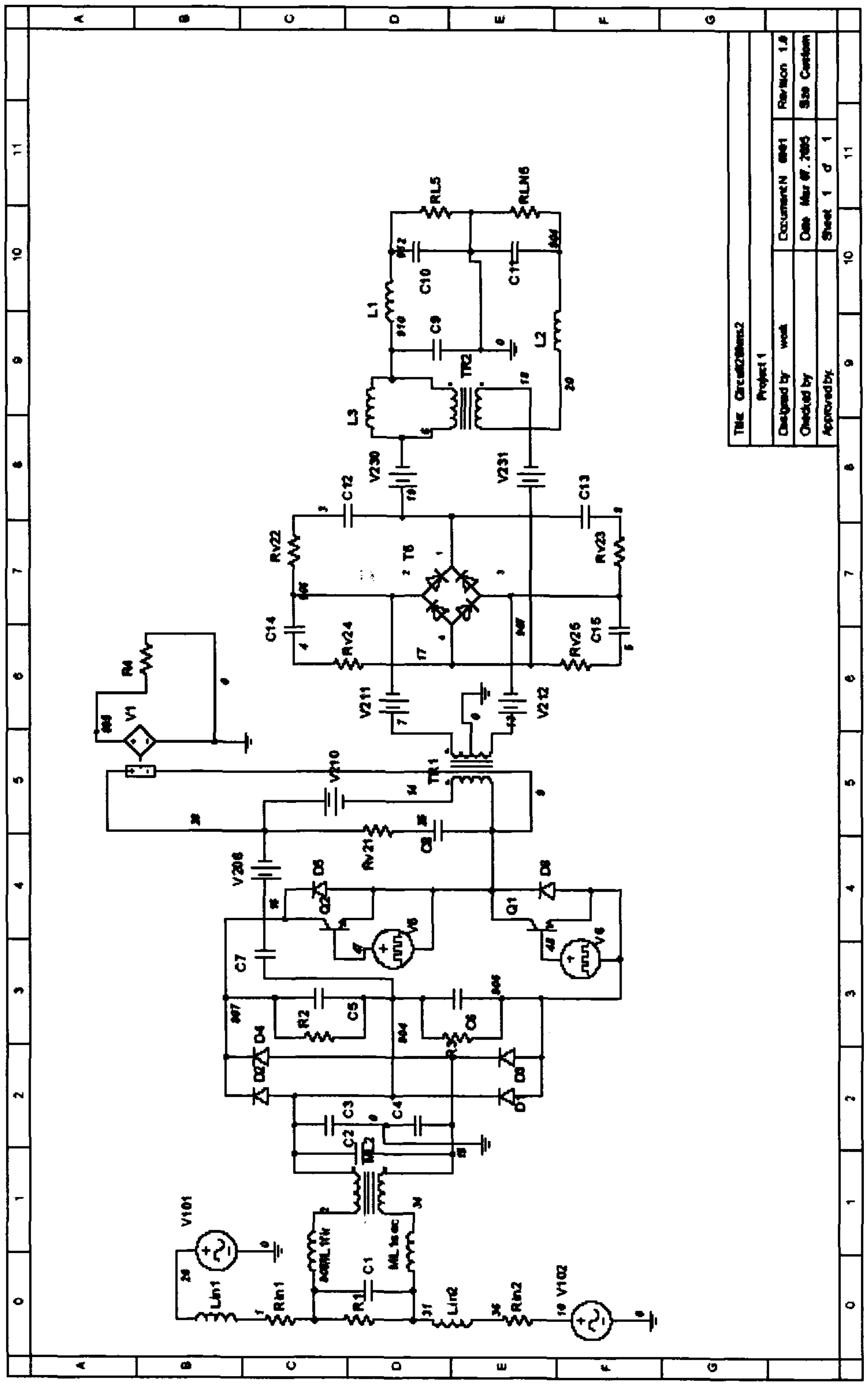
Appendix H Netlist File of SMPS

```
**Circuit200ms2
* Electronics Workbench
* This file was created by:
*     multiSIM to SPICE netlist routine
* Generated by: cyxmn
* 1/8/2006 2:37:29 PM
xTR1 14 9 7 13 0 TRBASE10__TRANSFORMER__98
cC7 804 16 2.0E-6
xML2 804 2 15 34 TL2__TRANSFORMER__5
vV212 13 907 dc 0
+
+
xTR2 910 6 18 20 TRMUL__TRANSFORMER__68
eV1 905 0 28 9 1
qQ1 9 48 806 2SC2655__BJT_NPN__681
qQ2 807 47 9 2SC2655__BJT_NPN__681
rR1 801 31 510k
rR4 905 0    1000 resR4
.model resR4
+r(
+
+  )
xT5 19 906 17 907 MBRB2545__TRANSFORMER__73
dD4 15 807 1N5406__TRANSFORMER__72
dD2 804 807 1N5406__TRANSFORMER__72
dD1 806 804 IDEALD__DIODE__1__20
dD3 806 15 IDEALD__DIODE__1__20
cC5 807 804 0.00068
cC6 804 806 0.00068
rR2 807 804    330000 resR2
.model resR2
+r(
+
+  )
rR3 804 806    330000 resR3
.model resR3
+r(
+   TC2=0
+  )
vV101 26 0 dc 0    ac 1 0
+   distof1 1 0
+   distof2 1 0
+ sin(0 169.71 60 0 0 0)
vV102 10 0 dc 0    ac 1 0
+   distof1 1 0
+   distof2 1 0
+ sin(0 2.828 60 0 0 0)
rRin1 1 801    0.01 resRin1
.model resRin1
+r(
+
+  )
lLin1 26 1 1e-006
rRin2 10 36    0.01 resRin2
.model resRin2
+r(
+
+  )
lLin2 36 31 1e-006
lL3 6 910 1e-005
cC9 910 0 0.0033
lL1 910 912 1e-006
cC10 912 0 0.0033
rRL5 912 0    0.3 resRL5
.model resRL5
+r(
+
+  )
cC11 0 904 0.00047
rRLN5 0 904    1 resRLN5
.model resRLN5
+r(
+
+  )
rRv21 28 25    47 resRv21
.model resRv21
+r(
+
+  )
cC8 25 9 2e-009
dD5 9 807 IDEALD__DIODE__1__8
dD6 806 9 IDEALD__DIODE__1__9
vV5 47 9 dc 0
+
+
+ pulse(0 5 0 1e-009
+   1e-009 1.8e-005
+   4e-005)
vV6 48 806 dc 0
+
+
+ pulse(0 5 2e-005 1e-009
+   1e-009 1.8e-005
+   4e-005)
cC12 19 3 1e-008
rRv22 3 906    4.7 resRv22
.model resRv22
+r(
+
+ )
cC14 906 4 1e-008
rRv24 4 17   4.7 resRv24
.model resRv24
+r(
+
+ )
cC15 907 5 1e-008
rRv25 5 17   4.7 resRv25
.model resRv25
+r(
+
+ )
cC13 19 8 1e-008
rRv23 8 907   4.7 resRv23
.model resRv23
+r(
+
+ )
cC3 804 0 2e-009
cC4 0 15 2e-009
cC1 801 31 2.2e-007
cC2 804 15 2.2e-007
lML1fir 801 2 1e-005
lML1sec 31 34 1e-005
lL2 20 904 1e-006
vV206 16 28 dc 0
+
+
vV230 19 6 dc 0
+
+
vV231 17 18 dc 0
+
+
vV211 7 906 dc 0
+
+
vV210 28 14 dc 0
+
+
.SUBCKT TRBASE10__TRANSFORMER__98 1 2 3 4 5
*        EWB         Version        4        -
TRBASE10__TRANSFORMER__98ransformer
Model
* n= 20 Le= 3.6-006 Lm= 6e5 Rp= 1e-006 Rs= 5e-007
    Rp     1    6 1e-006ohm
    Rs1 12    3 5e-007ohm
    Rs2 13    4 5e-007ohm
    Le     6    7 3e-006H
    Lm     7    2 6e5H
    Ls1      10    12 3e-009H
    Ls2      11    13 3e-009H
    E1     9   5   7   2 0.05
    E2     5   8   7   2 0.05
    V1     9 10 DC 0V
    V2     11 8 DC 0V
    F1     7    2 V1 0.05
    F2     7    2 V2 0.05
.ENDS
.SUBCKT TL2__TRANSFORMER__5 1 2 3 4
*   *1, 2-- primary winding,   *3,4-- secondary terminal
Rs1 1 11 1e-003
Rl2 3 31 1e-003
L1   11 2 3e-003
L2   31 4 3e-003
K12   L1 L2 9.99e-001
.ENDS
.SUBCKT TRMUL__TRANSFORMER__68 1 2 3 4
*   *1, 2-- primary winding,   *3,4-- secondary terminal
Rs1 1 11 1e-003
Rl2 31 3 1e-003
L1   11 2 6e-006
L2   31 4 6e-006
K12   L1 L2 9.833e-001
.ENDS
.subckt MBRB2545__TRANSFORMER__73 1 2 3 4
d1 4 1 DMBRB2545__TRANSFORMER__73
d2 2 1 DMBRB2545__TRANSFORMER__73
d3 3 4 DMBRB2545__TRANSFORMER__73
d4 3 2 DMBRB2545__TRANSFORMER__73
.MODEL        DMBRB2545__TRANSFORMER__73 D
+ is=2e-05
+ rs= 0.00133491
+ n= 1.18777
+ tt= 1e-09
+ cjo= 1e-011
+ vj= 0.7
+ m= 0.5
+ eg= 0.6
+ xti= 4.547
+ fc= 0.5
+ bv= 45
+ ibv= 4e-5
+ kf= 0
+ af= 1
.ends MBRB2545__TRANSFORMER__73
.MODEL        1N5406__TRANSFORMER__72 D
+     IS =   2.68e-12
+     RS =   0.00731
+     N =   1.17
+     TT =   1.44e-5
+     CJO =   124e-12
+     VJ =   0.6
+     M =   0.5
+   EG =  1.11
+   XTI =  3.0
+   KF =  0
+   AF =  1
+   FC =  0.5
+   BV =  900
+   IBV =  10e-6
+   TNOM =  27
.MODEL IDEALD__DIODE__1__20   D
+   (
+   IS =  2.68e-12
+   RS =  0.00731
+   N =  1.17
+   TT =  1.44e-5
+   CJO =  124e-12
+   VJ =  0.6
+   M =  0.5
+   EG =  1.11
+   XTI =  3.0
+   KF =  0
+   AF =  1
+   FC =  0.5
+   BV =  900
+   IBV =  10e-6
+   TNOM =  27
+ )
.MODEL IDEALD__DIODE__1__8   D
+   (
+   IS =  1.0e-14
+   RS =  0.01
+   N =  1
+   TT =  0
+   CJO =  0
+   VJ =  0.1
+   M =  0.5
+   EG =  1.11
+   XTI =  3.0
+   KF =  0
+   AF =  1
+   FC =  0.5
+   BV =  1.0e30
+   IBV =  1.0e-3
+   TNOM =  27
+ )
.MODEL IDEALD__DIODE__1__9   D
+   (
+   IS =  1.0e-14
+   RS =  0.01
+   N =  1
+   TT =  0
+   CJO =  0
+   VJ =  0.1
+   M =  0.5
+   EG =  1.11
+   XTI =  3.0
+   KF =  0
+   AF =  1
+   FC =  0.5
+   BV =  1.0e30
+   IBV =  1.0e-3
+   TNOM =  27
+ )
.MODEL      2SC2655__BJT_NPN__681 NPN
+ IS=1.925e-014 NF=1.000e+000 ISE=4.331e-016
+ NE=1.053e+000 BF=1.687e+002 BR=1.000e+000
+ IKF=2.000e+000 VAF=3.000e+001 VAR=2.000e+001
+ EG=1.110e+000 XTI=3.000e+000 XTB=0.000e+000
+ RC=3.500e-001 RB=2.000e+000 RE=0.000e+000
+ CJE=7.5e-11, MJE=0.33 VJE=0.75
+ CJC=5.5e-11, MJC=0.33 VJC=0.75
```

Appendix I SMPS Circuit in PSCAD



Appendix J SMPS Circuit in Multisim



Appendix K Matlab Codes

I. Algorithm I

```
clear;  clc;  disp('1Start');  tic
global Rvector1 Cvector1 Lvector1 TL1 TL2 Rvector2 Cvector2 Lvector2 TR1 TR2 YLLM YRLM YRLME
global ZLD ZRD ZRQ ZRC BrL BrR MidYRtranx MIdYRtranxE ZRCE BrE
global SDL SDR SQRcom IonL IonR Vlink deltaT
global TLstep TRstep TRE TswRtemp TswLtemp NumofPoints2
global VLinc VRinc VLMinc VRMinc ILpre IRpre deltaV XL XR IL IR VonL VonR
global freq T4saw T4Q1th T4Q2th Dmax Mag Phy w TLstep TRstep

% Basic time step. time step TLstep/TRstep=integer TRstep/TRmin=integer
TLstep=2e-6; TRstep=2e-7; TRE=TRstep/2; Time=50e-3; deltaT=5e-15; deltaV=1e-12;
Tstart2rec=Time-0.05; Trec4L=1*TLstep; Trec4R=1*TRstep;
NumofStart2Rec=round(Tstart2rec/TLstep)+1; NumofPoints1=round(Time/TLstep)+1;
NumofPoints2=round(TLstep/TRstep); %NumofPoints3=round(TRstep/TRmin);
Step4RecL=round(Trec4L/TLstep); Step4RecR=round(Trec4R/TRstep);
%PWM control part
Dmax=0.9; freq=25e3; T4PWM=floor(1/freq/TRstep/2)*TRstep*2; T4saw=T4PWM/TRstep/2;
T4Q1th=floor(T4saw*Dmax); T4Q2th=T4Q1th+T4saw;
VLinc=zeros(14,2); VRinc=zeros(19,2); VLMinc=zeros(4,2); VRMinc=zeros(2,2);
w=2*pi*60; Phy=[0;0]; Mag=[120*sqrt(2); 2*sqrt(2)];
% Rectifier part
Rs=2e-2; R1=510e3; R2=330e3; R3=330e3; Rvector1=[Rs/2, R1, R2, R3]; Ls=2e-6;
Lvector1=[Ls/2]; C1=220e-9; C2=220e-9; C3=2e-9; C4=2e-9; C5=680e-6; C6=680e-6;
Cvector1=[C1, C2, C3, C4, C5, C6]; TL1=[10e-6, 10e-6, 0e-6, 0e-6, 0e-7, 0];
TL2=[3.0e-3, 3.0e-3, 2.997e-3, 1e-3, 1e-3, 0]; Rdon=0.0119; Rdoff=5e11; ZLD=[Rdon, Rdoff];
% DC-DC converter
C7=2e-6; C8=2e-9; C9=3300e-6; C10=3300e-6; C11=470e-6; C=10e-9;
Cvector2=[C5, C6, C7, C8, C9, C10, C11, C, C, C, C];
R4=47; RL5=0.3; RLN5=1; R=4.7; Rvector2=[R4, R, R, R, R, RL5, RLN5, 0.01];
L1=1e-6;L2=1e-6; L3=10e-6; Lvector2=[L1, L2, L3];
La=3e-6; Lb=3e-9; Lc=3e-9; a12=20; a13=20; TR1=[La, Lb, Lc, a12, a13, 1e-6, 5e-7, 5e-7];
TR2=[6e-6,6e-6,5.9e-6, 1e-6, 1e-6, 0];
Rd1on=0.002043; Rd1off=5e11;   Rd2on=0.0253; Rd2off=5e11; Q1on=0.3601; Q1off=100e6;
D1=[Rd1on, Rd1off]; D2=[Rd2on, Rd2off]; ZRD=[D1; D2]; ZRQ=[Q1on, Q1off];
VonL=0.8217; VonR=[0.3883, 0.8131, 0.0349]; IonL=VonL/ZLD(1); IonR=VonR./[ZRD(1,1),ZRD(2,1),ZRQ(1)];

VLfid=fopen('VLdata1.xls','w'); ILfid=fopen('ILdata1.xls','w');
VRfid=fopen('VRdata1.xls','w'); IRfid=fopen('IRdata1.xls','w');
XL=zeros(12,3);IL=zeros(16,3);ZL=zeros(8,1);XR=zeros(20,3);IR=zeros(27,3);ZR=zeros(12,1);
Vlink=[(XL(7,3)-XL(4,3)+XR(1,3)-XR(2,3))/2; (XL(4,3)-XL(6,3)+XR(2,3)-XR(3,3))/2];
D=impedance(1); ZLC=D{1}; ZLL=D{2}; YTL1=D{3}; YTL2=D{4}; BrL=D{5};
D=impedance(2); ZRC=D{1}; ZRL=D{2}; YRLM=D{3}; BrR=D{4}; YTR2=D{5}; MidYRtranx=D{6};
D=impedance(3); ZRCE=D{1}; ZRLE=D{2}; YRLME=D{3}; BrE=D{4}; YTR2E=D{5}; MidYRtranxE=D{6};
SDL=zeros(4,4); SDR=zeros(8,4); SQRcom=zeros(2,2);
SDLstack=ones(size(SDL(:,3)')); SDRstack=ones(size(SDR(:,3)'));
% Generate Y matrix for rectifier
D=sysmatrix(1); YL=D{1}; BL=D{2}; BLT=-BL'; DL=D{3}; invYL=inv(YL); invDL(:,:,1)=inv(DL);
```

```
PL3(:,:,1)=invDL(:,:,1)*BLT*invYL*BL; PL2(:,:,1)=invDL(:,:,1)*BLT; PL1=invYL*BL;
IPL3=eye(size(PL3(:,:,1))); PL4(:,:,1)=inv(IPL3+PL3(:,:,1));
EL(:,:,1)=invYL-PL1*PL4(:,:,1)*PL2(:,:,1)*invYL; GL(:,:,1)=PL4(:,:,1)*PL2(:,:,1)*invYL;
D=sysmatrix(2); YR=D{1}; BR=D{2}; BRT=-BR'; DR=D{3}; invYR=inv(YR); invDR(:,:,1)=inv(DR);
PR3(:,:,1)=invDR(:,:,1)*BRT*invYR*BR; PR2(:,:,1)=invDR(:,:,1)*BRT; PR1=invYR*BR;
IPR3=eye(size(PR3(:,:,1))); PR4(:,:,1)=inv(IPR3+PR3(:,:,1));
ER(:,:,1)=invYR-PR1*PR4(:,:,1)*PR2(:,:,1)*invYR; GR(:,:,1)=PR4(:,:,1)*PR2(:,:,1)*invYR;

%main program
n=1; k=1; signL=switchstate(n); signR=switchstate(n-1,k);
SDL(:,[1,2])=SDL(:,[3,4]); SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(1)=SQRcom(2);
VLinc(:,1)=VLinc(:,2); VLMinc(:,1)=VLMinc(:,2); VRinc(:,1)=VRinc(:,2); VRMinc(:,1)=VRMinc(:,2);
if NumofStart2Rec==1
  fprintf(VLfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', XL(1:8,3)');
  fprintf(ILfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IL(1:10,3)');
  fprintf(ILfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n',IL(11:16,3)' );
  fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',XR(1:10,3)');
  fprintf(VRfid,'%4.6f %4.6f %4.6f\n', XR(11:13,3)');
  fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(1:10,3)');
  fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(11:20,3)');
  fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', IR(21:27,3)');
end;
cond=0;
if SDL(:,1)'==SDLstack(1,:)
  Lstate=1;
else cond=1; Lstate=2;
end;
if cond==1
  SDLstack(Lstate,:)=SDL(:,1)';
  PL3(:,:,Lstate)=diag(SDL(:,2))*PL3(:,:,1); PL2(:,:,Lstate)=diag(SDL(:,2))*PL2(:,:,1);
  PL4(:,:,Lstate)=inv(IPL3+PL3(:,:,Lstate)); invDL(:,:,Lstate)=diag(SDL(:,2))*invDL(:,:,1);
  EL(:,:,Lstate)=invYL-PL1*PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
  GL(:,:,Lstate)=PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
end;
cond=0;
if SDR(:,1)'==SDRstack(1,:)
  Rstate=1;
else cond=1; Rstate=2;
end;
if cond==1
  SDRstack(Rstate,:)=SDR(:,1)';
  PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1); PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
  PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate)); invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
  ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
  GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
end;
getini=0;   %if getini=1, TLM models associated Backward Euler rule are used to start the simulation
%Simulation Procedure
for n=2:NumofPoints1          %Loop control according to the L-subnetwork time step.
  dV=[0;0];
  % Do the L-subnetwork calculation
  XL(:,[1,2])=XL(:,[2,3]); IL(:,[1,2])=IL(:,[2,3]); VLinc(:,1)=VLinc(:,2);
  VLMinc(:,1)=VLMinc(:,2); VLinc(11:12,2)=Vlink-VLinc(11:12,1);
```

```
ZL=histsrc(0,n);  XL(:,3)=[EL(:,:,Lstate);GL(:,:,Lstate)]*ZL;  IL(:,3)=current(1); signL=switchstate(n);
if SDL(:,1)==SDL(:,3)
  TswL=[TLstep TLstep TLstep TLstep];
else TswL=switchtime(1);
end;
TswLstd=[TLstep TLstep TLstep TLstep]; TswLtemp=TswL;
% Sort the switching time
if abs(TswL-TswLstd)<=deltaT        %state unchanged or changed at the calculation point
  IL11=linspace(IL(11,2),IL(11,3),NumofPoints2+1);        %linear interpolation
  IL12=linspace(IL(12,2),IL(12,3),NumofPoints2+1);
  if signL==1
    cond=1; SDL(:,[1,2])=SDL(:,[3,4]);
    for iloop=1:length(SDLstack(:,1))
      if SDL(:,1)'==SDLstack(iloop,:)
        Lstate=iloop; cond=0; break;
      end;
    end;
    if cond==1
      Lstate=iloop+1; SDLstack(Lstate,:)=SDL(:,1)'; PL3(:,:,Lstate)=diag(SDL(:,2))*PL3(:,:,1);
      PL2(:,:,Lstate)=diag(SDL(:,2))*PL2(:,:,1); PL4(:,:,Lstate)=inv(IPL3+PL3(:,:,Lstate));
      EL(:,:,Lstate)=invYL-PL1*PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
      GL(:,:,Lstate)=PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
    end;
  end;
  for k=2:NumofPoints2+1
    if getini<0.5
      XR(:,[1,2])=XR(:,[2,3]); IR(:,[1,2])=IR(:,[2,3]); VRinc(:,1)=VRinc(:,2);   VRMinc(:,1)=VRMinc(:,2);
      VRinc(4:5,2)=Vlink-VRinc(4:5,1);  ZR=histsrc(0,n-1,k);  XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
      IR(:,3)=current(2); signR=switchstate(n-1,k);
      if SDR(:,1)==SDR(:,3)
        TswR=[TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep];
      else   TswR=switchtime(2);
      end;
      TswRstd=[TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep]; TswRtemp=TswR;
      if abs(TswR-TswRstd)<deltaT        %state unchanged or changed at the calculation point
        dV=dV+[(IR(4,3)+IR(4,2)+IL11(k)+IL11(k-1))*ZRC(1);   (IR(5,3)+IR(5,2)+IL12(k)+IL12(k-1))*ZRC(2)];
        if signR==1
          SDR(1:6,[1,2])=SDR(1:6,[3,4]);
          if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
               | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
            getini=initial(n,k); %   getini=0;   %if getini=0, then only tranditional TLM models are used
          end;
          SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(:,1)=SQRcom(:,2); cond=1;
          for iloop=1:length(SDRstack(:,1))
            if SDR(:,1)'==SDRstack(iloop,:)
              Rstate=iloop; cond=0; break;
            end;
          end;
          if cond==1
            Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
            PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);   PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
            PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));   invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
            ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
          end;
        end;
        SQRcom(:,1)=SQRcom(:,2);
      else          %Switching action happens in   R-sub.   for (if abs(TswR-TswRstd)<deltaT)
        TswRtemp=[0,TswR]; TswRseq=[1,2,3,4,5,6,7,8];
        for p=2:9
          TswRcomp=TswRtemp(p);
          for q=p+1:9
            if TswRcomp > TswRtemp(q)
              TswRcomp=TswRtemp(q); TswRtemp(q)=TswRtemp(p); TswRtemp(p)=TswRcomp;
              seq=TswRseq(q-1); TswRseq(q-1)=TswRseq(p-1); TswRseq(p-1)=seq;
            end;
          end;
          if abs(TswRcomp-TRstep)<deltaT
            break;
          end;
        end;     %for p=2:8
        if abs(TswRtemp(9)-TRstep)>deltaT
          TswRtemp(10)=TRstep;
        end;
        IRtemp=IR(:,2); XRtemp=XR(:,2); m=0; TRini=0;
        while (~isequal(SDR(:,1),SDR(:,3)) | abs(TRini-TRstep)>deltaT)
          m=m+1; swRnum=0;
          %find the number of times that the switching happens at the same time
          for na=m+1:length(TswRtemp)
            if (abs(TswRtemp(na)-TswRtemp(m+1)) < deltaT)
              swRnum=swRnum+1;
            else break;
            end;
          end;
          %Obtain the state values at switching time t-
          TswRstep=TswRtemp(m+1)-TswRtemp(m);
          %using interpolation to get the state value at switching time t-
          XR(:,3)=XR(:,2)+(XR(:,3)-XR(:,2))/TRstep*TswRstep;
          IR(:,3)=IR(:,2)+(IR(:,3)-IR(:,2))/TRstep*TswRstep;
          VRinc(:,2)=VRinc(:,1)+(VRinc(:,2)-VRinc(:,1))/TRstep*TswRstep;
          VRMinc(:,2)=VRMinc(:,1)+(VRMinc(:,2)-VRMinc(:,1))/TRstep*TswRstep;
          dV=dV+[(IR(4,3)+IR(4,2)+(IL11(k-1)+(IL11(k)-IL11(k-1))...
               *(TswRtemp(m+1)+TswRtemp(m))/TRstep))*TswRstep/Cvector2(1)/2;
            (IR(5,3)+IR(5,2)+(IL12(k-1)+(IL12(k)-IL12(k-1))...
               *(TswRtemp(m+1)+TswRtemp(m))/TRstep))*TswRstep/Cvector2(2)/2];
          TRini=TswRtemp(m+1);
          %renew switch states
          for na=m:m+swRnum-1
            %Obtain the state values at switching time t+
            if (TswRseq(na)==7 | TswRseq(na)==8) & abs(TRini-TRstep)<deltaT
               if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
                   | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
                 getini=initial(n,k); % getini=0;    %if getini=0, then only tranditional TLM models are used
               end;
               SQRcom(:,1)=SQRcom(:,2);
            end;
            SDR(TswRseq(na),[1,2])=SDR(TswRseq(na),[3,4]);
          end;
          if abs(TRini-TRstep) > deltaT
            m=m+swRnum-1; XR(:,2)=XR(:,3); IR(:,2)=IR(:,3); VRinc(:,1)=VRinc(:,2);
            VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink-VRinc(4:5,1); cond=1;
            for iloop=1:length(SDRstack(:,1))
              if SDR(:,1)'==SDRstack(iloop,:)
                Rstate=iloop; cond=0; break;
              end;
            end;
            if cond==1
              Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
              PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);   PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
              PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));   invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
              ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
              GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            end;
            ZR=histsrc(m+1,n-1,k);   XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
            IR(:,3)=current(2); signR=switchstate(n-1,k);
            if SDR(:,1)==SDR(:,3)
              TswRtemp(m+2:m+9)=TRstep; TswRseq(m+1:m+8)=[1:8];
            else
              TswRini=switchtime(2);   TswRiniseq=[1,2,3,4,5,6,7,8];
              for p=1:8
                TswRcomp=TswRini(p); seq=TswRiniseq(p);
                for q=p+1:8
                  if TswRcomp>TswRini(q)
                    TswRcomp=TswRini(q); TswRini(q)=TswRini(p); TswRini(p)=TswRcomp;
                    seq=TswRiniseq(q);TswRiniseq(q)=TswRiniseq(p);TswRiniseq(p)=seq;
                  end;
                 end;
                 if TswRcomp<=TRstep-TswRtemp(m+1)
                  TswRtemp(m+p+1)=TswRcomp+TswRtemp(m+1);TswRseq(m+p)=TswRiniseq(p);
                 else
                  TswRtemp(m+p+1)=TRstep; TswRseq(m+p)=TswRiniseq(p);
                   SDR(TswRiniseq(p),[3,4])=SDR(TswRiniseq(p),[1,2]);
                 end;
              end;           %for (for p=1:8)
            end;           %for if isequal(SDR(:,1),SDR(:,3))
          end;
        end;           %for while (SDR(:,1)~=SDR(:,3) | abs(TRini-TRstep)<deltaT)
        IR(:,2)=IRtemp; XR(:,2)=XRtemp; cond=1;
        for iloop=1:length(SDRstack(:,1))
          if SDR(:,1)'==SDRstack(iloop,:)
            Rstate=iloop; cond=0; break;
          end;
        end;
        if cond==1
          Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
          PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);     PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
          PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));     invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
          ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
          GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
          end;
        end;
else
  VRMinc(1:2,2)=2*VRMinc(1:2,2);     VRinc(1:3,2)=2*VRinc(1:3,2);
  VRinc(4:9,2)=VRinc(4:9,2);   VRinc(10:12,2)=2*VRinc(10:12,2);
  VRinc(13,2)=VRinc(13,2); VRinc(14:15,2)=2*VRinc(14:15,2); VRinc(16:19,2)=VRinc(16:19,2);
  for k1=1:2
    XR(:,[1,2])=XR(:,[2,3]); IR(:,[1,2])=IR(:,[2,3]); VRinc(:,1)=VRinc(:,2);
    VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink/2;
    ZR=histsrc(-1,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
    IR(:,3)=current(3); signR=switchstate(n-1,k);
    if SDR(:,1)==SDR(:,3)
      TswR=[TRE,TRE,TRE,TRE,TRE,TRE,TRE,TRE];
    else   TswR=switchtime(3);
    end;
    TswRstd=[TRE,TRE,TRE,TRE,TRE,TRE,TRE,TRE]; TswRtemp=TswR;
    if abs(TswR-TswRstd)<deltaT        %state unchanged or changed at the calculation point
      dV=dV+[(IR(4,3)+IL11(k))*ZRC(1); (IR(5,3)+IL12(k))*ZRC(2)];    VRinc(4:5,2)=Vlink/2;
      if signR==1
        SDR(1:6,[1,2])=SDR(1:6,[3,4]);
        if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
             | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
          getini=initial(n,k);    %     getini=0;    %if getini=0, then only tranditional TLM models are used
        end;
        SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(:,1)=SQRcom(:,2); cond=1;
        for iloop=1:length(SDRstack(:,1))
          if SDR(:,1)'==SDRstack(iloop,:)
            Rstate=iloop; cond=0; break;
          end;
        end;
        if cond==1
          Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
          PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1); PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
          PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate)); invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
          ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
          GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
        end;
      end;
      SQRcom(:,1)=SQRcom(:,2);
    else           %Switching action happens in   R-sub.   for (if abs(TswR-TswRstd)<deltaT)
      TswRtemp=[0,TswR]; TswRseq=[1,2,3,4,5,6,7,8];
      for p=2:9
        TswRcomp=TswRtemp(p);
        for q=p+1:9
          if TswRcomp > TswRtemp(q)
            TswRcomp=TswRtemp(q); TswRtemp(q)=TswRtemp(p); TswRtemp(p)=TswRcomp;
            seq=TswRseq(q-1); TswRseq(q-1)=TswRseq(p-1); TswRseq(p-1)=seq;
          end;
        end;
        if abs(TswRcomp-TRE)<deltaT
          break;
        end;
      end;     %for p=2:8
```

```
if abs(TswRtemp(9)-TRE)>deltaT
  TswRtemp(10)=TRE;
end;
IRtemp=IR(:,2); XRtemp=XR(:,2); m=0; TRini=0;
while (~isequal(SDR(:,1),SDR(:,3)) | abs(TRini-TRE)>deltaT)
  m=m+1; swRnum=0;
  %find the number of times that the switching happens at the same time
  for na=m+1:length(TswRtemp)
    if (abs(TswRtemp(na)-TswRtemp(m+1)) < deltaT)
      swRnum=swRnum+1;
    else break;
    end;
  end;
  %Obtain the state values at switching time t-
  TswRstep=TswRtemp(m+1)-TswRtemp(m);
  %using interpolation to get the state value at switching time t-
  XR(:,3)=XR(:,2)+(XR(:,3)-XR(:,2))/TRE*TswRstep;
  IR(:,3)=IR(:,2)+(IR(:,3)-IR(:,2))/TRE*TswRstep;
  VRinc(:,2)=VRinc(:,1)+(VRinc(:,2)-VRinc(:,1))/TRE*TswRstep;
  VRMinc(:,2)=VRMinc(:,1)+(VRMinc(:,2)-VRMinc(:,1))/TRE*TswRstep;
  dV=dV+[(IR(4,3)+(IL11(k-1)+(IL11(k)-IL11(k-1)) *TswRtemp(m+1)/TRE))*TswRstep/Cvector2(1);
    (IR(5,3)+(IL12(k-1)+(IL12(k)-IL12(k-1))*TswRtemp(m+1)/TRE))*TswRstep/Cvector2(2)];
  VRinc(4:5,2)=Vlink/2;    TRini=TswRtemp(m+1);
  %renew switch states
  for na=m:m+swRnum-1
    %Obtain the state values at switching time t+
    if (TswRseq(na)==7 | TswRseq(na)==8) & abs(TRini-TRE)<deltaT
      if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
          | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
        getini=initial(n,k);    %    getini=0;    %if getini=0, then only tranditional TLM models are used
      end;
      SQRcom(:,1)=SQRcom(:,2);
    end;
    SDR(TswRseq(na),[1,2])=SDR(TswRseq(na),[3,4]);
  end;
  if abs(TRini-TRE) > deltaT
    m=m+swRnum-1; XR(:,2)=XR(:,3); IR(:,2)=IR(:,3); VRinc(:,1)=VRinc(:,2);
    VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink/2; cond=1;
    for iloop=1:length(SDRstack(:,1))
      if SDR(:,1)'==SDRstack(iloop,:)
        Rstate=iloop; cond=0; break;
      end;
    end;
    if cond==1
      Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
      PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);    PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
      PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));    invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
      ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
      GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
    end;
    ZR=histsrc(-1,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
    IR(:,3)=current(3); signR=switchstate(n-1,k);
    if SDR(:,1)==SDR(:,3)
```

```
                    TswRtemp(m+2:m+9)=TRE; TswRseq(m+1:m+8)=[1:8];
                  else
                    TswRini=switchtime(3); TswRiniseq=[1,2,3,4,5,6,7,8];
                    for p=1:8
                      TswRcomp=TswRini(p); seq=TswRiniseq(p);
                      for q=p+1:8
                        if TswRcomp>TswRini(q)
                          TswRcomp=TswRini(q); TswRini(q)=TswRini(p); TswRini(p)=TswRcomp;
                          seq=TswRiniseq(q);TswRiniseq(q)=TswRiniseq(p);TswRiniseq(p)=seq;
                        end;
                      end;
                      if TswRcomp<=TRE-TswRtemp(m+1)
                        TswRtemp(m+p+1)=TswRcomp+TswRtemp(m+1);TswRseq(m+p)=TswRiniseq(p);
                      else
                        TswRtemp(m+p+1)=TRE; TswRseq(m+p)=TswRiniseq(p);
                        SDR(TswRiniseq(p),[3,4])=SDR(TswRiniseq(p),[1,2]);
                      end;
                    end;        %for (for p=1:8)
                  end;          %for if isequal(SDR(:,1),SDR(:,3))
                end;
              end;          %for while (SDR(:,1)~=SDR(:,3) | abs(TRini-TRE)<deltaT)
              IR(:,2)=IRtemp; XR(:,2)=XRtemp; cond=1;
              for iloop=1:length(SDRstack(:,1))
                if SDR(:,1)'==SDRstack(iloop,:)
                  Rstate=iloop; cond=0; break;
                end;
              end;
              if cond==1
                Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
                PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);    PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
                PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));    invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
                ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
                GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
              end;
            end;
          end;
          VRMinc(1:2,2)=VRMinc(1:2,2)/2;   VRinc(1:3,2)=VRinc(1:3,2)/2;
          VRinc(10:12,2)=VRinc(10:12,2)/2;   VRinc(14:15,2)=VRinc(14:15,2)/2;
          getini=0; %if getini=1, then only TLM models associated with Backward Euler are used
        end;
        if n>NumofStart2Rec & rem(k-1,Step4RecR)==0
          fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',XR(1:9,3)');
          fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f\n', XR(10:13,3)');
          fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(1:9,3)');
          fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(10:19,3)');
          fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', IR(20:27,3)');
        end;
      end;
    else
      TswLseq=[1,2,3,4]; TswLtemp=[0,TswL];
      %sorting the switching time
      for j=2:5
        TswLcomp=TswLtemp(j);
        for k=j+1:5
          if TswLcomp > TswLtemp(k)
            TswLcomp=TswLtemp(k); TswLtemp(k)=TswLtemp(j); TswLtemp(j)=TswLcomp;
            seq=TswLseq(k-1); TswLseq(k-1)=TswLseq(j-1); TswLseq(j-1)=seq;
          end;
        end;
        if abs(TswLcomp-TLstep)<deltaT
          break;
        end;
      end;
      j=0; TLini=0; ILtemp=IL(:,2); XLtemp=XL(:,2);
      while (~isequal(SDL(:,1),SDL(:,3)) | abs(TLini-TLstep)>deltaT)
        j=j+1; swLnum=0;
        %Finding how mnay times the switching actions happen at the same time
        for na=j+1:length(TswLtemp)
          if (abs(TswLtemp(na)-TswLtemp(j+1)) < deltaT)
            swLnum=swLnum+1;
          else break;
          end;
        end;
        TswLstep=TswLtemp(j+1)-TswLtemp(j);
        XL(:,3)=XL(:,2)+(XL(:,3)-XL(:,2))/TLstep*TswLstep;
        IL(:,3)=IL(:,2)+(IL(:,3)-IL(:,2))/TLstep*TswLstep;
        VLinc(:,2)=VLinc(:,1)+(VLinc(:,2)-VLinc(:,1))/TLstep*TswLstep;
        VLMinc(:,2)=VLMinc(:,1)+(VLMinc(:,2)-VLMinc(:,1))/TLstep*TswLstep;
        if abs(TswLtemp(j+1)-TswLtemp(j))>TRstep/2
          IL11(round(TswLtemp(j)/TRstep)+1:round(TswLtemp(j+1)/TRstep)+1)...
            =linspace(IL(11,2),IL(11,3),round(TswLstep/TRstep+1));
          IL12(round(TswLtemp(j)/TRstep)+1:round(TswLtemp(j+1)/TRstep)+1)...
            =linspace(IL(12,2),IL(12,3),round(TswLstep/TRstep+1));
          %R-subnetwork calculation
          for k=round(TswLtemp(j)/TRstep)+2:round(TswLtemp(j+1)/TRstep)+1
            if getini<0.5
              XR(:,[1,2])=XR(:,[2,3]); IR(:,[1,2])=IR(:,[2,3]); VRinc(:,1)=VRinc(:,2); VRMinc(:,1)=VRMinc(:,2);
              VRinc(4:5,2)=Vlink-VRinc(4:5,1); ZR=histsrc(0,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
              IR(:,3)=current(2); signR=switchstate(n-1,k);
              if SDR(:,1)==SDR(:,3)
                TswR=[TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep];
              else    TswR=switchtime(2);
              end;
              TswRstd=[TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep]; TswRtemp=TswR;
              if abs(TswR-TswRstd)<deltaT         %state unchanged or changed at the calculation point
                dV=dV+[(IR(4,3)+IR(4,2)+IL11(k)+IL11(k-1))*ZRC(1);   (IR(5,3)+IR(5,2)+IL12(k)+IL12(k-1))*ZRC(2)];
                if signR==1
                  if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
                       | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
                    getini=initial(n,k); %    getini=0; %if getini=0, then only tranditional TLM models are used
                  end;
                  SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(:,1)=SQRcom(:,2);    cond=1;
                  for iloop=1:length(SDRstack(:,1))
                    if SDR(:,1)'==SDRstack(iloop,:)
                      Rstate=iloop; cond=0; break;
                    end;
          end;
          if cond==1
            Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
            PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);   PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
            PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));   invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
            ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
          end;
        end;
        SQRcom(:,1)=SQRcom(:,2);
      else          %Switching action happens in   R-sub.   for (if abs(TswR-TswRstd)<deltaT)
        TswRtemp=[0,TswR]; TswRseq=[1,2,3,4,5,6,7,8];
        for p=2:9
          TswRcomp=TswRtemp(p);
          for q=p+1:9
            if TswRcomp > TswRtemp(q)
              TswRcomp=TswRtemp(q); TswRtemp(q)=TswRtemp(p); TswRtemp(p)=TswRcomp;
              seq=TswRseq(q-1); TswRseq(q-1)=TswRseq(p-1); TswRseq(p-1)=seq;
            end;
          end;
          if abs(TswRcomp-TRstep)<deltaT
            break;
          end;
        end;      %for p=2:8
        if abs(TswRtemp(9)-TRstep)>deltaT
          TswRtemp(10)=TRstep;
        end;
        IRtemp=IR(:,2); XRtemp=XR(:,2); m=0; TRini=0;
        while (~isequal(SDR(:,1),SDR(:,3)) | abs(TRini-TRstep)>deltaT)
          m=m+1; swRnum=0;
          %find the number of times that the switching happens at the same time
          for na=m+1:length(TswRtemp)
            if (abs(TswRtemp(na)-TswRtemp(m+1)) < deltaT)
              swRnum=swRnum+1;
            else break;
            end;
          end;
          %Obtain the state values at switching time t-
          TswRstep=TswRtemp(m+1)-TswRtemp(m);
          %using interpolation to get the state value at switching time t-
          XR(:,3)=XR(:,2)+(XR(:,3)-XR(:,2))/TRstep*TswRstep;
          IR(:,3)=IR(:,2)+(IR(:,3)-IR(:,2))/TRstep*TswRstep;
          VRinc(:,2)=VRinc(:,1)+(VRinc(:,2)-VRinc(:,1))/TRstep*TswRstep;
          VRMinc(:,2)=VRMinc(:,1)+(VRMinc(:,2)-VRMinc(:,1))/TRstep*TswRstep;
          dV=dV+[(IR(4,3)+IR(4,2)+(IL11(k-1)+(IL11(k)-IL11(k-1))...
              *(TswRtemp(m+1)+TswRtemp(m))/TRstep))*TswRstep/Cvector2(1)/2;
            (IR(5,3)+IR(5,2)+(IL12(k-1)+(IL12(k)-IL12(k-1))...
              *(TswRtemp(m+1)+TswRtemp(m))/TRstep))*TswRstep/Cvector2(2)/2];
          TRini=TswRtemp(m+1);
          %renew switch states
          for na=m:m+swRnum-1
            %Obtain the state values at switching time t+
            if (TswRseq(na)==7 | TswRseq(na)==8) & abs(TRini-TRstep)<deltaT
              if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
                    | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
                getini=initial(n,k); %   getini=0; %if getini=0, then only tranditional TLM models are used
              end;
              SQRcom(:,1)=SQRcom(:,2);
            end;
            SDR(TswRseq(na),[1,2])=SDR(TswRseq(na),[3,4]);
          end;
          if abs(TswRtemp(m+1)-TRstep) > deltaT
            m=m+swRnum-1; XR(:,2)=XR(:,3); IR(:,2)=IR(:,3); VRinc(:,1)=VRinc(:,2);
            VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink-VRinc(4:5,1); cond=1;
            for iloop=1:length(SDRstack(:,1))
              if SDR(:,1)'==SDRstack(iloop,:)
                Rstate=iloop; cond=0; break;
              end;
            end;
            if cond==1
              Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
              PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);     PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
              PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));     invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
              ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
              GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            end;
            ZR=histsrc(m+1,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
            IR(:,3)=current(2); signR=switchstate(n-1,k);
            if SDR(:,1)==SDR(:,3)
              TswRtemp(m+2:m+9)=TRstep; TswRseq(m+1:m+8)=[1:8];
            else
              TswRini=switchtime(2); TswRiniseq=[1,2,3,4,5,6,7,8];
              for p=1:8
                TswRcomp=TswRini(p); seq=TswRiniseq(p);
                for q=p+1:8
                  if TswRcomp>TswRini(q)
                    TswRcomp=TswRini(q); TswRini(q)=TswRini(p); TswRini(p)=TswRcomp;
                    seq=TswRiniseq(q);TswRiniseq(q)=TswRiniseq(p);TswRiniseq(p)=seq;
                  end;
                end;
                if TswRcomp<=TRstep-TswRtemp(m+1)
                  TswRtemp(m+p+1)=TswRcomp+TswRtemp(m+1);TswRseq(m+p)=TswRiniseq(p);
                else
                  TswRtemp(m+p+1)=TRstep; TswRseq(m+p)=TswRiniseq(p);
                  SDR(TswRiniseq(p),[3,4])=SDR(TswRiniseq(p),[1,2]);
                end;
              end;          %for (for p=1:8)
            end;          %for if isequal(SDR(:,1),SDR(:,3))
          end;
        end;          %for while (SDR(:,1)~=SDR(:,3) | abs(TRini-TRstep)<deltaT)
        IR(:,2)=IRtemp; XR(:,2)=XRtemp; cond=1;
        for iloop=1:length(SDRstack(:,1))
          if SDR(:,1)'==SDRstack(iloop,:)
            Rstate=iloop; cond=0; break;
          end;
        end;
```

```
            if cond==1
               Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
               PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1); PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
               PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate)); invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
               ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
               GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            end;
         end;
      else
VRMinc(1:2,2)=2*VRMinc(1:2,2); VRinc(1:3,2)=2*VRinc(1:3,2); VRinc(4:9,2)=VRinc(4:9,2);
VRinc(10:12,2)=2*VRinc(10:12,2); VRinc(13,2)=VRinc(13,2); VRinc(14:15,2)=2*VRinc(14:15,2);
         VRinc(16:19,2)=VRinc(16:19,2);
         for k1=1:2
            XR(:,[1,2])=XR(:,[2,3]); IR(:,[1,2])=IR(:,[2,3]); VRinc(:,1)=VRinc(:,2);
            VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink/2;
            ZR=histsrc(-1,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
            IR(:,3)=current(3); signR=switchstate(n-1,k);
            if SDR(:,1)==SDR(:,3)
               TswR=[TRE,TRE,TRE,TRE,TRE,TRE,TRE,TRE];
            else TswR=switchtime(3);
            end;
            TswRstd=[TRE,TRE,TRE,TRE,TRE,TRE,TRE,TRE]; TswRtemp=TswR;
            if abs(TswR-TswRstd)<deltaT       %state unchanged or changed at the calculation point
               dV=dV+[(IR(4,3)+IL11(k))*ZRC(1); (IR(5,3)+IL12(k))*ZRC(2)]; VRinc(4:5,2)=Vlink/2;
               if signR==1
                  SDR(1:6,[1,2])=SDR(1:6,[3,4]);
                  if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
                        | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
                     getini=initial(n,k); %    getini=0;    %if getini=0, then only tranditional TLM models are used
                  end;
                  SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(:,1)=SQRcom(:,2); cond=1;
                  for iloop=1:length(SDRstack(:,1))
                     if SDR(:,1)'==SDRstack(iloop,:)
                        Rstate=iloop; cond=0; break;
                     end;
                  end;
                  if cond==1
                     Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
                     PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);   PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
                     PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate)); invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
                     ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
                     GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
                  end;
               end;
               SQRcom(:,1)=SQRcom(:,2);
            else           %Switching action happens in   R-sub.   for (if abs(TswR-TswRstd)<deltaT)
               TswRtemp=[0,TswR]; TswRseq=[1,2,3,4,5,6,7,8];
               for p=2:9
                  TswRcomp=TswRtemp(p);
                  for q=p+1:9
                     if TswRcomp > TswRtemp(q)
                        TswRcomp=TswRtemp(q); TswRtemp(q)=TswRtemp(p); TswRtemp(p)=TswRcomp;
                        seq=TswRseq(q-1); TswRseq(q-1)=TswRseq(p-1); TswRseq(p-1)=seq;
```

```
        end;
      end;
      if abs(TswRcomp-TRE)<deltaT
        break;
      end;
    end;    %for p=2:8
    if abs(TswRtemp(9)-TRE)>deltaT
      TswRtemp(10)=TRE;
    end;
    IRtemp=IR(:,2); XRtemp=XR(:,2); m=0; TRini=0;
    while (~isequal(SDR(:,1),SDR(:,3)) | abs(TRini-TRE)>deltaT)
      m=m+1; swRnum=0;
      %find the number of times that the switching happens at the same time
      for na=m+1:length(TswRtemp)
        if (abs(TswRtemp(na)-TswRtemp(m+1)) < deltaT)
          swRnum=swRnum+1;
        else break;
        end;
      end;
      %Obtain the state values at switching time t-
      TswRstep=TswRtemp(m+1)-TswRtemp(m);
      %using interpolation to get the state value at switching time t-
      XR(:,3)=XR(:,2)+(XR(:,3)-XR(:,2))/TRE*TswRstep;
      IR(:,3)=IR(:,2)+(IR(:,3)-IR(:,2))/TRE*TswRstep;
      VRinc(:,2)=VRinc(:,1)+(VRinc(:,2)-VRinc(:,1))/TRE*TswRstep;
      VRMinc(:,2)=VRMinc(:,1)+(VRMinc(:,2)-VRMinc(:,1))/TRE*TswRstep;
      dV=dV+[(IR(4,3)+(IL11(k-1)+(IL11(k)-IL11(k-1))*TswRtemp(m+1)/TRE))*TswRstep/Cvector2(1);
        (IR(5,3)+(IL12(k-1)+(IL12(k)-IL12(k-1))*TswRtemp(m+1)/TRE))*TswRstep/Cvector2(2)];
      VRinc(4:5,2)=Vlink/2; TRini=TswRtemp(m+1);
      %renew switch states
      for na=m:m+swRnum-1
        %Obtain the state values at switching time t+
        if (TswRseq(na)==7 | TswRseq(na)==8) & abs(TRini-TRE)<deltaT
          if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
              | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
            getini=initial(n,k); %    getini=0; %if getini=0, then only tranditional TLM models are used
          end;
          SQRcom(:,1)=SQRcom(:,2);
        end;
        SDR(TswRseq(na),[1,2])=SDR(TswRseq(na),[3,4]);
      end;
      if abs(TRini-TRE) > deltaT
        m=m+swRnum-1; XR(:,2)=XR(:,3); IR(:,2)=IR(:,3); VRinc(:,1)=VRinc(:,2);
        VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink/2; cond=1;
        for iloop=1:length(SDRstack(:,1))
          if SDR(:,1)'==SDRstack(iloop,:)
            Rstate=iloop; cond=0; break;
          end;
        end;
        if cond==1
          Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
          PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1); PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
          PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));   invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
```

```
                    ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
                    GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
                  end;
                  ZR=histsrc(-1,n-1,k);    XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
                  IR(:,3)=current(3); signR=switchstate(n-1,k);
                  if SDR(:,1)==SDR(:,3)
                    TswRtemp(m+2:m+9)=TRE; TswRseq(m+1:m+8)=[1:8];
                  else
                    TswRini=switchtime(3); TswRiniseq=[1,2,3,4,5,6,7,8];
                    for p=1:8
                      TswRcomp=TswRini(p); seq=TswRiniseq(p);
                      for q=p+1:8
                        if TswRcomp>TswRini(q)
                          TswRcomp=TswRini(q); TswRini(q)=TswRini(p); TswRini(p)=TswRcomp;
                          seq=TswRiniseq(q);TswRiniseq(q)=TswRiniseq(p);TswRiniseq(p)=seq;
                        end;
                      end;
                      if TswRcomp<=TRE-TswRtemp(m+1)
                        TswRtemp(m+p+1)=TswRcomp+TswRtemp(m+1);TswRseq(m+p)=TswRiniseq(p);
                      else
                        TswRtemp(m+p+1)=TRE; TswRseq(m+p)=TswRiniseq(p);
                        SDR(TswRiniseq(p),[3,4])=SDR(TswRiniseq(p),[1,2]);
                      end;
                    end;          %for (for p=1:8)
                  end;          %for if isequal(SDR(:,1),SDR(:,3))
                end;
              end;          %for while (SDR(:,1)~=SDR(:,3) | abs(TRini-TRE)<deltaT)
              IR(:,2)=IRtemp; XR(:,2)=XRtemp; cond=1;
              for iloop=1:length(SDRstack(:,1))
                if SDR(:,1)'==SDRstack(iloop,:)
                  Rstate=iloop; cond=0; break;
                end;
              end;
              if cond==1
                Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
                PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);     PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
                PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));     invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
                ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
                GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
              end;
            end;
          end;
          VRMinc(1:2,2)=VRMinc(1:2,2)/2;
          VRinc(1:3,2)=VRinc(1:3,2)/2;
          VRinc(10:12,2)=VRinc(10:12,2)/2;
          VRinc(14:15,2)=VRinc(14:15,2)/2;
          getini=0; %if getini=1, then only TLM models associated with Backward Euler are used
        end;
        if n>NumofStart2Rec & rem(k-1,Step4RecR)==0
          fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f',XR(1:9,3)');
          fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f\n', XR(10:13,3)');
          fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f',IR(1:9,3)');
          fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f',IR(10:19,3)');
          fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', IR(20:27,3)');
        end;
      end;
    end;
    IL11(round(TswLtemp(j+1)/TRstep)+1)=IL(11,3);
    IL12(round(TswLtemp(j+1)/TRstep)+1)=IL(12,3);
    for na=j:j+swLnum-1
      SDL(TswLseq(na),[1,2])=SDL(TswLseq(na),[3,4]);
      TswLtemp(na+1)=round(TswLtemp(na+1)/TRstep)*TRstep;
    end;
    TLini=TswLtemp(j+1);
    if abs(TLini-TLstep)>deltaT
      XL(:,2)=XL(:,3); IL(:,2)=IL(:,3); VLinc(:,1)=VLinc(:,2); VLMinc(:,1)=VLMinc(:,2);
      VLinc(11:12,2)=Vlink-VLinc(11:12,1); cond=1;
      for iloop=1:length(SDLstack(:,1))
        if SDL(:,1)'==SDLstack(iloop,:)
          Lstate=iloop; cond=0; break;
        end;
      end;
      if cond==1
        Lstate=iloop+1; SDLstack(Lstate,:)=SDL(:,1)';
        PL3(:,:,Lstate)=diag(SDL(:,2))*PL3(:,:,1); PL2(:,:,Lstate)=diag(SDL(:,2))*PL2(:,:,1);
        PL4(:,:,Lstate)=inv(IPL3+PL3(:,:,Lstate)); invDL(:,:,Lstate)=diag(SDL(:,2))*invDL(:,:,1);
        EL(:,:,Lstate)=invYL-PL1*PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
        GL(:,:,Lstate)=PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
      end;
      ZL=histsrc(j,n); XL(:,3)=[EL(:,:,Lstate);GL(:,:,Lstate)]*ZL; IL(:,3)=current(1); signL=switchstate(n);
      j=j+swLnum-1;
      if SDL(:,1)==SDL(:,3)
        TswLtemp(j+2:j+5)=TLstep; TswLseq(j+1:j+4)=[1,2,3,4];
      else
        TswLini=switchtime(1);
        TswLiniseq=[1,2,3,4];
        for p=1:4
          TswLcomp=TswLini(p); seq=TswLiniseq(p);
          for q=p+1:4
            if TswLcomp>TswLini(q)
              TswLcomp=TswLini(q); TswLini(q)=TswLini(p); TswLini(p)=TswLcomp;
              seq=TswLiniseq(q); TswLiniseq(q)=TswLiniseq(p); TswLiniseq(p)=seq;
            end;
          end;
          if TswLcomp<=TLstep-TswLtemp(j+1)
            TswLtemp(j+p+1)=TswLcomp+TswLtemp(j+1);TswLseq(j+p)=seq;
          else
            TswLtemp(j+p+1)=TLstep; TswLseq(j+p)=seq; SDL(seq,[3,4])=SDL(seq,[1,2]);
          end;
        end;
      end;
    end;
  end;
  IL(:,2)=ILtemp; XL(:,2)=XLtemp; cond=1; SDL(:,[1,2])=SDL(:,[3,4]);
  for iloop=1:length(SDLstack(:,1))
    if SDL(:,1)'==SDLstack(iloop,:)
```

```
            Lstate=iloop; cond=0; break;
         end;
      end;
      if cond==1
         Lstate=iloop+1; SDLstack(Lstate,:)=SDL(:,1)';
         PL3(:,:,Lstate)=diag(SDL(:,2))*PL3(:,:,1);PL2(:,:,Lstate)=diag(SDL(:,2))*PL2(:,:,1);
         PL4(:,:,Lstate)=inv(IPL3+PL3(:,:,Lstate));
         invDL(:,:,Lstate)=diag(SDL(:,2))*invDL(:,:,1);
         EL(:,:,Lstate)=invYL-PL1*PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
         GL(:,:,Lstate)=PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
      end;
   end;
   Vlink=Vlink+dV;
   VRinc(4:5,2)=(Vlink-[IR(4,3)*ZRC(1);IR(5,3)*ZRC(2)])/2;
   VLinc(11:12,2)=(Vlink-[IL(11,3)*ZLC(5);IL(12,3)*ZLC(6)])/2;
   if n>NumofStart2Rec & rem(n-1,Step4RecL)==0
      fprintf(VLfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', XL(1:8,3)');
      fprintf(ILfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IL(1:10,3)');
      fprintf(ILfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n',IL(11:16,3)' );
   end;
end;
T1=toc; fclose(VLfid); fclose(ILfid); fclose(VRfid); fclose(IRfid);
T1fid=fopen('T1data1.xls','w'); fprintf(T1fid,'%12.3f',T1); fclose(T1fid);
disp('1Program End');
```

II. TrapwithBE3

```
clear; clc; disp('TrapWithBE Start'); tic
global Rvector1 Cvector1 Lvector1 TL1 TL2 Rvector2 Cvector2 Lvector2 TR1 TR2 YLLM YRLM
global YRLME ZLD ZRD ZRQ ZRC BrL BrR MidYRtranx MIdYRtranxE ZRCE BrE
global SDL SDR SQRcom IonL IonR Vlink deltaT
global TLstep TRstep TRE TswRtemp TswLtemp NumofPoints2
global VLinc VRinc VLMinc VRMinc ILpre IRpre deltaV XL XR IL IR VonL VonR
global freq T4saw T4Q1th T4Q2th Dmax Mag Phy w TLstep TRstep

% Basic time step. time step TLstep/TRstep=integer TRstep/TRmin=integer
TLstep=20e-6; TRstep=4e-7; TRE=TRstep/2; Time=50e-3; deltaT=5e-15; deltaV=1e-12;
Tstart2rec=Time-0.05; Trec4L=1*TLstep; Trec4R=1*TRstep;
NumofStart2Rec=round(Tstart2rec/TLstep)+1; NumofPoints1=round(Time/TLstep)+1;
NumofPoints2=round(TLstep/TRstep);
Step4RecL=round(Trec4L/TLstep); Step4RecR=round(Trec4R/TRstep);
%PWM control part
Dmax=0.9; freq=25e3; T4PWM=floor(1/freq/TRstep/2)*TRstep*2; T4saw=T4PWM/TRstep/2;
T4Q1th=floor(T4saw*Dmax); T4Q2th=T4Q1th+T4saw; VLinc=zeros(14,2); VRinc=zeros(19,2);
VLMinc=zeros(4,2); VRMinc=zeros(2,2);
w=2*pi*60; Phy=[0;0]; Mag=[120*sqrt(2); 2*sqrt(2)]; wpwm=2*pi*720;
% Rectifier part
Rs=2e-2; R1=510e3; R2=330e3; R3=330e3; Rvector1=[Rs/2, R1, R2, R3]; Ls=2e-6;
Lvector1=[Ls/2]; C1=220e-9; C2=220e-9; C3=2e-9; C4=2e-9; C5=680e-6; C6=680e-6;
Cvector1=[C1, C2, C3, C4, C5, C6]; TL1=[10e-6, 10e-6, 0e-6, 0e-6, 0e-7, 0];
TL2=[3.0e-3, 3.0e-3, 2.997e-3, 1e-3, 1e-3, 0]; Rdon=0.0119; Rdoff=5e11; ZLD=[Rdon, Rdoff];
% DC-DC converter
```

```
C7=2e-6; C8=2e-9; C9=3300e-6; C10=3300e-6; C11=470e-6; C=10e-9;
Cvector2=[C5, C6, C7, C8, C9, C10, C11, C, C, C, C];
R4=47; RL5=0.3; RLN5=1; R=4.7; Rvector2=[R4, R, R, R, R, RL5, RLN5, 0.01];
L1=1e-6;L2=1e-6; L3=10e-6; Lvector2=[L1, L2, L3];
La=3e-6; Lb=3e-9; Lc=3e-9; a12=20; a13=20; TR1=[La, Lb, Lc, a12, a13, 1e-6, 5e-7, 5e-7];
TR2=[6e-6,6e-6,5.9e-6, 1e-6, 1e-6, 0];
Rd1on=0.002043; Rd1off=5e11;   Rd2on=0.0253; Rd2off=5e11; Q1on=0.3601; Q1off=100e6;
D1=[Rd1on, Rd1off]; D2=[Rd2on, Rd2off]; ZRD=[D1; D2]; ZRQ=[Q1on, Q1off];
VonL=0.8217; VonR=[0.3883, 0.8131, 0.0349]; IonL=VonL/ZLD(1); IonR=VonR./[ZRD(1,1),ZRD(2,1),ZRQ(1)];

VLfid=fopen('VLdata3.xls','w'); ILfid=fopen('ILdata3.xls','w');
VRfid=fopen('VRdata3.xls','w'); IRfid=fopen('IRdata3.xls','w');
XL=zeros(12,3);IL=zeros(16,3);ZL=zeros(8,1);XR=zeros(20,3);IR=zeros(27,3);ZR=zeros(12,1);
Vlink=[(XL(7,3)-XL(4,3)+XR(1,3)-XR(2,3))/2; (XL(4,3)-XL(6,3)+XR(2,3)-XR(3,3))/2];
D=impedance(1); ZLC=D{1}; ZLL=D{2}; YTL1=D{3}; YTL2=D{4}; BrL=D{5};
D=impedance(2); ZRC=D{1}; ZRL=D{2}; YRLM=D{3}; BrR=D{4}; YTR2=D{5}; MidYRtranx=D{6};
D=impedance(3); ZRCE=D{1}; ZRLE=D{2}; YRLME=D{3}; BrE=D{4}; YTR2E=D{5}; MidYRtranxE=D{6};

SDL=zeros(4,4); SDR=zeros(8,4); SQRcom=zeros(2,2);
SDLstack=ones(size(SDL(:,3)')); SDRstack=ones(size(SDR(:,3)'));
D=sysmatrix(1); YL=D{1}; BL=D{2}; BLT=-BL'; DL=D{3}; invYL=inv(YL); invDL(:,:,1)=inv(DL);
PL3(:,:,1)=invDL(:,:,1)*BLT*invYL*BL; PL2(:,:,1)=invDL(:,:,1)*BLT; PL1=invYL*BL;
IPL3=eye(size(PL3(:,:,1))); PL4(:,:,1)=inv(IPL3+PL3(:,:,1));
EL(:,:,1)=invYL-PL1*PL4(:,:,1)*PL2(:,:,1)*invYL; GL(:,:,1)=PL4(:,:,1)*PL2(:,:,1)*invYL;
D=sysmatrix(2); YR=D{1}; BR=D{2}; BRT=-BR'; DR=D{3}; invYR=inv(YR); invDR(:,:,1)=inv(DR);
PR3(:,:,1)=invDR(:,:,1)*BRT*invYR*BR; PR2(:,:,1)=invDR(:,:,1)*BRT; PR1=invYR*BR;
IPR3=eye(size(PR3(:,:,1))); PR4(:,:,1)=inv(IPR3+PR3(:,:,1));
ER(:,:,1)=invYR-PR1*PR4(:,:,1)*PR2(:,:,1)*invYR; GR(:,:,1)=PR4(:,:,1)*PR2(:,:,1)*invYR;

%main program
n=1; k=1; signL=switchstate(n); signR=switchstate(n-1,k);
SDL(:,[1,2])=SDL(:,[3,4]); SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(1)=SQRcom(2);
VLinc(:,1)=VLinc(:,2); VLMinc(:,1)=VLMinc(:,2); VRinc(:,1)=VRinc(:,2); VRMinc(:,1)=VRMinc(:,2);
if NumofStart2Rec==1
   fprintf(VLfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', XL(1:8,3)');
   fprintf(ILfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IL(1:10,3)');
   fprintf(ILfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n',IL(11:16,3)' );
   fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',XR(1:10,3)');
   fprintf(VRfid,'%4.6f %4.6f %4.6f\n', XR(11:13,3)');
   fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(1:10,3)');
   fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(11:20,3)');
   fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', IR(21:27,3)');
end;
cond=0;
if SDL(:,1)'==SDLstack(1,:)
   Lstate=1;
else cond=1; Lstate=2;
end;
if cond==1
   SDLstack(Lstate,:)=SDL(:,1)';
   PL3(:,:,Lstate)=diag(SDL(:,2))*PL3(:,:,1); PL2(:,:,Lstate)=diag(SDL(:,2))*PL2(:,:,1);
   PL4(:,:,Lstate)=inv(IPL3+PL3(:,:,Lstate)); invDL(:,:,Lstate)=diag(SDL(:,2))*invDL(:,:,1);
   EL(:,:,Lstate)=invYL-PL1*PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
```

```
  GL(:,:,Lstate)=PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
end;
cond=0;
if SDR(:,1)'==SDRstack(1,:)
  Rstate=1;
else cond=1; Rstate=2;
end;
if cond==1
  SDRstack(Rstate,:)=SDR(:,1)';
  PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1); PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
  PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate)); invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
  ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
  GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
end;
getini=0;   %if getini=1, TLM models   associated Backward Euler rule are used to start the simulation
%Simulation Procedure
for n=2:NumofPoints1              %Loop control according to the L-subnetwork time step.
  % Do the L-subnetwork calculation
  XL(:,[1,2])=XL(:,[2,3]); IL(:,[1,2])=IL(:,[2,3]); VLinc(:,1)=VLinc(:,2);
  VLMinc(:,1)=VLMinc(:,2); VLinc(11:12,2)=Vlink-VLinc(11:12,1);
  ZL=histsrc(0,n); XL(:,3)=[EL(:,:,Lstate);GL(:,:,Lstate)]*ZL; IL(:,3)=current(1); signL=switchstate(n);
  if SDL(:,1)==SDL(:,3)
    TswL=[TLstep TLstep TLstep TLstep];
  else TswL=switchtime(1);
  end;
  TswLstd=[TLstep TLstep TLstep TLstep]; TswLtemp=TswL;
  % Sort the switching time
  if abs(TswL-TswLstd)<=deltaT      %state unchanged or changed at the calculation point
    IL11=linspace(IL(11,2),IL(11,3),NumofPoints2+1);     %linear interpolation
    IL12=linspace(IL(12,2),IL(12,3),NumofPoints2+1);
    if signL==1
      cond=1; SDL(:,[1,2])=SDL(:,[3,4]);
      for iloop=1:length(SDLstack(:,1))
        if SDL(:,1)'==SDLstack(iloop,:)
          Lstate=iloop; cond=0; break;
        end;
      end;
      if cond==1
        Lstate=iloop+1; SDLstack(Lstate,:)=SDL(:,1)'; PL3(:,:,Lstate)=diag(SDL(:,2))*PL3(:,:,1);
        PL2(:,:,Lstate)=diag(SDL(:,2))*PL2(:,:,1);   PL4(:,:,Lstate)=inv(IPL3+PL3(:,:,Lstate));
        EL(:,:,Lstate)=invYL-PL1*PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
        GL(:,:,Lstate)=PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
      end;
    end;
    for k=2:NumofPoints2+1
      if getini<0.5
        XR(:,[1,2])=XR(:,[2,3]); IR(:,[1,2])=IR(:,[2,3]); VRinc(:,1)=VRinc(:,2); VRMinc(:,1)=VRMinc(:,2);
        VRinc(4:5,2)=Vlink-VRinc(4:5,1); ZR=histsrc(0,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
        IR(:,3)=current(2); signR=switchstate(n-1,k);
        if SDR(:,1)==SDR(:,3)
          TswR=[TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep];
        else   TswR=switchtime(2);
        end;
```

```
TswRstd=[TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep]; TswRtemp=TswR;
if abs(TswR-TswRstd)<deltaT        %state unchanged or changed at the calculation point
  Vlink=[(IR(4,3)+IR(4,2)+IL11(k)+IL11(k-1))*ZRC(1); (IR(5,3)+IR(5,2)+IL12(k)+IL12(k-1))*ZRC(2)]+Vlink;
  VRinc(4:5,2)=(Vlink-[IR(4,3)*ZRC(1);IR(5,3)*ZRC(2)])/2;
  if signR==1
    SDR(1:6,[1,2])=SDR(1:6,[3,4]);
    if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
        | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
      getini=initial(n,k);
    end;
    SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(:,1)=SQRcom(:,2); cond=1;
    for iloop=1:length(SDRstack(:,1))
      if SDR(:,1)'==SDRstack(iloop,:)
        Rstate=iloop; cond=0; break;
      end;
    end;
    if cond==1
      Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
      PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);  PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
      PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));  invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
      ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
      GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
    end;
  end;
  SQRcom(:,1)=SQRcom(:,2);
else          %Switching action happens in   R-sub.   for (if abs(TswR-TswRstd)<deltaT)
  TswRtemp=[0,TswR]; TswRseq=[1,2,3,4,5,6,7,8];
  for p=2:9
    TswRcomp=TswRtemp(p);
    for q=p+1:9
      if TswRcomp > TswRtemp(q)
        TswRcomp=TswRtemp(q); TswRtemp(q)=TswRtemp(p); TswRtemp(p)=TswRcomp;
        seq=TswRseq(q-1); TswRseq(q-1)=TswRseq(p-1); TswRseq(p-1)=seq;
      end;
    end;
    if abs(TswRcomp-TRstep)<deltaT
      break;
    end;
  end;    %for p=2:8
  if abs(TswRtemp(9)-TRstep)>deltaT
    TswRtemp(10)=TRstep;
  end;
  IRtemp=IR(:,2); XRtemp=XR(:,2); m=0; TRini=0;
  while (~isequal(SDR(:,1),SDR(:,3))) | abs(TRini-TRstep)>deltaT)
    m=m+1; swRnum=0;
    %find the number of times that the switching happens at the same time
    for na=m+1:length(TswRtemp)
      if (abs(TswRtemp(na)-TswRtemp(m+1)) < deltaT)
        swRnum=swRnum+1;
      else break;
      end;
    end;
    %Obtain the state values at switching time t-
```

```
TswRstep=TswRtemp(m+1)-TswRtemp(m);
%using interpolation to get the state value at switching time t-
XR(:,3)=XR(:,2)+(XR(:,3)-XR(:,2))/TRstep*TswRstep;
IR(:,3)=IR(:,2)+(IR(:,3)-IR(:,2))/TRstep*TswRstep;
VRinc(:,2)=VRinc(:,1)+(VRinc(:,2)-VRinc(:,1))/TRstep*TswRstep;
VRMinc(:,2)=VRMinc(:,1)+(VRMinc(:,2)-VRMinc(:,1))/TRstep*TswRstep;
Vlink=[(IR(4,3)+IR(4,2)+(IL11(k-1)+(IL11(k)-IL11(k-1))...
      *(TswRtemp(m+1)+TswRtemp(m))/TRstep))*TswRstep/Cvector2(1)/2;
   (IR(5,3)+IR(5,2)+(IL12(k-1)+(IL12(k)-IL12(k-1))...
      *(TswRtemp(m+1)+TswRtemp(m))/TRstep))*TswRstep/Cvector2(2)/2]+Vlink;
VRinc(4:5,2)=(Vlink-[IR(4,3)*ZRC(1);IR(5,3)*ZRC(2)])/2;
TRini=TswRtemp(m+1);
%renew switch states
for na=m:m+swRnum-1
   %Obtain the state values at switching time t+
   if (TswRseq(na)==7 | TswRseq(na)==8) & abs(TRini-TRstep)<deltaT
      if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
         | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
         getini=initial(n,k);
      end;
      SQRcom(:,1)=SQRcom(:,2);
   end;
   SDR(TswRseq(na),[1,2])=SDR(TswRseq(na),[3,4]);
end;
if abs(TRini-TRstep) > deltaT
   m=m+swRnum-1; XR(:,2)=XR(:,3); IR(:,2)=IR(:,3); VRinc(:,1)=VRinc(:,2);
   VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink-VRinc(4:5,1); cond=1;
   for iloop=1:length(SDRstack(:,1))
      if SDR(:,1)'==SDRstack(iloop,:)
         Rstate=iloop; cond=0; break;
      end;
   end;
   if cond==1
      Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
      PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);     PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
      PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));     invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
      ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
      GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
   end;
   ZR=histsrc(m+1,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
   IR(:,3)=current(2); signR=switchstate(n-1,k);
   if SDR(:,1)==SDR(:,3)
      TswRtemp(m+2:m+9)=TRstep; TswRseq(m+1:m+8)=[1:8];
   else
      TswRini=switchtime(2);   TswRiniseq=[1,2,3,4,5,6,7,8];
      for p=1:8
         TswRcomp=TswRini(p); seq=TswRiniseq(p);
         for q=p+1:8
            if TswRcomp>TswRini(q)
               TswRcomp=TswRini(q); TswRini(q)=TswRini(p); TswRini(p)=TswRcomp;
               seq=TswRiniseq(q);TswRiniseq(q)=TswRiniseq(p);TswRiniseq(p)=seq;
            end;
         end;
```

```
              if TswRcomp<=TRstep-TswRtemp(m+1)
                 TswRtemp(m+p+1)=TswRcomp+TswRtemp(m+1);TswRseq(m+p)=TswRiniseq(p);
              else
                 TswRtemp(m+p+1)=TRstep; TswRseq(m+p)=TswRiniseq(p);
                 SDR(TswRiniseq(p),[3,4])=SDR(TswRiniseq(p),[1,2]);
              end;
            end;          %for (for p=1:8)
          end;          %for if isequal(SDR(:,1),SDR(:,3))
        end;
      end;          %for while (SDR(:,1)~=SDR(:,3) | abs(TRini-TRstep)<deltaT)
      IR(:,2)=IRtemp; XR(:,2)=XRtemp; cond=1;
      for iloop=1:length(SDRstack(:,1))
        if SDR(:,1)'==SDRstack(iloop,:)
          Rstate=iloop; cond=0; break;
        end;
      end;
      if cond==1
        Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
        PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);   PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
        PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));     invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
        ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
        GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
      end;
    end;
  else
    VRMinc(1:2,2)=2*VRMinc(1:2,2);   VRinc(1:3,2)=2*VRinc(1:3,2); VRinc(4:9,2)=VRinc(4:9,2);
    VRinc(10:12,2)=2*VRinc(10:12,2); VRinc(13,2)=VRinc(13,2); VRinc(14:15,2)=2*VRinc(14:15,2);
    VRinc(16:19,2)=VRinc(16:19,2);
    for k1=1:2
      XR(:,[1,2])=XR(:,[2,3]); IR(:,[1,2])=IR(:,[2,3]); VRinc(:,1)=VRinc(:,2);
      VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink/2;
      ZR=histsrc(-1,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
      IR(:,3)=current(3); signR=switchstate(n-1,k);
      if SDR(:,1)==SDR(:,3)
        TswR=[TRE,TRE,TRE,TRE,TRE,TRE,TRE,TRE];
      else   TswR=switchtime(3);
      end;
      TswRstd=[TRE,TRE,TRE,TRE,TRE,TRE,TRE,TRE]; TswRtemp=TswR;
      if abs(TswR-TswRstd)<deltaT        %state unchanged or changed at the calculation point
        Vlink=[(IR(4,3)+IL11(k))*ZRC(1); (IR(5,3)+IL12(k))*ZRC(2)]+Vlink;
        VRinc(4:5,2)=Vlink/2;
        if signR==1
          SDR(1:6,[1,2])=SDR(1:6,[3,4]);
          if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
               | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
            getini=initial(n,k);
          end;
          SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(:,1)=SQRcom(:,2); cond=1;
          for iloop=1:length(SDRstack(:,1))
            if SDR(:,1)'==SDRstack(iloop,:)
              Rstate=iloop; cond=0; break;
            end;
          end;
          if cond==1
            Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
            PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1); PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
            PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));   invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
            ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
          end;
        end;
        SQRcom(:,1)=SQRcom(:,2);
      else          %Switching action happens in   R-sub.   for (if abs(TswR-TswRstd)<deltaT)
        TswRtemp=[0,TswR]; TswRseq=[1,2,3,4,5,6,7,8];
        for p=2:9
          TswRcomp=TswRtemp(p);
          for q=p+1:9
            if TswRcomp > TswRtemp(q)
              TswRcomp=TswRtemp(q); TswRtemp(q)=TswRtemp(p); TswRtemp(p)=TswRcomp;
              seq=TswRseq(q-1); TswRseq(q-1)=TswRseq(p-1); TswRseq(p-1)=seq;
            end;
          end;
          if abs(TswRcomp-TRE)<deltaT
            break;
          end;
        end;      %for p=2:8
        if abs(TswRtemp(9)-TRE)>deltaT
          TswRtemp(10)=TRE;
        end;
        IRtemp=IR(:,2); XRtemp=XR(:,2); m=0; TRini=0;
        while (~isequal(SDR(:,1),SDR(:,3)) | abs(TRini-TRE)>deltaT)
          m=m+1; swRnum=0;
          %find the number of times that the switching happens at the same time
          for na=m+1:length(TswRtemp)
            if (abs(TswRtemp(na)-TswRtemp(m+1)) < deltaT)
              swRnum=swRnum+1;
            else break;
            end;
          end;
          %Obtain the state values at switching time t-
          TswRstep=TswRtemp(m+1)-TswRtemp(m);
          %using interpolation to get the state value at switching time t-
          XR(:,3)=XR(:,2)+(XR(:,3)-XR(:,2))/TRE*TswRstep;
          IR(:,3)=IR(:,2)+(IR(:,3)-IR(:,2))/TRE*TswRstep;
          VRinc(:,2)=VRinc(:,1)+(VRinc(:,2)-VRinc(:,1))/TRE*TswRstep;
          VRMinc(:,2)=VRMinc(:,1)+(VRMinc(:,2)-VRMinc(:,1))/TRE*TswRstep;
          Vlink=[(IR(4,3)+(IL11(k-1)+(IL11(k)-IL11(k-1))...
                *TswRtemp(m+1)/TRE))*TswRstep/Cvector2(1);
            (IR(5,3)+(IL12(k-1)+(IL12(k)-IL12(k-1))...
                *TswRtemp(m+1)/TRE))*TswRstep/Cvector2(2)]+Vlink;
          VRinc(4:5,2)=Vlink/2;
          TRini=TswRtemp(m+1);
          %renew switch states
          for na=m:m+swRnum-1
            %Obtain the state values at switching time t+
            if (TswRseq(na)==7 | TswRseq(na)==8) & abs(TRini-TRE)<deltaT
              if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
                  | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
                getini=initial(n,k);
              end;
              SQRcom(:,1)=SQRcom(:,2);
            end;
            SDR(TswRseq(na),[1,2])=SDR(TswRseq(na),[3,4]);
          end;
          if abs(TRini-TRE) > deltaT
            m=m+swRnum-1; XR(:,2)=XR(:,3); IR(:,2)=IR(:,3); VRinc(:,1)=VRinc(:,2);
            VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink/2; cond=1;
            for iloop=1:length(SDRstack(:,1))
              if SDR(:,1)'==SDRstack(iloop,:)
                Rstate=iloop; cond=0; break;
              end;
            end;
            if cond==1
              Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
              PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);   PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
              PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));   invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
              ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
              GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            end;
            ZR=histsrc(-1,n-1,k);   XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
            IR(:,3)=current(3); signR=switchstate(n-1,k);
            if SDR(:,1)==SDR(:,3)
              TswRtemp(m+2:m+9)=TRE; TswRseq(m+1:m+8)=[1:8];
            else
              TswRini=switchtime(3);     TswRiniseq=[1,2,3,4,5,6,7,8];
              for p=1:8
                TswRcomp=TswRini(p); seq=TswRiniseq(p);
                for q=p+1:8
                  if TswRcomp>TswRini(q)
                    TswRcomp=TswRini(q); TswRini(q)=TswRini(p); TswRini(p)=TswRcomp;
                    seq=TswRiniseq(q);TswRiniseq(q)=TswRiniseq(p);TswRiniseq(p)=seq;
                  end;
                end;
                if TswRcomp<=TRE-TswRtemp(m+1)
                  TswRtemp(m+p+1)=TswRcomp+TswRtemp(m+1);TswRseq(m+p)=TswRiniseq(p);
                else
                  TswRtemp(m+p+1)=TRE; TswRseq(m+p)=TswRiniseq(p);
                  SDR(TswRiniseq(p),[3,4])=SDR(TswRiniseq(p),[1,2]);
                end;
              end;          %for (for p=1:8)
            end;          %for if isequal(SDR(:,1),SDR(:,3))
          end;
        end;          %for while (SDR(:,1)~=SDR(:,3) | abs(TRini-TRE)<deltaT)
        IR(:,2)=IRtemp; XR(:,2)=XRtemp; cond=1;
        for iloop=1:length(SDRstack(:,1))
          if SDR(:,1)'==SDRstack(iloop,:)
            Rstate=iloop; cond=0; break;
          end;
        end;
```

```
            if cond==1
              Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
              PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);     PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
              PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));    invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
              ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
              GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            end;
          end;
        end;
        VRMinc(1:2,2)=VRMinc(1:2,2)/2; VRinc(1:3,2)=VRinc(1:3,2)/2; VRinc(10:12,2)=VRinc(10:12,2)/2;
        VRinc(14:15,2)=VRinc(14:15,2)/2;
        getini=0;    %if getini=1, then only TLM models associated with Backward Euler rule are used
      end;
      if n>NumofStart2Rec & rem(k-1,Step4RecR)==0
        fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',XR(1:9,3)');
        fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f\n', XR(10:13,3)');
        fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(1:9,3)');
        fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(10:19,3)');
        fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', IR(20:27,3)');
      end;
    end;
  else
    TswLseq=[1,2,3,4]; TswLtemp=[0,TswL];
    %sorting the switching time
    for j=2:5
      TswLcomp=TswLtemp(j);
      for k=j+1:5
        if TswLcomp > TswLtemp(k)
          TswLcomp=TswLtemp(k); TswLtemp(k)=TswLtemp(j); TswLtemp(j)=TswLcomp;
          seq=TswLseq(k-1); TswLseq(k-1)=TswLseq(j-1); TswLseq(j-1)=seq;
        end;
      end;
      if abs(TswLcomp-TLstep)<deltaT
        break;
      end;
    end;
    j=0; TLini=0; ILtemp=IL(:,2); XLtemp=XL(:,2);
    while (~isequal(SDL(:,1),SDL(:,3)) | abs(TLini-TLstep)>deltaT)
      j=j+1; swLnum=0;
      %Finding how mnay times the switching actions happen at the same time
      for na=j+1:length(TswLtemp)
        if (abs(TswLtemp(na)-TswLtemp(j+1)) < deltaT)
          swLnum=swLnum+1;
        else break;
        end;
      end;
      TswLstep=TswLtemp(j+1)-TswLtemp(j);
      XL(:,3)=XL(:,2)+(XL(:,3)-XL(:,2))/TLstep*TswLstep;
      IL(:,3)=IL(:,2)+(IL(:,3)-IL(:,2))/TLstep*TswLstep;
      VLinc(:,2)=VLinc(:,1)+(VLinc(:,2)-VLinc(:,1))/TLstep*TswLstep;
      VLMinc(:,2)=VLMinc(:,1)+(VLMinc(:,2)-VLMinc(:,1))/TLstep*TswLstep;
      if abs(TswLtemp(j+1)-TswLtemp(j))>TRstep/2
        IL11(round(TswLtemp(j)/TRstep)+1:round(TswLtemp(j+1)/TRstep)+1)...
```

```
              =linspace(IL(11,2),IL(11,3),round(TswLstep/TRstep+1));
            IL12(round(TswLtemp(j)/TRstep)+1:round(TswLtemp(j+1)/TRstep)+1)...
              =linspace(IL(12,2),IL(12,3),round(TswLstep/TRstep+1));
            %R-subnetwork calculation
            for k=round(TswLtemp(j)/TRstep)+2:round(TswLtemp(j+1)/TRstep)+1
              if getini<0.5
XR(:,[1,2])=XR(:,[2,3]); IR(:,[1,2])=IR(:,[2,3]); VRinc(:,1)=VRinc(:,2);    VRMinc(:,1)=VRMinc(:,2);
                VRinc(4:5,2)=Vlink-VRinc(4:5,1);   ZR=histsrc(0,n-1,k);       XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
                IR(:,3)=current(2); signR=switchstate(n-1,k);
                if SDR(:,1)==SDR(:,3)
                  TswR=[TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep];
                else   TswR=switchtime(2);
                end;
                TswRstd=[TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep,TRstep]; TswRtemp=TswR;
                if abs(TswR-TswRstd)<deltaT        %state unchanged or changed at the calculation point
                  Vlink=[(IR(4,3)+IR(4,2)+IL11(k)+IL11(k-1))*ZRC(1);
                    (IR(5,3)+IR(5,2)+IL12(k)+IL12(k-1))*ZRC(2)]+Vlink;
                  VRinc(4:5,2)=(Vlink-[IR(4,3)*ZRC(1);IR(5,3)*ZRC(2)])/2;
                  if signR==1
                    if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
                        | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
                      getini=initial(n,k);
                    end;
                    SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(:,1)=SQRcom(:,2);
                    cond=1;
                    for iloop=1:length(SDRstack(:,1))
                      if SDR(:,1)'==SDRstack(iloop,:)
                        Rstate=iloop; cond=0; break;
                      end;
                    end;
                    if cond==1
                      Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
                      PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1); PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
                      PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate)); invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
                      ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
                      GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
                    end;
                  end;
                  SQRcom(:,1)=SQRcom(:,2);
                else          %Switching action happens in   R-sub.   for (if abs(TswR-TswRstd)<deltaT)
                  TswRtemp=[0,TswR]; TswRseq=[1,2,3,4,5,6,7,8];
                  for p=2:9
                    TswRcomp=TswRtemp(p);
                    for q=p+1:9
                      if TswRcomp > TswRtemp(q)
                        TswRcomp=TswRtemp(q); TswRtemp(q)=TswRtemp(p); TswRtemp(p)=TswRcomp;
                        seq=TswRseq(q-1); TswRseq(q-1)=TswRseq(p-1); TswRseq(p-1)=seq;
                      end;
                    end;
                    if abs(TswRcomp-TRstep)<deltaT
                      break;
                    end;
                  end;     %for p=2:8
```

```
if abs(TswRtemp(9)-TRstep)>deltaT
  TswRtemp(10)=TRstep;
end;
IRtemp=IR(:,2); XRtemp=XR(:,2); m=0; TRini=0;
while (~isequal(SDR(:,1),SDR(:,3)) | abs(TRini-TRstep)>deltaT)
  m=m+1; swRnum=0;
  %find the number of times that the switching happens at the same time
  for na=m+1:length(TswRtemp)
    if (abs(TswRtemp(na)-TswRtemp(m+1)) < deltaT)
      swRnum=swRnum+1;
    else break;
    end;
  end;
  %Obtain the state values at switching time t-
  TswRstep=TswRtemp(m+1)-TswRtemp(m);
  %using interpolation to get the state value at switching time t-
  XR(:,3)=XR(:,2)+(XR(:,3)-XR(:,2))/TRstep*TswRstep;
  IR(:,3)=IR(:,2)+(IR(:,3)-IR(:,2))/TRstep*TswRstep;
  VRinc(:,2)=VRinc(:,1)+(VRinc(:,2)-VRinc(:,1))/TRstep*TswRstep;
  VRMinc(:,2)=VRMinc(:,1)+(VRMinc(:,2)-VRMinc(:,1))/TRstep*TswRstep;
  Vlink=[(IR(4,3)+IR(4,2)+(IL11(k-1)+(IL11(k)-IL11(k-1))...
      *(TswRtemp(m+1)+TswRtemp(m))/TRstep))*TswRstep/Cvector2(1)/2;
    (IR(5,3)+IR(5,2)+(IL12(k-1)+(IL12(k)-IL12(k-1))...
      *(TswRtemp(m+1)+TswRtemp(m))/TRstep))*TswRstep/Cvector2(2)/2]+Vlink;
  VRinc(4:5,2)=(Vlink-[IR(4,3)*ZRC(1);IR(5,3)*ZRC(2)])/2;
  TRini=TswRtemp(m+1);
  %renew switch states
  for na=m:m+swRnum-1
    %Obtain the state values at switching time t+
    if (TswRseq(na)==7 | TswRseq(na)==8) & abs(TRini-TRstep)<deltaT
      if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
          | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
        getini=initial(n,k);
      end;
      SQRcom(:,1)=SQRcom(:,2);
    end;
    SDR(TswRseq(na),[1,2])=SDR(TswRseq(na),[3,4]);
  end;
  if abs(TswRtemp(m+1)-TRstep) > deltaT
    m=m+swRnum-1; XR(:,2)=XR(:,3); IR(:,2)=IR(:,3); VRinc(:,1)=VRinc(:,2);
    VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink-VRinc(4:5,1); cond=1;
    for iloop=1:length(SDRstack(:,1))
      if SDR(:,1)'==SDRstack(iloop,:)
        Rstate=iloop; cond=0; break;
      end;
    end;
    if cond==1
      Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
      PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1); PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
      PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate)); invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
      ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
      GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
    end;
```

```
            ZR=histsrc(m+1,n-1,k); XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
            IR(:,3)=current(2); signR=switchstate(n-1,k);
            if SDR(:,1)==SDR(:,3)
               TswRtemp(m+2:m+9)=TRstep; TswRseq(m+1:m+8)=[1:8];
            else
               TswRini=switchtime(2); TswRiniseq=[1,2,3,4,5,6,7,8];
               for p=1:8
                  TswRcomp=TswRini(p); seq=TswRiniseq(p);
                  for q=p+1:8
                     if TswRcomp>TswRini(q)
                        TswRcomp=TswRini(q); TswRini(q)=TswRini(p); TswRini(p)=TswRcomp;
                        seq=TswRiniseq(q);TswRiniseq(q)=TswRiniseq(p);TswRiniseq(p)=seq;
                     end;
                  end;
                  if TswRcomp<=TRstep-TswRtemp(m+1)
                     TswRtemp(m+p+1)=TswRcomp+TswRtemp(m+1);TswRseq(m+p)=TswRiniseq(p);
                  else
                     TswRtemp(m+p+1)=TRstep; TswRseq(m+p)=TswRiniseq(p);
                     SDR(TswRiniseq(p),[3,4])=SDR(TswRiniseq(p),[1,2]);
                  end;
               end;          %for (for p=1:8)
            end;          %for if isequal(SDR(:,1),SDR(:,3))
         end;
      end;          %for while (SDR(:,1)~=SDR(:,3) | abs(TRini-TRstep)<deltaT)
      IR(:,2)=IRtemp; XR(:,2)=XRtemp; cond=1;
      for iloop=1:length(SDRstack(:,1))
         if SDR(:,1)'==SDRstack(iloop,:)
            Rstate=iloop; cond=0; break;
         end;
      end;
      if cond==1
         Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
         PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);  PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
         PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate)); invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
         ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
         GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
      end;
   end;
else
   VRMinc(1:2,2)=2*VRMinc(1:2,2);  VRinc(1:3,2)=2*VRinc(1:3,2); VRinc(4:9,2)=VRinc(4:9,2);
   VRinc(10:12,2)=2*VRinc(10:12,2); VRinc(13,2)=VRinc(13,2); VRinc(14:15,2)=2*VRinc(14:15,2);
   VRinc(16:19,2)=VRinc(16:19,2);
   for k1=1:2
      XR(:,[1,2])=XR(:,[2,3]); IR(:,[1,2])=IR(:,[2,3]); VRinc(:,1)=VRinc(:,2);
      VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink/2;
      ZR=histsrc(-1,n-1,k);   XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
      IR(:,3)=current(3); signR=switchstate(n-1,k);
      if SDR(:,1)==SDR(:,3)
         TswR=[TRE,TRE,TRE,TRE,TRE,TRE,TRE,TRE];
      else   TswR=switchtime(3);
      end;
      TswRstd=[TRE,TRE,TRE,TRE,TRE,TRE,TRE,TRE]; TswRtemp=TswR;
      if abs(TswR-TswRstd)<deltaT        %state unchanged or changed at the calculation point
```

```
  Vlink=[(IR(4,3)+IL11(k))*ZRC(1); (IR(5,3)+IL12(k))*ZRC(2)]+Vlink;
  VRinc(4:5,2)=Vlink/2;
  if signR==1
    SDR(1:6,[1,2])=SDR(1:6,[3,4]);
    if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
        | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
      getini=initial(n,k);
    end;
    SDR(:,[1,2])=SDR(:,[3,4]); SQRcom(:,1)=SQRcom(:,2); cond=1;
    for iloop=1:length(SDRstack(:,1))
      if SDR(:,1)'==SDRstack(iloop,:)
        Rstate=iloop; cond=0; break;
      end;
    end;
    if cond==1
      Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
      PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);  PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
      PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));  invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
      ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
      GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
    end;
  end;
  SQRcom(:,1)=SQRcom(:,2);
else           %Switching action happens in   R-sub.   for (if abs(TswR-TswRstd)<deltaT)
  TswRtemp=[0,TswR]; TswRseq=[1,2,3,4,5,6,7,8];
  for p=2:9
    TswRcomp=TswRtemp(p);
    for q=p+1:9
      if TswRcomp > TswRtemp(q)
        TswRcomp=TswRtemp(q); TswRtemp(q)=TswRtemp(p); TswRtemp(p)=TswRcomp;
        seq=TswRseq(q-1); TswRseq(q-1)=TswRseq(p-1); TswRseq(p-1)=seq;
      end;
    end;
    if abs(TswRcomp-TRE)<deltaT
      break;
    end;
  end;     %for p=2:8
  if abs(TswRtemp(9)-TRE)>deltaT
    TswRtemp(10)=TRE;
  end;
  IRtemp=IR(:,2); XRtemp=XR(:,2); m=0; TRini=0;
  while (~isequal(SDR(:,1),SDR(:,3)) | abs(TRini-TRE)>deltaT)
    m=m+1; swRnum=0;
    %find the number of times that the switching happens at the same time
    for na=m+1:length(TswRtemp)
      if (abs(TswRtemp(na)-TswRtemp(m+1)) < deltaT)
        swRnum=swRnum+1;
      else break;
      end;
    end;
    %Obtain the state values at switching time t-
    TswRstep=TswRtemp(m+1)-TswRtemp(m);
    %using interpolation to get the state value at switching time t-
```

```
XR(:,3)=XR(:,2)+(XR(:,3)-XR(:,2))/TRE*TswRstep;
IR(:,3)=IR(:,2)+(IR(:,3)-IR(:,2))/TRE*TswRstep;
VRinc(:,2)=VRinc(:,1)+(VRinc(:,2)-VRinc(:,1))/TRE*TswRstep;
VRMinc(:,2)=VRMinc(:,1)+(VRMinc(:,2)-VRMinc(:,1))/TRE*TswRstep;
Vlink=[(IR(4,3)+(IL11(k-1)+(IL11(k)-IL11(k-1))...
    *TswRtemp(m+1)/TRE))*TswRstep/Cvector2(1);
  (IR(5,3)+(IL12(k-1)+(IL12(k)-IL12(k-1))...
    *TswRtemp(m+1)/TRE))*TswRstep/Cvector2(2)]+Vlink;
VRinc(4:5,2)=Vlink/2;
TRini=TswRtemp(m+1);
%renew switch states
for na=m:m+swRnum-1
  %Obtain the state values at switching time t+
  if (TswRseq(na)==7 | TswRseq(na)==8) & abs(TRini-TRE)<deltaT
    if (XR(1,3)>VonR(3)+deltaV & SQRcom(1,2)~=SQRcom(1,1))...
        | (-XR(3,3)>VonR(3)+deltaV & SQRcom(2,2)~=SQRcom(2,1))
      getini=initial(n,k);
    end;
    SQRcom(:,1)=SQRcom(:,2);
  end;
  SDR(TswRseq(na),[1,2])=SDR(TswRseq(na),[3,4]);
end;
if abs(TRini-TRE) > deltaT
  m=m+swRnum-1; XR(:,2)=XR(:,3); IR(:,2)=IR(:,3); VRinc(:,1)=VRinc(:,2);
  VRMinc(:,1)=VRMinc(:,2); VRinc(4:5,2)=Vlink/2; cond=1;
  for iloop=1:length(SDRstack(:,1))
    if SDR(:,1)'==SDRstack(iloop,:)
      Rstate=iloop; cond=0; break;
    end;
  end;
  if cond==1
    Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
    PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);  PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
    PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));  invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
    ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
    GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
  end;
  ZR=histsrc(-1,n-1,k);  XR(:,3)=[ER(:,:,Rstate);GR(:,:,Rstate)]*ZR;
  IR(:,3)=current(3); signR=switchstate(n-1,k);
  if SDR(:,1)==SDR(:,3)
    TswRtemp(m+2:m+9)=TRE; TswRseq(m+1:m+8)=[1:8];
  else
    TswRini=switchtime(3);  TswRiniseq=[1,2,3,4,5,6,7,8];
    for p=1:8
      TswRcomp=TswRini(p); seq=TswRiniseq(p);
      for q=p+1:8
        if TswRcomp>TswRini(q)
          TswRcomp=TswRini(q); TswRini(q)=TswRini(p); TswRini(p)=TswRcomp;
          seq=TswRiniseq(q);TswRiniseq(q)=TswRiniseq(p);TswRiniseq(p)=seq;
        end;
      end;
      if TswRcomp<=TRE-TswRtemp(m+1)
        TswRtemp(m+p+1)=TswRcomp+TswRtemp(m+1);TswRseq(m+p)=TswRiniseq(p);
```

```
                    else
                      TswRtemp(m+p+1)=TRE; TswRseq(m+p)=TswRiniseq(p);
                      SDR(TswRiniseq(p),[3,4])=SDR(TswRiniseq(p),[1,2]);
                    end;
                  end;          %for (for p=1:8)
                end;          %for if isequal(SDR(:,1),SDR(:,3))
              end;
            end;          %for while (SDR(:,1)~=SDR(:,3) | abs(TRini-TRE)<deltaT)
            IR(:,2)=IRtemp; XR(:,2)=XRtemp; cond=1;
            for iloop=1:length(SDRstack(:,1))
              if SDR(:,1)'==SDRstack(iloop,:)
                Rstate=iloop; cond=0; break;
              end;
            end;
            if cond==1
              Rstate=iloop+1; SDRstack(Rstate,:)=SDR(:,1)';
              PR3(:,:,Rstate)=diag(SDR(:,2))*PR3(:,:,1);   PR2(:,:,Rstate)=diag(SDR(:,2))*PR2(:,:,1);
              PR4(:,:,Rstate)=inv(IPR3+PR3(:,:,Rstate));   invDR(:,:,Rstate)=diag(SDR(:,2))*invDR(:,:,1);
              ER(:,:,Rstate)=invYR-PR1*PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
              GR(:,:,Rstate)=PR4(:,:,Rstate)*PR2(:,:,Rstate)*invYR;
            end;
          end;
        end;
        VRMinc(1:2,2)=VRMinc(1:2,2)/2;   VRinc(1:3,2)=VRinc(1:3,2)/2;
        VRinc(10:12,2)=VRinc(10:12,2)/2; VRinc(14:15,2)=VRinc(14:15,2)/2;
        getini=0;   %if getini=1, then only TLM models associated with Backward Euler rule are used
      end;
      if n>NumofStart2Rec & rem(k-1,Step4RecR)==0
        fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',XR(1:9,3)');
        fprintf(VRfid,'%4.6f %4.6f %4.6f %4.6f\n', XR(10:13,3)');
        fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(1:9,3)');
        fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IR(10:19,3)');
        fprintf(IRfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', IR(20:27,3)');
      end;
    end;
  end;
  IL11(round(TswLtemp(j+1)/TRstep)+1)=IL(11,3);
  IL12(round(TswLtemp(j+1)/TRstep)+1)=IL(12,3);
  for na=j:j+swLnum-1
    SDL(TswLseq(na),[1,2])=SDL(TswLseq(na),[3,4]);
    TswLtemp(na+1)=round(TswLtemp(na+1)/TRstep)*TRstep;
  end;
  VLinc(11:12,2)=(Vlink-[IL(11,3)*ZLC(5);IL(12,3)*ZLC(6)])/2; TLini=TswLtemp(j+1);
  if abs(TLini-TLstep)>deltaT
    XL(:,2)=XL(:,3); IL(:,2)=IL(:,3); VLinc(:,1)=VLinc(:,2); VLMinc(:,1)=VLMinc(:,2);
    VLinc(11:12,2)=Vlink-VLinc(11:12,1); cond=1;
    for iloop=1:length(SDLstack(:,1))
      if SDL(:,1)'==SDLstack(iloop,:)
        Lstate=iloop; cond=0; break;
      end;
    end;
    if cond==1
      Lstate=iloop+1; SDLstack(Lstate,:)=SDL(:,1)';
```

```
                PL3(:,:,Lstate)=diag(SDL(:,2))*PL3(:,:,1); PL2(:,:,Lstate)=diag(SDL(:,2))*PL2(:,:,1);
                PL4(:,:,Lstate)=inv(IPL3+PL3(:,:,Lstate)); invDL(:,:,Lstate)=diag(SDL(:,2))*invDL(:,:,1);
                EL(:,:,Lstate)=invYL-PL1*PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
                GL(:,:,Lstate)=PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
            end;
            ZL=histsrc(j,n); XL(:,3)=[EL(:,:,Lstate);GL(:,:,Lstate)]*ZL; IL(:,3)=current(1); signL=switchstate(n);
            j=j+swLnum-1;
            if SDL(:,1)==SDL(:,3)
                TswLtemp(j+2:j+5)=TLstep; TswLseq(j+1:j+4)=[1,2,3,4];
            else
                TswLini=switchtime(1);
                TswLiniseq=[1,2,3,4];
                for p=1:4
                    TswLcomp=TswLini(p); seq=TswLiniseq(p);
                    for q=p+1:4
                        if TswLcomp>TswLini(q)
                            TswLcomp=TswLini(q); TswLini(q)=TswLini(p); TswLini(p)=TswLcomp;
                            seq=TswLiniseq(q); TswLiniseq(q)=TswLiniseq(p); TswLiniseq(p)=seq;
                        end;
                    end;
                    if TswLcomp<=TLstep-TswLtemp(j+1)
                        TswLtemp(j+p+1)=TswLcomp+TswLtemp(j+1);TswLseq(j+p)=seq;
                    else
                        TswLtemp(j+p+1)=TLstep; TswLseq(j+p)=seq; SDL(seq,[3,4])=SDL(seq,[1,2]);
                    end;
                end;
            end;
        end;
    end;
    IL(:,2)=ILtemp; XL(:,2)=XLtemp; cond=1; SDL(:,[1,2])=SDL(:,[3,4]);
    for iloop=1:length(SDLstack(:,1))
        if SDL(:,1)'==SDLstack(iloop,:)
            Lstate=iloop; cond=0; break;
        end;
    end;
    if cond==1
        Lstate=iloop+1; SDLstack(Lstate,:)=SDL(:,1)';
        PL3(:,:,Lstate)=diag(SDL(:,2))*PL3(:,:,1);PL2(:,:,Lstate)=diag(SDL(:,2))*PL2(:,:,1);
        PL4(:,:,Lstate)=inv(IPL3+PL3(:,:,Lstate));
        invDL(:,:,Lstate)=diag(SDL(:,2))*invDL(:,:,1);
        EL(:,:,Lstate)=invYL-PL1*PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
        GL(:,:,Lstate)=PL4(:,:,Lstate)*PL2(:,:,Lstate)*invYL;
    end;
  end;
  VLinc(11:12,2)=(Vlink-[IL(11,3)*ZLC(5);IL(12,3)*ZLC(6)])/2;
  if n>NumofStart2Rec & rem(n-1,Step4RecL)==0
      fprintf(VLfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n', XL(1:8,3)');
      fprintf(ILfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f %4.6f ',IL(1:10,3)');
      fprintf(ILfid,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\n',IL(11:16,3)' );
  end;
end;
T1=toc; fclose(VLfid); fclose(ILfid); fclose(VRfid); fclose(IRfid);
T1fid=fopen('T1data3.xls','w'); fprintf(T1fid,'%12.3f',T1); fclose(T1fid);
```

```
disp('TrapWithBE End');
```

III. Impedance

```
function D=impedance(subnetwork)
%subnetwork == 1 give out the impedance of L-subnetwork
%subnetwork == 2 give out the impedance of R-subnetwork
%subnetwork == 3 give out the impedance of R-subnetwork (TRmin)
global Rvector1 Cvector1 Lvector1 TL1 TL2 Rvector2 Cvector2 Lvector2 TR1 TR2 YLLM YRLM
global TLstep TRstep TRE

switch subnetwork,
  case 1
    ZLC=TLstep./(2*Cvector1);   ZLL=2*Lvector1/TLstep;
    ZLM1=2*TL1(1:3)/TLstep; ZTL1=[ZLM1(1)+TL1(4), ZLM1(3)+TL1(6); ZLM1(3)+TL1(6), ZLM1(2)+TL1(5)];
    YTL1(:,:,1)=inv(ZTL1); YTL1(:,:,2)=-[ZLM1(1),0;0,ZLM1(3)]*YTL1(:,:,1);
    YTL1(:,:,3)=-[ZLM1(3),0;0,ZLM1(2)]*YTL1(:,:,1);
    ZLM2=2*TL2(1:3)/TLstep; ZTL2=[ZLM2(1)+TL2(4), ZLM2(3)+TL2(6); ZLM2(3)+TL2(6), ZLM2(2)+TL2(5)];
    YTL2(:,:,1)=inv(ZTL2); YTL2(:,:,2)=-[ZLM2(1),0;0,ZLM2(3)]*YTL2(:,:,1);
    YTL2(:,:,3)=-[ZLM2(3),0;0,ZLM2(2)]*YTL2(:,:,1);
    BrL(1,1:6)=[1, -2, -ZLL(1), ZLL(1)-Rvector1(1), -1, ZLL(1)]/(Rvector1(1)+ZLL(1));
    BrL(2,1:6)=[1, -2, -ZLL(1), ZLL(1)-Rvector1(1), -1, ZLL(1)]/(Rvector1(1)+ZLL(1));
    BrL(3:4,1:6)=[YTL1(:,:,1), YTL1(:,:,2), YTL1(:,:,3)];
    BrL(5:6,1:6)=[YTL2(:,:,1), YTL2(:,:,2), YTL2(:,:,3)];
    BrL(7,1:4)=[(Rvector1(2)+ZLC(1))/(Rvector1(2)*ZLC(1)), -2/ZLC(1), 1, -1];
    BrL(8,1:4)=[1/ZLC(2), -2/ZLC(2), 1, -1];
    BrL(9,1:4)=[1/ZLC(3), -2/ZLC(3), 1, -1];
    BrL(10,1:4)=[1/ZLC(4), -2/ZLC(4), 1, -1];
    BrL(11,1:4)=[1/ZLC(5), -2/ZLC(5), 1, -1];
    BrL(12,1:4)=[1/ZLC(6), -2/ZLC(6), 1, -1];
    D={ZLC,ZLL,YTL1,YTL2,BrL};
  case 2
    ZRC=TRstep./(2*Cvector2);   ZRL=2*Lvector2/TRstep;
    T1=TR1; T1(1:3)=2*T1(1:3)/TRstep; T=T1; T(1:3)=T1(1:3)+T1(6:8);
    De(1)=[T(4)^2*T(5)^2*T(2)*T(3)+T(5)^2*T(3)*T(1)+T(4)^2*T(2)*T(1)];
    De(2)=[T1(4)^2*T1(5)^2*T1(2)*T1(3)+T1(5)^2*T1(3)*T1(1)+T1(4)^2*T1(2)*T1(1)];
    YRtranx(:,:,1)=[T(5)^2*T(3)+T(4)^2*T(2), -T(4)*T(5)^2*T(3),-T(5)*T(4)^2*T(2);
      -T(4)*T(5)^2*T(3),T(4)^2*T(5)^2*T(3)+T(4)^2*T(1), -T(4)*T(5)*T(1);
      -T(4)^2*T(5)*T(2), -T(4)*T(5)*T(1),T(4)^2*T(5)^2*T(2)+T(5)^2*T(1)]/De(1);
    YRtranx(:,:,2)=[T1(1),0,0;0,T1(2),0;0,0,T1(3)]*YRtranx(:,:,1);
    YRtranx(:,:,3)=[T(1),0,0;0,T(2),0;0,0,T(3)];
    MidYRtranx(1,:)=[T(4)^2*T(5)^2*T(2)*T(3),T(4)*T(5)^2*T(3)*T(1),T(5)*T(4)^2*T(2)*T(1)]/De(1);
    MidYRtranx(2,:)=[T1(4)^2*T1(5)^2*T1(2)*T1(3),T1(4)*T1(5)^2*T1(3)*T1(1),T1(5)*T1(4)^2*T1(2)*T1(1)]/De(2);
    ZRM2=2*TR2(1:3)/TRstep; ZTR2=[ZRM2(1)+TR2(4), ZRM2(3)+TR2(6); ZRM2(3)+TR2(6), ZRM2(2)+TR2(5)];
    YTR2(:,:,1)=inv(ZTR2); YTR2(:,:,2)=-[ZRM2(1),0;0,ZRM2(3)]*YTR2(:,:,1);
    YTR2(:,:,3)=-[ZRM2(3),0;0,ZRM2(2)]*YTR2(:,:,1);
    BrR(1,1:4)=[1/ZRL(1), -2/ZRL(1), -1, 1];
    BrR(2,1:4)=[1/ZRL(2), -2/ZRL(2), -1, 1];
    BrR(3,1:4)=[1/ZRL(3), -2/ZRL(3), -1, 1];
    BrR(4,1:4)=[1/ZRC(1), -2/ZRC(1), 1, -1];
    BrR(5,1:4)=[1/ZRC(2), -2/ZRC(2), 1, -1];
    BrR(6,1:4)=[1/ZRC(3), -2/ZRC(3), 1, -1];
    BrR(7,1:4)=[1, -2, ZRC(4), Rvector2(1)-ZRC(4)]/(Rvector2(1)+ZRC(4));
    BrR(8,1:4)=[1/ZRC(5), -2/ZRC(5), 1, -1];
    BrR(9,1:4)=[1/ZRC(6), -2/ZRC(6), 1, -1];
    BrR(10:12,1:3)=YRtranx(:,:,1);
    BrR(13,1:4)=[1/ZRC(7), -2/ZRC(7), 1, -1];
    BrR(14:15,1:6)=[YTR2(:,:,1), YTR2(:,:,2), YTR2(:,:,3)];
    BrR(16,1:4)=[1, -2, ZRC(8), Rvector2(2)-ZRC(8)]/(Rvector2(2)+ZRC(8));
    BrR(17,1:4)=[1, -2, ZRC(9), Rvector2(3)-ZRC(9)]/(Rvector2(3)+ZRC(9));
    BrR(18,1:4)=[1, -2, ZRC(10), Rvector2(4)-ZRC(10)]/(Rvector2(4)+ZRC(10));
    BrR(19,1:4)=[1, -2, ZRC(11), Rvector2(5)-ZRC(11)]/(Rvector2(5)+ZRC(11));
    D={ZRC,ZRL,YRtranx,BrR,YTR2,MidYRtranx};
  case 3
    ZRCE=TRstep./(4*Cvector2); ZRC=ZRCE*2;   ZRLE=4*Lvector2/TRstep; ZRL=ZRLE/2;
    T1=TR1; T1(1:3)=2*T1(1:3)/TRstep;   T=T1; T(1:3)=T1(1:3)+T1(6:8);
    De(1)=[T(4)^2*T(5)^2*T(2)*T(3)+T(5)^2*T(3)*T(1)+T(4)^2*T(2)*T(1)];
    De(2)=[T1(4)^2*T1(5)^2*T1(2)*T1(3)+T1(5)^2*T1(3)*T1(1)+T1(4)^2*T1(2)*T1(1)];
    YRtranxE(:,:,1)=[T(5)^2*T(3)+T(4)^2*T(2), -T(4)*T(5)^2*T(3),-T(5)*T(4)^2*T(2);
      -T(4)*T(5)^2*T(3),T(4)^2*T(5)^2*T(3)+T(4)^2*T(1), -T(4)*T(5)*T(1);
      -T(4)^2*T(5)*T(2), -T(4)*T(5)*T(1),T(4)^2*T(5)^2*T(2)+T(5)^2*T(1)]/De(1);
    YRtranxE(:,:,2)=[T1(1),0,0;0,T1(2),0;0,0,T1(3)]*YRtranxE(:,:,1);
    YRtranxE(:,:,3)=[T(1),0,0;0,T(2),0;0,0,T(3)];
    MidYRtranxE(1,:)=[T(4)^2*T(5)^2*T(2)*T(3),T(4)*T(5)^2*T(3)*T(1),T(5)*T(4)^2*T(2)*T(1)]/De(1);
    MidYRtranxE(2,:)=[T1(4)^2*T1(5)^2*T1(2)*T1(3),T1(4)*T1(5)^2*T1(3)*T1(1),T1(5)*T1(4)^2*T1(2)*T1(1)]/De(2);
    ZRM2E=4*TR2(1:3)/TRstep;       ZTR2E=[ZRM2E(1)/2+TR2(4),       ZRM2E(3)/2+TR2(6);       ZRM2E(3)/2+TR2(6),
  ZRM2E(2)/2+TR2(5)];
    YTR2E(:,:,1)=inv(ZTR2E); YTR2E(:,:,2)=-[ZRM2E(1)/2,0;0,ZRM2E(3)/2]*YTR2E(:,:,1);
    YTR2E(:,:,3)=-[ZRM2E(3)/2,0;0,ZRM2E(2)/2]*YTR2E(:,:,1);
    BrE(1,1:4)=[1/ZRL(1), -1/ZRL(1), -1, 1];
    BrE(2,1:4)=[1/ZRL(2), -1/ZRL(2), -1, 1];
    BrE(3,1:4)=[1/ZRL(3), -1/ZRL(3), -1, 1];
    BrE(4,1:4)=[1/ZRC(1), -2/ZRC(1), 1/2, 0];
    BrE(5,1:4)=[1/ZRC(2), -2/ZRC(2), 1/2, 0];
    BrE(6,1:4)=[1/ZRC(3), -2/ZRC(3), 1/2, 0];
    BrE(7,1:4)=[1, -2, ZRC(4)/2, Rvector2(1)]/(Rvector2(1)+ZRC(4));
    BrE(8,1:4)=[1/ZRC(5), -2/ZRC(5), 1/2, 0];
    BrE(9,1:4)=[1/ZRC(6), -2/ZRC(6), 1/2, 0];
    BrE(10:12,1:3)=YRtranxE(:,:,1);
    BrE(13,1:4)=[1/ZRC(7), -2/ZRC(7), 1/2, 0];
    BrE(14:15,1:6)=[YTR2E(:,:,1), YTR2E(:,:,2), YTR2E(:,:,3)];
    BrE(16,1:4)=[1, -2, ZRC(8)/2, Rvector2(2)]/(Rvector2(2)+ZRC(8));
    BrE(17,1:4)=[1, -2, ZRC(9)/2, Rvector2(3)]/(Rvector2(3)+ZRC(9));
    BrE(18,1:4)=[1, -2, ZRC(10)/2, Rvector2(4)]/(Rvector2(4)+ZRC(10));
    BrE(19,1:4)=[1, -2, ZRC(11)/2, Rvector2(5)]/(Rvector2(5)+ZRC(11));
    D={ZRCE, ZRLE,YRtranxE,BrE,YTR2E, MidYRtranxE};
end;
```

IV. Sysmatrix

```
function D=sysmatrix(subnetwork)
%subnetwork == 1/2 give out the impedance matirx of L-subnetwork/R-subnetwork
global Rvector1 Rvector2 ZLD ZRD ZRQ YRLM BrL BrR
```

```
switch subnetwork,
  case 1
    % Generate Y matrix for rectifier
    YL=zeros(8,8);
    YL(1,2)=BrL(3,2)-BrL(7,1);YL(1,3)=-BrL(3,1);YL(1,8)=-BrL(3,2);
    YL(1,1)=BrL(1,1)-YL(1,2)-YL(1,3)-YL(1,8);
    YL(2,1)=BrL(4,1)-BrL(7,1);YL(2,3)=-BrL(4,1);YL(2,8)=-BrL(4,2);
    YL(2,2)=BrL(2,1)+BrL(7,1)+BrL(4,2);
    YL(3,1)=-BrL(3,1);YL(3,2)=-BrL(3,2);YL(3,4)=-BrL(5,1);YL(3,5)=-BrL(5,2);
    YL(3,8)=BrL(3,2)+BrL(5,2); YL(3,3)=BrL(3,1)+BrL(5,1);
    YL(4,3)=-BrL(5,1);YL(4,5)=BrL(5,2)-BrL(8,1);YL(4,6)=-1/ZLD(2)-1/Rvector1(4)-BrL(12,1);
    YL(4,7)=-1/ZLD(2)-1/Rvector1(3)-BrL(11,1); YL(4,8)=-BrL(5,2);
    YL(4,4)=-YL(4,3)+BrL(9,1)-YL(4,5)-YL(4,6)-YL(4,7)-YL(4,8);
    YL(5,3)=-BrL(6,1); YL(5,4)=BrL(6,1)-BrL(8,1);YL(5,6)=-1/ZLD(2);YL(5,7)=-1/ZLD(2);
    YL(5,8)=-BrL(6,2);YL(5,5)=BrL(10,1)-YL(5,6)-YL(5,7)-YL(5,8)+BrL(8,1);
    YL(6,4)=YL(4,6);YL(6,5)=YL(5,6);;YL(6,6)=-YL(6,4)-YL(6,5);
    YL(7,4)=YL(4,7);YL(7,5)=YL(5,7);YL(7,7)=-YL(7,4)-YL(7,5);
    YL(8,1)=-BrL(4,1); YL(8,2)=-BrL(4,2); YL(8,3)=BrL(4,1)+BrL(6,1);
    YL(8,4)=-BrL(6,1);YL(8,5)=-BrL(6,2); YL(8,8)=BrL(4,2)+BrL(6,2);
    Btemp=zeros(8,4); Btemp(4:7,:)=[-1,1,0,0; 0,0,-1,1; 1,0,1,0; 0,-1,0,-1];
    Dtemp=diag([-ZLD(1), -ZLD(1) -ZLD(1), -ZLD(1)]);
    D={YL,Btemp,Dtemp};
  case 2
    YR=zeros(12,12);
    YR(1,2)=-BrR(4,1);YR(1,1)=-YR(1,2)+1/ZRQ(2)+1/ZRD(2,2);
    YR(2,1)=YR(1,2);YR(2,3)=-BrR(5,1); YR(2,5)=-BrR(6,1); YR(2,2)=-YR(2,1)-YR(2,3)-YR(2,5);
    YR(3,2)=YR(2,3); YR(3,3)=-YR(3,2)+1/ZRQ(2)+1/ZRD(2,2);
    YR(4,11)=-BrR(2,1); YR(4,4)=-YR(4,11)+BrR(13,1)+1/Rvector2(7);
    YR(5,2)=YR(2,5); YR(5,6)=YRLM(1,2,1); YR(5,7)=-YRLM(1,3,1);
    YR(5,5)=-YR(5,2)+YRLM(1,1,1)+BrR(7,1);
    YR(6,5)=YR(5,6); YR(6,7)=-YRLM(2,3,1); YR(6,8)=-1/ZRD(1,2)-BrR(16,1);
    YR(6,9)=-1/ZRD(1,2)-BrR(18,1); YR(6,6)=YRLM(2,2,1)-YR(6,8)-YR(6,9);
    YR(7,5)=YR(5,7); YR(7,6)=YR(6,7); YR(7,8)=-1/ZRD(1,2)-BrR(17,1);
    YR(7,9)=-1/ZRD(1,2)-BrR(19,1); YR(7,7)=-YR(7,8)-YR(7,9)+YRLM(3,3,1);
    YR(8,6)=YR(6,8); YR(8,7)=YR(7,8); YR(8,9)=-BrR(14,2); YR(8,11)=BrR(14,2);
    YR(8,10)=-BrR(14,1)-BrR(3,1); YR(8,8)=-YR(8,6)-YR(8,7)-YR(8,10);
    YR(9,6)=YR(6,9); YR(9,7)=YR(7,9); YR(9,8)=-BrR(15,1); YR(9,10)=BrR(15,1);
    YR(9,11)=-BrR(15,2); YR(9,9)=-YR(9,6)-YR(9,7)-YR(9,11);
    YR(10,8)=YR(8,10);YR(10,9)=BrR(14,2);YR(10,11)=-BrR(14,2);YR(10,12)=-BrR(1,1);
    YR(10,10)=-YR(10,8)-YR(10,12)+BrR(8,1);
    YR(11,8)=BrR(15,1);YR(11,9)=YR(9,11); YR(11,10)=-BrR(15,1);
    YR(11,4)=-BrR(2,1); YR(11,11)=-YR(11,9)-YR(11,4);
    YR(12,10)=YR(10,12); YR(12,12)=-YR(12,10)+BrR(9,1)+1/Rvector2(6);
    Btemp=zeros(12,8);   Btemp(1:3,5:8)=[-1,0,1,0; 0,0,0,0; 0,1,0,-1];
    Btemp(6:9,1:4)=[1,0,-1,0; 0,1,0,-1; -1,-1,0,0; 0,0,1,1];
    Dtemp=diag([-ZRD(1),-ZRD(1),-ZRD(1),-ZRD(1),-ZRD(2),-ZRD(2),-ZRQ(1),-ZRQ(1)]);
    D={YR,Btemp,Dtemp};
end;
```

V. Histsrc

```
function [Zsource]=histsrc(swseq,varargin)
```

```
%calculate the current of two subnetworks
%nargin==1, Current source of L-subnetwork nargin ==2, Current source of R-subnetwork
%nargin==2, swseq<0: Current source of R-subnetwork (TRE)
global TR1 ZLD ZRD ZRQ YLLM YRLM SDL SDR BrL BrR IR TswLtemp BrE YRLME
global IonL IonR Mag Phy w TLstep VLinc VRinc VLMinc VRMinc XL XR ILpre IRpre

switch nargin,
  case 2,
    %giving out the equivalent current source of L-subnetwork.
    k=varargin{1};
    if swseq==0
      Vs(:,1)=Mag.*sin(w*(k-2)*TLstep+Phy); Vs(:,2)=Mag.*sin(w*(k-1)*TLstep+Phy);
    else
      Vs(:,1)=Mag.*sin(w*(k-2)*TLstep+Phy+w*TswLtemp(swseq));
      Vs(:,2)=Mag.*sin(w*(k-1)*TLstep+Phy+w*TswLtemp(swseq+1));
    end;
    VLinc(1,2)=BrL(1,3)*(-XL(1,2))+BrL(1,4)*VLinc(1,1)+BrL(1,6)*(-Vs(1,1));
    ILpre(1,1)=BrL(1,2)*VLinc(1,2)+BrL(1,5)*(-Vs(1,2));
    VLinc(2,2)=BrL(2,3)*(-XL(2,2))+BrL(2,4)*VLinc(2,1)+BrL(2,6)*(-Vs(2,1));
    ILpre(2,1)=BrL(2,2)*VLinc(2,2)+BrL(2,5)*(-Vs(2,2));
    temp=[BrL(3:4,3:4);BrL(3:4,5:6)]*([XL(1,2)-XL(3,2);XL(2,2)-XL(8,2)]-2*VLinc(3:4,1)...
      -2*VLMinc(1:2,1))-[VLinc(3,1);VLMinc(1,1);VLMinc(2,1);VLinc(4,1)];
    VLinc(3:4,2)=temp([1,4],1); VLMinc(1:2,2)=temp([2,3],1);
    ILpre(3:4,1)=-2*BrL(3:4,1:2)*(VLinc(3:4,2)+VLMinc(1:2,2));
    temp=[BrL(5:6,3:4);BrL(5:6,5:6)]*([XL(3,2)-XL(4,2);XL(8,2)-XL(5,2)]-2*VLinc(5:6,1)...
      -2*VLMinc(3:4,1))-[VLinc(5,1);VLMinc(3,1);VLMinc(4,1);VLinc(6,1)];
    VLinc(5:6,2)=temp([1,4],1); VLMinc(3:4,2)=temp([2,3],1);
    ILpre(5:6,1)=-2*BrL(5:6,1:2)*(VLinc(5:6,2)+VLMinc(3:4,2));
    VLinc(7:10,2)=BrL(7:10,3).*[XL(1,2)-XL(2,2); XL(4,2)-XL(5,2);XL(4,2); XL(5,2)] +BrL(7:10,4).*VLinc(7:10,1);
    ILpre(7:12,1)=BrL(7:12,2).*VLinc(7:12,2);
    IDLpre(1:4,1)=[SDL(1,2)*IonL; SDL(2,2)*IonL; SDL(3,2)*IonL; SDL(4,2)*IonL];
    Zsource=[ILpre(1)-ILpre(3)-ILpre(7); ILpre(2)-ILpre(4)+ILpre(7);
      ILpre(3)-ILpre(5); ILpre(5)-ILpre(8)-ILpre(9)+ILpre(11)-ILpre(12)-IDLpre(1,1)+IDLpre(2,1);
      ILpre(6)-ILpre(10)-IDLpre(3,1)+IDLpre(4,1)+ILpre(8); ILpre(12)+IDLpre(1,1)+IDLpre(3,1);
      -ILpre(11)-IDLpre(2,1)-IDLpre(4,1); ILpre(4)-ILpre(6)];
  case 3,
    if swseq>-0.5
      n=varargin{1}; k=varargin{2}-1;
      %giving out the equivalent current source of R-subnetwork.
      VRinc(1:3,2)=BrR(1:3,3).*[XR(10,2)-XR(12,2); XR(11,2)-XR(4,2); XR(8,2)-XR(10,2)]+BrR(1:3,4).*VRinc(1:3,1);
      IRpre(1:3,1)=BrR(1:3,2).*VRinc(1:3,2);
      VRinc(6:9,2)=BrR(6:9,3).*[XR(5,2)-XR(2,2); XR(5,2); XR(10,2); XR(12,2)]+BrR(6:9,4).*VRinc(6:9,1);
      IRpre(4:9,1)=BrR(4:9,2).*VRinc(4:9,2);
      VRinc(10:12,2)=-YRLM(:,:,2)*([XR(5,2);XR(6,2);-XR(7,2)]-2*VRinc(10:12,1))...
        -VRinc(10:12,1);
      IRpre(10:12,1)=-2*YRLM(:,:,1)*VRinc(10:12,2);
      VRinc(13,2)=-BrR(13,3)*XR(4,2)+BrR(13,4)*VRinc(13,1);
      IRpre(13,1)=BrR(13,2).*VRinc(13,2);
      temp=[BrR(14:15,3:4);BrR(14:15,5:6)]*([XR(8,2)-XR(10,2);XR(11,2)-XR(9,2)]...
        -2*VRinc(14:15,1)-2*VRMinc(1:2,1))-[VRinc(14,1);VRMinc(1,1);VRMinc(2,1);VRinc(15,1)];
      VRinc(14:15,2)=temp([1,4],1); VRMinc(1:2,2)=temp([2,3],1);
      IRpre(14:15,1)=-2*BrR(14:15,1:2)*(VRinc(14:15,2)+VRMinc(1:2,2));
      VRinc(16:19,2)=BrR(16:19,3).*[XR(6,2)-XR(8,2); XR(7,2)-XR(8,2); XR(9,2)-XR(6,2);
```

```
        XR(9,2)-XR(7,2)]+BrR(16:19,4).*VRinc(16:19,1);
      IRpre(16:19,1)=BrR(16:19,2).*VRinc(16:19,2);
      IDRpre(1:8,1)=[SDR(1,2)*IonR(1); SDR(2,2)*IonR(1); SDR(3,2)*IonR(1); SDR(4,2)*IonR(1);
        SDR(5,2)*IonR(2); SDR(6,2)*IonR(2); SDR(7,2)*IonR(3); SDR(8,2)*IonR(3)];
      Zsource=[-IRpre(4)+IDRpre(7,1)-IDRpre(5,1); IRpre(4)-IRpre(5)+IRpre(6);
        IRpre(5)+IDRpre(6,1)-IDRpre(8,1); IRpre(2)+IRpre(13);
        -IRpre(6)-IRpre(7)-IRpre(10); -IRpre(11)-IRpre(16)+IRpre(18)+IDRpre(1,1)-IDRpre(3,1);
        IRpre(12)-IRpre(17)+IRpre(19)+IDRpre(2,1)-IDRpre(4,1);
        -IRpre(14)+IRpre(16)+IRpre(17)-IDRpre(1,1)-IDRpre(2,1)-IRpre(3,1);
        IRpre(15)-IRpre(18)-IRpre(19)+IDRpre(3,1)+IDRpre(4,1);
        -IRpre(1)-IRpre(8)+IRpre(14)+IRpre(3,1); -IRpre(2)-IRpre(15); IRpre(1)-IRpre(9)];
    else
      n=varargin{1}; k=varargin{2}-1;
      %giving out the equivalent current source of R-subnetwork.
      VRinc(1:3,2)=BrE(1:3,3).*[XR(10,2)-XR(12,2); XR(11,2)-XR(4,2); XR(8,2)-XR(10,2)]+BrE(1:3,4).*VRinc(1:3,1);
      IRpre(1:3,1)=BrE(1:3,2).*VRinc(1:3,2);
      VRinc(6:9,2)=BrE(6:9,3).*[XR(5,2)-XR(2,2); XR(5,2); XR(10,2); XR(12,2)] +BrE(6:9,4).*VRinc(6:9,1);
      IRpre(4:9,1)=BrE(4:9,2).*VRinc(4:9,2);
      VRinc(10:12,2)=-YRLME(:,:,2)*([XR(5,2);XR(6,2);-XR(7,2)]-VRinc(10:12,1));
      IRpre(10:12,1)=-YRLME(:,:,1)*VRinc(10:12,2);
      VRinc(13,2)=-BrE(13,3)*XR(4,2)+BrE(13,4)*VRinc(13,1);
      IRpre(13,1)=BrE(13,2).*VRinc(13,2);
      temp=[BrE(14:15,3:4);BrE(14:15,5:6)]*([XR(8,2)-XR(10,2);XR(11,2)-XR(9,2)]-VRinc(14:15,1)-VRMinc(1:2,1));
      VRinc(14:15,2)=temp([1,4],1); VRMinc(1:2,2)=temp([2,3],1);
      IRpre(14:15,1)=-BrE(14:15,1:2)*(VRinc(14:15,2)+VRMinc(1:2,2));
      VRinc(16:19,2)=BrE(16:19,3).*[XR(6,2)-XR(8,2); XR(7,2)-XR(8,2); XR(9,2)-XR(6,2);
        XR(9,2)-XR(7,2)]+BrE(16:19,4).*VRinc(16:19,1);
      IRpre(16:19,1)=BrE(16:19,2).*VRinc(16:19,2);
      IDRpre(1:8,1)=[SDR(1,2)*IonR(1); SDR(2,2)*IonR(1); SDR(3,2)*IonR(1); SDR(4,2)*IonR(1);
        SDR(5,2)*IonR(2); SDR(6,2)*IonR(2); SDR(7,2)*IonR(3); SDR(8,2)*IonR(3)];
      Zsource=[-IRpre(4)+IDRpre(7,1)-IDRpre(5,1); IRpre(4)-IRpre(5)+IRpre(6);
        IRpre(5)+IDRpre(6,1)-IDRpre(8,1); IRpre(2)+IRpre(13);
        -IRpre(6)-IRpre(7)-IRpre(10); -IRpre(11)-IRpre(16)+IRpre(18)+IDRpre(1,1)-IDRpre(3,1);
        IRpre(12)-IRpre(17)+IRpre(19)+IDRpre(2,1)-IDRpre(4,1);
        -IRpre(14)+IRpre(16)+IRpre(17)-IDRpre(1,1)-IDRpre(2,1)-IRpre(3,1);
        IRpre(15)-IRpre(18)-IRpre(19)+IDRpre(3,1)+IDRpre(4,1);
        -IRpre(1)-IRpre(8)+IRpre(14)+IRpre(3,1); -IRpre(2)-IRpre(15); IRpre(1)-IRpre(9)];
    end;
end;
```

VI. Current

```
function [Amper]=current(subnetwork)
%calculate the current of two subnetworks
%subnetwork==1, Current of L-subnetwork %subnetwork==2, Current of R-subnetwork %subnetwork==3, Current of R-subnetwork
global ZLD ZRD ZRQ YRLM XL XR ILpre IRpre BrL BrR IonL IonR SDL SDR BrE YRLME

% varargin;
switch subnetwork,
  case 1
    IDLpre(1:4,1)=[SDL(1,2)*IonL; SDL(2,2)*IonL; SDL(3,2)*IonL; SDL(4,2)*IonL];
    Amper=[-XL(1,3)*BrL(1,1)+ILpre(1);    -XL(2,3)*BrL(2,1)+ILpre(2);
```

```
        (XL(1,3)-XL(3,3))*BrL(3,1)+(XL(2,3)-XL(8,3))*BrL(3,2)+ILpre(3);
        (XL(1,3)-XL(3,3))*BrL(4,1)+(XL(2,3)-XL(8,3))*BrL(4,2)+ILpre(4);
        (XL(3,3)-XL(4,3))*BrL(5,1)+(XL(8,3)-XL(5,3))*BrL(5,2)+ILpre(5);
        (XL(3,3)-XL(4,3))*BrL(6,1)+(XL(8,3)-XL(5,3))*BrL(6,2)+ILpre(6);
        (XL(1,3)-XL(2,3))*BrL(7,1)+ILpre(7); (XL(4,3)-XL(5,3))*BrL(8,1)+ILpre(8);
        XL(4,3)*BrL(9,1)+ILpre(9); XL(5,3)*BrL(10,1)+ILpre(10);
        (XL(7,3)-XL(4,3))*BrL(11,1)+ILpre(11); (XL(4,3)-XL(6,3))*BrL(12,1)+ILpre(12);
        (XL(6,3)-XL(4,3))/ZLD(2)+XL(9,3)-IDLpre(1,1);
        (XL(4,3)-XL(7,3))/ZLD(2)+XL(10,3)-IDLpre(2,1);
        (XL(6,3)-XL(5,3))/ZLD(2)+XL(11,3)-IDLpre(3,1);
        (XL(5,3)-XL(7,3))/ZLD(2)+XL(12,3)-IDLpre(4,1)];
case 2
    IDRpre(1:8,1)=[SDR(1,2)*IonR(1); SDR(2,2)*IonR(1); SDR(3,2)*IonR(1); SDR(4,2)*IonR(1);
        SDR(5,2)*IonR(2); SDR(6,2)*IonR(2); SDR(7,2)*IonR(3); SDR(8,2)*IonR(3)];
    Amper=[(XR(10,3)-XR(12,3))*BrR(1,1)+IRpre(1);   (XR(11,3)-XR(4,3))*BrR(2,1)+IRpre(2);
        (XR(8,3)-XR(10,3))*BrR(3,1)+IRpre(3); (XR(1,3)-XR(2,3))*BrR(4,1)+IRpre(4);
        (XR(2,3)-XR(3,3))*BrR(5,1)+IRpre(5); (XR(5,3)-XR(2,3))*BrR(6,1)+IRpre(6);
        XR(5,3)*BrR(7,1)+IRpre(7); XR(10,3)*BrR(8,1)+IRpre(8); XR(12,3)*BrR(9,1)+IRpre(9);
        YRLM(:,:,1)*[XR(5,3);XR(6,3);-XR(7,3)]+IRpre(10:12,1);
        -XR(4,3)*BrR(13,1)+IRpre(13);
        (XR(8,3)-XR(10,3))*BrR(14,1)+(XR(11,3)-XR(9,3))*BrR(14,2)+IRpre(14);
        (XR(8,3)-XR(10,3))*BrR(15,1)+(XR(11,3)-XR(9,3))*BrR(15,2)+IRpre(15);
        (XR(6,3)-XR(8,3))*BrR(16,1)+IRpre(16);(XR(7,3)-XR(8,3))*BrR(17,1)+IRpre(17);
        (XR(9,3)-XR(6,3))*BrR(18,1)+IRpre(18);(XR(9,3)-XR(7,3))*BrR(19,1)+IRpre(19);
        (XR(6,3)-XR(8,3))/ZRD(1,2)+XR(13,3)-IDRpre(1,1);
        (XR(7,3)-XR(8,3))/ZRD(1,2)+XR(14,3)-IDRpre(2,1);
        (XR(9,3)-XR(6,3))/ZRD(1,2)+XR(15,3)-IDRpre(3,1);
        (XR(9,3)-XR(7,3))/ZRD(1,2)+XR(16,3)-IDRpre(4,1);
        -XR(1,3)/ZRD(2,2)+XR(17,3)-IDRpre(5,1);
        XR(3,3)/ZRD(2,2)+XR(18,3)-IDRpre(6,1);
        XR(1,3)/ZRQ(2)+XR(19,3)-IDRpre(7,1);
        -XR(3,3)/ZRQ(2)+XR(20,3)-IDRpre(8,1)];
case 3
    IDRpre(1:8,1)=[SDR(1,2)*IonR(1); SDR(2,2)*IonR(1); SDR(3,2)*IonR(1); SDR(4,2)*IonR(1);
        SDR(5,2)*IonR(2); SDR(6,2)*IonR(2); SDR(7,2)*IonR(3); SDR(8,2)*IonR(3)];
    Amper=[(XR(10,3)-XR(12,3))*BrE(1,1)+IRpre(1);   (XR(11,3)-XR(4,3))*BrE(2,1)+IRpre(2);
        (XR(8,3)-XR(10,3))*BrE(3,1)+IRpre(3); (XR(1,3)-XR(2,3))*BrE(4,1)+IRpre(4);
        (XR(2,3)-XR(3,3))*BrE(5,1)+IRpre(5); (XR(5,3)-XR(2,3))*BrE(6,1)+IRpre(6);
        XR(5,3)*BrE(7,1)+IRpre(7); XR(10,3)*BrE(8,1)+IRpre(8);XR(12,3)*BrE(9,1)+IRpre(9);
        YRLME(:,:,1)*[XR(5,3);XR(6,3);-XR(7,3)]+IRpre(10:12,1);
        -XR(4,3)*BrE(13,1)+IRpre(13);
        (XR(8,3)-XR(10,3))*BrE(14,1)+(XR(11,3)-XR(9,3))*BrE(14,2)+IRpre(14);
        (XR(8,3)-XR(10,3))*BrE(15,1)+(XR(11,3)-XR(9,3))*BrE(15,2)+IRpre(15);
        (XR(6,3)-XR(8,3))*BrE(16,1)+IRpre(16);(XR(7,3)-XR(8,3))*BrE(17,1)+IRpre(17);
        (XR(9,3)-XR(6,3))*BrE(18,1)+IRpre(18);(XR(9,3)-XR(7,3))*BrE(19,1)+IRpre(19);
        (XR(6,3)-XR(8,3))/ZRD(1,2)+XR(13,3)-IDRpre(1,1);
        (XR(7,3)-XR(8,3))/ZRD(1,2)+XR(14,3)-IDRpre(2,1);
        (XR(9,3)-XR(6,3))/ZRD(1,2)+XR(15,3)-IDRpre(3,1);
        (XR(9,3)-XR(7,3))/ZRD(1,2)+XR(16,3)-IDRpre(4,1);
        -XR(1,3)/ZRD(2,2)+XR(17,3)-IDRpre(5,1);
        XR(3,3)/ZRD(2,2)+XR(18,3)-IDRpre(6,1);
        XR(1,3)/ZRQ(2)+XR(19,3)-IDRpre(7,1);
        -XR(3,3)/ZRQ(2)+XR(20,3)-IDRpre(8,1)];
```

```
end;
```

VII. Switchstate

```
function [sign]=switchstate(varargin)
%determine the state of switches %nargin=1,determine the state of L-subnetwork; nargin=2,determine the state of R-subnetwork
global XL XR SDL SDR SQRcom VonL VonR deltaV T4saw T4Q1th T4Q2th NumofPoints2

sign=0;
switch nargin,
  case 1,                       %L-subnetwork
    n=varargin{1};
    if SDL(1,1)==1
      if XL(6,3)-XL(4,3)>=VonL-deltaV
        SDL(1,3:4)=[1,1];           %on-state
      else SDL(1,3:4)=[-1,0];       %off-state
      end;
    elseif XL(6,3)-XL(4,3)>=VonL+deltaV
      SDL(1,3:4)=[1,1];             %on-state
    else  SDL(1,3:4)=[-1,0];        %off-state
    end;
    if SDL(1,1)~=SDL(1,3)
      sign=sign+1;
    end;
    if SDL(3,1)==1
      if XL(6,3)-XL(5,3)>=VonL-deltaV
        SDL(3,3:4)=[1,1];           %on-state
      else SDL(3,3:4)=[-1,0];       %off-state
      end;
    elseif XL(6,3)-XL(5,3)>=VonL+deltaV
      SDL(3,3:4)=[1,1];             %on-state
    else  SDL(3,3:4)=[-1,0];        %off-state
    end;
    if SDL(3,1)~=SDL(3,3)
      sign=sign+1;
    end;
    if SDL(2,1)==1
      if XL(4,3)-XL(7,3)>=VonL-deltaV
        SDL(2,3:4)=[1,1];           %on-state
      else SDL(2,3:4)=[-1,0];       %off-state
      end;
    elseif XL(4,3)-XL(7,3)>=VonL+deltaV
      SDL(2,3:4)=[1,1];             %on-state
    else  SDL(2,3:4)=[-1,0];        %off-state
    end;
    if SDL(2,1)~=SDL(2,3)
      sign=sign+1;
    end;
    if SDL(4,1)==1
      if XL(5,3)-XL(7,3)>=VonL-deltaV
        SDL(4,3:4)=[1,1];           %on-state
      else SDL(4,3:4)=[-1,0];       %off-state
      end;
    elseif XL(5,3)-XL(7,3)>=VonL+deltaV
      SDL(4,3:4)=[1,1];              %on-state
    else   SDL(4,3:4)=[-1,0];            %off-state
    end;
    if SDL(4,1)~=SDL(4,3)
      sign=sign+1;
    end;
  case 2,                        %R-subnetwork
    n=varargin{1}; k=varargin{2};
    if SDR(1,1)==1
      if XR(6,3)-XR(8,3)>=VonR(1)-deltaV
        SDR(1,3:4)=[1,1];              %on-state
      else SDR(1,3:4)=[-1,0];            %off-state
      end;
    elseif XR(6,3)-XR(8,3)>=VonR(1)+deltaV
      SDR(1,3:4)=[1,1];              %on-state
    else   SDR(1,3:4)=[-1,0];            %off-state
    end;
    if SDR(1,1)~=SDR(1,3)
      sign=sign+1;
    end;
    if SDR(2,1)==1
      if XR(7,3)-XR(8,3)>=VonR(1)-deltaV
        SDR(2,3:4)=[1,1];              %on-state
      else SDR(2,3:4)=[-1,0];            %off-state
      end;
    elseif XR(7,3)-XR(8,3)>=VonR(1)+deltaV
      SDR(2,3:4)=[1,1];              %on-state
    else   SDR(2,3:4)=[-1,0];            %off-state
    end;
    if SDR(2,1)~=SDR(2,3)
      sign=sign+1;
    end;
    if SDR(3,1)==1
      if XR(9,3)-XR(6,3)>=VonR(1)-deltaV
        SDR(3,3:4)=[1,1];              %on-state
      else SDR(3,3:4)=[-1,0];            %off-state
      end;
    elseif XR(9,3)-XR(6,3)>=VonR(1)+deltaV
      SDR(3,3:4)=[1,1];              %on-state
    else   SDR(3,3:4)=[-1,0];            %off-state
    end;
    if SDR(3,1)~=SDR(3,3)
      sign=sign+1;
    end;
    if SDR(4,1)==1
      if XR(9,3)-XR(7,3)>=VonR(1)-deltaV
        SDR(4,3:4)=[1,1];              %on-state
      else SDR(4,3:4)=[-1,0];            %off-state
      end;
    elseif XR(9,3)-XR(7,3)>=VonR(1)+deltaV
      SDR(4,3:4)=[1,1];              %on-state
```

```
else   SDR(4,3:4)=[-1,0];              %off-state
end;
if SDR(4,1)~=SDR(4,3)
  sign=sign+1;
end;
if SDR(5,1)==1
  if -XR(1,3)>=VonR(2)-deltaV
    SDR(5,3:4)=[1,1];                  %on-state
  else SDR(5,3:4)=[-1,0];              %off-state
  end;
elseif -XR(1,3)>=VonR(2)+deltaV
  SDR(5,3:4)=[1,1];                    %on-state
else   SDR(5,3:4)=[-1,0];              %off-state
end;
if SDR(5,1)~=SDR(5,3)
  sign=sign+1;
end;
if SDR(6,1)==1
  if XR(3,3)>=VonR(2)-deltaV
    SDR(6,3:4)=[1,1];                  %on-state
  else SDR(6,3:4)=[-1,0];              %off-state
  end;
elseif XR(3,3)>=VonR(2)+deltaV
  SDR(6,3:4)=[1,1];                    %on-state
else   SDR(6,3:4)=[-1,0];              %off-state
end;
if SDR(6,1)~=SDR(6,3)
  sign=sign+1;
end;
if n>=1
  t=rem((n-1)*NumofPoints2+k-1,2*T4saw);
  if t<T4Q1th
    SQRcom(1:2,2)=[1;-1]; SDR(8,3:4)=[-1,0];
    if XR(1,3)>VonR(3)
      SDR(7,3:4)=[1,1];
    else   SDR(7,3:4)=[-1,0];
    end;
  elseif t>=T4saw & t<T4Q2th
    SQRcom(1:2,2)=[-1;1]; SDR(7,3:4)=[-1,0];
    if -XR(3,3)>VonR(3)
      SDR(8,3:4)=[1,1];
    else SDR(8,3:4)=[-1,0];
    end;
  else SQRcom(:,2)=[-1;-1]; SDR(7:8,3:4)=[-1,0;-1,0];
  end;
else   SQRcom(:,2)=[-1;-1]; SDR(7:8,3:4)=[-1,0;-1,0];
end;
if SDR(7,1)~=SDR(7,3)
  sign=sign+1;
end;
if SDR(8,1)~=SDR(8,3)
  sign=sign+1;
end;
```

```
end;
if sign>0
   sign=1;
else   sign=0;
end;
```

VIII. Switchtime

```
function [Tsw]=switchtime(subnetwork)
%determine the state of switches
%nargin=1, determine the state of L-subnetwork /=2, determine the state of R-subnetwork
%nargin=3, determine the state of R-subnetwork (TRE)
global TLstep TRstep TRE XL XR VonL VonR deltaV SDL SDR SQRcom deltaT

switch subnetwork,
   case 1,                        %L-subnetwork
      if SDL(1,1)~=SDL(1,3) & abs(XL(6,3)-XL(4,3)-VonL)>=deltaV
         Tsw(1)=(VonL-(XL(6,2)-XL(4,2)))*TLstep/(XL(6,3)-XL(4,3)-(XL(6,2)-XL(4,2)));
      else   Tsw(1)=TLstep;
      end;
      if abs(Tsw(1))<deltaT
         Tsw(1)=0;
      end;
      if SDL(2,1)~=SDL(2,3) & abs(XL(4,3)-XL(7,3)-VonL)>=deltaV
         Tsw(2)=(VonL-(XL(4,2)-XL(7,2)))*TLstep/(XL(4,3)-XL(7,3)-(XL(4,2)-XL(7,2)));
      else Tsw(2)=TLstep;
      end;
      if abs(Tsw(2))<deltaT & abs(XL(6,3)-XL(4,3)-(XL(6,2)-XL(4,2)))>2*VonR(1)
         Tsw(2)=0;
      end;
      if SDL(3,1)~=SDL(3,3) & abs(XL(6,3)-XL(5,3)-VonL)>=deltaV
         Tsw(3)=(VonL-(XL(6,2)-XL(5,2)))*TLstep/(XL(6,3)-XL(5,3)-(XL(6,2)-XL(5,2)));
      else Tsw(3)=TLstep;
      end;
      if abs(Tsw(3))<deltaT
         Tsw(3)=0;
      end;
      if SDL(4,1)~=SDL(4,3) & abs(XL(5,3)-XL(7,3)-VonL)>=deltaV
         Tsw(4)=(VonL-(XL(5,2)-XL(7,2)))*TLstep/(XL(5,3)-XL(7,3)-(XL(5,2)-XL(7,2)));
      else Tsw(4)=TLstep;
      end;
      if abs(Tsw(4))<deltaT
         Tsw(4)=0;
      end;
   case 2,
      if SDR(1,1)~=SDR(1,3) & abs(XR(6,3)-XR(8,3)-VonR(1))>=deltaV
         Tsw(1)=(VonR(1)-(XR(6,2)-XR(8,2)))*TRstep/(XR(6,3)-XR(8,3)-(XR(6,2)-XR(8,2)));
      else Tsw(1)=TRstep;
      end;
      if abs(Tsw(1))<deltaT
         Tsw(1)=0;
      end;
```

```
if SDR(2,1)~=SDR(2,3) & abs(XR(7,3)-XR(8,3)-VonR(1))>=deltaV
  Tsw(2)=(VonR(1)-(XR(7,2)-XR(8,2)))*TRstep/(XR(7,3)-XR(8,3)-(XR(7,2)-XR(8,2)));
else Tsw(2)=TRstep;
end;
if abs(Tsw(2))<deltaT
  Tsw(2)=0;
end;
if SDR(3,1)~=SDR(3,3) & abs(XR(9,3)-XR(6,3)-VonR(1))>=deltaV
  Tsw(3)=(VonR(1)-(XR(9,2)-XR(6,2)))*TRstep/(XR(9,3)-XR(6,3)-(XR(9,2)-XR(6,2)));
else Tsw(3)=TRstep;
end;
if abs(Tsw(3))<deltaT
  Tsw(3)=0;
end;
if SDR(4,1)~=SDR(4,3) & abs(XR(9,3)-XR(7,3)-VonR(1))>=deltaV
  Tsw(4)=(VonR(1)-(XR(9,2)-XR(7,2)))*TRstep/(XR(9,3)-XR(7,3)-(XR(9,2)-XR(7,2)));
else Tsw(4)=TRstep;
end;
if abs(Tsw(4))<deltaT
  Tsw(4)=0;
end;
if SDR(5,1)~=SDR(5,3) & abs(-XR(1,3)-VonR(2))>=deltaV
  Tsw(5)=(VonR(2)+XR(1,2))*TRstep/(-XR(1,3)+XR(1,2));
else Tsw(5)=TRstep;
end;
if abs(Tsw(5))<deltaT
  Tsw(5)=0;
end;
if SDR(6,1)~=SDR(6,3) & abs(XR(3,3)-VonR(2))>=deltaV
  Tsw(6)=(VonR(2)-XR(3,2))*TRstep/(XR(3,3)-XR(3,2));
else Tsw(6)=TRstep;
end;
if abs(Tsw(6))<deltaT
  Tsw(6)=0;
end;
ini=0;
if SDR(7,1)==SDR(7,3)
  Tsw(7)=TRstep;
elseif SQRcom(1,1)==-1
  Tsw(7)=TRstep;
else
  Tsw(7)=(VonR(3)-XR(1,2))*TRstep/(XR(1,3)-XR(1,2));
  if abs(Tsw(7))<deltaT
    Tsw(7)=0;
  end;
  if Tsw(7)>TRstep | Tsw(7)<0
    Tsw(7)=TRstep;
  end;
end;
if SDR(8,1)==SDR(8,3)
  Tsw(8)=TRstep;
elseif SQRcom(2,1)==-1
  Tsw(8)=TRstep;
```

```
    else
      Tsw(8)=(VonR(3)+XR(3,2))*TRstep/(-XR(3,3)+XR(3,2));
      if abs(Tsw(8))<deltaT
        Tsw(8)=0;
      end;
      if Tsw(8)>TRstep | Tsw(8)<0
        Tsw(8)=TRstep;
      end;
    end;
  case 3,
    if SDR(1,1)~=SDR(1,3) & abs(XR(6,3)-XR(8,3)-VonR(1))>=deltaV
      Tsw(1)=(VonR(1)-(XR(6,2)-XR(8,2)))*TRE/(XR(6,3)-XR(8,3)-(XR(6,2)-XR(8,2)));
    else Tsw(1)=TRE;
    end;
    if abs(Tsw(1))<deltaT
      Tsw(1)=0;
    end;
    if SDR(2,1)~=SDR(2,3) & abs(XR(7,3)-XR(8,3)-VonR(1))>=deltaV
      Tsw(2)=(VonR(1)-(XR(7,2)-XR(8,2)))*TRE/(XR(7,3)-XR(8,3)-(XR(7,2)-XR(8,2)));
    else Tsw(2)=TRE;
    end;
    if abs(Tsw(2))<deltaT
      Tsw(2)=0;
    end;
    if SDR(3,1)~=SDR(3,3) & abs(XR(9,3)-XR(6,3)-VonR(1))>=deltaV
      Tsw(3)=(VonR(1)-(XR(9,2)-XR(6,2)))*TRE/(XR(9,3)-XR(6,3)-(XR(9,2)-XR(6,2)));
    else Tsw(3)=TRE;
    end;
    if abs(Tsw(3))<deltaT
      Tsw(3)=0;
    end;
    if SDR(4,1)~=SDR(4,3) & abs(XR(9,3)-XR(7,3)-VonR(1))>=deltaV
      Tsw(4)=(VonR(1)-(XR(9,2)-XR(7,2)))*TRE/(XR(9,3)-XR(7,3)-(XR(9,2)-XR(7,2)));
    else Tsw(4)=TRE;
    end;
    if abs(Tsw(4))<deltaT
      Tsw(4)=0;
    end;
    if SDR(5,1)~=SDR(5,3) & abs(-XR(1,3)-VonR(2))>=deltaV
      Tsw(5)=(VonR(2)+XR(1,2))*TRE/(-XR(1,3)+XR(1,2));
    else Tsw(5)=TRE;
    end;
    if abs(Tsw(5))<deltaT
      Tsw(5)=0;
    end;
    if SDR(6,1)~=SDR(6,3) & abs(XR(3,3)-VonR(2))>=deltaV
      Tsw(6)=(VonR(2)-XR(3,2))*TRE/(XR(3,3)-XR(3,2));
    else Tsw(6)=TRE;
    end;
    if abs(Tsw(6))<deltaT
      Tsw(6)=0;
    end;
    ini=0;
    if SDR(7,1)==SDR(7,3)
      Tsw(7)=TRE;
    elseif SQRcom(1,1)==-1
      Tsw(7)=TRE;
    else
      Tsw(7)=(VonR(3)-XR(1,2))*TRE/(XR(1,3)-XR(1,2));
      if abs(Tsw(7))<deltaT
        Tsw(7)=0;
      end;
      if Tsw(7)>TRE | Tsw(7)<0
        Tsw(7)=TRE;
      end;
    end;
    if SDR(8,1)==SDR(8,3)
      Tsw(8)=TRE;
    elseif SQRcom(2,1)==-1
      Tsw(8)=TRE;
    else
      Tsw(8)=(VonR(3)+XR(3,2))*TRE/(-XR(3,3)+XR(3,2));
      if abs(Tsw(8))<deltaT
        Tsw(8)=0;
      end;
      if Tsw(8)>TRE | Tsw(8)<0
        Tsw(8)=TRE;
      end;
    end;
end;
```

IX. Initial

```
function [getini]=initial(n,k)
global XR IR SDR SQRcom IRpre ZRD ZRQ VonR IonR Rvector2 MidYRtranx BrR TR1 Vlink ZRC VRinc YRLM

SDini=SDR(5:8,:); Vsw=[-XR(1,3); XR(3,3); XR(1,3); -XR(3,3)]; SQRini=SQRcom; Icomp=zeros(4,1);
IDRpre(1:4,1)=[SDR(5,2)*IonR(2); SDR(6,2)*IonR(2); SDR(7,2)*IonR(3); SDR(8,2)*IonR(3)]; start=1;
while start>0.5 | ~isequal(SDini(:,1),SDini(:,3))
  start=0;  SDini(:,[1,2])=SDini(:,[3,4]);
  if SDini(1,1)==-1
    R5=ZRD(2,2);
  else R5=ZRD(2,1)*ZRD(2,2)/(ZRD(2,1)+ZRD(2,2));
  end;
  if SDini(2,1)==-1
    R6=ZRD(2,2);
  else R6=ZRD(2,1)*ZRD(2,2)/(ZRD(2,1)+ZRD(2,2));
  end;
  if SDini(3,1)==-1
    R7=ZRQ(2);
  else R7=ZRQ(1)*ZRQ(2)/(ZRQ(1)+ZRQ(2));
  end;
  if SDini(4,1)==-1
    R8=ZRQ(2);
  else R8=ZRQ(1)*ZRQ(2)/(ZRQ(1)+ZRQ(2));
```

```
    end;
    if SDR(5,1)==SDini(1,1)
      Icomp(1)=0;
    elseif SDR(5,1)==-1 & SDini(1,1)==1
      Icomp(1)=(Vsw(1)-VonR(2))/ZRD(2,1); %enter node 1
    else Icomp(1)=-XR(17,3)+IDRpre(1,1);
    end;
    if SDR(6,1)==SDini(2,1)
      Icomp(2)=0;
    elseif SDR(6,1)==-1 & SDini(2,1)==1
      Icomp(2)=(Vsw(2)-VonR(2))/ZRD(2,1); %enter node 0
    else Icomp(2)=-XR(18,3)+IDRpre(2,1);
    end;
    if SQRini(1,1)==1 & SQRini(1,2)==-1 & SDR(7,1)==1
      Icomp(3)=XR(19,3)-IDRpre(3,1);     %enter node 1
    elseif SQRini(1,1)==-1 & SQRini(1,2)==1 & SDini(3,1)==1
      Icomp(3)=-(Vsw(3)-VonR(3))/ZRQ(1);
    else Icomp(3)=0;
    end;
    if SQRini(2,1)==1 & SQRini(2,2)==-1 & SDR(8,1)==1
      Icomp(4)=XR(20,3)-IDRpre(4,1);     %enter node 0
    elseif SQRini(2,1)==-1 & SQRini(2,2)==1 & SDini(4,1)==1
      Icomp(4)=-(Vsw(4)-VonR(3))/ZRQ(1);
    else Icomp(4)=0;
    end;
    R=1/(1/R5+1/R6+1/R7+1/R8+1/Rvector2(1)); V=(Icomp(1)-Icomp(2)+Icomp(3)-Icomp(4))*R;
    if -V+Vsw(1)>=VonR(2)
      SDini(1,[3,4])=[1,1];
    else SDini(1,[3,4])=[-1,0];
    end;
    if V+Vsw(2)>=VonR(2)
      SDini(2,[3,4])=[1,1];
    else SDini(2,[3,4])=[-1,0];
    end;
    if V+Vsw(3)>=VonR(3) & SQRini(1,2)==1
      SDini(3,[3,4])=[1,1];
    else SDini(3,[3,4])=[-1,0];
    end;
    if -V+Vsw(4)>=VonR(3) & SQRini(2,2)==1
      SDini(4,[3,4])=[1,1];
    else SDini(4,[3,4])=[-1,0];
    end;
  end;
  XR([1:3,5],3)=XR([1:3,5],3)+V; Vsw=[-XR(1,3); XR(3,3); XR(1,3); -XR(3,3)];
  if SDini(1,1)==1
    XR(17,3)=Vsw(1)/ZRD(2,1);   Isw(1)=XR(17,3)-IonR(2)+Vsw(1)/ZRD(2,2);
  else XR(17,3)=0; Isw(1)=Vsw(1)/ZRD(2,2);
  end;
  if SDini(2,1)==1
    XR(18,3)=Vsw(2)/ZRD(2,1);   Isw(2)=XR(18,3)-IonR(2)+Vsw(2)/ZRD(2,2);
  else XR(18,3)=0; Isw(2)=Vsw(2)/ZRD(2,2);
  end;
  if SDini(3,1)==1
```

```
   XR(19,3)=Vsw(3)/ZRQ(1);   Isw(3)=XR(19,3)-IonR(3)+Vsw(3)/ZRQ(2);
else XR(19,3)=0; Isw(3)=Vsw(3)/ZRQ(2);
end;
if SDini(4,1)==1
   XR(20,3)=Vsw(4)/ZRQ(1);   Isw(4)=XR(20,3)-IonR(3)+Vsw(4)/ZRQ(2);
else XR(20,3)=0; Isw(4)=Vsw(4)/ZRQ(2);
end;
IR(4,3)=Isw(1)-Isw(3); IR(5,3)=Isw(2)-Isw(4);   IR(7,3)=IR(7,3)+V/Rvector2(1); IR(6,3)=IR(6,3)-V/Rvector2(1);
VRinc(4:5,2)=([XR(1,3)-XR(2,3);XR(2,3)-XR(3,3)]-[IR(4,3)*ZRC(1);IR(5,3)*ZRC(2)])/2;
VRinc(6,2)=(XR(5,3)-XR(2,3)-ZRC(3)*IR(6,3))/2; VRinc(7,2)=(IR(7,3)-BrR(7,1)*XR(5,3))/BrR(7,2);
VRN=MidYRtranx(2,:)*([XR(5,3);XR(6,3);-XR(7,3)]-TR1(1,6:8)'.*IR(10:12,3));
VRinc(10:12,2)=([XR(5,3);XR(6,3);-XR(7,3)]-VRN*[1;1/TR1(4);1/TR1(5)]-YRLM(:,:,3)*IR(10:12,3))/2;
SDR(5:8,:)=SDini;   getini=1;   % if getini=0, only tranditional TLM models are used.
```

Reference

[1]. Power Sources Manufacturers Association, Inc., *Handbook of Standardized Terminology for the Power Sources Industry, 2nd ed.*, 1995.

[2]. Ned Mohan, Tore M. Undeland, William P. Robbins, *Power electronics: Convert, Application, and Design* (2nd ed.), John Wiley & Sons, Inc. 1995.

[3]. H. W. Dommel, "Digital computer solution of electromagnetic transients in single and multiple networks," *IEEE Trans. Power Apparatus Syst.*, vol. 88, No. 4, pp. 388-399, April 1969.

[4]. H. Jin, "Behavior-mode simulation of power electronic circuits," *IEEE Trans. Power Electronics*, vol. 12, issue: 3, pp. 443-452, May 1997.

[5]. S. R. Naidu, A. M. N. Lima, "A new approach for the simulation of power electronic circuits," *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, issue: 9, pp. 1317-1324, Sept. 2002

[6]. L.Salazar, G. Joos; "PSPICE simulation of three-phase inverters by means of switching functions," *IEEE Trans.Power Electronics*, vol. 9, issue: 1, pp. 35-42, Jan. 1994

[7]. I. Budihardjo, P. O. Lauritzen, K. Y. Wong, R. B. Darling, H. A. Mantooth, "Defining standard performance levels for power semiconductor devices," in *Proc. 1995 IEEE Industry Applications Annual Conf.*, vol. 2, pp. 1084-1090.

[8]. H. R. Visser, P. P. J. van den Bosch, "Modeling of periodically switching networks," in *Proc. 1991 Power Electronics Specialists Annual Conf.*, pp. 67-73.

[9]. P. R. Mugur, J. Roudet, J. C. Crebier, "Power electronic converter EMC analysis through state variable approach techniques," *IEEE Trans. Electromagnetic Compatibility*, vol. 43, issue: 2, pp. 229-238, May 2001.

[10]. S. Y. R. Hui, S. Morrall, "Generalised associated discrete circuit model for switching devices," *IEE Proc. Science, Measurement and Technology*, vol. 141, issue: 1, pp. 57-64, Jan. 1994

[11]. K. K. Fung, S. Y. R. Hui, "Improved TLM link model for reactive circuit components," IEE Proc. Science, Measurement and Technology, vol. 143, issue: 5, pp. 341-344, Sept. 1996.

[12]. P. Pejovic, D. Maksimovic, "A Method for Fast Time-Domain Simulation of Networks with Switches," *IEEE Trans. Power Electronics*, vol. 9, issue 4, pp. 449-456, July 1994.

[13]. A. B. Yildiz, N. Abut, "An Efficient Approach to Modelling and Analysis of Power Electronic Circuits," in *Proc. 1998 Applied Power Electronics Annual Conference and Exposition*, vol. 1, pp. 344-349.

[14]. K. K. Fung, S. Y. R. Hui, "Fast simulation of multistage power electronic systems with widely separated operating frequencies," *IEEE Trans. Power Electronics*, vol. 11, issue: 3, pp. 405-412, May 1996

[15]. S. Y. R. Hui, C. Christopoulos, "Discrete transform technique for solving nonlinear circuits

and equations," *IEE Proc. Science, Measurement and Technology*, vol. 139, issue: 6, pp. 321-328, Nov. 1992.

[16]. S. Y. R. Hui, C. Christopoulos, "Discrete transform technique for solving coupled integro-differential equations in digital computers," *IEE Proc. Science, Measurement and Technology*, vol. 138, issue: 5, pp. 273-280, Sept. 1991.

[17]. S. Y. R. Hui, K. K. Fung, C. Christopoulos, "Decoupled simulation of multi-stage power electronic systems using transmission-line links," *IEEE Trans. Power Electronics*, vol. 9, issue 1, pp. 85-91, Jan. 1994.

[18]. K. K. Fung,; S. Y. R. Hui, "Improved TLM Link Model for Reactive Circuit Components," *IEE Proc. Science, Measurement and Technology*, vol. 143, issue 5, pp. 341-344, Sept. 1996.

[19]. J. R. Marti, J. Lin, "Suppression of numerical oscillations in the EMTP power systems," *IEEE Trans. Power Systems*, vol. 4, issue 2, pp.739 - 747, May 1989.

[20]. C. C. Chan, K. -T. Chau, "A Fast and Exact Time-Domain Simulation of Switched-Mode Power regulators," *IEEE Trans. Industrial Electronics*, vol. 39, issue 4, pp.341-350 Aug. 1992.

[21]. H. Selhi, C. Christopoulos, "Generalised TLM switch model for power electronics applications," *IEE Proc. Science, Measurement and Technology*, vol. 145, issue 3, pp. 101-104, May 1998

[22]. B. K. H. Wong, H. Chung, "Time-domain simulation of power electronics circuits using state variable quadratic extrapolations," *IEEE Trans.Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, issue: 6, pp. 751-756, June 1999.

[23]. B. K. H. Wong, H. S. Chung, "An efficient technique for the time-domain simulation of power electronic circuits," *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, issue: 4, pp.364-376, April 1998.

[24]. H. S. -H. Chung, S. Y. R. Hui, K. Y. Mak, "Time domain simulation of power electronics circuits using embedded companion models," in *Proc. 1999 IEEE International Industrial Electronics Symposium*, vol. 1, pp. 226-231.

[25]. B. K. H. Wong, K. K. Tse, H. Chung, "A fast time-domain simulation algorithm for the power electronic circuits," in *Proc. 1996 IEEE International Industrial Electronics, Control, and Instrumentation Conf.*, vol. 2, pp.659-664.

[26]. E. Chiarantoni, G. Fornarelli, S. Vergura, "A new method for efficient time-domain simulation of power electronic circuits," in *Proc. 2002 IEEE International Circuits and Systems Symposium*, vol. 5, pp. 37-40.

[27]. S. A. Sudha, Chandrasekaran A, V. Rajagopalan, "New approach to switch modelling in the analysis of power electronic systems," *IEE Proc. Electric Power Applications*, vol. 140, issue 2, pp. 115-123, March 1993.

[28]. C. Baroncelli, P. Maranesi, R. Rossi, "Linear Models of Diode-Capacitor Rectifiers and Multipliers," in *Proc. 1996 IEEE Power Electronics Specialists Annual Conf.*, vol. 1, pp.

818-822.

[29]. P. W. Lehn, M. R. Irvani, "Discrete Time Modeling and Control of the Voltage Source Converter for Improved Disturbance Rejection," *IEEE Trans. Power Electronics*, vol. 14, issue 6, pp. 1028-1036, Nov. 1999.

[30]. B. De Kelper, L. A. Dessaint, V. Q. Do, J. -C Soumagne, "An algorithm for accurate switching representation in fixed-step simulation of power electronics," in *Proc. 2000 IEEE Power Engineering Society Winter Meeting*, vol. 1, pp. 762-767.

[31]. B. De Kelper, L. A. Dessaint, K. Al-Haddad, H. Nakra, "A comprehensive approach to fixed-step simulation of switched circuits," *IEEE Trans. Power Electronics*, vol. 17, issue 2, pp. 216-224, March 2002.

[32]. V. A. Niemela, "Leakage-impedance model for multiple-winding transformers," in *Proc. 2000 IEEE Power Electronics Specialists Annual Conf.*, vol. 1, pp. 264-269.

[33]. Chung-Wen Ho, A. Ruehli, P. Brennan, " The modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems*, vol. 22, issue 6, pp.504-509, Jun 1975.

[34]. Kijun Lee, Song-Bai Park, "Reduced modified nodal approach to circuit analysis," *IEEE Trans. Circuits and Systems*, vol. 32, issue 10, pp.1056-1060, Oct 1985.

[35]. Fung-Yue Chang; "The unified nodal approach to circuit analysis," in *Proc. 1997 IEEE International Circuits and Systems Symposium*, vol. 2, pp.849-852.

[36]. N. Femia, "A fast method to find the state of switches after forced commutations in switching converters," in *Proc. 1996 IEEE Power Electronics Specialists Annual Conf.*, vol. 2, pp.1356-1362.

[37]. N. Femia, "The master-slave connection matrix: a tool for improving the simulation of switched RLC networks," in *Proc. 1997 International Industrial Electronics, Control and Instrumentation Conf.*, vol. 2, pp.633 – 638.

[38]. N. Femia, G. Spagnuolo, M. Vitelli, "Unified analysis of synchronous commutations in switching converters," *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, issue 8, pp. 1150-1166, Aug. 2002

[39]. M. O. Faruque, V. Dinavahi, Wilsun Xu, "Algorithms for the accounting of multiple switching events in digital simulation of power-electronic systems," *IEEE Trans. Power Delivery*, vol. 20, issue 2, pp. 1157-1167, April 2005.

[40]. J. Mahseredjian, G. Joos, "Solution methods for varying topology networks," in Proc. 2002 IEEE Power Engineering Society Winter Meeting, vol. 1, pp. 479-483.

[41]. J. Vlach, A. Opal, J. Wojciechowski, "Simulation of networks with inconsistent initial conditions," in *Proc. 1993 IEEE International Circuits and Systems Symposium*, vol.3, pp. 1627-1630.

[42]. M. Demir, A. Bekir Yildiz, "Defining of the initial conditions in the switching circuits," in *Proc. 2004 IEEE Mediterranean Electrotechnical Conf.*, vol. 1, pp.59 – 62.

[43]. Michael J. Cloud, Edward Rothwell, *Electromagnetics* [Online]. Available: http://www.engnetbase.com

[44]. J. D. Glover, M. S. Sarma, *Power System Analysis and Design* (3rd ed.), Brooks/Cole, Thompson Learning, 2002, ISBN 0-534-95367-0

[45]. Matthew N. O. Sadiku, *Numerical Techniques in Electromagnetics* (2nd ed.), [Online]. Available: http://www.engnetbase.com.

[46]. D. Maksimovic, A. M. Stankovic, V. J. Thottuvelil, G. C. Verghese, "Modeling and simulation of power electronic converters," *Proceedings of the IEEE* vol. 89, issue 6, pp.898-912, June 2001

[47]. A. Opal, "The transition matrix for linear circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, issue 5, pp. 427-436, May 1997.

[48]. Y. Kang, J. G. Lacy, "Conversion of MNA equations to state variable form for nonlinear dynamical circuits", *Electronics Letters*, vol. 28, issue 13, pp.1240 – 1241, June 1992

[49]. S. Natarajan, "A systematic method for obtaining state equations using MNA", *IEE Proc. Circuits, Devices and Systems*, vol. 138, issue 3, pp.341 – 346, June 1991.

[50]. *EMTDC User's Guide 402*, Winnipeg, MB, Canada: Manitoba HVDC Research Center

[51]. *PSCAD User's Guide 402*, Winnipeg, MB, Canada: Manitoba HVDC Research Center

[52]. Balabanian, Norman , *Electric Circuits*, McGraw-Hill, 1994.

[53]. Samuel M. Goldwassser; *Nots on the troubleshooting and repair of small switchmode power supplies*, [Online],Availabe: http://an.hitchcock.org/repairfaq/sam/smpsfaq.htm.

[54]. M. H. Nagrial, A. Hellany, "Radiated and conducted EMI emission SMPS sources, causes and predictions," in *Proc. 2001 IEEE INMIC conf.*, pp. 54 - 61

[55]. Xin Wu, M. H. Pong, C. M. Lee, Z. M. Qian, "Reduction of EMI by electric field method," in *Proc. 1999 Applied Power Electronics Conference and Exposition*, vol. 1, pp. 135-138

[56]. *Multisim 2001 Component help*, Electronics Workbench, National Instruments Corporation.

[57]. J. B. Brown, C. E. Smith, "SPICE-A system analysis program?" in *Proc., 1993 Twenty-Fifth Southeastern System Theory Symposium*, pp.195-199.