10-1-1991

Data Structures and Sparsity in a Digital Simulation of an HVDC Link

Mark Timothy Luehmann
Purdue University

Data Structures and Sparsity in a Digital Simulation of an HVDC Link

Mark Timothy Luehmann

TR-EE 91-44
October 1991

DATA STRUCTURES AND SPARSITY IN A DIGITAL SIMULATION OF AN HVDC LINK

A Thesis

Submitted to the Faculty

of

Purdue University

by

Mark Timothy Luehmann

In Partial Fulfillmnt of the

Requirements for the Degree

of

Master of Science in Electrical Engineering

December 1991

to Sally Goldberg and Martin Stob

ACKNOWLEDGEMENTS

Although this thesis has my name on the title page, it should have a number of names there with mine. Were it not for the editing of Sally Goldberg, Martin Stob, John and Candace Suriano, Tom Roettger, and Professor Ong, this thesis would still be in the beginning stages. Also many thanks go to John Suriano for his help in drawing figures, Tom Roettger for his consultations, and Sally Goldberg for all her formatting wizardry. I would like to thank Professor Ong for his time and support in advising me on the development, programming, and editing of this work. I would also like to thank Professor Lin and Professor Ogborn for serving as members on my advisory committee.

Most of all I need to say thanks for the support given to me during the past two months from my parents and family, Sally Goldberg, and Martin Stob. Without their support, I would have scrapped this long ago. One last acknowledgement goes to God who has blessed me through this work and through the friends who have supported me.

TABLE OF CONTENTS

Page

Page

APPENDICES

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Luehmann, Mark Timothy. MSEE, Purdue University, December 1991. Data Structures and Sparcity in a Digital Simulation of an HVDC Link. Major Professor: Chee-Mun Ong.

An earlier algorithm for the detatiled simulation of the High Voltage Direct Current (HVDC) link using the tensor approach was modified to improve the runtime speed. The runtime, approximately eleven percent that of the original algorithm, was improved by employing sparsity techniques, restructuring the state matrix, and condensing subroutines. The readability of the code was accomplished by a modular programming approach. This research was done for a two-terminal simulation, but could be expanded and modified to include multi-terminal simulations.

CHAPTER 1. INTRODUCTION

This thesis documents a time efficient and modular approach to simulate a detailed model of an HVDC system. Time savings was achieved by employing sparsity techniques and by taking advantage of the structure of the state equation coefficient matrix. Modularity was employed to allow for better readability and for ease of future maintenance of the program. These implementations enhance the program's use as a tool in future research.

1.1 Brief History of HVDC Developments

Direct current was the basis of early discoveries and applications in electricity. Early scientists Galvani, Volta, Ohm, and Ampere worked with and described electricity in terms of direct current. The early applications of power transmission included those by Thomas Edison, who, using DC power, was the first in the world to build a central power station on Pearl Street in New York City. Two primary developments promoted the growth of Alternating Current transmission over DC transmission. The first of these was the induction machine which became a less expensive, reliable workhorse. The second the transformer. Although direct current applications were almost completely superceded by other forms of energy, it was still used in some applications, one of which was long distance power transmission. The use of DC power in transmission stemmed from

lower line losses. The early HVDC systems used DC generators, frequently hydro powered, connected in series to boost the voltage to a transmission level. In 1903, Peter Cooper Hewitt invented the mercury-arc rectifier thus enabling the use of AC generation for DC transmission. By the 1930's a control grid had been added to the arc valves, and the first HVDC line had been built by the New York Power and Light Corporation and General Electric. In the 1940's AC-DC links were being used in steel mills to convert power between 25Hz and 60Hz operating frequencies. In 1954, the first commercial HVDC line was built for 20 miles of underwater power transmission between the Swedish mainland and Gotland. By the mid-1980's, there were about 26 HVDC systems operating in the world with eight more being built. This growth in HVDC was due to the replacement of mercury-arc valves with much cheaper, more reliable, solid state thyristors in the 1970's[1]. The new thyristors, sometimes referred to as Silicon Controlled Rectifiers (SCR), made HVDC an economic feasibility for over-land applications longer than 400 miles and for underwater applications longer than 20 miles[2].

1.2 Simulation Techniques for the HVDC Systems

The designers of HVDC systems needed tools for the initial design and the analysis of the systems. The first method developed was a scaled down version of the actual system. Much information could be obtained, but the ability to modify the test system parameters was very difficult[3]. These scaled models were also expensive to build.

Another method of simulation, developed later, utilized the analog computer. This type of detailed simulation gave good results for investigations lasting up to about one second of real time [4]. As the analog simulations often did not employ a canonical representation of the system equations, it was cumbersome to change the structure of the system[3]. A variation of the analog simulation simulation was developed to modularize the system. Called a "parity simulation," it was configured to have topological parity with test system. With this technique, the HVDC system and the model were divided into common segments so that parameters could be more easily modified[5].

1.2.1 Digital Simulations

With the development of digital computers, new methods were developed to model HVDC systems. This exploration was done because digital computers offered the flexibility to simulate different system parameters. The digital simulations were developed for extensive evaluation of new and existing HVDC systems. Some of the applications of the digital simulation include:

- examination of system disburances[6],
- development of new control schemes[7,8,9],
- study of expanded multiterminal systems[10,11,12],
- minimization of transmission losses[13],
- enhancement of load studies[14], and
- examination of bulk transfer.

Depending on the study, different types of models may be employed. Some models were developed to study long term transients; other models were developed to examine short term transients. The accuracy of an average value

model is often sufficient for a long term transient study, whereas a detailed model provides a more refined view used for short term transient studies.

A particular representation of the system should be based on the type of study which must be done. There are models which vary between the extremes of expressing the entire HVDC system as a load and emulating the converter circuit in detail. The models could be divided into two subheadings - average models and detailed models. The average models vary in detail from a steady state load to the inclusion of the converter controls. Some of these models include:

- the steady-state load model[15]
- the varying load model[15]
- the load model with steady-state DC control[15]
- the quasi-steady-state model[16]
- the refined quasi-steady-state model[17]

More refined models use a combination of a dynamic control scheme and DC circuit response with the steady state converter model. These models are all simulated using an average value model of the converter. Another method is to model both the converter and DC system in detail. This type of model is needed to perform a variety of studies including those related to harmonics and short term transients. A detailed model of the HVDC link presents difficulties in simulating the changing topologies of the converters.

1.2.2 Detailed Simulation

Detailed digital simulations of HVDC systems require a close tracking of the topology of each converter in the system so the describing differential equations correctly reflect the converter topology. The formation of the correct set of

differential equations presents the primary difficulty in this approach.

The formation of the converter state equations can be done in a number of ways. One method is to predict every converter topology for all conditions[18,19]. The state equations which describe each of the topologies are stored in memory and implemented during the simulation as required. This method is costly from a memory point of view because of the large number of possible converter topologies including those which reflect abnormal conditions imposed on the converter. Although ample memory is available in disk form, more time is required to access the information from disk than from random access memory (RAM). Thus, the large number of circuit topologies which must be saved may also cause slower run times due to the accessing time of the information.

Another method of implementation is to use a fixed topology method. When a thyristor is on, the resistance of that thyristor is very low; when a thyristor is off, the resistance of that thyristor is very high. This method is used in many applications such as SPICE and GRAYCE[20]. It is applicable for cases in which a circuit topology changes only slightly or when the simulation is intended for general circuit applications. This method of simulating switching does not work well when there is too great a difference between the on and off resistances used for the thyristors. If the magnitude of the difference becomes too large, the state matrix which describes the circuit becomes stiff. The mathematical model may not be numerically stable, or an implicit integration routine may be necessary to retain a sufficiently large region of stability[20].

Another approach is to use a general method to form a set of independent equations that accurately reflect the instantaneous topology of the system. In this approach a new set of converter state equations would be developed each time the states of the thyristors change. There have been multiple methods developed to form the set of describing equations. One method uses the formation of a tree based on a predetermined hierarchy. In this method, zero-current sources are substituted for non-conducting thyristors, and zero-voltage sources are substituted for conducting thyristors [21].

A similar approach implements Kron reduction to simulate the model[12]. The original set of converter equations is set up as if all of the thyristors were conducting. This method reduces the describing equations to only the independent equations for the present converter topology based on knowledge of the thyristor states. This approach employs a general method of obtaining an independent set of equations not restricted to predetermined converter topologies[22]. This approach of modelling the converter was selected for this project.

Detailed models have been used in the development of other digital simulations. Hybrid simulations use the average value models to simulate the system for long term DC transients and the detailed model for the short term, more immediate transients. Recently, parallel processing ideas have been used to digitally simulate the HVDC system in real-time[3].

1.3 Motivation

A flexible, time efficient HVDC simulation tool was desirable for reducing analysis time. A working program was developed in the mid 1980's that used the tensor analysis method to simulate an HVDC system[12]. It was subsequently expanded to be compatible with the Electromagnetic Transient Program (EMPT)[10,23]. Using the methods developed in these efforts, a more modular approach was sought which would also be more time efficient.

If only the converter were needed to accurately model the HVDC system, the program would run very quickly since only two or three state equations are required to fully model the converter. In an accurate representation of the HVDC system, the surrounding AC system must also be modeled. This includes the Thevenin representation of the AC system and the filters used to suppress the harmonics produced by the converters. Typically for a 12-pulse HVDC system, each of the three phases requires an eleventh harmonic filter, a thirteenth harmonic filter, a high pass filter, a capacitor bank, and a Thevenin line impedance. Each of the filters requires two state equations; the capacitor bank and the Thevenin impedance require one state equation for each. Thus eight state equations needed to describe each phase of the AC system. For the entire three phase system 24 state equations are needed.

For the two-terminal HVDC link, there are 48 state equations needed to describe the AC system voltages and currents, one for the DC line voltage, and a variable number (usually two or three) to describe each converter. Therefore,

between 53 and 55 state equations are required to model a 12-pulse HVDC link. The simulation of a model with this many state equations is very time-consuming and difficult to manage.

The algorithm can also be difficult to program in a fashion which would provide flexibility to subsequent users. Therefore, a modular approach was sought which would allow flexibility for users to modify any part of the system model. This flexibility would provide the researcher a more useful tool.

1.4 Objective

The objective of this thesis is to modify the previous algorithm to make it a more efficient and useful tool. The usefulness of the program can be enhanced by adhering to a modular programming approach to ensure ease of reading and maintaining the code. The efficiency of the program could be enhanced through the restructuring of the state equation coefficient matrix and employing sparsity techniques. These objectives would be best met in using a language that supports the use of records and structured variable declarations. Therefore, the final choice was to write a new program in C to achieve a more efficient and useful tool.

1.5 Outline of Thesis

Chapter 2 of this thesis reviews and explains the mathematical development of both the AC state equations and the set of converter loop equations. Chapter 2 also explains the extension of the tensor mathematical development into a modular algorithmic approach. The results, given in Chapter 3, shows time

savings over its predecessor, the validity of the model, and a transient response of the model. Chapter 4 gives a summary of what was done and recommendations for future work.

CHAPTER 2. MATHEMATICAL AND STRUCTURAL DEVELOPMENT

2.1 Introduction

An explanation of the system structure, a mathematical development of the simulation, and an overview of the algorithm is presented in this chapter. It is first necessary to define the system which is being simulated. Figure 2.1 shows a one-line diagram of the complete two-terminal system which contains two converters, a DC line, and the AC networks connected to each of the converters. The AC network consists of a Thevenin equivalent source and line impedance, a capacitor bank, and three sets of filters. Also included are two three phase transformers: one is Y-Y connected, and the other is Y-Δ connected so that there is a 30 degree phase difference in the AC voltages to the top and bottom bridges of each converter. Diagrams of the converter are shown in Figure 2.2 and Figure 2.3.

2.1.1 Thyristor Ordering

There are two different orderings of the thyristors used in the program. The first, shown in Figure 2.2, is the firing order; the second, shown in Figure 2.3, is the state order. The purpose of the state order is to provide a numbering scheme to label the thyristors in the program. The thyristor voltage and current

variables are indexed using the state order, whereas the ring counter uses the firing order.

Figure 2.2 has the thyristors labeled in firing order. The firing order is based on the commutation voltages. The commutation voltage is the voltage that transfers the current of the thyristor which is presently conducting to the next thyristor in the firing order. The commutation voltage for thyristor 2 in Figure 2.3 is the voltage between the A phase and the B phase sources. When the commutation voltage becomes positive, the next thyristor may start to conduct if there is a gate voltage provided from the controller.

2.1.2 Detailed Converter Modelling

The differential equations which describe the HVDC link could be set up in a variety of ways. One way would be to have a separate set of equations for each of the possible converter operating conditions. It would be difficult to define all possible topologies of the converter including all possible fault conditions. [18,19]

Another method is to use a SPICE-like approach; each thyristor is assigned multiple values of resistance. When the thyristor is conducting, it has a small impedance. When the thyristor is not conducting, it has a large impedance. Because of the large difference in the order of magnitude between the on and off impedances, the resulting numerical problem is stiff. The integration method used must have a large region of absolute stability. Explicit methods are not suitable for stiff problems because their region of absolute stability is usually small. [24]

A different method, based on tensor analysis, was utilized. A connection matrix, based on thyristor states, relates a comprehensive set of converter state equations to an independent set of equations that defines the present converter topology. This method affords a more general solution than the first approach without the numerical stiffness found in the second approach. The variables and development of a structured mathematical solution is done in the following sections[12,22].

2.2 Mathematical Development

This section provides the mathematical development of the algorithm implemented and the important variables being used. Figure 2.1 shows the detailed model of the DC link which is being simulated. The HVDC system can be divided into three parts: the AC system, the DC system, and the converter. The AC system voltages and currents require 24 state equations. Since the AC system components do not physically change over the course of the simulation, the state equations remain fixed. This is not true in the general case because it is possible to switch on a new capacitor bank or to adjust the taps of the transformer for added control. In this work the control for changing the taps of the transformer has not been incorporated because the simulation runs of interest are short relative to the delay in the tap-changing control. The converters are slightly more difficult to model in time because the number and the coefficients of the related equations change each time a thyristor changes states. Although the DC line voltage equation does not change, the location of its state equation

coefficients within the A matrix vary as the converter loop equation variables change.

The mathematical steps used to simulate one time step of the model are illustrated in the flowchart in Figure 2.4.

2.2.1 The Converter Equations

Although it is not possible to separate the converter equations from the rest of the system equations, the following section focuses primarily on building and solving the converter equations. The converter equations are dependent on the AC system equations. The link currents are updated based on the latest values of the AC currents. After the development of the converter equations, the AC variables are discussed.

2.2.1.1 Loop Equations

In a tensor approach, the initial set of state equations is developed assuming all thyristors were conducting. Figure 2.5 displays the tree and link diagram of a 12 pulse converter. Nine branches, shown as solid lines in the figure, were arbitrarily chosen to be the tree. Based on this tree, ten loop equations fully describe the converter. These equations can be written in the following form:

$$V_{loop} = Z_{loop} I_{loop} \tag{2.1}$$

where

$$V_{loop} = \begin{bmatrix} e^{Y}_{ac} \\ e^{Y}_{bc} \\ e^{Y}_{ca} \\ e^{Y}_{cb} \\ -V_d \\ .... \\ e^{\Delta}_{ca} \\ e^{\Delta}_{bc} \\ e^{\Delta}_{ca} \\ -e^{\Delta}_{bc} \\ e^{\Delta}_{ca} + e^{\Delta}_{bc} + e^{\Delta}_{ca} \end{bmatrix} = \begin{bmatrix} V^{Y}_{loop} \\ .... \\ V^{\Delta}_{loop} \end{bmatrix}, \quad I_{loop} = \begin{bmatrix} i_{link1} \\ i_{link2} \\ i_{link3} \\ i_{link4} \\ i_{link5} \\ .... \\ i_{link6} \\ i_{link7} \\ i_{link8} \\ i_{link9} \\ i_{link10} \end{bmatrix} = \begin{bmatrix} I^{Y}_{loop} \\ .... \\ I^{\Delta}_{loop} \end{bmatrix} \tag{2.2}$$

and

$$Z_{loop} = \begin{bmatrix} Z^{Y}_{loop} & Z_c \\ Z^{t}_{c} & Z^{\Delta}_{loop} \end{bmatrix} \tag{2.3}$$

The superscript Y denotes the Y-Y portion of the converter, thyristors one through six in state order. The superscript Δ denotes the Y-Δ portion of the converter, thyristors seven through twelve in state order.

I_{loop} is the set of all the loop currents which are shown as links in Figure 2.5. Z_{loop} is the impedance matrix which relates the set of voltages in V_{loop} to the loop currents. The actual values for the Z_{loop} matrix along with an example derivation can be found in Appendix B. V_{loop} is the set of voltages which are used to determine the link currents. In this notation, $e_{ac} = e_a - e_c$.

2.2.1.2 The Connection Matrix

The connection matrix is developed to reduce the original set of loop equations to only those applicable to the present converter topology [22]. It

relates the loop currents (the set of currents from the links shown in Figure 2.5) to the new set describing the present converter topology as follows:

$$I_{loop} = C_n I_n \tag{2.4}$$

where I_{loop} is the set of loop currents, I_n is the reducted set of currents of the present converter topology, and C_n is the connection matrix. A detailed description of the logic and the formation of the connection matrix is given in Appendix C.

2.2.1.3 Reduction of the Loop Equations

Substituting Equation 2.4 into Equation 2.1 gives a new expression for V_{loop}, that is

$$V_{loop} = Z_{loop} C_n I_n. \tag{2.5}$$

Pre-multiplying both sides by the transpose of C_n gives

$$C_n^t V_{loop} = C_n^t Z_{loop} C_n I_n. \tag{2.6}$$

Defining

$$V_n = C_n^t V_{loop} \tag{2.7}$$

and

$$Z_n = C_n^t Z_{loop} C_n, \tag{2.8}$$

the final result is

$$V_n = Z_n I_n. \tag{2.9}$$

The connection matrix is a 10×2 matrix when there is no commutation between thyristors. Under this condition, the result will be a 2×2 Z_n matrix. For each

commutation occurring, one additional column exists in the connection matrix in order to form a complete set of loop equations. Therefore, under normal conditions with overlap of less than 60 electrical degrees, the converter can be fully described using no more than three of the loop equations.

2.2.1.4 The Converter State Equations

The next step is to manipulate Equation 2.9 into state equation form. Previously, the variable Z has been used to represent the impedance. Since this is a dynamic simulation, the original values of R and L must be used. Replacing Z_n by its operational form,

$$Z_n = R_n + p\,L_n.$$

Using this substitution in Equation 2.9 gives

$$V_n = (R_n + p\,L_n)I_n. \tag{2.10}$$

Expanded, this is

$$V_n = R_n I_n + p\,L_n I_n. \tag{2.11}$$

Putting the differential on the left side gives

$$p\,L_n I_n = V_n - R_n I_n. \tag{2.12}$$

Finally, pre-multiplying both sides by L_n yields the final result.

$$p\,I_n = L_n^{-1}V_n - L_n^{-1}R_n I_n \tag{2.13}$$

The loop equations are now in state equation form. Since they are in the same form as the AC state equations their coefficients can be added to the state matrix.

2.2.1.5 System State Equation Integration

After the A matrix has been formed, the integration is performed. The integration technique employed is the fourth order Runge-Kutta method. By using the tensor approach, the need for an implicit integration was eliminated because the numerical problem is not stiff.

An implicit method requires an inversion of the A matrix to obtain a value for the variable at the next time step. An explicit method predicts the value of the variable at the next step based on previous values. By employing the explicit integration method, computation time can be saved without a loss of accuracy.

2.2.1.6 Loop Currents

The relationship between I_{loop} and I_n was defined in Equation 2.4. The system state variables were updated during the integration. The state variables solved for in the converter state equations were the values of I_n. So by the original definition the loop currents can be determined using,

$$I_{loop} = C_n I_n. \tag{2.4}$$

2.2.1.7 Loop Voltages

To update the loop voltages, another transformation matrix must be developed. The definition of V_{loop} given in Equation 2.2 was

$$V_{loop} = \begin{bmatrix} e^{Y}_{ac} \\ e^{Y}_{bc} \\ e^{Y}_{ca} \\ e^{Y}_{cb} \\ -V_d \\ \cdots\cdots \\ e^{\Delta}_{ca} \\ e^{\Delta}_{bc} \\ e^{\Delta}_{ca} \\ -e^{\Delta}_{bc} \\ e^{\Delta}_{ca} + e^{\Delta}_{bc} + e^{\Delta}_{ca} \end{bmatrix}$$

The primary voltages of the transformers are found as part of the AC state equation solution integration. Multiplying the primary voltage by the appropriate turns ratio, t, gives the following set of loop voltages. For example, the first equation would be

$$e^{Y}_{ac} = e^{Y}_{a,\text{secondary}} - e^{Y}_{c,\text{secondary}} \tag{2.14}$$

$$e^{Y}_{ac} = t * (e^{Y}_{a,\text{primary}} - e^{Y}_{c,\text{primary}}).$$

For the same level of secondary voltage as the Y-Y transformer, the turns ratio of the Y-Δ transformer must be $\sqrt{3}\,t$. This yields:

$$e^{\Delta}_{ac} = e^{Y}_{a,\text{secondary}} - e^{Y}_{c,\text{secondary}} \tag{2.15}$$

$$e^{Y}_{ac} = t\sqrt{3}\ (e^{\Delta}_{a,\text{primary}} - e^{Y}_{c,\text{primary}}).$$

Extending this for all the loop voltages,

$$\begin{bmatrix} e_{ac}^{Y} \\ e_{bc}^{Y} \\ e_{ca}^{Y} \\ e_{cb}^{Y} \\ -V_d \\ \cdots\cdot \\ e_{ca}^{\Delta} \\ e_{bc}^{\Delta} \\ e_{ca}^{\Delta} \\ -e_{bc}^{\Delta} \\ e_{ca}^{\Delta} + e_{bc}^{\Delta} + e_{ca}^{\Delta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \\ \cdots\cdot & \cdots\cdot & \cdots\cdot \\ 0 & 0 & -\sqrt{3} \\ 0 & \sqrt{3} & 0 \\ 0 & 0 & \sqrt{3} \\ 0 & -\sqrt{3} & 0 \\ \sqrt{3} & \sqrt{3} & \sqrt{3} \end{bmatrix} \begin{bmatrix} V_{a,primary} \\ V_{b,primary} \\ V_{c,primary} \end{bmatrix} \qquad (2.16)$$

$$V_{loop} = T_v V_{primary}.$$

2.2.1.8 Thyristor Voltages

Since the assumption is that the thyristors are ideal, the voltage across those conducting is zero. The other thyristor voltages must be calculated to determine the state of the thyristor at the next time step. The simulation of the converters has been built on Equation 2.1.

$$V_{loop} = Z_{loop} I_{loop}$$

The loop equations were derived assuming that the thyristors were all conducting. If this were true, the voltage across all the thyristors would be zero; however, all of the thyristors do not conduct at the same time. A set of voltage equations must be developed to account for the voltage across the thyristors in the loops defined in Equation 2.1. This difference due to the voltage across the thyristors is defined as V_x.

$$V_x = V_{loop} - Z_{loop} I_{loop} \tag{2.17}$$

Now, expressing V_x and V_{loop} in terms of the thyristor voltages gives

$$V_x = \begin{bmatrix} v_1^Y - v_3^Y \\ v_2^Y - v_3^Y \\ v_4^Y - v_6^Y \\ v_5^Y - v_6^Y \\ v_3^Y + v_6^Y \\ \cdots\cdots \\ v_1^\Delta - v_3^\Delta \\ v_2^\Delta - v_3^\Delta \\ v_4^\Delta - v_6^\Delta \\ v_5^\Delta - v_6^\Delta \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ v_3^\Delta + v_6^\Delta \\ \cdots\cdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.18}$$

Based on Equation 2.18, a general incidence matrix can be developed to relate all the thyristor voltages to the voltage difference of the loop equations, V_x. Transcribing the coefficients from Equation 2.18 into matrix form yields:

$$D = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.19}$$

Since it is the set of thyristor voltages being sought, $V_{thyristor}$ must be expressed in terms of V_x. To formulate this expression, it is necessary to reduce D.

Each column of D corresponds to a thyristor and each thyristor that is conducting, has a voltage of zero across it. Therefore, since the columns of D

corresponding to conducting thyristors will always be multiplied by zero, they can be eliminated.

Each row of the D matrix corresponds to a loop equation. Equation 2.1 was developed using the assumption that all thyristors were conducting. V_x was developed only because some of the thyristors were not conducting. Therefore, since link currents corresponding to a conducting thyristor have V_x elements equal to zero, these rows can also be eliminated. So the final result will be:

$$V_{x,n} = D_n V_{thyristor,n}. \tag{2.20}$$

After the row and column eliminations, D becomes a square matrix which can be inverted. Now the voltages of the thyristors not conducting can be determined using

$$V_{thyristor,n} = D_n^{-1} V_{x,n}. \tag{2.21}$$

2.2.1.9 Rectifier Control

The subroutine for calculating rectifier control variables is divided into two sections. The first contains error detection and amplification, and the second contains thyristor firing control. The control scheme is illustrated in Figure 2.6.

The first section of the subroutine employs a PI controller to produce a normalized output signal based on the error between the actual DC current and its reference value. The signal is processed through a proportional gain and an integral gain. The amplified errors are summed together and sent to a limiter

which reduces system overcompensation. This controller, illustrated in Figure 2.7, is used as an input to the second part of the rectifier control.

The second part of the subroutine uses a ramp function to determine the firing interval. Whenever the fixed-slope ramp function crosses the control level, the ramp function is reset to zero. Figure 2.8 illustrates what this looks like over time; it is a sawtooth waveform. The x-axis is time, and the y-axis represents the number of electrical degrees since the last thyristor was fired. When the converter is operating under steady state conditions, a thyristor should be fired every 30 electrical degrees because each of the twelve thyristors should fire once every cycle, or once every 360 electrical degrees. The dashed line running parallel to the x-axis is the steady state firing angle, or $\frac{\pi}{6}$ radians.

The current in the DC line depends on the length of time the thyristors conduct. When the delay angle is longer, the thyristor conducts for less time; when the thyristor conducts for less time, less current flows in the DC line. Therefore, if the current were to be higher than the reference current at the last time step, the controller produces an error signal which causes the firing of the next thyristor to be delayed longer. Conversely, if the current was low at the last time step, then the next thyristor is turned on earlier.

The purpose of the output error signal of the PI controller is to adjust the firing time. If the actual DC current from the rectifier is high, then the output from the PI controller is negative.

$$\text{Error}=\frac{(I_{DC}^{reference}-I_{DC}^{actual})}{I_{DC}^{reference}} \tag{2.22}$$

Since the current is high, the thyristor firing angle must be increased. The angle can be increased by subtracting the output of the PI controller from the steady state, $\frac{\pi}{6}$ firing angle. This moves the point of intersection between the ramp and the firing level. It is at the new point of intersection that the next thyristor will be fired. The curved, dashed line shown in Figure 2.9 is the difference between the steady state firing angle and the error out of the PI controller.

The two horizontal, dotted lines in Figure 2.9 show the maximum and minimum firing angles. If the curved line dips below the minimum value, the next thyristor will not be fired until the ramp has intersected the minimum firing angle line. If the curved line exceeds the maximum value, the next thyristor will automatically fire when the ramp reaches the maximum firing angle line.

The final step of the routine advances the ring counter in a circular queue. A set of four consecutive 1's, or "on" values, are moved in firing order as illustrated in Figure 2.10. The 1's correspond to the conducting thyristors, and the 0's correspond to the non-conducting thyristors.

2.2.1.10 Inverter Control

The inverter control subroutine is very similar to that of the rectifier control subroutine. (See Figure 2.11) The inverter subroutine uses a proportional controller to amplify the error signal. The error signal is the difference in the actual extinction angle, γ_{act}, and the reference extinction angle, γ_{ref}.

Before the error can be calculated, the actual extinction angle must be found. This calculation requires that the commutation voltages be monitored for downward zero crossings. When one is detected, the time the last thyristor was turned off is subtracted from the current time. By converting this time to radians, γ_{act} can be determined. The error is calculated, amplified by the proportional gain, and sent to the second part of the procedure which determines when the next thyristor should be fired.

The next thyristor firing is determined by using an Inter-firing-period (IFP) [25]. It is very similar to the ramp/sawtooth type of method used in the rectifier control. The IFP is the time between firings of the thyristor. Under steady state conditions the IFP is 30 electrical degrees, or $\frac{\pi}{6}$ radians.

The output from the proportional controller is added to the IFP. When the current time reaches the sum of the IFP and the error time, the next thyristor is ready to be fired. The ring counter is then advanced. The firing order of the thyristors in the inverter is exactly the same as that for the rectifier. (See Figure 2.10)

2.2.1.11 Thyristor States

Each thyristor state must be checked every time-step to determine its status. The first step checks for thyristors that are turning off. This requires the values of the thyristor currents. If the thyristor state is on and its current has gone negative since the previous time step, , the thyristor should be turned off.

The thyristor currents must be calculated based on the updated independent link currents. The following equation expresses this relationship.

$$\begin{bmatrix} I_{thy1} \\ I_{thy2} \\ I_{thy3} \\ I_{thy4} \\ I_{thy5} \\ I_{thy6} \\ I_{thy7} \\ I_{thy8} \\ I_{thy9} \\ I_{thy10} \\ I_{thy11} \\ I_{thy12} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 \end{bmatrix} \begin{bmatrix} I_{link1} \\ I_{link2} \\ I_{link3} \\ I_{link4} \\ I_{link5} \\ I_{link6} \\ I_{link7} \\ I_{link8} \\ I_{link9} \\ I_{link10} \end{bmatrix} \tag{2.23}$$

The second step is to check for thyristors which are presently off, but should be turned on. Turning a thyristor on takes no mathematical calculations. If the ring counter element is on, then the corresponding thyristor is turned on.

If there were no changes in the thyristor states, then the calculations needed for this time step are complete. The next step in the solution is to integrate the state equations for the next time step. If there was a change in the thyristor states, then the state equations no longer reflect the current converter circuit, and the converter state equations must be rebuilt. This rebuilding is accomplished using the connection matrix based on the new set of thyristor states to reduce the loop equations.

It is also necessary to revise the state variables and to update the loop currents. After new converter state equations have been calculated and the state variables updated, the process begins the next time step with another integration.

2.2.2 AC and DC System Equations

From a physical perspective, the AC system modelled consists of a Thevenin equivalent of the AC system looking back into the system from the converter transformer primaries, and the AC filters. The AC filters in parallel with the converter are tuned to remove the 11th and 13th harmonics as well as other high frequency harmonics.

Examining the circuit from a mathematical perspective, both loop and nodal equations must be developed to form an independent set of equations which describe the AC voltages and currents. The only known voltage in the AC system is the Thevenin voltage, which is the forcing function of the system. All other values of current and voltage can be found through the set of state equations. The following are the differential equations needed to simulate the AC side of the converter. Only the A phase equations of these equations are only shown. The equations for the other two phases are similar.

AC Phase Current in the AC equivalent:

$$\frac{d}{dt}i_{ac} = (\frac{1}{L_{th}})(e_{th} - R_{th}\,i_{th} - v_{ap}). \tag{2.24}$$

AC Phase Voltage at the Primary of the Converter Transformer:

$$\frac{d}{dt}v_{ac} = (\frac{1}{C_c})[i_{ac} - i_{f11} - i_{f13} - i_{ac,conv} - i_{hp} - (\frac{1}{R_{hp}})(v_{ac} - v_{hp})]. \tag{2.25}$$

11th Harmonic Filter Current:

$$\frac{d}{dt}i_{f11} = (\frac{1}{L_{f11}})(v_{ac} - v_{c11} - R_{f11}\,i_{f11}). \tag{2.26}$$

11th Harmonic Filter Voltage:

$$\frac{d}{dt}v_{c11} = (\frac{1}{C_{f11}})(i_{f11}). \tag{2.28}$$

13th Harmonic Filter Current:

$$\frac{d}{dt}i_{f13} = (\frac{1}{L_{f13}})(v_{ac}-v_{c13}-R_{f13}i_{f13}). \tag{2.27}$$

13th Harmonic Filter Voltage:

$$\frac{d}{dt}v_{c13} = (\frac{1}{C_{f13}})(i_{f13}). \tag{2.29}$$

High Pass Filter Current:

$$\frac{d}{dt}i_{hp} = (\frac{1}{L_{hp}R_{hp}})(v_{ac}-v_{hp}). \tag{2.30}$$

High Pass Filter Voltage:

$$\frac{d}{dt}v_{hp} = (\frac{1}{C_{hp}})i_{hp} + (\frac{1}{R_{hp}C_{hp}})(v_{ac}-v_{hp}). \tag{2.31}$$

where

L_{th} is the Thevenin inductance of the AC system,

R_{th} is the Thevenin resistance of the AC system,

e_{th} is the Thevenin voltage of the AC system,

i_{th} is the Thevenin current of the AC system,

v_{ap} is the phase voltage of the capacitor bank,

C_c is the capacitance of the capacitor bank,

i_{f11} is the current in the eleventh harmonic filter

i_{f13} is the current in the thirteenth harmonic filter

i_{hp} is the current in the inductor of the high pass filter,

v_{c11} is the capacitor voltage in the eleventh harmonic filter,

v_{c13} is the capacitor voltage in the thirteenth harmonic filter,

v_{hp} is the voltage across the capacitor of the high pass filter,

L_{f11} is the inductance of the eleventh harmonic filter,

L_{f13} is the inductance of the thirteenth harmonic filter,

L_{hp} is the inductance of the high pass filter,

R_{f11} is the resistance of the eleventh harmonic filter,

R_{f13} is the resistance of the thirteenth harmonic filter,

R_{hp} is the resistance of the high pass filter,

C_{f11} is the capacitance of the eleventh harmonic filter,

C_{f13} is the capacitance of the thirteenth harmonic filter,

C_{hp} is the capacitance of the high pass filter,

There is also a state equation to describe the DC line voltage:

$$\frac{d}{dt}V_{dc} = \frac{1}{C_{dc}}(I_{dc,rect} - I_{dc,inv}). \tag{2.32}$$

The DC network is represented by a "T" network with a shunt capacitor at the center to develop the DC line voltage. This DC line voltage equation is dependent on the DC line currents. The line currents are part of the converter equations. Since the indices of the DC line currents sometimes change when the set of conducting thyristors change, the voltage equation must be rebuilt when the converter state equations are rebuilt. The values of the coefficients will not change, only their locations in the A matrix.

2.3 The State Equation Coefficient Matrix

The following sections discuss the structure of the state matrix and its implementation in terms of sparsity and data structure. Each set of converter equations is developed separately and then added to the set of AC state equations. On examining the contents of the A matrix, the sparsity of the matrix is apparent. If the sparsity of the matrix is to be exploited, a new means of storing the elements of the array must be developed. Conceptually, it can still be thought of as a two dimensional array, although, it is not stored in a two dimensional array.

2.3.1 Structure of the State Matrix

Figure 2.12 shows a broad view of the layout. The first 24 rows are the state equations describing the AC system associated with the rectifier. The next 24 rows are the state equations describing the AC system associated with the inverter. The 49th row is the equation describing the DC line voltage. The next four rows are reserved for the rectifier state equations. The last four rows are reserved for the inverter state equations. Under normal conditions, a maximum of three of the rows are used for the rectifier and three for the inverter.

Another way to separate the A matrix, shown in Figure 2.13, is to divide it into blocks. Row one is the A phase AC current at the converter. Column one is multiplied by the A phase AC current at the last time step. The A matrix can be viewed as a grid in which blocks correspond to physical groups. For example, the upper left 24×24 block corresponds to the AC system connected to the rectifier.

The 24×24 block directly to the right and the 24×24 block directly below the upper left block are filled with zeros because there is no AC interconnection between the two AC systems, only through the DC line.

By looking at the matrix in grid form, it is possible to categorize the blocks into two groups. The first group consists of those blocks which will not change over the course of the simulation. In this case it is the upper left 49×49 block. The second group would be those blocks which change over the course of the simulation. This includes the rest of the matrix which is the set of rows and columns corresponding to the converter equation variables. The second group could be broken down further between converters. In the two terminal case, it would be broken into two parts: the rectifier and the inverter.

Table 2.1 Row Description of A Matrix

Rows	Description
Rectifier AC equations:	
1-3	A,B,C phase AC currents at the converter
4-6	A,B,C phase AC voltages at the converter
7-9	A,B,C phase AC voltages across the 11th harmonic filter
10-12	A,B,C phase AC voltages across the 13th harmonic filter
13-15	A,B,C phase AC voltages across the high pass filter
16-18	A,B,C phase AC currents through the 11th harmonic filter
19-21	A,B,C phase AC currents through the 13th harmonic filter
22-24	A,B,C phase AC currents through the high pass filter
Inverter AC equations:	
25-48	Same as rows 1-24
DC line equation:	
49	The voltage across the DC line capacitance
Rectifier equations:	
50-53	Reduced loop equations for rectifier
Inverter equations:	
50-53	Reduced loop equations for inverter

is well over 11,000 multiplications. The time step being used is one quarter of an electrical degree. This translates one second of simulated time into 86,400 iterations. One multiplication on a Sun IPC (Sparc I) takes approximately 2.1 microseconds, which would yield a savings of almost 17 minutes of CPU time for a half second of simulated time.

This discussion emphasizes that there should be a way of multiplying only the non-zero elements in the array, while ignoring the zero elements.

2.3.3 Data Structure of the State Matrix

Since sparsity theoretically yields a valuable time savings, a method must be developed to multiply only the non-zero elements. Since this is a specific application and the structure of the system matrix is known, only the non-zero elements in the matrix need to be stored. This would require a different structure than a two-dimensional array.

This was the approach that was taken in this work. A one dimensional array of records was used to store the non-zero coefficients. The record has the structure shown in Figure 2.15. Since only non-zero elements are stored, there is no need to check the matrix for non-zero elements. All of the necessary information is in the record.

Since the A matrix was divided into three sections, the multiplication was also divided into three sections. This division was used to recalculate the converter state equation coefficients and load them into the A matrix without affecting the unchanged coefficients. Although this three-section approach is

efficient in terms of CPU time, it does not lend itself well to handling an unknown number of converters.

The Runge-Kutta integration technique requires four multiplications of the A matrix by four one-dimensional arrays. The multiplication begins by initializing the resultant matrix. As each record is read from the array, the value of the coefficient is multiplied by the "column of A" column element in the one dimensional array. That product is added to the "row of A" column in the resultant matrix.

2.4 Algorithmic Overview

The algorithm used to simulate the two-terminal HVDC system was developed using a modular approach. The algorithm was divided into computer tasks. Each major task, such as the integration and the control, was implemented in a separate subroutine. Other aspects of modularity were also investigated.

Modularity by function was incorporated when possible. Function is defined as a physical process of the converter or AC system. For example, the control of the physical system can be replaced or modified while the rest of the physical system remains unchanged. Therefore, it would be beneficial to have the ability to replace or to modify only the control routine for the rectifier or inverter bridge. This type of modularity allows for a straight forward substitution of subroutines to incorporate a change in the physical system.

Another form of modularity was in the flow of the program. This form of modularity locates related subroutines or lines of code together in the same section of the program. It also ensures that independent tasks are located in separate sections. In order to make the algorithm more readable and maintainable, related pieces of code were kept physically close together within the algorithm. The algorithm, broken down by flow, is shown in Figure 2.16. The flow of the program is separated into six large tasks.

2.4.1 System Parameter Initialization

The flow of the program begins with the initialization of the system parameters, parameters unique to the particular HVDC system being modelled. The parameters include voltage levels, resistances, capacitances, and inductances, which are used by the algorithm. It is during this stage that the constant matrices and variables are initialized. For example, the full set of rectifier and converter loop equations are formed. The transformation matrices used to calculate thyristor voltages and loop voltages are also formed.

2.4.2 Operating Condition Initialization

The second stage in the program flow is the initialization of the operating conditions. Contained in the operating conditions are the initial states of the thyristors, values of the control variables, values of the state variables, filter values, and other variables necessary for the start of the simulation. The second stage initializes the state variables using the physical values loaded in the first stage.

There are two options given in the initialization of the operating conditions. The first option is to begin the simulation using a pre-defined set of thyristor states. The initial operating condition variables are described in Appendix A. The second option is to continue from a previous run. In this case, the restart conditions are read from a recovery file that has been created at the end of a previous simulation. It contains all the information necessary to begin the simulation from the last time step of that previous simulation. All these variables are listed in Appendix A.

2.4.3 Converter State Equation Initialization

After the operating conditions and AC state equations are initialized, the converter equations must be developed. This stage of the program flow initializes the matrices which describe the topology, reduces the loop equations, and manipulates the reduced set of loop equations into state equation form. These tasks are based entirely on the states of the thyristors. Formation of the converter state equations must be repeated each time the topology of the converter changes.

2.4.4 Main Loop

The fourth stage of the program flow is the main loop of the simulation. It contains the integration of the state equations, the updating of the loop currents, loop source voltages, and thyristor voltages, and the control subroutines for the converters. It is the most time consuming part of the simulation because it is repeated every time step.

2.4.5 Thyristor State Changes

When a thyristor state changes, the reduced set of loop equations is no longer valid. The new converter topology dictates that the converter state equations be reformed. This is done by reforming the connection matrix, reducing the loop equations, and manipulating the reduced loop equations into state equation form. Re-forming the converter state equations is separated from the main loop because it is necessary to do this only after a change in the converter topology. Computing time would be wasted if the state equations were recalculated every time step.

2.4.6 Exiting the Program

The user has three choices when exiting the program:

- to continue the simulation,
- to end the simulation after saving the final operating conditions as a starting point for another simulation, or
- to end the simulation without saving the final operating conditions.

These six stages make up the flow of the algorithm. The first three are related to the initialization, the fourth and fifth comprise the main loop, and the sixth allows the user to exit the simulation.

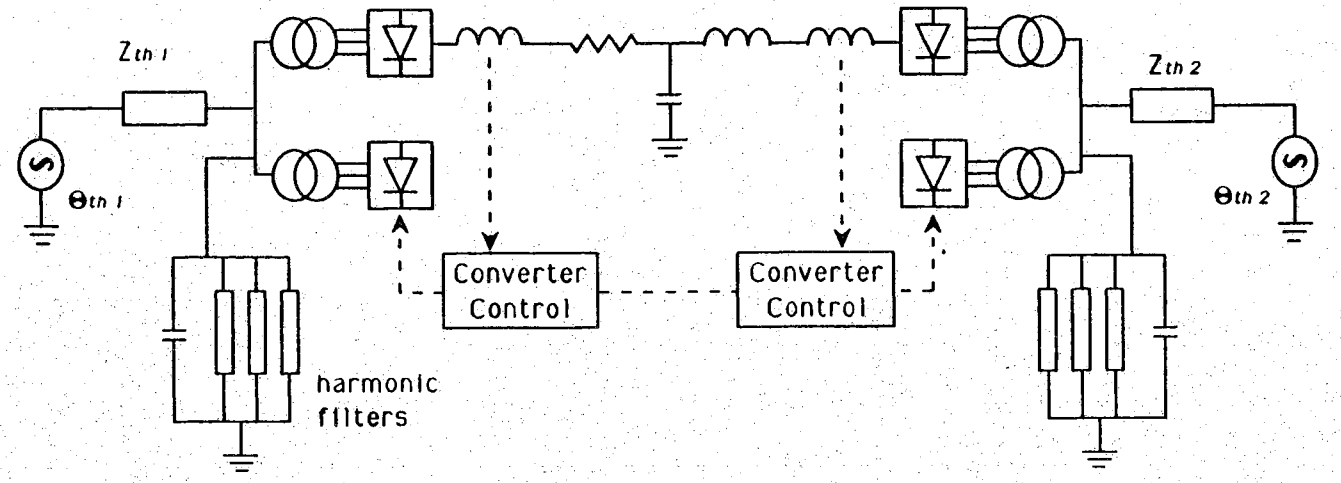


Figure 2.1 Detailed Model of a DC Link

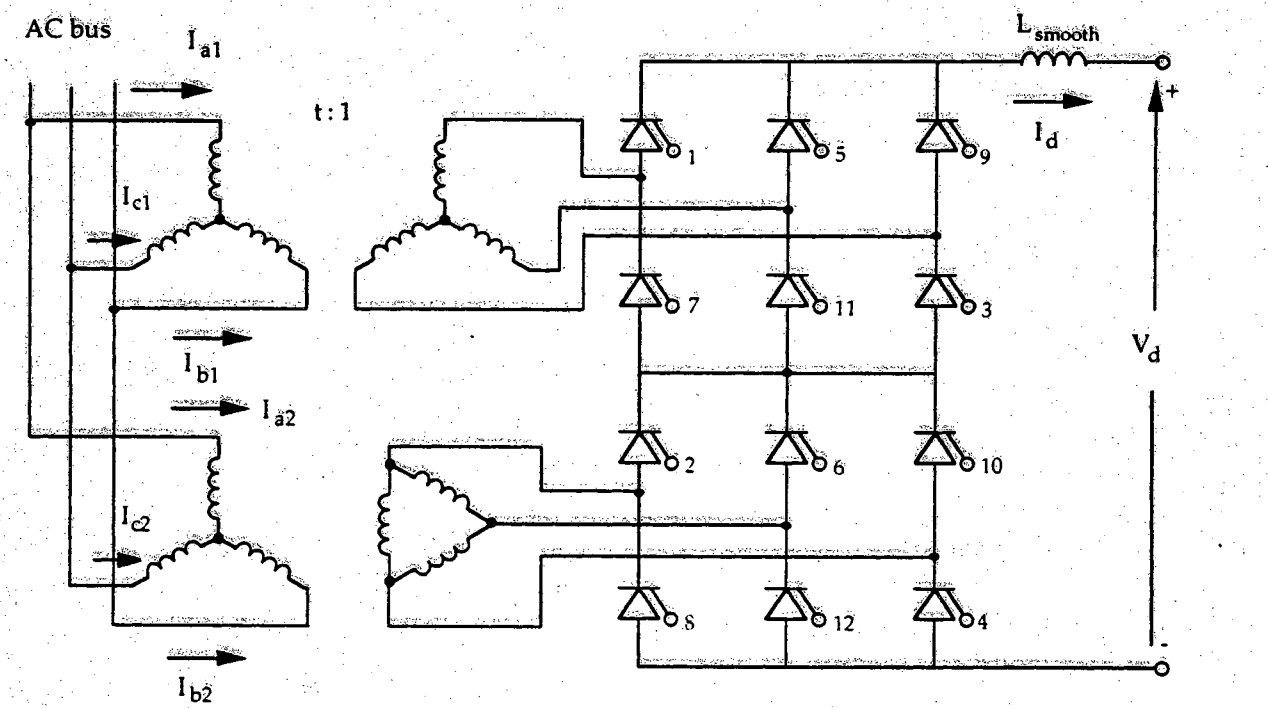


Figure 2.2 12-Pulse Converter Diagram in Firing Order

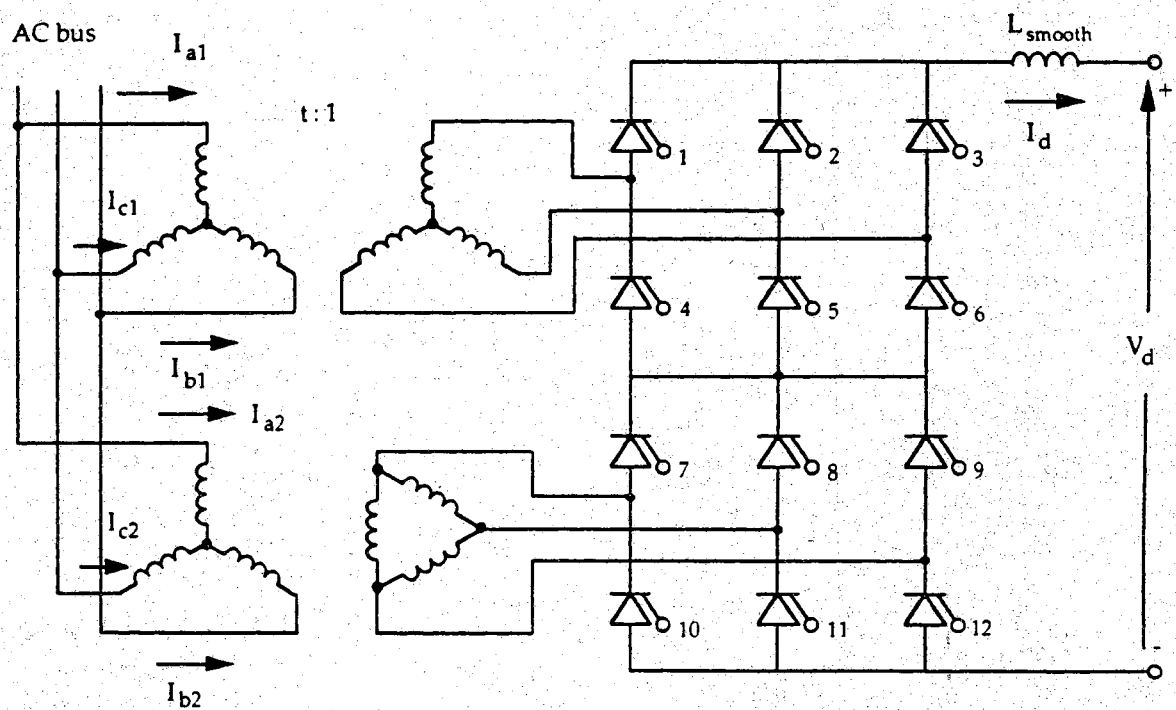


Figure 2.3 12-Pulse Converter Diagram in State Order

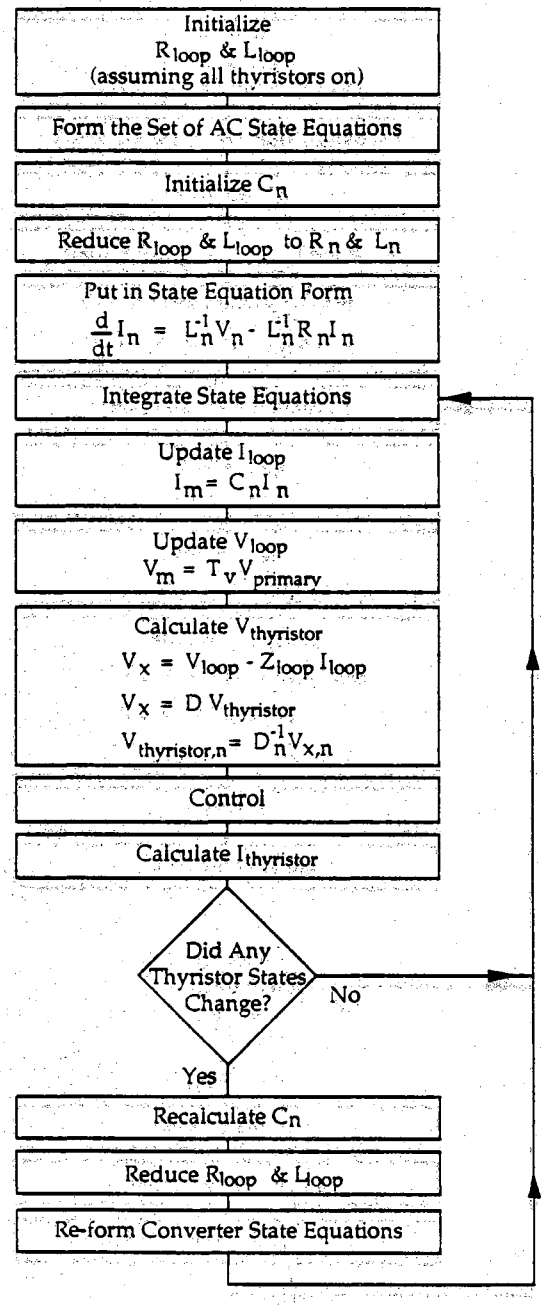


Figure 2.4 Flow Chart of Mathematical Tasks for One Time Step

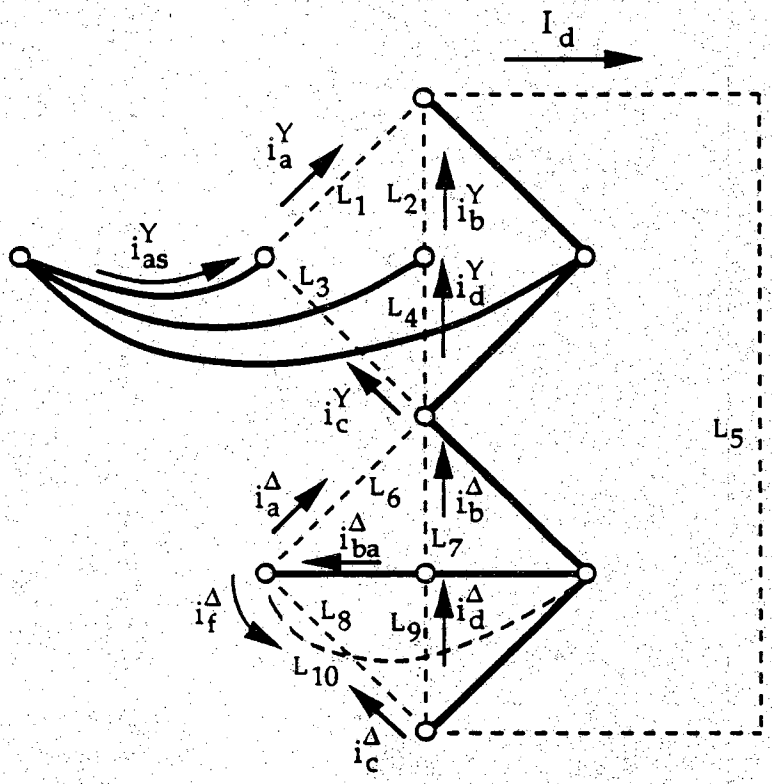


Figure 2.5 Loop Diagram for a 12-Pulse Converter

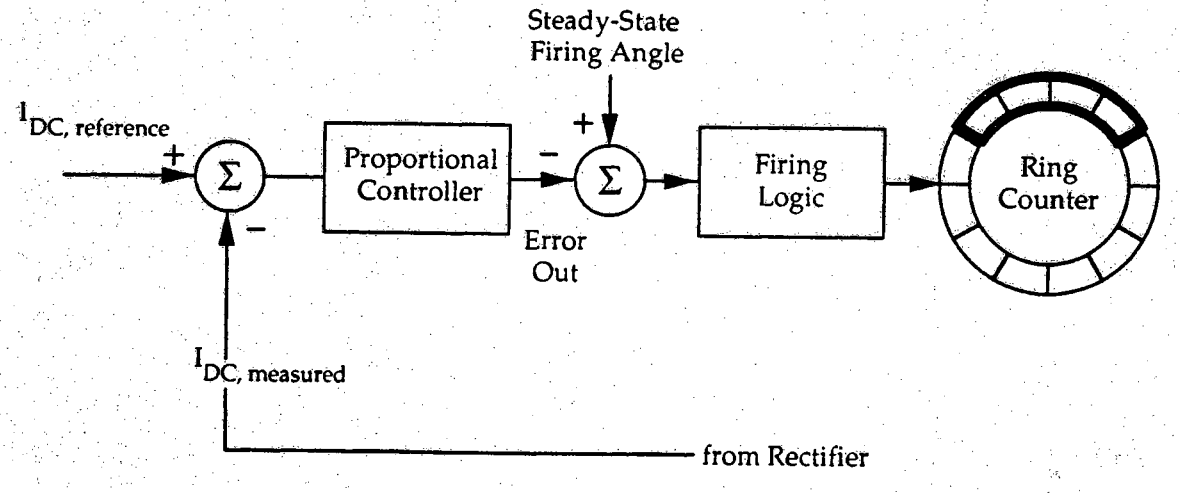


Figure 2.6 Rectifier Control System

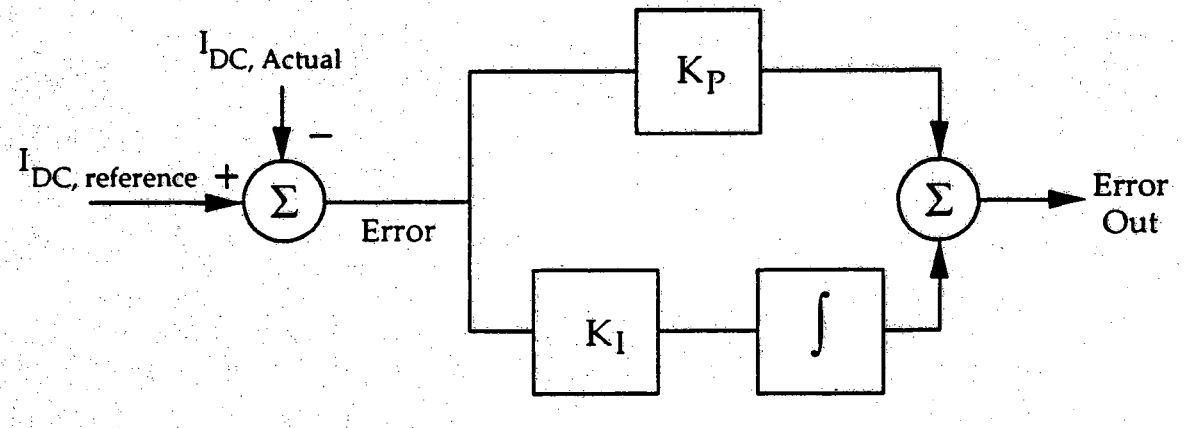


Figure 2.7 PI Controller for Rectifier

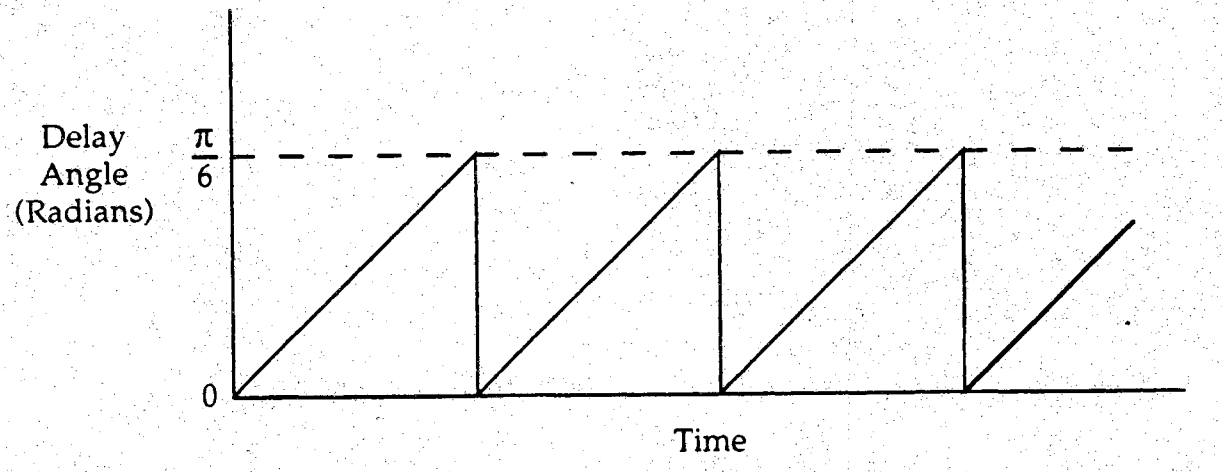


Figure 2.8 Steady State Generation of Firing Pulses

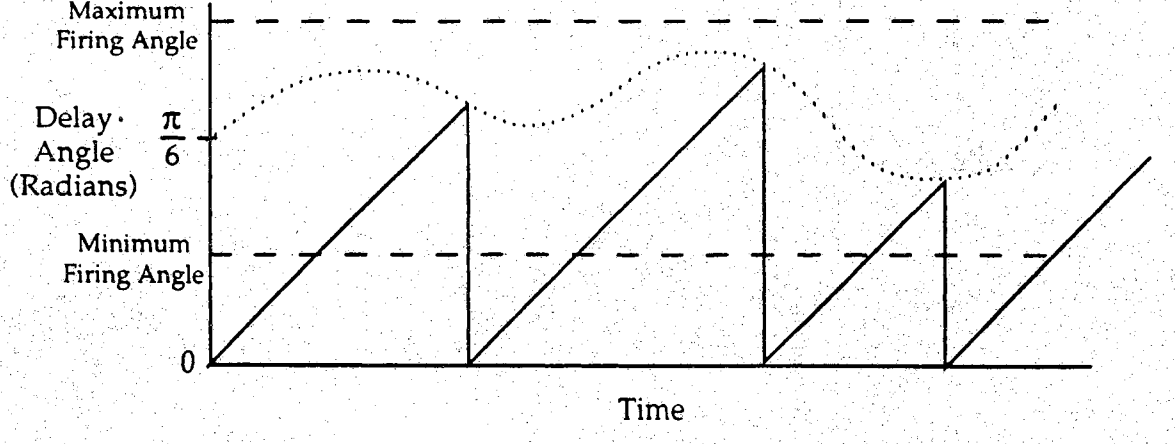


Figure 2.9 Generation of Firing Pulses with Error from PI Controller

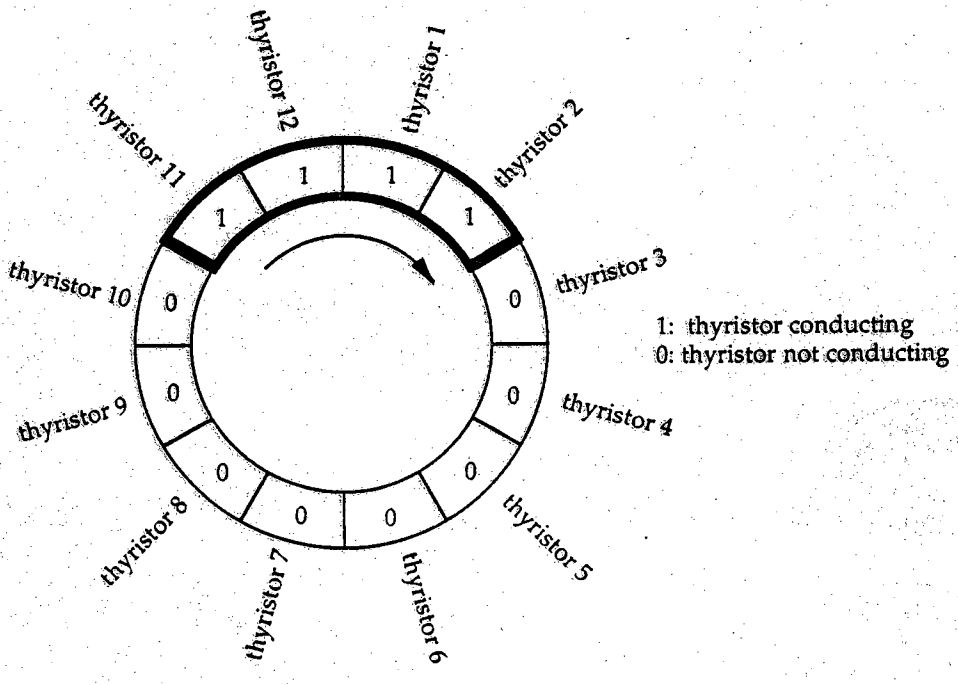


Figure 2.10 Ring Counter

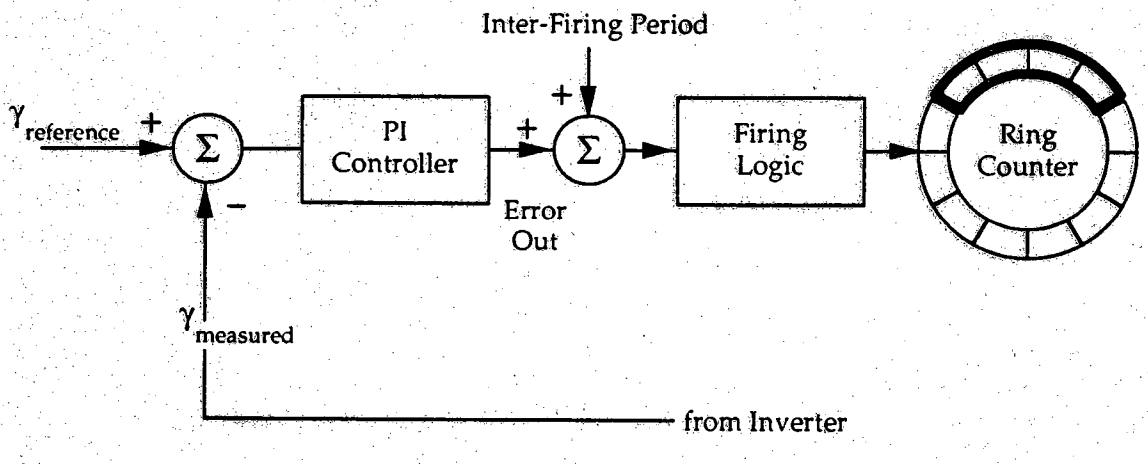


Figure 2.11 Inverter Control System

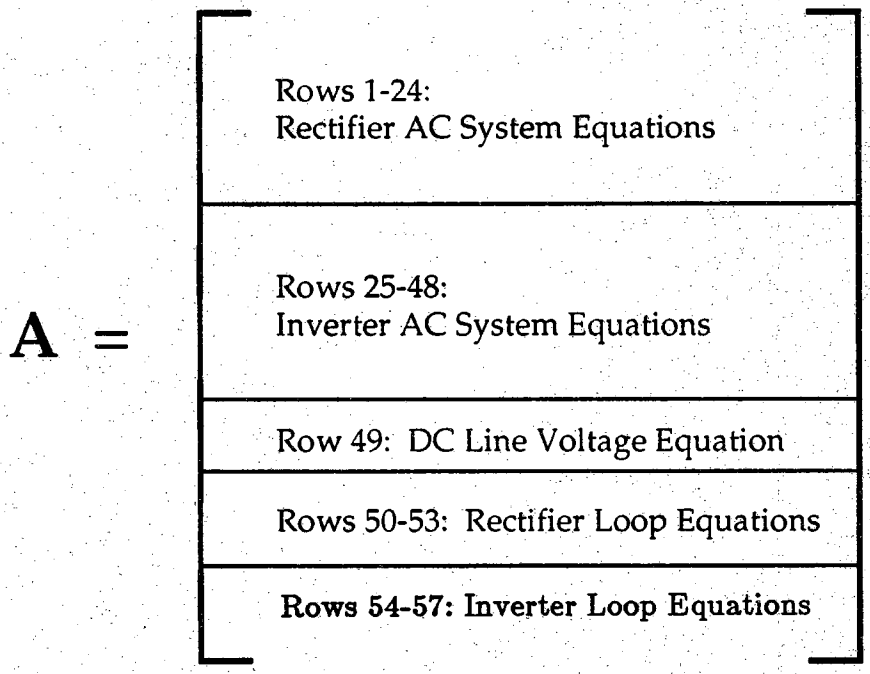
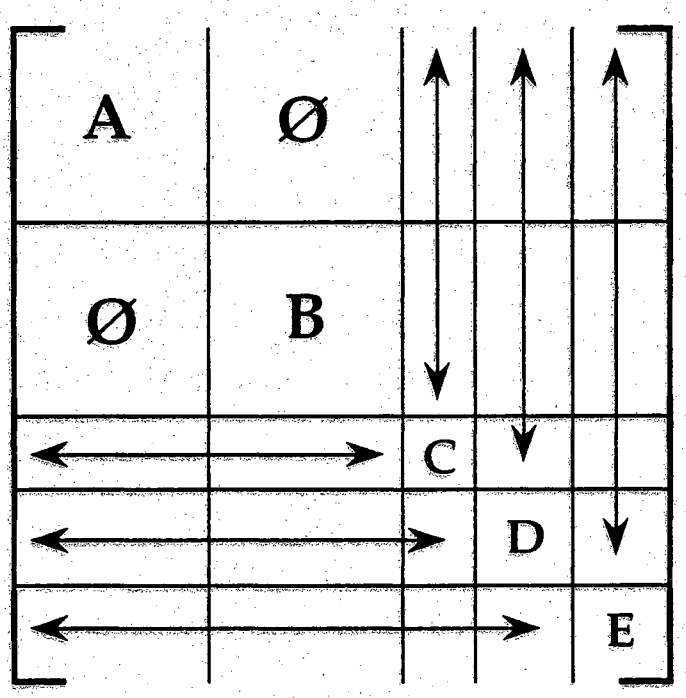


Figure 2.12 Separation of A Matrix by Rows



A - Rectifier AC System

B - Inverter AC System

C - DC Line Voltage

D - Rectifier → Vary with Thyristor States

E - Inverter →

Figure 2.13 Separation of A Matrix by Grid

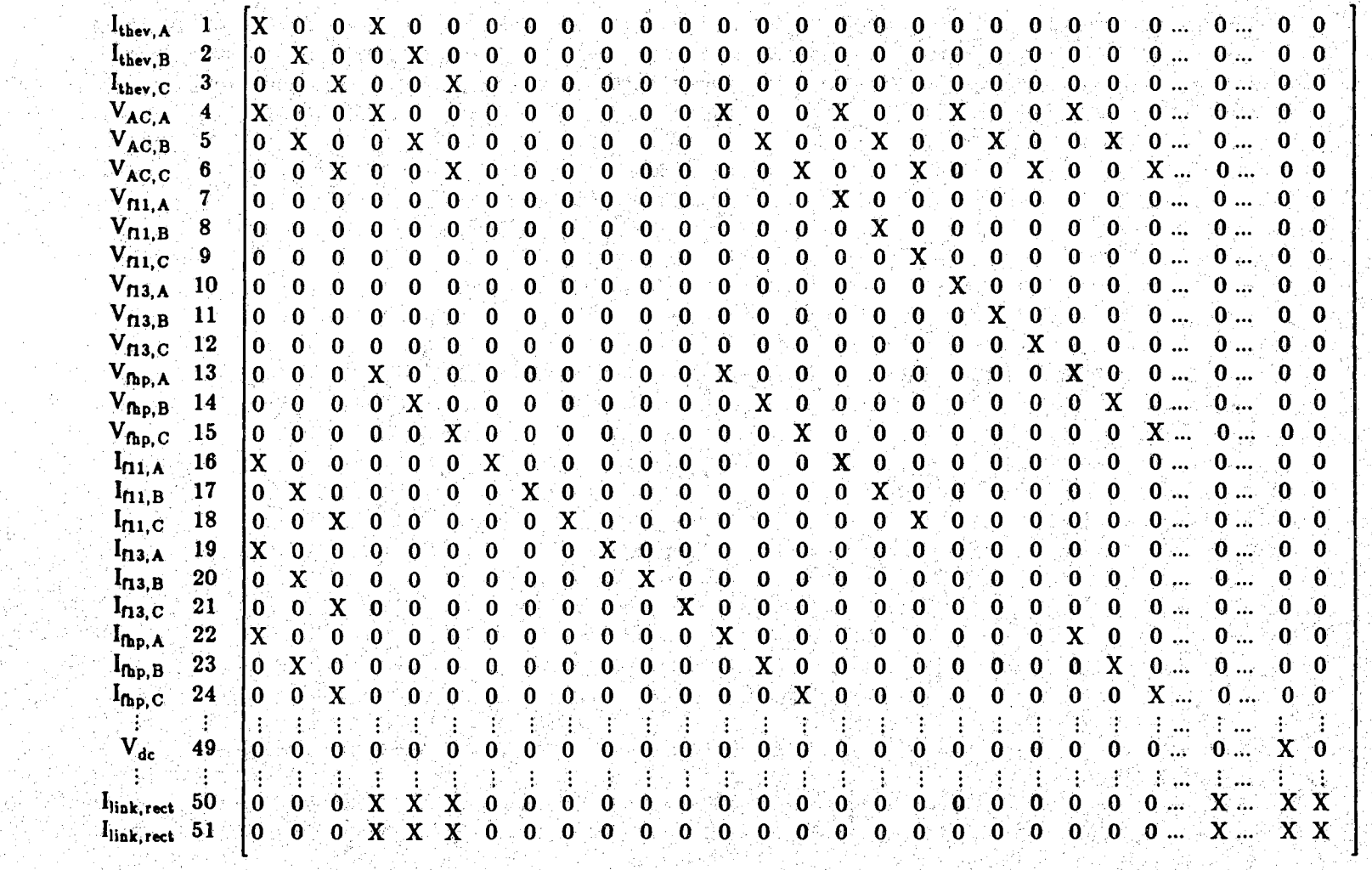


Figure 2.14 Rectifier Portion of the A Matrix

row of A	column of A	value of coefficient

Figure 2.15 Structure of Record Used to Store Elements of A Matrix

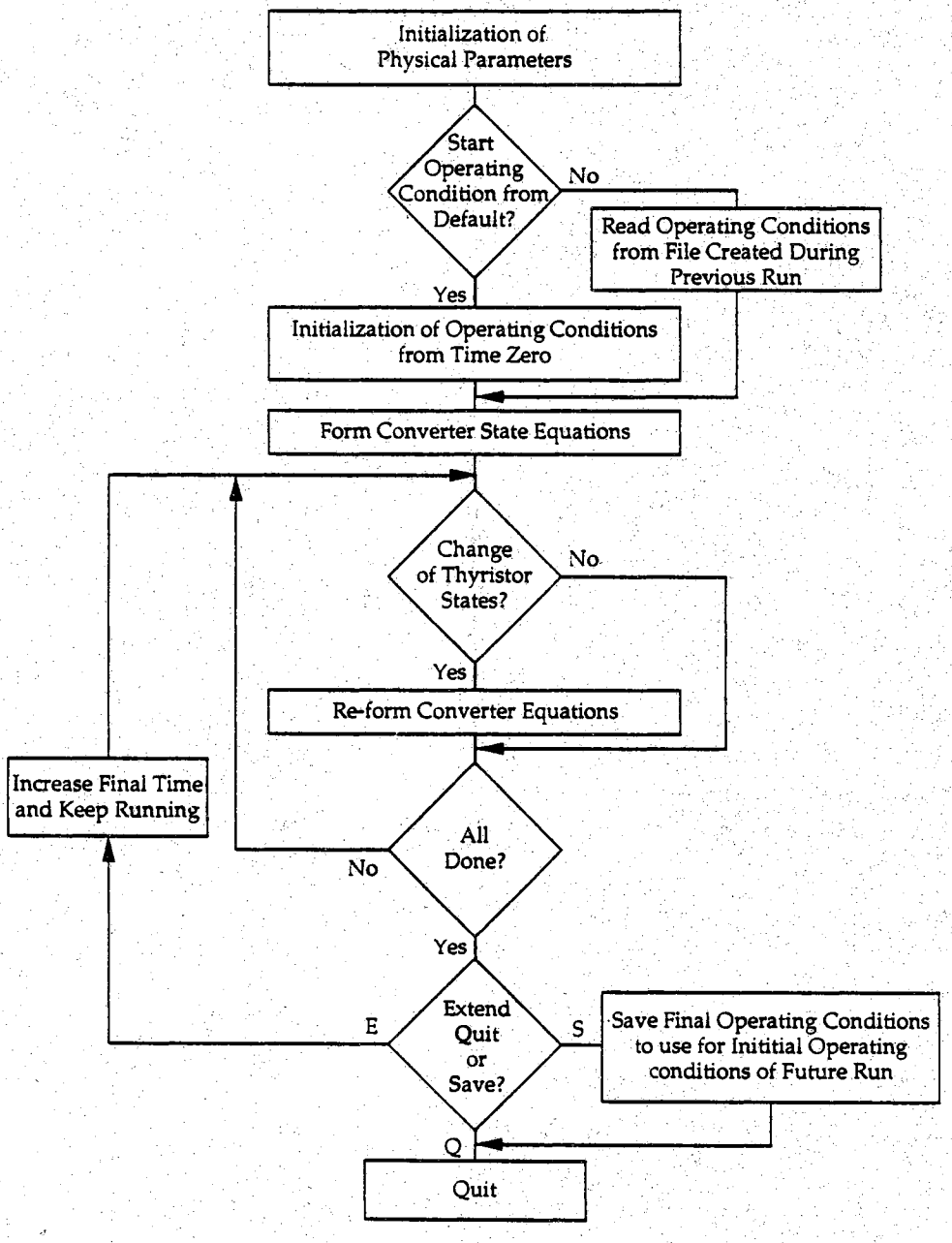


Figure 2.16 Flow Chart of Algorithm Divided into Large Tasks

CHAPTER 3. RESULTS

3.1 Introduction

The goal of this thesis was to employ time saving techniques to an existing detailed HVDC algorithm to make it more efficient and useful. To verify the results, the validity of both time savings and the model itself must be tested. The claims of time savings are verified through the use of a compiler option called "prof", or Profile. Profile keeps a record of the amount of CPU time spent inside of each subroutine. Steady state results from the program are shown, and simulated values are compared with expected values. The expected values are calculated using formulas derived for steady state conditions. Lastly, the transient study shows the reaction of the system to a step change in the reference DC current at the rectifier.

3.2 Time Savings

The results of the profile study are shown in Tables 3.1 and 3.2. In the tables, the column of data used to analyze the performance of individual subroutines is the "ms/call", or the average execution time per call in milliseconds. The value in this column only reflects the CPU time required to execute the statements within the corresponding subroutine. It does not include

Table 3.1 Profile Results from Original Program

time	cumsecs	#call	ms/call	name
61.7	144.13	1439	100.16	integ
19.6	189.95	5758	7.96	ginv
2.7	216.41	2878	2.22	tvolt
1.6	220.17	11512	0.33	multi
0.9	222.25	17268	0.12	filter
0.9	224.24	1	1990.00	MAIN
0.4	227.70	1439	0.61	current
0.3	229.22	2878	0.26	state
0.1	231.79	1439	0.12	firing
0.1	231.94	48	3.12	reduce
0.0	232.70	1439	0.06	gprec
0.0	233.21	1	40.00	init
0.0	233.54	48	0.42	tensor

Table 3.2 Profile Results from New Program

time	cumsecs	#call	ms/call	name
20.9	5.42	2878	1.88	CalcVthy
18.7	10.27	5756	0.84	multiply
12.1	13.40	1439	2.18	integ
9.7	15.92	2928	0.86	Invert
6.9	17.72	17268	0.10	filter
5.5	19.14	1	1420.00	main
1.3	23.50	2878	0.12	CalcState
1.3	23.84	1439	0.24	InvCtrl
1.0	24.09	1439	0.17	buildB
0.5	24.78	50	2.40	formAvar
0.4	24.88	50	2.00	reduceRL
0.3	25.21	1439	0.05	RectCtrl
0.0	25.93	50	0.20	formCn
0.0	25.94	1	0.00	InitStates
0.0	25.94	1	0.00	formAfixed
0.0	25.94	1	0.00	formD
0.0	25.94	1	0.00	formRL
0.0	25.94	1	0.00	formTx

the time required to perform any subroutines which are called from within the given subroutine. The main routine is called only once. The actual time spent performing statements in the main routine shown in Table 3.2 was 1420 milliseconds, but it took the program 25.94 seconds to run. The rest of the time was spent within other subroutines which were called from the main routine.

Table 3.1 shows the results for the original Fortran algorithm, and Table 3.2 displays the results for the C algorithm. These results were obtained from runs on a Sun IPC (Sparc I) workstation. Each program was run for one full electrical cycle, or $\frac{1}{60}$ of a second. The time step for each simulation was one quarter of an electrical degree, or 11.6 microseconds. The original simulation took 233.54 CPU seconds to simulate one electrical cycle. The new simulation required only 25.94 seconds of CPU time. There is a time savings of almost an order of magnitude.

3.2.1 Integration Time Savings

The primary time savings occurred within the integration routine. The subroutines from each routine were different in terms of the tasks that were done for each algorithm. Therefore, to examine the time savings in the integration, the set of subroutines which performed the same task must be studied for the Fortran and C versions.

In the Fortran version, both the A and B matrices were formed within the integration algorithm. The "multi" routine was performed eight times within the

integration to form the coverter state equations. The integration in the Fortran version required 100.16 ms plus eight times 0.33 ms for each iteration. One other subroutine was used within the integration subroutine to calculate the converter state equations. The subroutine "ginv" was performed once per iteration and required 7.96 ms. This is a total of 110.76 ms of CPU time for one integration.

In order to be consistent in the comparison between the two versions, the subroutines which perform the same tasks as in the Fortran version must be analyzed for the C version. For the same function to occur in the C integration as in the Fortran integration, "integ", "multiply", "formAvar", "formAfixed", "buildB", and "Invert" must all be examined. The integration requires 2.18 ms from "integ" plus four executions of "multiply". "multiply" is used to form the product of the state matrix and the state variables. The total CPU time required for the actual integration was 5.54 ms per iteration. (Note: "multi" in the Fortran program and "multiply" in the C program do not perform the same tasks and therefore should not be compared.)

The formation of the A matrix must also analyzed since it is part of the task performed in the Fortran version. The formation of the A matrix in the C version is done in two steps. The first step is the formation of the AC state equation portion whose coefficient will not change. This is done in "formAfixed" which takes less than 0.01 ms to perform. Step two is the formation of the converter state equations. These state equations must be formed every time the thyristor states of the converters change. This task is performed using

"formAvar", "formCn", "reduceRL", and "Invert" which were executed 50 times during the simulation. To make a meaningful comparison with the Fortran version, an equivalent time duration must be calculated in milliseconds per iteration. Therefore, $\frac{50}{1439}$ times the sum of the CPU time required of each subroutine gives the equivalent CPU time expended for each iteration. That result gives the equivalent time duration of 0.19 ms per iteration spent on reforming the converter state equations.

The last routine which must be made part of the comparison is the calculation of the B matrix. This is done in "buildB" and requires 0.17 ms per iteration. Summing the CPU times for these subroutines gives a total of 5.90 ms per iteration.

From Chapter 2, the expected savings of employing sparsity in the integration was examined. If one multiplication took 70 cycles on a 33 MHz machine, and if approximately 11,000 multiplications could be saved in one iteration, it was calculated that the estimated savings would be 23 milliseconds per iteration. The actual savings was 110.76 ms minus 5.90 ms which is 104.86 ms of CPU time for every iteration. This savings is much greater than anticipated.

3.2.2 Reasons for Time Savings

Although the sparsity techniques employed supplied a large part of the time savings, the elimination of re-forming the A matrix at every time step also saved

much CPU time. In Chapter 2 it was shown that there are 166 coefficients which do not change over the course of the simulation. There are another 14 to 32 coefficients for the converter state equations which vary only with the change in the converter topology. But there are no coefficients which needed to be changed every time step. In fact, by examining Table 3.2, it can be seen that the state equations were calculated only 50 times for both the rectifier and the inverter. Therefore, the converter equation coefficients in the A matrix were calculated 1389 times more than necessary in the Fortran program, and the AC system coefficients were calculated 1438 times more than necessary.

Another large difference can be seen in the time it took to invert the L matrix for the converter state equations. The subroutine "ginv" uses a specialized singular value decomposition algorithm which requires far more calculations than does the Gauss-Jordan inversion employed in "Invert". This can be seen in the savings of over 7 ms per iteration due to the replacement of "ginv" by "Invert".

Although there are other time savings which can be identified by comparing Tables 3.1 and 3.2, The majority of the time savings resulted from the sparsity techniques and the elimination of re-forming the A matrix. The overall time savings was nearly 90%.

3.3 Starting from a Recovery File

A recovery file was implemented to allow the user to begin the simulation from the end condition of a previous simulation. This feature is especially useful for multiple runs from the same starting conditions because it eliminates the need

to wait for the transients in the simulation to settle before starting a new study. The following results were obtained using the recovery file. All of the values used for the recovery file are described in Appendix A.

A simulation was run to illustrate the length of time it requires for the initial transients to settle. The steady state value of DC current should be about 1.98 kA since the reference current was set to 1.98 kA. By examining Figure 3.1, it can be seen that the simulation requires approximately 0.28 seconds for the initial transients to settle. This is equivalent to approximately 24,200 iterations of the program, 17 electrical cycles, or 7.25 minutes of CPU time. This can be shortened by using a closer set of initial conditions.

3.4 Steady State Results

The validity of the model must be tested. A straight forward approach is to compare the simulated steady state results with the corresponding values predicted from formulas[1]. One such formula for the average DC voltage at the rectifier terminals is

$$V_{dc,ave} = V_{do} \cos(\alpha) - R_c \, I_d. \tag{3.1}$$

where:

V_{do} is the open circuit voltage, $\frac{6\sqrt{3}}{\pi} E_m \, t$,

E_m is the primary phase AC peak line to neutral voltage,

t is the turns ration of the transformer,

I_d is the DC current,

α is the delay angle,

R_c is the commutating resistance, $\frac{6\ X_l}{\pi}$,

X_l is the leakage reactance of the transformer.

Table 3.3 Values Used to Calculate Average DC Voltage

Variable	Value	
	Rectifier	Inverter
E_m	188.525	188.549
t (turns ratio)	0.8859	0.8475
L_l (transformer inductance)	0.2793	0.03052
α	13.996	--
γ	--	17.337
I_d	1.98	1.98

Calculating V_{do} using the peak voltage from Figure 3.9 and the transformer turns ratio found in Table 3.3, yields a value of 536 kV. The value shown in Table 3.3 for the transformer inductance must be multiplied by the AC frequency to find the reactance. The value of R_c is 20.1 Ω. Figure 3.6 displays the DC voltage waveform of the rectifier. An exact average value is not shown, but 496.3 kV matches very closely with the displayed waveform.

Another formula to test the DC voltage at the inverter terminals is:

$$V_{dc,ave} = V_{do}\ \cos(\gamma) - R_c\ I_d \tag{3.2}$$

γ is the extinction angle of the inverter. The other variables correspond to the same physical definitions described above for the inverter. The value for V_{do} is

528 kV, R_c is 22.0 Ω which yields a value of 461.1 kV for the average DC voltage. Figure 3.7 displays the DC voltage at the inverter terminals. The average value calculated compares favorably with the center of the waveform shown in Figure 3.7.

3.5 Transient Results

A case was run to demonstrate the transient response of the HVDC system, and in particular, the action of the converter control. A step change in the reference current from 1.98 kA to 1.89 kA was put into the rectifier controller. The results are shown in Figures 3.16 through 3.19. The DC current took only 0.3 seconds to settle to its new value.

It is difficult to analyze the transient conditions. Using Equation 3.1, V_{do} and R_c should remain constant. One would think that since the rectifier control varies the firing angle, α would increase so that I_d would decrease. As α is increased the average voltage level drops which causes α to decrease. This oscillation occurs in the firing angle until the DC current reaches the reference value. In this case, the controller oscillates once.

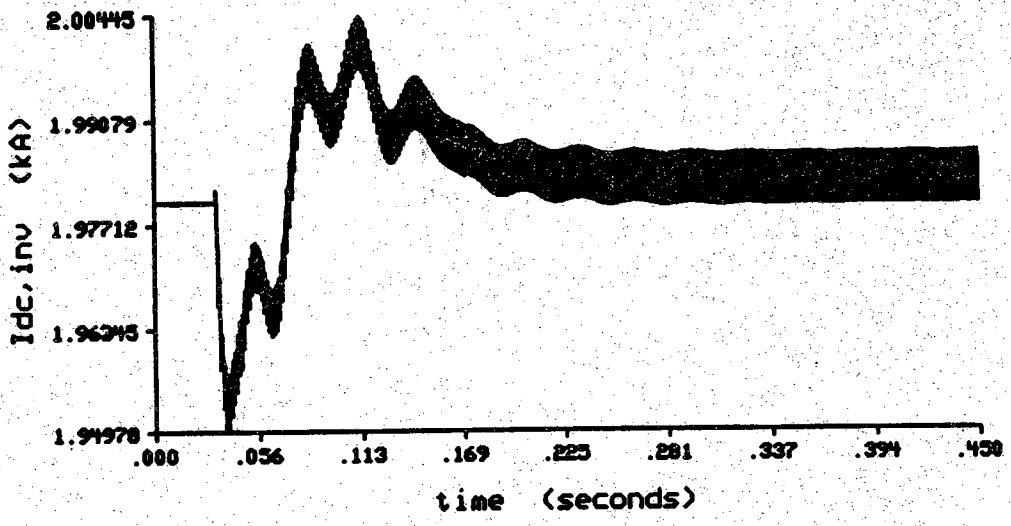


Figure 3.1 Initial Transient Based on Load Flow Solution, I_{dc}

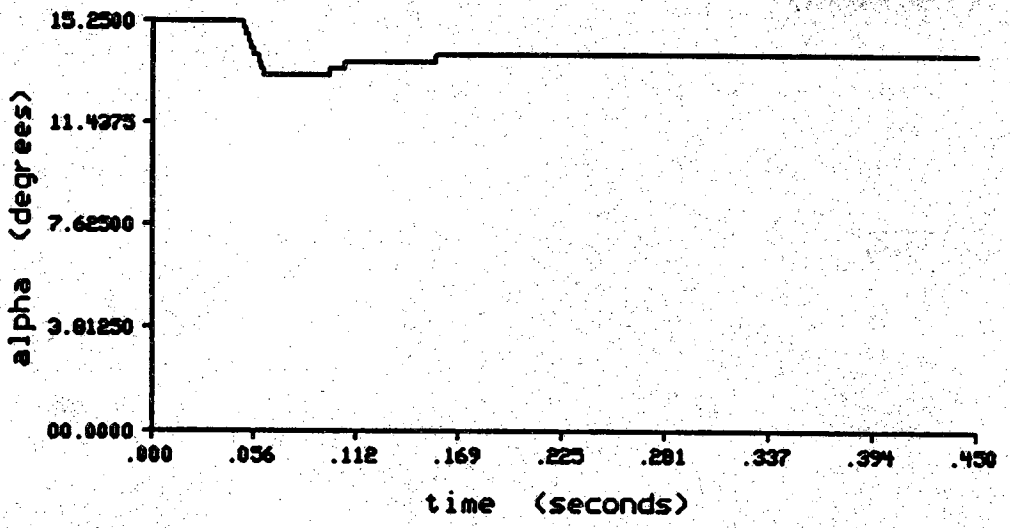


Figure 3.2 Initial Transient Based on Load Flow Solution, α

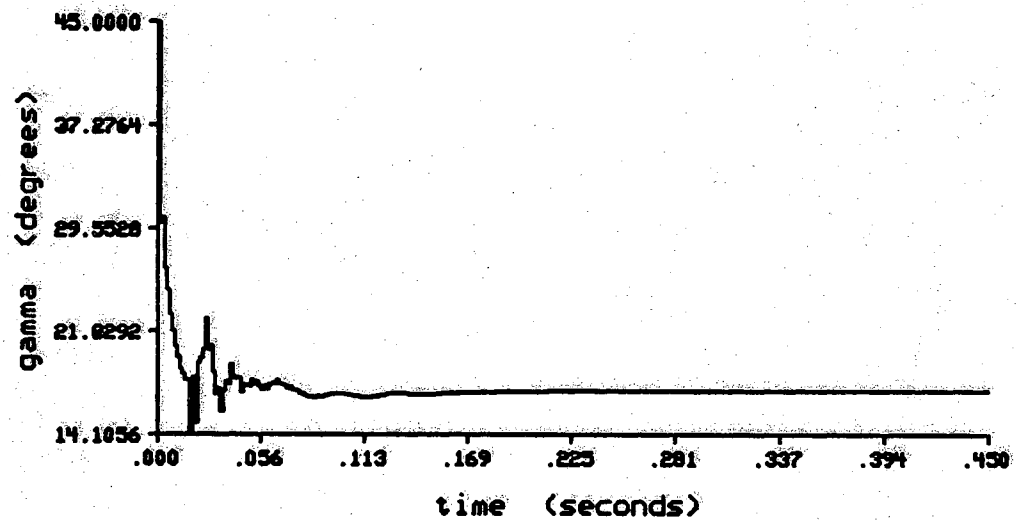


Figure 3.3 Initial Transient Based on Load Flow Solution, γ

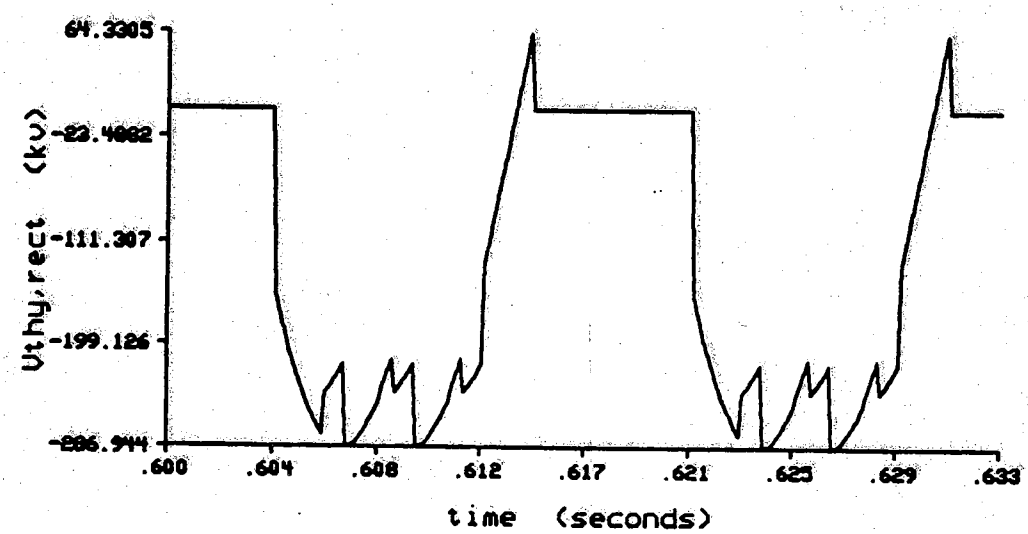


Figure 3.4 Steady State, Thyristor Voltage at Rectifier

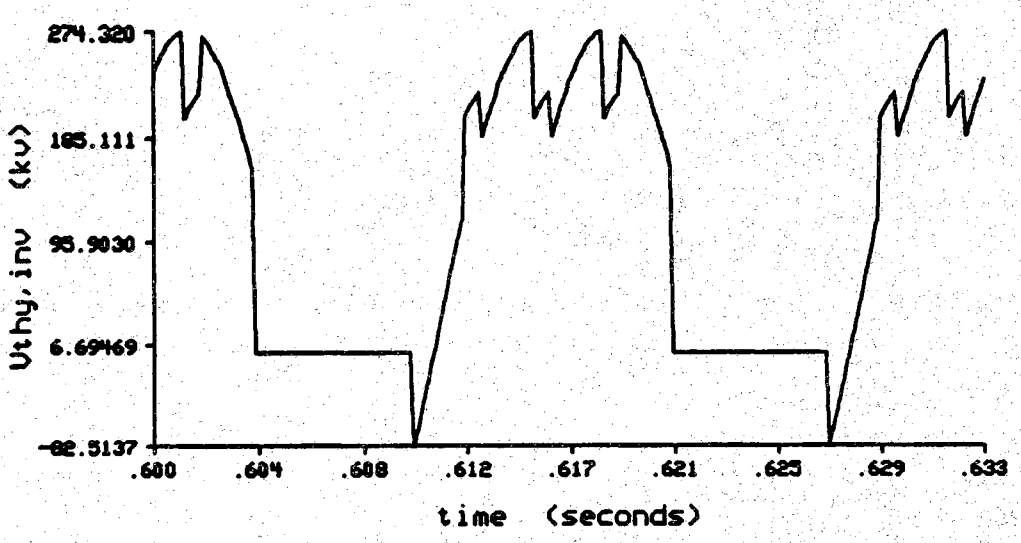


Figure 3.5 Steady State, Thyristor Voltage at Inverter

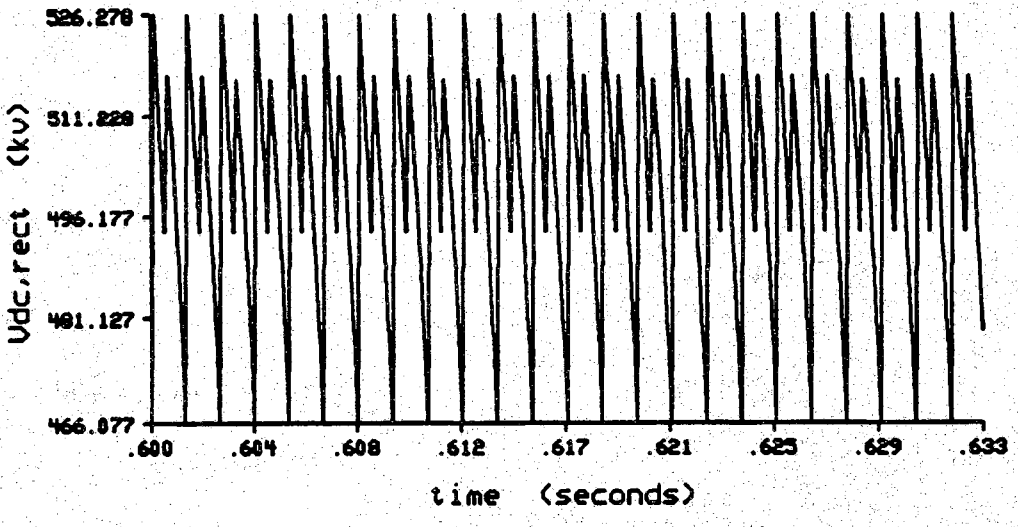


Figure 3.6 Steady State, DC Voltage at the Rectifier

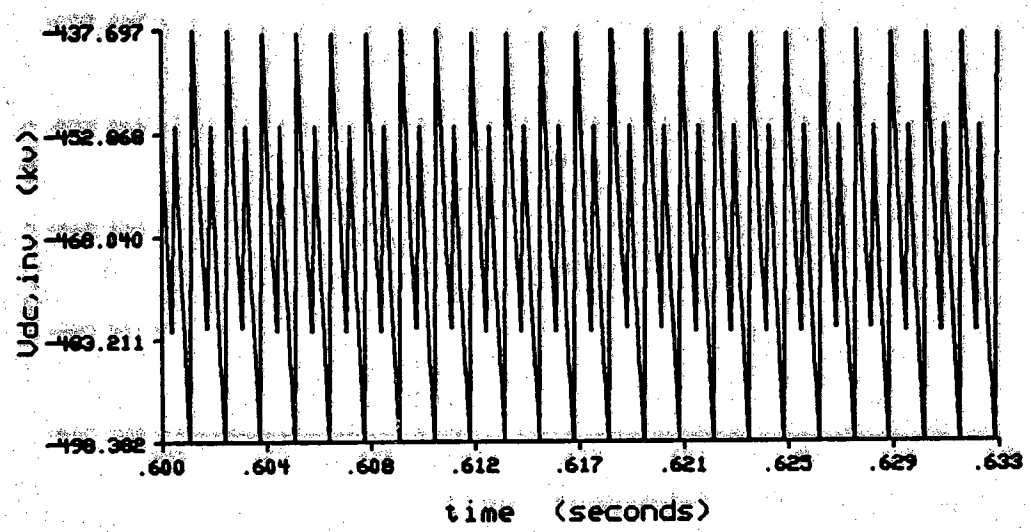


Figure 3.7 Steady State, DC Voltage at the Inverter

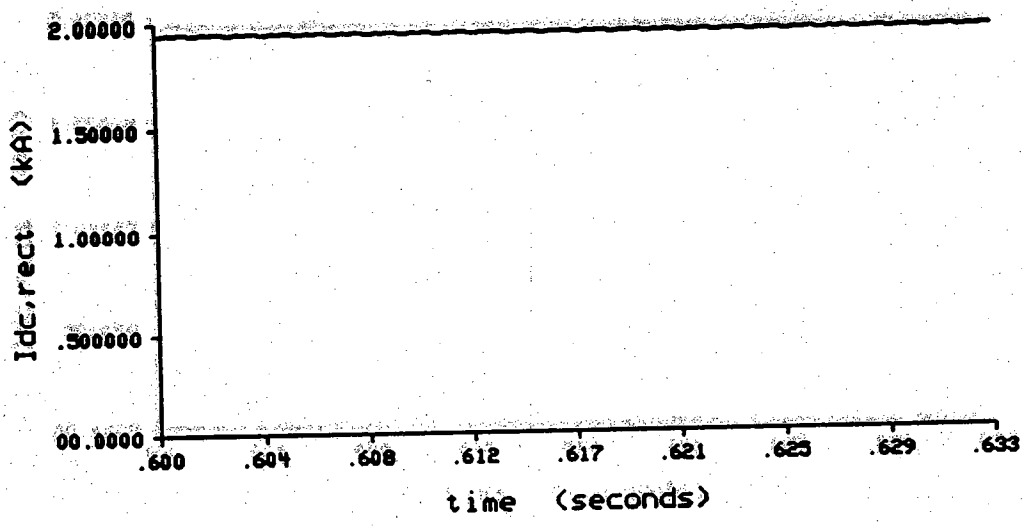


Figure 3.8 Steady State, DC Current at the Rectifier

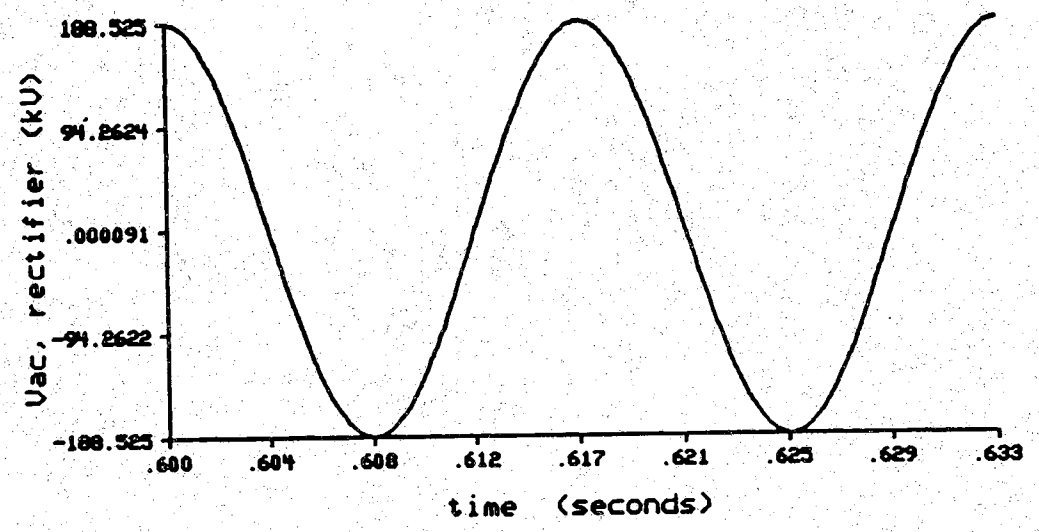


Figure 3.9 Steady State, AC Voltage at the Transformer Primary, Rectifier

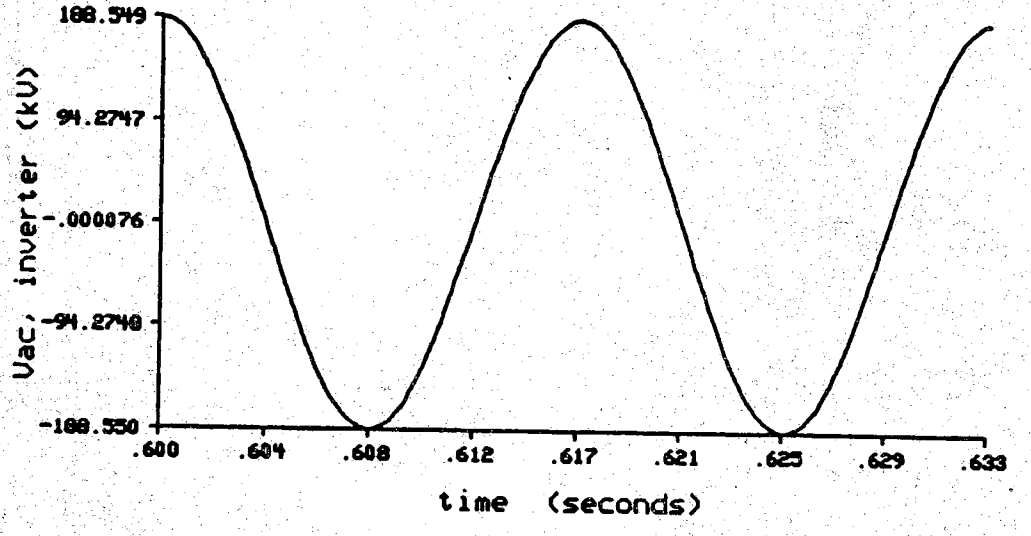


Figure 3.10 Steady State, AC Voltage at the Transformer Primary, Inverter

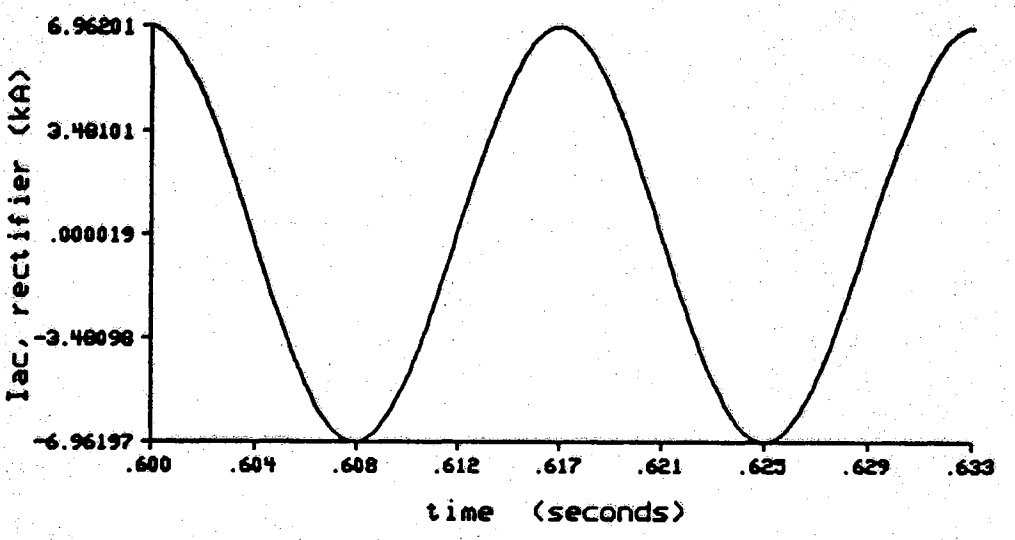


Figure 3.11 Steady State, AC Thevenin Current, Rectifier

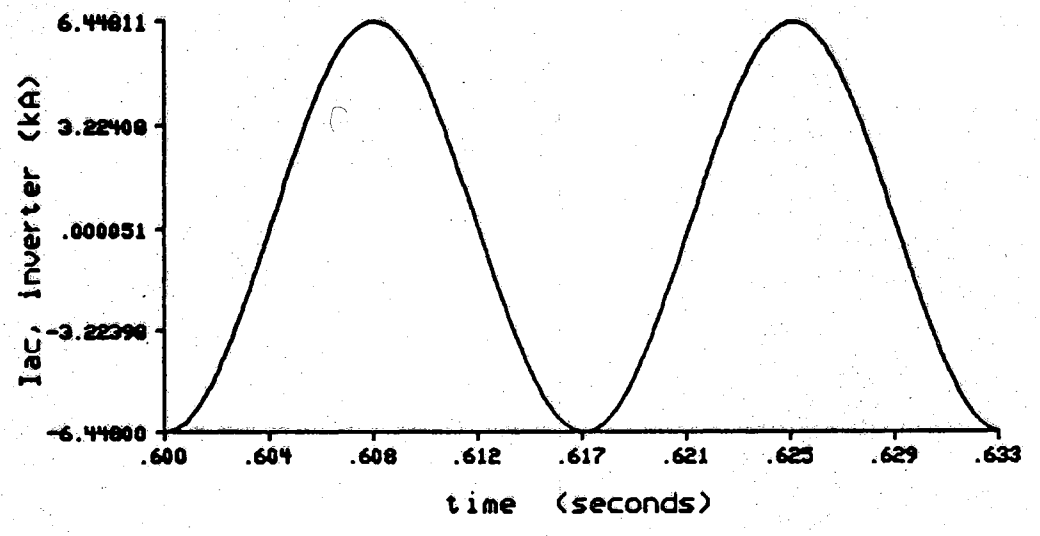


Figure 3.12 Steady State, AC Thevenin Current, Inverter

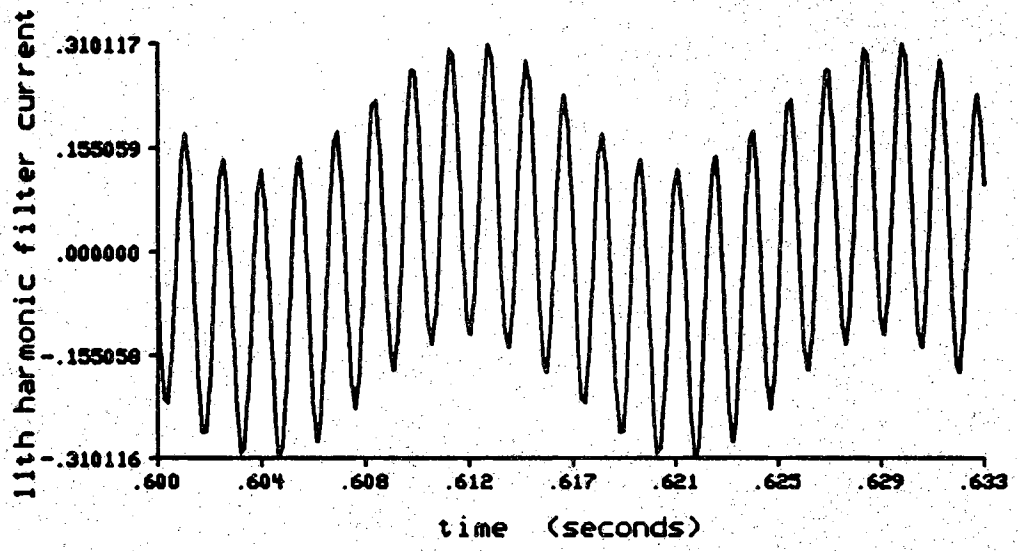


Figure 3.13 Steady State, 11th Harmonic Filter Current, Rectifier

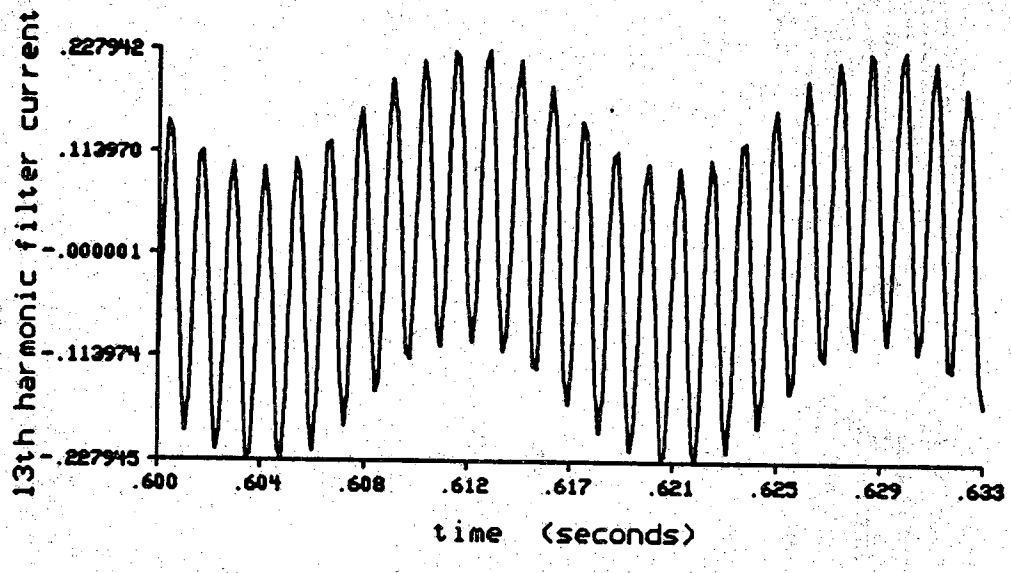


Figure 3.14 Steady State, 13th Harmonic Filter Current, Rectifier

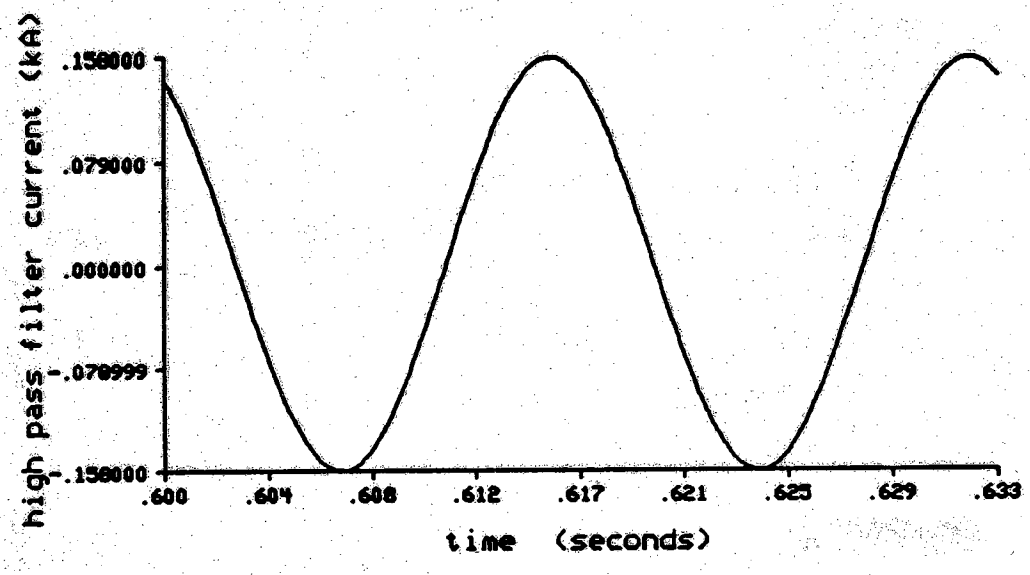


Figure 3.15 Steady State, High Pass Filter Current, Rectifier

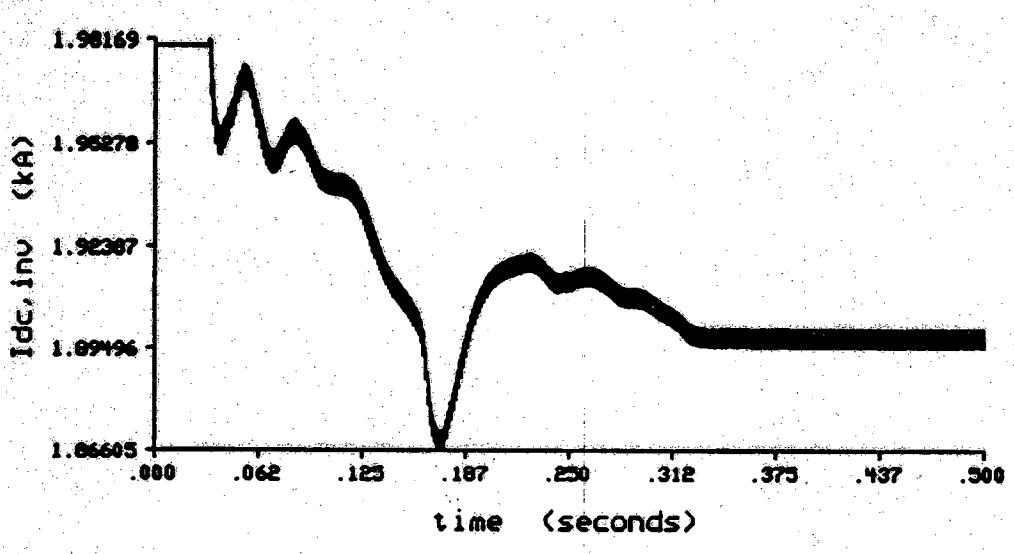


Figure 3.16 Transient Based on Change in Reference Current, I_{dc}

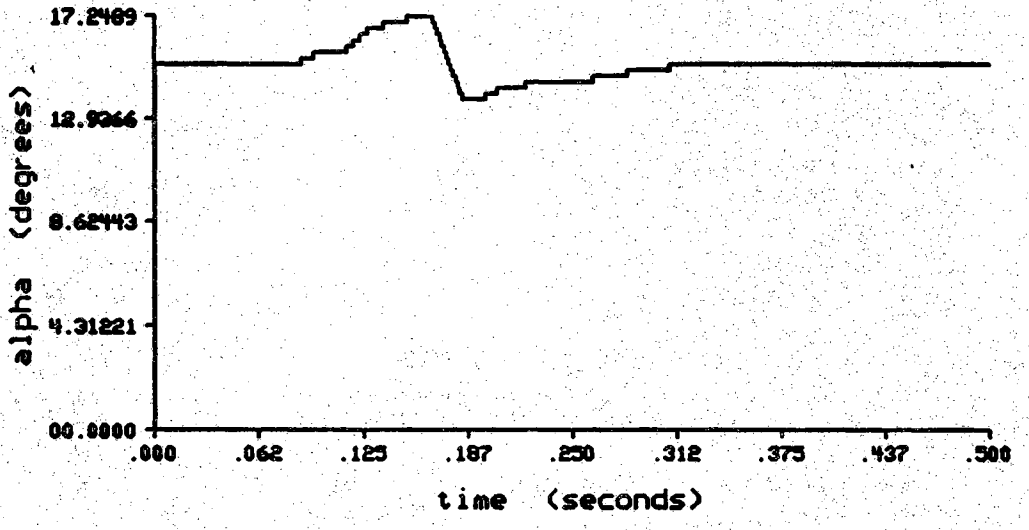


Figure 3.17 Transient Based on Change in Reference Current, α

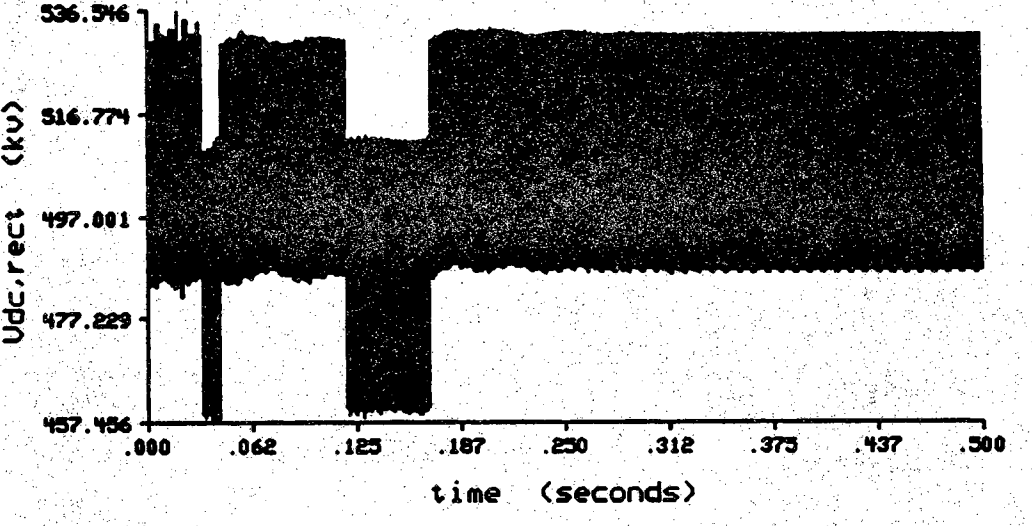


Figure 3.18 Transient Based on Change in Reference Current, V_{dc}

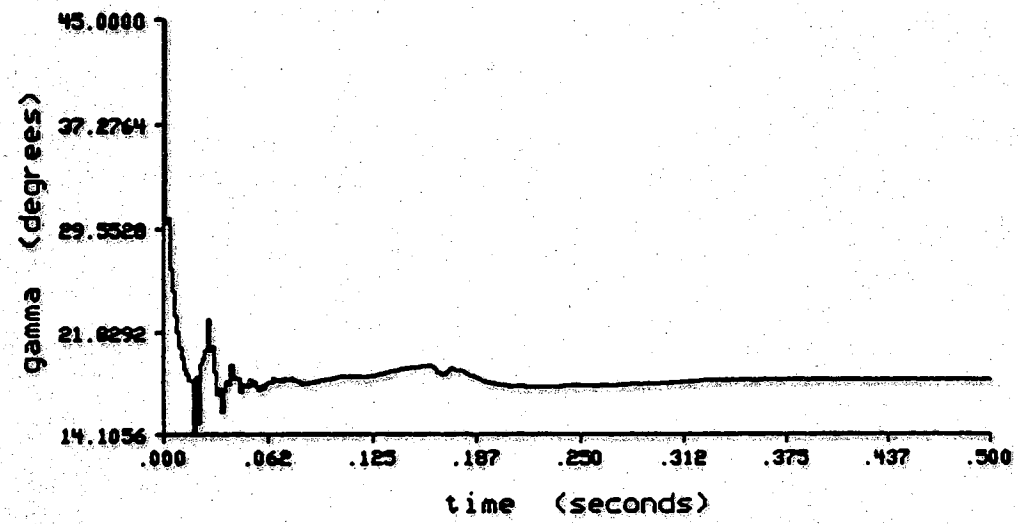


Figure 3.19 Transient Based on Change in Reference Current, γ

CHAPTER 4. CONCLUSIONS AND RECOMMENDATIONS

4.1 Conclusions

A detailed HVDC simulation can be used to conduct a variety of studies when designing a new HVDC link or analyzing an existing system. These studies include bulk power transfer, analysis of harmonic components on either the AC or DC systems, and analysis of transient conditions. The main objective of this thesis was to develop a versatile and efficient simulation tool for these purposes. The primary benefits derived from this research are the time savings in simulation runs and the ease of maintaining the program.

4.1.1 Time Savings

The time savings resulting from modifying the original algorithm were substantial and can be attributed to several factors. The most important factor was the implementation of sparsity techniques. By implementing sparsity, over 11,000 multiplications were eliminated every iteration.

Another time saving was the result of re-structuring the A matrix. Previously, the entire A matrix was re-formed every time step. By dividing the A matrix into three sections, only the sections corresponding to a change in the circuit topology would have to be re-formed. The state equations of the AC

systems connected to the rectifier or to the inverter remain constant during the simulation. Therefore, the AC system coefficients are calculated once for the entire simulation. The rectifier and inverter state equations change each time the topology of the converter changes. Therefore, it is necessary to calculate the state equation coefficients only at the time of the topology change. The actual time savings achieved using the sparsity techniques and the restructured A matrix was 104.86 milliseconds per iteration using a Sun IPC (Sparc I) workstation. This translates into more than 15 minutes of CPU time saved for every 0.1 seconds of simulated time.

The streamlined implementation of the control schemes as well as other routines all contributed to an overall time savings. All the changes implemented gave a total time savings of 20 minutes and 45 seconds of CPU time for every 0.1 seconds of simulated time on a Sun IPC workstation. The new algorithm ran in approximately eleven percent of the time of the program written without these time saving techniques.

4.1.2 Modularity

An objective of the thesis was to make the algorithm readable and maintainable. This was achieved by modularizing the flow, function, and task of the program. Modularity of the flow was accomplished by keeping related tasks within the same section of code. Code pertaining to each independent task was grouped together. By grouping the tasks sequentially, rather than intermixing lines of code for the tasks, the algorithm became more readable. In the algorithm

there are six stages in the flow. The first three stages comprise the initialization processes, the following two stages form the main loop, and the last stage terminates the program.

The second modularity was by function. Here, function describes a physical process of the converter, i.e. the control is a physical process within the converter. An example of how this modularity assists in the simulation is in the replacement of an existing controller by a different one through the substitution of a single subroutine.

The last type of modularity was by task, or by subroutine. To improve readability and maintainability, subroutines were created for two purposes: to perform tasks which were done repeatedly and to condense the main routine into a comprehensible form.

4.1.3 Advantages of the C Language

The programming language C enabled a straight forward implementation of sparsity and modular programming. C contains a data structure type called a record. It was much easier to develop the sparsity techniques using records. These records could hold both the location of the element in the A matrix and its value in a single data structure. This eliminated the need for a set of arrays to hold the row, column, and value of the element. Another feature that C possesses is the ability to dynamically store data.

The structure of the C language facilitated documentation of the program. Comments can be on the same line as program statements. Since C is case sensitive, variables using upper and lower case letters can be uniquely defined. The necessity to declare each variable provides a convenient place to verbally describe that variable.

4.2 Recommendations

The purpose of this project was to create a simulation that would be a time and memory efficient tool for a two terminal link. In the future, however, more work will be done with multiterminal systems. Most of these recommendations stem from the need for the multiterminal case.

4.2.1 Layout of the State Matrix

Further modularity in the structure of the state matrix should be pursued. Presently, the approach is to divide the A matrix into three sections. The first part of the matrix houses the coefficients that do not change their values or locations during the course of the simulation. The second section holds the coefficients calculated for the rectifier topology. The third section holds the coefficients calculated for the inverter topology.

An alternative approach would be to divide the A matrix based on each converter rather than by fixed and changing coefficients. This approach would provide an easier indexing scheme for the multiterminal case. Rather than separating the AC state equations from the converter state equations, it would be more beneficial to keep the converter equations and the associated AC system

equations in the same block of the A matrix. By modularizing the A matrix in this way, there would be a constant block size reserved for the coefficients of each converter and its associated AC system. This would make it easier to index the state variables for each converter in the multiterminal case. Another advantage would be the easy elimination or addition of a converter to the system.

4.2.2 Data Structure of the State Matrix

Presently, each of the three sections of the A matrix is stored in a separate array of records because the rectifier and inverter could change topologies at different times. One difficulty with this method is that the creation of the array of records must be done statically in the declaration section of the program. Hence, if a new converter and DC line were to be added to the system, another array would have to be declared.

An alternative approach using the array structure would be to store all of the coefficients in one array of records. This method would eliminate the need to declare and dimension a new array of records each time a new converter was added to the system, but the dimension of the single array would still need to be re-dimensioned. Another problem arises because the number of coefficients required to model the converter is not static. A dynamic method of eliminating the old coefficients and reading in the new coefficients is needed. A linked list can accomplish this.

By using a linked list, one data structure can replace the three or more arrays used in the multiterminal case. Pointers can be used to keep track of the

start and end of each converter's coefficient section. When the topology of a converter changes, the pointers can be used to eliminate the old coefficients and insert the new coefficients to the end of the list.

4.2.3 Indexing of the Loop Equations

A problem arises when the set of loop equations is reduced to the independent set of loop currents. The index given to the DC link in the reduced set of loop equations varies between two values.

Under normal operating conditions, when the full set of loop equations are reduced, the DC link current is always one of the two or three loop currents chosen to be an independent current. (The boolean logic explanation of the connection matrix is in Appendix C.) Thus, the DC link current index in the reduced set of equations takes on the values of one or two during normal operation. If the DC link current were permanently chosen as the first link in the reduced set, then it would also be indexed one in the reduced set. There would be no to recalculate its index.

4.2.4 Change the Indexing Method of the Converters

Each converter presently has an index associated with it. The rectifier's index is zero, and the inverter's index is one. The indices are used for a variety of purposes including the control schemes and the multiplication of the A matrix. Indexing the converters from 1 to N, where N is the number of converters, would be more consistent with the rest of the indexing schemes used in the thryristor state numbering, the state variable numbering, etc.

4.2.5 Use of Transformer Taps in System Control

Presently the transformer tap settings are fixed. Changing the transformer taps would require recalculating parts of the AC state equations involving the turns ratio, the current transformation matrix, and the voltage transformation matrix. Additionally, another control loop to change the tap settings would have to be developed.

4.2.6 Summary

This work has been done to provide a fast, flexible, and useful tool to simulate a two-terminal HVDC system. The sparsity techniques and modular programming were implemented with two-terminal systems in mind; however, these techniques may be extended for multi-terminal systems.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] E.W. Kimbark, *Direct Current Transmission*, Vol. 1, Wiley-Interscience, New York, 1971.

[2] D.P. Carroll. "HVDC Transmission Systems," course notes, *Purdue University*, West Lafayette, Indiana.

[3] O.Rathjen. "Digital Real-Time Simulation of a HVDC System," EPE Aachen, 1989, pp. 883-888.

[4] H.A. Peterson, P.C. Krause, J.F. Luini, and C.H. Thomas. "An Analog Computer Study of a Parallel AC and DC Power System," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-85, No. 3, March 1966, pp. 191-209.

[5] J.G. Kassakian. "Simulating Power Electronic Systems - A New Approach," *IEEE Procedings*, Vol. 67, No. 10.

[6] M.D. Heffernan, K.S. Turner, J. Arrillaga, and C.P. Arnold. "Computation of AC-DC System Disturbances - Part I. Interactive Coordinaton of Generator and Convertor Transient Models," IEEE 1981 Winter Meeting, 81 WM 059-5

[7] B. Zhou. "Steady State Sstability analysis of HVDC Systems with Digital Control," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-102, No. 6, June 1983, pp. 1764-1770.

[8] A.E. Hammad, D.A. Woodford, and R.M. Mathur. "AC Voltage Control at an HVDC Terminal," IEEE Canadian Conference on Communicatons and Power, Montreal, October, 1978 pp.341-344.

[9] L.A.S. Pillotto, M. Roitman and J.E.R. Alves. "Digital Control of HVDC Converters," *IEEE Transactions on Power Systems* , Vol. 4, No. 2, May 1989, pp. 704-711.

[10] C.T. Liu. "Dynamic Studies of Multiterminal AC-DC Systems," Ph.D. Thesis, *Purdue University*, West Lafayette, May 1988.

[11] C.M. Ong and A. Hamzei-Nejad. "Digital Simulation of Multiterminal HVDC Systems for Transient Stability Studies Using a Simplified DC System Representation," IEEE Paper 84SM588-0, PES Summer Meeting, Seattle, July 15-20, 1984.

[12] Alireza Hamzei-Nejad. "Dynamic Studies of Multiterminal DC-AC Systems," Ph.D. Thesis, Purdue University, West Lafayette, December 1984.

[13] S.S. Chen, C.N. Lu, and C.M. Ong. "Reactive and DC Power Dispatch to Minimize Transmission Losses of an Integrated AC-DC Power System," CIGRE Symposium, Boston 1987.

[14] B.K. Johnson, F.P. deMello, and J.M. Undrill. "Representation of DC Converters in Load Flow and Stability Calculation," IEEE 1980 Summer Meeting, 80 SM 586-8.

[15] J. Reeve and R. Adapa. "Evaluation of Developments in DC Models For AC/DC Transient Stability Programs," Cigre Symposium, Boston, 1987.

[16] N. Sato et al. "Multiterminal HVDC System Representation in a Transient Stability Program," *IEEE Transactions on Power Apparatus and Systems*, Vol. 99, no. 5, Sept./Oct. 1980, pp.1927-1935.

[17] R. Adapa and J. Reeve. "Diagnosis of the Response of a DC System to Symmetrical and Non-Symmetrical Faults," Proceedings of IEEE MONTECH Conference on HVDC Power Transmission, September 1986, pp 35-38.

[18] S.Williams and I.R. Smith. "Fast Digital Computation of 3-phase Thyristor Bridge Circuits," *Proc. IEE*, Vol. 120, No. 7, 1973.

[19] J. Milias-Argitis and G. Galanos. "Dynamic Simulation of HVDC Transmission Systems," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-92, No. 8, May/June 1976, pp. 2056-2065.

[20] P.M. Lin. "Computer Aided Circuit Analysis," course notes, *Purdue University* West Lafayette, Indiana.

[21] G.R. Berube and B.T. Ooi. "Fast Periodic Solution For Twelve-Pulse Converter" *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-102, No. 7, July 1983, pp. 1994-2003.

[22] C.M. Ong, C.T. Liu, and C.N. Lu. "Generation of Connection Matrices for Digital Simulation of Converter Circuits Using the Tensor Approach," *IEEE Transactions on Power Systems*, Vol. PWRS-2, No. 4, November 1987, pp. 906-912.

[23] C.T. Liu and C.M. Ong. "EMTP Compatible HVDC and SVC Circuit Modules," IASTED International Conference, Taiwan, March 1991.

[24] L.O. Chua and P.M. Lin. *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[25] J.P. Sucena Paiva, C.M. Amaral Alegria, and L.L. Freris. "On-Line Microcomputer Control of Converter Systems," *Proceedings of the Seventh Power Systems Computations Conference*, PSCC 81, Lausanne, Switzerland, pp. 454-461, Westbury House, July 1981.

[26] R.W. Hornbeck. *Numerical Methods* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[27] J.L. Hay, J.S. Bhatti, and N.G. Hingorani. "Simplified Dynamic Simulation of HVDC System By Digital Computer," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-90, No. 2, March/April 1971, pp. 859-864.

[28] J. Reeve and R. Adapa. "A New Approach to Dynamic Analysis of AC Networks Incorporating Detailed Modeling of DC Systems. Part I: Principles and Implementation," *IEEE Transactions on Power Delivery*, Vol. 3, No. 4, October 1988, pp. 2005-2011.

[29] J. Milias-Argitis, T.Zacharias, C. Hatziadoniu, and G.D. Galanos. "Transient Simulation of Integrated AC/DC Systems Part I: Converter Modeling and Simulation," *IEEE Transactions on Power Systems*, Vol. 3, No. 1, February 1988, pp. 166-172.

[30] D.A. Woodford, A.M. Gole, and R.W. Menzies. "Digital Simulation of DC links and AC Machines," IEEE 1982 Summer Meeting, 82 SM 480-2

[31] K.R. Padiyar, Sachchidanand, A.G. Kothari, S. Bhattacharyya, and A.Srivatava. "Study of HVDC Controls Through Efficient Dynamic Digital Simulation of Converters," IEEE 1989 Winter Meeting, 89 WM 113-2 PWRD.

[32] G.D. Breuer, J.F. Luini, and C.C. Young. "Studies of Large AC/DC Systems on the Digital Computer," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-85, No. 11, November 1966, pp. 1107-1116.

[33] H.A. Peterson and P.C. Krause. "A Direct- and Quadrature-Axis Representation of a Parallel AC and DC Power System," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-85, No. 3, March 1966, pp. 210-225.

[34] D.P. Carroll and C.M. Ong. "Coordinated Power Modulation in Multiterminal HVDC Systems," IEEE 1980 Summer Meeting, 80 SM 670-0.

[35] G. Giannakopoulos, N.A. Vovos, T. Maris, and A. Lygdis. "A Fast and Flexible Method for Transient Simulation of Integrated AC/DC Systems," *IEEE Transactions on Power Systems*, Vol.3, No. 4, November 1988, pp. 1784-1792.

[36] B. Franken and G. Andersson. "Analysis of HVDC Converters Connected to Weak AC Systems," IEEE, 1989, 89 Sum 697-4 PWRS.

[37] C Wong, N. Mohan, S.E. Wright and K.N. Mortensen. "HVDC Converter Terminals," *IEEE Transactions on Power Delivery*, Vol. 4, No. 4, October 1989.

[38] J.F. Clifford and Albert H Schmidt, Jr. "Digital Representation of a DC Transmission System and Its Controls," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-89, No. 1, January 1970, pp. 97-105.

[39] C.E. Grund, T.H. Lee, S.R. Lightpost, R.J. Piwko, R.V. Pohl, K. Mortensen, R.J. Newell, J. Reeve and D.A. Woodford. "Functional Mode of Two-Terminal HVDC Systems for Transient and Steady State Stability," IEEE Committee Report, IEEE 1983, 83 SM 464-5.

[40] N.A. Vovos and G.D. Galanos. "Enhancement of the Transient Stability of Integrated AC/DC Systems using Active and Reactive Power Modulation," *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-104, No. 7, July 1985, pp. 1696-1702.

[41] A.E. Hammad and W. Kohn. "A Computational Algorithm for assessing Voltage Stability of AC/DC Interconnections," *IEEE Transactions on Power Systems*, Vol. PWRS-1, No. 1, February 1986.

[42] J. Kauferle, R. Mey and Y. Rogowsky. "H.V.D.C. Stations Connected to Weak A.C. Systems," *IEEE Transactons of Power Apparatus and Systems*, Vol. PAS-89, No. 7, September/October 1970, pp. 1610-1617.

[43] W.F. Tinney and J.W. Walker. "Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization," *Proceedings of the IEEE*, Vol. 55, No. 11, November 1967, pp. 1801-1809.

[44] R.D. Berry. "An Optimal Ordering of Electronic Circuit Equations for a Sparse Matrix Solution," *IEEE Transactions on Circuit Theory*, Vol. CT-18, No. 1, January 1971, pp. 40-50.

[45] N. Christi, P. Lutzelberger and K.Sadek. "Design Aspects and Applications of Damped AC-Filters for HVDC and SVC Connected Weak AC-Systems," EPE Aachen, 1989, pp. 479-784.

APPENDICES

Appendix A

These variables are saved at the end of the simulation if the option is taken. At the beginning of a new simulation, these values are restored and used as starting conditions.

Description	Variable
• The present time	t
• Rectifier Control	
Error from PI controller	Errold
The time of the last thyristor firing	Tzero
Ramp function used for sawtooth waveform	ramp
Actual firing angle	Alphafired
Gate pulse for thyristor	pulse
• Inverter Control	
Actual extinction angle	Gammanow
Error of the Proportional controller	ErrPI
Firing angle at last time step	Phi1
Time used to check for firing time	Clock
The time between thyristor firings	Tifp
• State variables	state
• Rectifier & Inverter Thyristor States	thystate
• The last States of thea Converter thyristor states	lastthystate
• The time at which each thyristor last turned on	Ton
• The time at which each thyristor last turned off	Toff
• Commutation Voltages for converters	VVcom2i Vcom2r
• Filtered commutation voltage of converter	Vcomf2
• Filter Variables	y1m1 y1m2 y2m1 y2m2 xsave y3save

Appendix B

A full set of loop equations is developed to describe the converter circuit assuming all thyristors are in a conducting state. The entire 12-pulse converter can be modeled using 10 loop equations. The final result will have the form

$$\left[V_{loop}\right] = \left[Z_{loop}\right] \left[I_{loop}\right]$$

The relationship between the loop voltages and currents is defined by Z_{loop}. V_{loop} and I_{loop} can be divided into sections, one for each 6-pulse bridge of the converter. As a result of this division, Z_{loop} can also be conceptually separated into four divisions.

$$\left[Z_{loop}\right] = \begin{bmatrix} Z^{Y}_{loop} & Z_c \\ Z^{t}_{c} & Z^{\Delta}_{loop} \end{bmatrix}$$

Z^{Y}_{loop} relates the Y-Y bridge voltages to the loop currents which are the links shown in Figure B.1. Z^{Δ}_{loop} relates loop voltages to the loop currents for the Y−Δ bridge. Z_c is the coupling matrix relating the loop currents and voltages between the two bridges. The Z_{loop} matrix is comprised of elements which can be defined in terms of the thyristor and transformer impedances.

A full Y-Y bridge is shown in Figure B.2 with the impedances labeled. The Δ bridge model is similar, but the links are chosen as in Figure B.1.

One example is given to demonstrate how the loop equations were developed. The remaining equations were defined in the same manner using different loops. The loop corresponding to the first equation is shown in Figure

B.3. The rest of the bridge circuit is not shown for the sake of clarity.

Writing the loop equation clockwise in terms of impedances, currents and voltage sources gives

$$e_A^Y + (I_{link1} - I_{link3})Z_a^Y + I_{link1}Z_1^Y + (I_{link1} + I_{link2} - I_{link5})Z_3^Y + (I_{link1} + I_{link2} - I_{link3} - I_{link4})Z_c^Y - e_c^Y = 0$$

Putting the voltages on the left side alone, and rearranging the other terms by link currents gives

$$e_A^Y - e_c^Y = (Z_a^Y + Z_1^Y + Z_3^Y + Z_c^Y)I_{link1} + (Z_3^Y + Z_c^Y)I_{link2} - (Z_a^Y + Z_c^Y)I_{link3} - Z_c^Y I_{link4} - Z_3^Y I_{link5}$$

Putting this in matrix form

$$\begin{bmatrix} e_{ac}^Y \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} (Z_a^Y + Z_1^Y + Z_3^Y + Z_c^Y) & (Z_3^Y + Z_c^Y) & -(Z_a^Y + Z_c^Y) & (-Z_c^Y) & (-Z_3^Y) \\ \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & & \end{bmatrix} \begin{bmatrix} I_{link1} \\ I_{link2} \\ I_{link3} \\ I_{link4} \\ I_{link5} \end{bmatrix}$$

Therefore

$$Z_{11} = Z_a^Y + Z_1^Y + Z_3^Y + Z_c^Y$$

$$Z_{12} = Z_3^Y + Z_c^Y$$

$$Z_{13} = -(Z_a^Y + Z_c^Y)$$

$$Z_{14} = -Z_c^Y$$

$$Z_{15} = -Z_3^Y$$

Repeating this process for the other nine loop equations gives the following

results. Only the non-zero elements are given, and the Z matrix is divided into the four quadrants defined earlier.

$$Z^Y_{22} = Z^Y_b + Z^Y_c + Z^Y_2 + Z^Y_3$$
$$Z^Y_{23} = -Z^Y_c$$
$$Z^Y_{24} = -Z^Y_b - Z^Y_c$$
$$Z^Y_{25} = -Z^Y_3$$

$$Z^Y_{33} = Z^Y_a + Z^Y_c + Z^Y_4 + Z^Y_6$$
$$Z^Y_{34} = Z^Y_c + Z^Y_6$$
$$Z^Y_{35} = -Z^Y_6$$

$$Z^Y_{44} = Z^7_b + Z^Y_c + Z^Y_5 + Z^Y_6$$
$$Z^Y_{45} = -Z^Y_6$$

$$Z^Y_{55} = Z^Y_3 + Z^Y_6 + Z_d$$

Z^Δ_{loop} is formed similarly, and the non-zero elements are given below.

$$Z^\Delta_{11} = Z^\Delta_1 + Z^\Delta_3$$
$$Z^\Delta_{12} = Z^\Delta_3$$
$$Z^\Delta_{15} = -Z^\Delta_c$$
$$Z^\Delta_{21} = Z^\Delta_b + Z^\Delta_3$$
$$Z^\Delta_{22} = Z^\Delta_b + Z^\Delta_2 + Z^\Delta_3$$
$$Z^\Delta_{23} = -Z^\Delta_b$$
$$Z^\Delta_{24} = -Z^\Delta_b$$
$$Z^\Delta_{25} = Z^\Delta_b$$
$$Z^\Delta_{33} = Z^\Delta_4 + Z^\Delta_6$$

$$Z^{\Delta}_{34} = Z^{\Delta}_{6}$$
$$Z^{\Delta}_{35} = Z^{\Delta}_{c}$$
$$Z^{\Delta}_{41} = -Z^{\Delta}_{b}$$
$$Z^{\Delta}_{42} = -Z^{\Delta}_{b}$$
$$Z^{\Delta}_{43} = Z^{\Delta}_{b} + Z^{\Delta}_{6}$$
$$Z^{\Delta}_{44} = Z^{\Delta}_{b} + Z^{\Delta}_{5} + Z^{\Delta}_{6}$$
$$Z^{\Delta}_{45} = -Z^{\Delta}_{b}$$
$$Z^{\Delta}_{51} = Z^{\Delta}_{a} + Z^{\Delta}_{b}$$
$$Z^{\Delta}_{52} = Z^{\Delta}_{b}$$
$$Z^{\Delta}_{51} = -Z^{\Delta}_{a} - Z^{\Delta}_{b}$$
$$Z^{\Delta}_{52} = -Z^{\Delta}_{b}$$
$$Z^{\Delta}_{51} = Z^{\Delta}_{a} + Z^{\Delta}_{b} + Z^{\Delta}_{c}$$

Z_c, the coupling matrix between the Y and Δ bridges, contains the following non-zero elements.

$$Z_{51} = - Z^{Y}_{3}$$
$$Z_{52} = - Z^{\Delta}_{3}$$
$$Z_{53} = - Z^{Y}_{6}$$
$$Z_{54} = - Z^{\Delta}_{6}$$

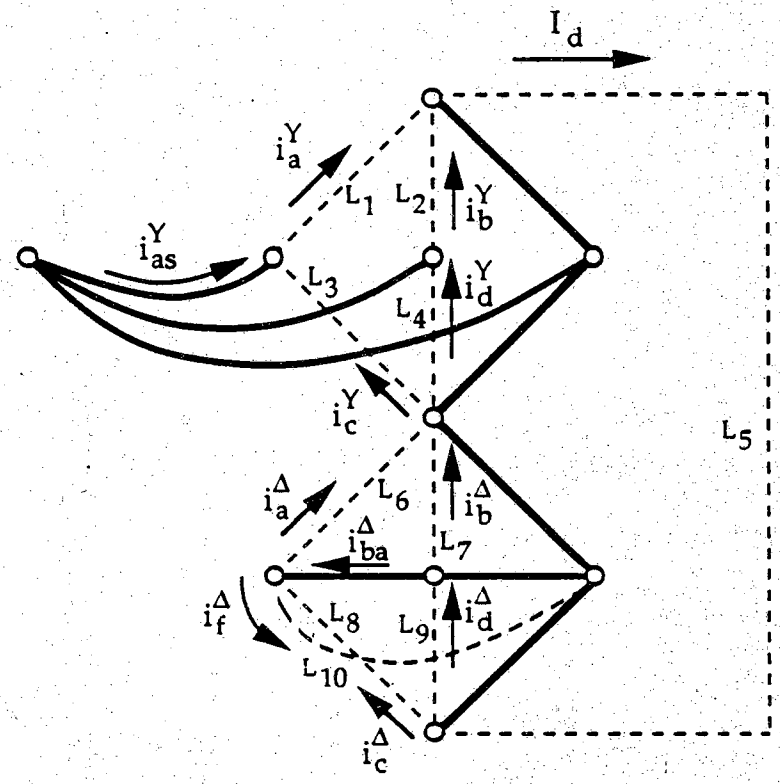


Figure B.1 Loop Diagram for a 12-Pulse Converter

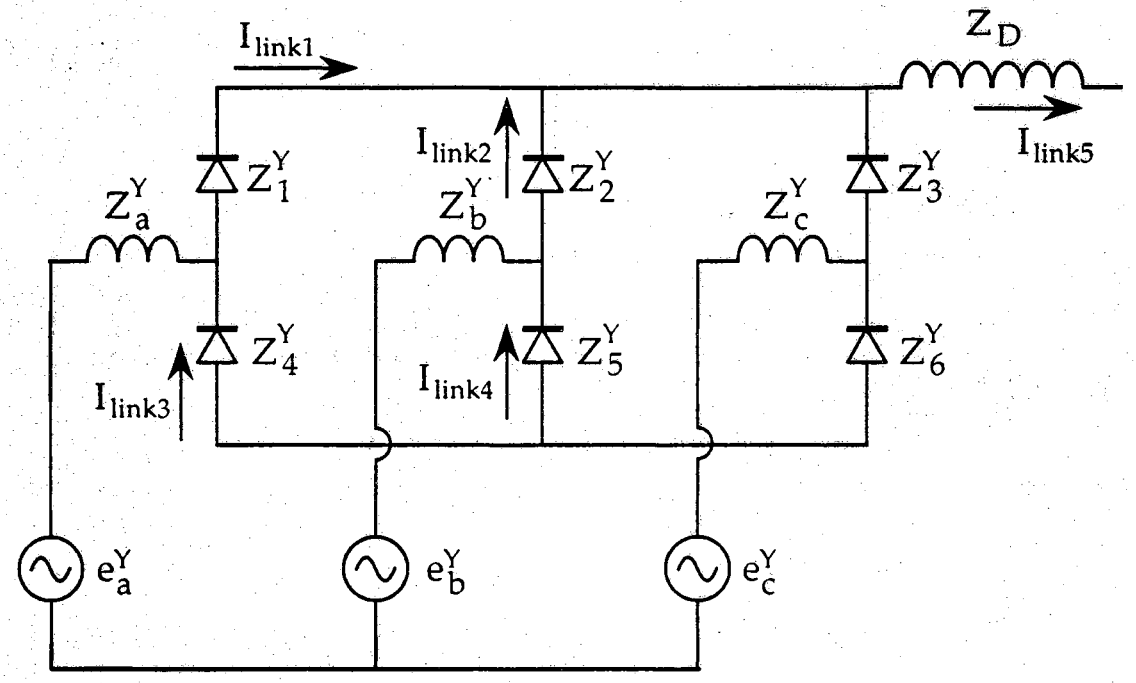


Figure B.2 12-Pulse Converter Diagram with Impedances

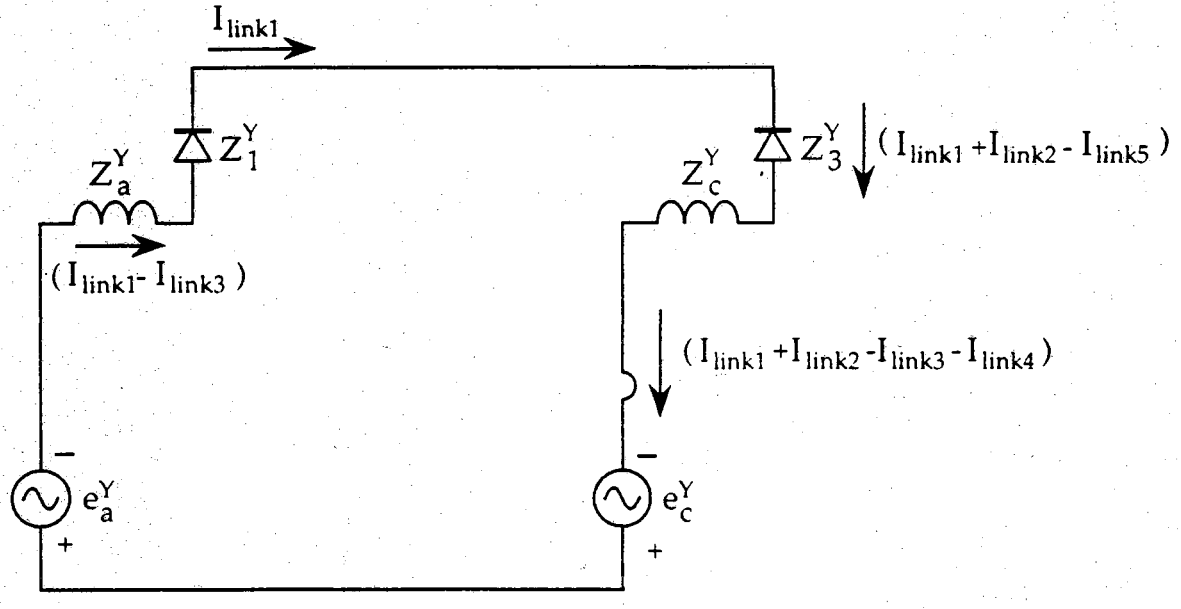


Figure B.3 Loop Diagram for First Loop Equation

Appendix C

The main program contains the initialization of system parameters, the initialization of operating conditions, the main loop subroutine calls, and the closing section. The main routine calls many other subroutines to do specific tasks. While the main routine contains all the stages of program flow, the other subroutines usually serve only one of the stages.

C.1 The Physical Parameter Initialization Subroutines

There are four subroutines called at the end of the system parameter initialization. The subroutine parameters are obtained exclusively from the system parameters.

C.1.1 Formation of the Converter Loop Equations

The subroutine "formRL" forms the full R and L matrices. The R and L matrices contain the coefficients of the loop equations which describe the converter. Both the R and L matrices are later reduced based on the states of the thyristors. A detailed derivation of the R and L matrices is given in Appendix B.

C.1.2 Creating Voltage and Current Transformations

The subroutine "formTx" forms transformation matrices which relate secondary AC voltages and currents to primary AC voltages and currents. The voltage transformation transforms the primary voltages such that the set of voltages used in the loop equations is returned. The current transformation relates the link currents to primary AC currents. The converter transformer

turns ratio is the only system parameter required.

C.1.3 Creating the Thyristor Voltage Transformation Matrix

The subroutine "formD" is used during the initialization of the system parameters. It forms a transformation matrix used to calculate the thyristor voltages. The transformation is based entirely upon the structure of the full converter circuit.

C.1.4 Creating the State Matrix for the AC Systems

The subroutine "formAfixed" is used to form part of the coefficient matrix for the state equations. There are 24 differential equations associated with each AC system. Since the physical properties are assumed to be constant throughout the simulation, the coefficients of these differential equations only need to be entered into the state matrix once for the entire simulation. This differs from the converter differential equations which change every time the converter topology changes.

C.2 Initialization of Operating Conditions

The subroutine "InitStates" initializes the state variables with the default values based on the physical parameters loaded during the first stage. Without a proper set of initial values, the simulation may have difficulty starting or may require a substantial time to reach steady state.

If the option to begin the simulation from the stored values of a previous run is chosen, the "InitStates" routine will not be used. The initial operating

conditions for the state variables recorded in the recovery file would be used instead.

C.3 Forming the Converter State Equations

This set of subroutines is used as part of the initialization and the reconstruction of the converter state equations. Therefore, they are executed only during the initialization process or during a change in the thyristor states of the converter.

C.3.1 Forming the Connection Matrix

The subroutine "formCn" forms the matrix which relates the full set of converter loop equations to the reduced set. The Boolean logic is employed to determine which of the links will be used. This logic is based on the present states of the thyristors. Figure C.2 shows the tree and link diagram upon which the logic is developed. The boolean equations to establish the various link conditions are given as:

$$
\begin{aligned}
\text{link}_1 &= x_1x_3 + x_1x_2 \\
\text{link}_2 &= x_2x_3 \\
\text{link}_3 &= x_4x_6 + x_4x_5 \\
\text{link}_4 &= x_5x_6 \\
\text{link}_5 &= x_1 + x_2 + x_3 \\
\text{link}_6 &= x_7x_9 + x_7x_8 \\
\text{link}_7 &= x_8x_9 \\
\text{link}_8 &= x_{10}x_{12} + x_{10}x_{11} \\
\text{link}_9 &= x_{11}x_{12} \\
\text{link}_{10} &= 1
\end{aligned}
$$

where x_j denotes the state of the thyristor j. The order of the thyristors is shown in Figure C.3.

The full-on circuit of a twelve pulse converter has ten loop currents corresponding to the ten links in Figure C.2. While inclusion of the entire set of links in the instantaneous network configuration is not a realistic condition, certain links are always included. For example, link 10, the delta transformer winding, is always on, as is link 5, the DC line. After determining which links will be used for the present topology, the connection matrix is built.

The reduced converter network connection matrix shows the relationship between the independent loop currents of the instantaneous converter topology and those of the completely connected network. Thus, generating the connection matrix begins by forming a 10 by 10 identity matrix. From here, the columns corresponding to those links not present in the reduced network are eliminated. Then it must be determined if any of the four branches containing a thyristor corresponds to a non-conducting thyristor. If this is the case, one of the other links must be made into a branch to replace this branch which is removed. When this substitution is completed, the connection matrix has been formed.

C.3.2 Reducing the Converter Loop Equations

Subroutine "ReduceRL" reduces the R_{loop} and L_{loop} matrices. In chapter 2 it was shown mathematically how the connection matrix reduced the set of loop equations to only those which were necessary to describe the converter.

$$Z_n = C_n^t \, Z_{loop} \, C_n$$

Reforming the equation for the time domain simulation gives:

$$Z_n = C_n^t(R_{loop} + pL_{loop})\ C_n$$

$$Z_n = C_n^t R_{loop} C_n + pC_n^t\ L_{loop} C_n.$$

Therefore, the connection matrix is used to reduce the R_{loop} and the L_{loop} matrices by multiplying them as:

$$R_n = C_n^t\ RC_n$$

and

$$L_n = C_n^t\ LC_n.$$

The reduction is done immediately after the connection matrix has been formed.

C.3.3 Forming the Converter State Equations

Subroutine "formAvar" forms the part of the state equation coefficient matrix, A, which holds the state equations for the given converter. The A matrix is divided into three sections for the two terminal simulations. One section describes the AC system whose equations do not change during the simulation; the second describes the rectifier; and the third describes the inverter.

This routine places the coefficients of both the rectifier and inverter state equations into the A matrix after the loop equations are reduced. The manipulation of the loop equations into state equation form occurs in the first part of "formAvar". (See Equation 2.13)

C.4 Main Loop Subroutines

Most of the subroutines are located in the main loop. This is also the stage of the program which is executed most often. Since these subroutines are executed most frequently, their efficiency is crucial.

C.4.1 Integrating the State Equations

The integration routine is used to integrate the state equations of the whole system. The numerical integration used for this simulation is a fourth order Runge-Kutta method. This is an explicit method of integration which predicts the value of the variable for the next time step based on previous values. This method of integration was chosen so that the state equation coefficient matrix would not have to be inverted as would be necessary if using an implicit method.

The integration is the first subroutine executed because the loop currents and voltages are updated based on the integrated values of the state variables at the most recent time step.

C.4.2 Calculating the Input to the System

State equations are written in the form:

$$\dot{x} = Ax + Bu$$

Called from within the integration procedure, "buildB" multiplies the input coefficient matrix, B, by the vector of system inputs, u. The B matrix is large as well as being sparse. It has a dimension of 57×3 for the two terminal case and is over 98% sparse. The input matrix, u, is comprised of the three-phase Thevenin voltage. The forcing function of the system is the 57×1 product of B and u, given by

$$Bu = \begin{bmatrix} 1/L_{Thev} & 0 & 0 \\ 0 & 1/L_{Thev} & 0 \\ 0 & 0 & 1/L_{Thev} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e^{A}_{Thev} \\ e^{B}_{Thev} \\ e^{C}_{Thev} \end{bmatrix}$$

This product is used in the integration routine.

C.4.3 The Multiplication of the State Matrix

The state matrix multiplication routine exploits the sparsity of A. It is developed to be used exclusively to multiply the n×n state matrix, A, with an n×1 matrix. The A matrix is formed by three parts in the case of the two terminal link. The multiplication is done in three steps because the A matrix is held in three separate arrays. This structuring was done so that one part of the A matrix could be changed without affecting the rest of the coefficients. It is the most efficient method computationally although it does not lend itself well to multiterminal simulations.

Since A is a very sparse matrix, only the non-zero elements were stored in an array of records. The record has the following structure shown in Figure C.1.

The sequence of steps for the multiplication begins by clearing the 57×1 resultant array. Next, the value of the coefficient in the record is multiplied by the "column of A" row in the 57×1 state variable matrix. This single product is added to the "row of A" row of the resultant array. The multiplication is repeated for each of the segments of A: the fixed equation coefficients, the inverter equation coefficients, and the rectifier equation coefficients.

C.4.4 Calculating the Thyristor Voltages

Subroutine "CalcVthy" calculates the voltages across the thyristors. The subroutine can be divided into four sections. First, the open-circuit loop voltages of the loops with non-conducting thyristors, V_x, are calculated. Next, the D matrix (the transformation between V_x and the thyristor voltages) is reduced based on the instantaneous topology. This is done by eliminating the columns corresponding to conducting thyristors and the rows corresponding to V_x's for which the loop remains closed. V_x is equal to zero for this case. Then the reduced D matrix, D_n, is inverted. Finally, the non-conducting thyristor voltages are calculated based on the product of D_n^{-1} and the open-circuit loops, $V_{x,n}$. The thyristor voltages are updated after the loop currents and voltages have been updated.

C.4.5 Controlling the Converters

A PI controller compares the DC current of the rectifier with a reference value. The difference is the input error. The PI controller delivers an output error signal to the second part of the control scheme. The second part of the control scheme, which determines when the next thyristor should be fired, is influenced by the output from the PI controller.

For the inverter, a proportional controller is used to monitor the error between the actual extinction angle, γ_{act}, and the desired extinction angle, γ_{des}. The output error from the proportional controller is sent to the firing section. The inverter control uses a different firing scheme than the rectifier.

C.4.6 Calculating the States of the Thyristors

Subroutine "CalcState" determines which thyristor states should be changed and tracks the time that each thyristor is turned on and off. the subroutine begins by calculating each thyristor current based on the loop currents. If the thyristor state is on and its predicted current becomes negative, then the thyristor state is turned off. A linear interpolation is performed to determine the time at which the thyristor turned off. This time is used to calculate γ_{actual} for the inverter.

If a thyristor is off, a different method is used to determine if it should be turned on. In the control schemes, a ring counter is maintained. The controllers determine when a thyristor should be turned on. If the ring counter element corresponding to a thyristor has a value of one, then the thyristor should be on. Therefore, if a thyristor is off, but the ring counter value is one, then the thyristor state is changed to be on. The time at which the thyristor was turned on is then recorded.

C.5 Utility Subroutines

There are two subroutines that are used as utilities in some of the subroutines.

C.5.1 A Matrix Inversion

A Gauss-Jordan matrix inversion which employs pivoting is used. It was written by Dr. Hornbeck at Carnegie-Mellon in 1975[26]. It has been translated

into C for this program. It is used in "formAvar" to compute L_n^{-1}. It was also used in "CalcVthy" to compute D_n^{-1}.

C.5.2 The Voltage Filter

This is a second-order digital filter taken from the original program written by Ali Hamzei-Nejad[12]. The purpose of the filter is to remove as much ripple as possible from the commutation voltages. The inverter control scheme watches for zero-crossings of the commutation voltage. Multiple zero-crossings due to ripples in the commutation voltage may cause problems in the control scheme. Therefore, the voltages are filtered so the ripples are eliminated.

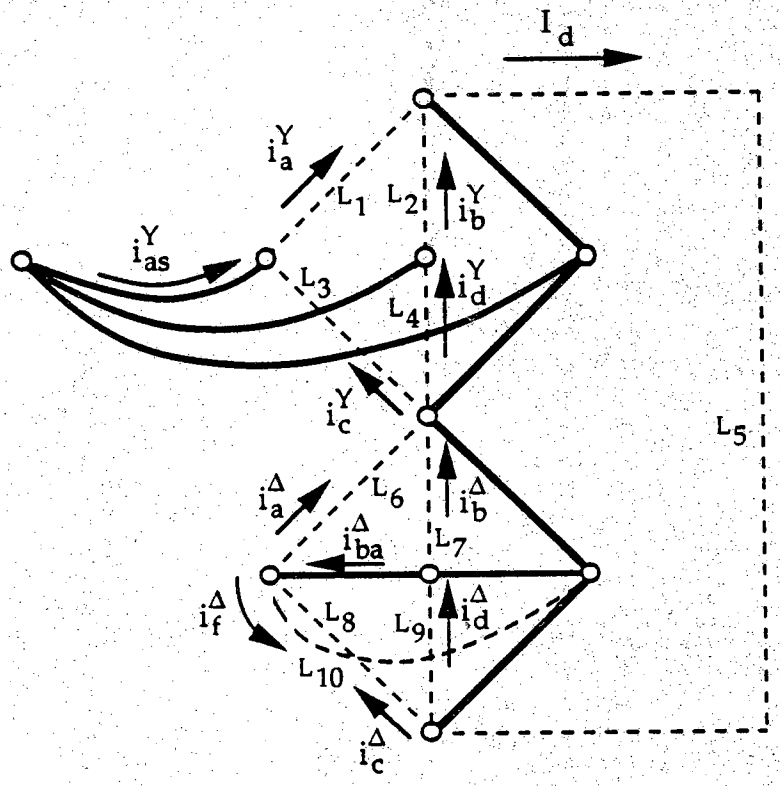


Figure C.1 Loop Diagram for a 12-Pulse Converter

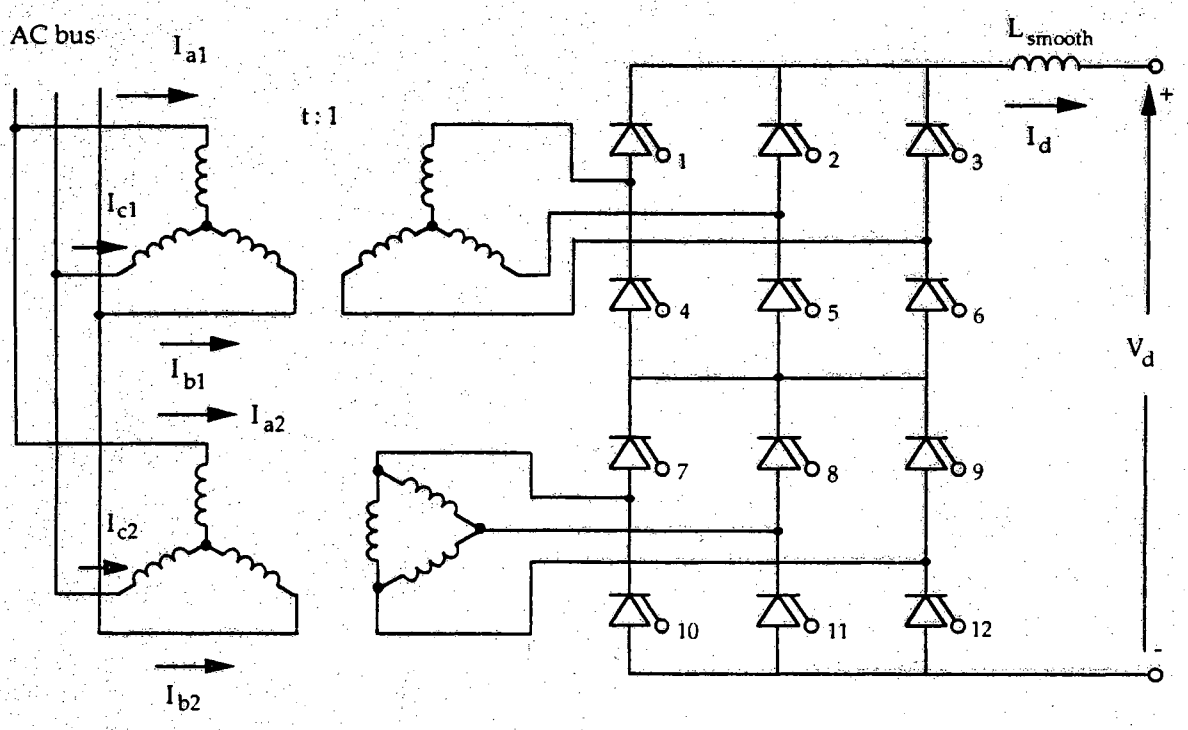


Figure C.2 12-Pulse Converter in State Order