USER'S GUIDE

A Comprehensive Resource for EMTDC

EMTDC

Transient Analysis for PSCAD Power System Simulation

MANITOBA·HVDC RESEARCH CENTRE

244 Cree Crescent, Winnipeg, Manitoba, Canada R3J 3W1

Print history:

- February 3, 2003 - Version 4.0.0, first printing.
- March 21, 2003 - Version 4.0.1
- July 17, 2003 - Version 4.0.2, second printing.
- April 30, 2004 - Version 4.1.0, third printing.
- April, 2005 - Version 4.2.0, fourth printing.

FOREWORD

What is EMTDC?

EMTDC is a powerful electro-magnetic transients simulation engine that has been evolving since the mid-1970s. Originally inspired by Dr. Hermann Dommel from his classic 1969 IEEE paper published in the Transactions of Power Apparatus and Systems, EMTDC development has been geared to many users in the power industry since inception.

Electro-magnetic transients simulation has changed greatly since the 1970's. The earliest versions of EMTDC were run at Manitoba Hydro on mainframe computers using punched cards. Only one or two cases a day could be submitted and run and so coding and program development were extremely slow compared with what can be accomplished today. As computers developed, more sophisticated file handling systems became available with text editors and cathode ray terminals. Today, the powerful personal desktop computers allow enable simulation that couldn't even be imagined 30 years ago.

Roots?

Many people have had a hand in developing EMTDC, and its graphical interface PSCAD. Many have come to refer to EMTDC, and PSCAD/EMTDC, as simply PSCAD. The original source code, and concepts developed in extending Dr. Dommel's equations to HVDC systems are the life's work of Dennis Woodford. Along with Dennis, many others have made significant contributions to PSCAD including the staff at the HVDC Research Centre, RTDS Technologies Inc, the University of Manitoba, Manitoba Hydro, and many other Research Centers and Universities around the world.

Who is using EMTDC?

PSCAD now has over 1000 licenses of the software worldwide. Many users in utility planning and operations, equipment manufacturers, consulting engineers, and research laboratories are all using this software on a day-to-day basis.

Where is EMTDC going?

EMTDC has proven to be very efficient and robust and is used to solve many engineering problems, some in novel ways. The Centre, in cooperation with the University of Manitoba and our users are continually advancing the EMTDC algorithm, components, models, and associated tools. We are constantly striving for more accuracy and speed to maintain our leading edge.

Our commitment

The Centre honors it commitment to excellence by ensuring that the PSCAD product is leading edge, and we are constantly looking for new ways to improve the technology to benefit our customers.

Paul Wilson, P. Eng.
Managing Director
Manitoba HVDC Research Centre
April 2005

Table of Contents

Table of Contents

About This Guide

The goal of this guide is to familiarize the PSCAD user with the EMTDC electromagnetic transients program, and to build a foundation for the continued study and simulation of electromagnetic transients using EMTDC.

The topics discussed in this guide are recommended for advanced PSCAD users. If you are using PSCAD for the first time, or your experience with PSCAD is limited, please review the PSCAD User's Guide before continuing on with these topics.

For more information on specific models, or the basic use and installation of PSCAD, please see the PSCAD User's Guide.

ACKNOWLEDGEMENTS

This guide is an accumulation of contributions from a variety of authors spanning the past decade. The Manitoba HVDC Research Centre would like to acknowledge and thank those who knowingly and unknowingly participated in its creation. In particular:

Dr. Ani Gole, *University of Manitoba*
Garth Irwin, *Electranix Corporation*
Dr. Om Nayak, *Nayak Corporation*
Dennis Woodford, *Electranix Corporation*

ORGANIZATION

The EMTDC User's Guide is organized in the following manner:

- **Chapter 1: Introduction** provides some answers to basic questions about EMTDC.
- **Chapter 2: Program Structure** discusses how the EMTDC main program is structured, including some important files involved.

- **Chapter 3: Electric Network Solution** describes how electrical elements and networks are represented in EMTDC. The topics range from lumped elements, to non-linear devices, switches and subsystems.

- **Chapter 4: Advanced Features** discusses the various features included in EMTDC to enhance speed and performance.

- **Chapter 5: Writing Your Own Models** provides an overview of important etiquette, formatting and features available for writing user models. Includes an example of a control type model.

- **Chapter 6: Interfacing Electric Models** describes the different methods for interfacing a user-written, electric type model to the EMTDC electric network solution.

- **Chapter 7: Transformers** provides a theoretical background to how transformers are developed and modeled in EMTDC.

- **Chapter 8: Rotating Machines** provides a theoretical background to how rotating machines are developed and modeled in EMTDC. This chapter also includes a general background on various, machine related control systems.

- **Chapter 9: Transmission Lines** provides a theoretical background to how transmission lines and cables are developed and modeled in EMTDC. Information on the Transmission Line and Cable Constants routines is also included.

- **Chapter 10: V2 Conversion Issues** discusses key issues for converting PSCAD V2 style, user-written Fortran subroutines. Includes information on filtering and manual tasks required for conversion.

- **References:** References to technical publications sited, as well as additional readings of interest.

- **Index:** Alphabetical index of keywords with page numbers.

DOCUMENTATION REQUIREMENTS

The following general conventions are followed throughout this manual:

References
References are cited using 'boxed brackets.' For example, referring to Reference #5 would appear as [5].

Text Boxes
All code examples will appear in text boxes, as shown below:

```
CODE
```

Notes
All note text will appear in text boxes, as shown below:

> **NOTE** text.

Chapter 1:

Introduction

One of the ways to understand the behaviour of complicated systems is to study the response when subjected to disturbances or parametric variations. Computer simulation is one way of producing these responses, which can be studied by observing time domain instantaneous values, time domain RMS values, or the frequency components of the response.

EMTDC is most suitable for simulating the time domain instantaneous responses (also popularly known as electromagnetic transients) of electrical systems.

The power of EMTDC is greatly enhanced by its state-of-the-art Graphical User Interface called PSCAD. PSCAD allows the user to graphically assemble the circuit, run the simulation, analyze the results, and manage the data in a completely integrated graphical environment.

WHAT IS EMTDC?

EMTDC (which stands for Electromagnetic Transients including DC) represents and solves differential equations (for both electromagnetic and electromechanical systems) in the time domain. Solutions are calculated based on a fixed time step, and its program structure allows for the representation of control systems, either with or without electromagnetic or electromechanical systems present.

The first lines of code were written in 1975, at Manitoba Hydro by Dennis Woodford (Executive Director of the Centre 1986 - 2001), out of a need for a simulation tool that was sufficiently powerful and flexible to study the Nelson River HVDC power system in Manitoba, Canada.

Following the success of this study, development of the program continued through the next two decades. Over this time, a full spectrum of professionally developed models was eventually accumulated (as needed for various simulation projects), in addition to various enhancements to the actual solution engine itself.

EMTDC now serves as the electromagnetic transients solution engine for the PSCAD family of products. PSCAD is used extensively for many types of AC and DC power simulation studies, including power electronics (FACTS), sub-synchronous resonance and lightning over-voltages (to name a few).

TIME VS. PHASOR DOMAIN SIMULATION

EMTDC is a class of simulation tool, which differs from phasor domain solution engines, such as load-flow and transient stability programs. These tools utilize steady-state equations to represent electrical circuits, but will actually solve the differential equations of machine mechanical dynamics.

EMTDC results are solved as instantaneous values in time, yet can be converted into phasor magnitudes and angles via built-in transducer and measurement functions in PSCAD - similar to the way real system measurements are performed.

Since load-flow and stability programs work with steady-state equations to represent the power system, they can output only fundamental frequency magnitude and phase information. EMTDC can duplicate the response of the power system at all frequencies, bounded only by the user-selected time step.

TYPICAL EMTDC STUDIES

EMTDC (with PSCAD) is used by engineers and scientists from utilities, manufacturers, consultants, and research/academic institutions, all over the world. It is used in planning, operation, design, commissioning, tender specification preparation, teaching, and advanced research.

The following is a sample of the types of studies routinely conducted using EMTDC:

- Contingency studies of AC networks consisting of rotating machines, exciters, governors, turbines, transformers, transmission lines, cables, and loads.
- Relay coordination.
- Transformer saturation effects.
- Over-voltages due to a fault or breaker operation.

- Insulation coordination of transformers, breakers and arrestors.
- Impulse testing of transformers.
- Sub-synchronous resonance (SSR) studies of networks with machines, transmission lines and HVDC systems.
- Evaluation of filter design.
- Harmonic analysis including resonance.
- Control system design and coordination of FACTS and HVDC, including STATCOM, VSC, etc.
- Variable speed drives of various types, including cycloconverters and transportation and ship drives.
- Optimal design of controller parameters.
- Industrial systems including compensation controllers, drives, electric furnaces, filters, etc.
- Investigation of new circuit and control concepts.
- Lightning strikes, faults or breaker operations.
- Steep front and fast front studies.
- Investigate the pulsing effects of diesel engines and wind turbines on electric networks.

EMTDC VS. OTHER EMTP-TYPE PROGRAMS

The electric network solution in EMTDC and some other EMTP-type programs, are based on the principles outlined in the classic 1969 paper by Hermann Dommel [1]. However, EMTDC has been independently developed from these other programs.

Virtually all power system models and techniques used in other EMTP-type programs are available in EMTDC. Some of the major differences between EMTDC and the other programs are listed as follows:

- Preparation and testing time is reduced due to the PSCAD Graphical User Interface.
- In EMTDC, many series and parallel electric elements are mathematically collapsed (such as an RLC branch) to reduce the amount of nodes and branches.
- The Optimal Ordering algorithm in EMTDC serves to increase LDU matrix decomposition speed.
- The Optimal Switch Ordering algorithm ensures that switching operations are very fast and efficient by moving switching elements to optimal conductance matrix positions.

- EMTDC utilizes subsystems, which takes advantage of the fact that the numerical solution of electric networks, separated by travelling wave transmission lines, are mathematically independent.
- An Interpolation algorithm is used in EMTDC to perform switching operations. This allows any switching event to occur at the exact switching instant, even if this instant is between time steps. This allows EMTDC to run at a larger time step (faster), yet maintain accurate results. Also, additional snubber circuits are not needed to address inherent numerical troubles.
- EMTDC uses a Chatter Removal algorithm (related to the Interpolation algorithm) to remove these unwanted oscillations.
- EMTDC does not restrict how circuit elements can be combined. Users can place any number of switching elements, sources, etc. in series or in parallel.
- EMTDC switching devices and sources can be ideal (i.e. 0 resistance) or non-ideal (where the user can enter the on/off resistance values).
- EMTDC users can easily write their own models, from very simple to very advanced. We provide an inherent interface to all main program variables and storage elements, which allows direct access for users.
- EMTDC users can write in Fortran, C and MATLAB.
- The EMTDC program takes advantage of the new Fortran 90/95 standard, which allows it to dynamically allocate memory at the beginning of each run.
- Initialization of systems with a Snapshot File. This initialization technique is very fast, and works for very large systems. It is the only practical method when highly non-linear systems (such as systems with HVDC and power electronics) are represented.
- Transmission Line and Cable models are superior in EMTDC.
- Full-time, professional support services are provided for EMTDC by the Manitoba HVDC Research Centre Inc.

EMTDC USER'S GROUP

The EMTDC Users' Group is an informal forum for PSCAD/EMTDC users worldwide. The group maintains a web site and email list server.

There are other regional user groups as well. Please contact the EMTDC Users' Group (details below) or the PSCAD web site (www.pscad.com/Agents) to find out if your region has one.

Membership
To become a member of the list server you should be an EMTDC user. There is no membership fee.

To add your name to this worldwide Users' Group list, contact the group coordinator, via email at emtdcug@ee.umanitoba.ca.

Facilities
The EMTDC users group maintains the following facilities:

Facility	Address
E-Mail List Server	emtdcug@ee.umanitoba.ca
Web Site	pscad_ug.ee.umanitoba.ca

E-Mail List Server
Users can submit their questions regarding PSCAD or EMTDC by sending an email to emtdcug@ee.umanitoba.ca. Your email will get distributed to all the members automatically. Members who have a reply to your query send an email to emtdcug@ee.umanitoba.ca. Often there may be multiple replies with multiple solutions to your problems.

The Manitoba HVDC Research Centre Inc. is also a member of the users group and hence PSCAD support staff at the Centre will get a copy of all the emails sent to the users group.

Web Site
The worldwide web site, maintained by the users' group, contains valuable contributions from users. These may include user-written models not yet available in the standard releases of PSCAD.

Technical Support
The Manitoba HVDC Research Centre Inc. is committed to providing the best technical support possible. Precedence however, is given to commercial PSCAD users. We can be reached at:

Facility	
E-mail	support@pscad.com
Phone	+1 (204) 989-1240
Fax	+1 (204) 453-5074
Web Site	www.pscad.com
Address	PSCAD Technical Support Services 244 Cree Crescent Winnipeg, Manitoba R3J 3W1 Canada

Chapter 2:

Program Structure

PROGRAM STRUCTURE BACKGROUND

When computer programs were first developed for power system analysis, the computers available were the old mainframe type used by the company for their accounts and billing. A team of specialists, whose function was to oversee all transactions involving the system, also serviced the system. The technical user was but a slave to the whims of these specialists, whose power lay in their ability to speak the system language: The interpretation of a mysterious jumble of words known only as 'JCL' was a jealously guarded secret. Consequently, the 'end user' (or more humiliatingly the 'client') was kept in hand and normally suffered through the computer programs so condescendingly provided.

These computer programs were supposedly structured to anticipate, as the programmers hoped, the every need of the technical user. Model construction and variation was accomplished by data entry. Unfortunately, it was impossible to anticipate all needs. Developments in power system technologies out-sped advancements in the computer programs. Because of these limitations in program flexibility, the programs always lagged in representation of the emerging power system. The creative potential of the technical specialist could not be fully realized with such program modeling restrictions.

Change was on the way. Minicomputers and later personal computers came on the scene with new operating systems, which made all the hocus-pocus associated with using large scientific programs on the old mainframes, obsolete. Virtual memory replaced overlays. New graduates, no longer intimidated by the computer, were more than capable of dealing with any programming situation. Today, the technical expert has control over both the computer and its programs and is intimately involved in adapting both hardware and software to meet his/her requirements for investigation and analysis.

EMTDC was developed for this new and more acceptable environment, first specifically for UNIX/Linux, and then on to Microsoft Windows operating systems. Model building of control and electric systems is no longer defined entirely by data entry - with the pro-

gram hidden, untouchable somewhere in the depths of the computer memory. Now the model is created by user Fortran statements. Time domain model components, such as AC machines, exciters, six-pulse valve groups etc., are modularized into subroutines for easy assembly by the user. The user is able to build his/her own subroutines if models are not available.

With the advent of PSCAD, the graphical user interface for EMTDC, there is for the most part, no longer any need for the user to build his/her own subroutines through coding. Subroutine assembly and insertion into EMTDC is now performed automatically by PSCAD. The responsibility of the user has been reduced to graphically assembling a given network, and in some cases, adding some user-defined code. However, there will still be a need in some instances, such as the conversion of older subroutines, where a good knowledge of the internal structure and methods of EMTDC is important.

MAIN PROGRAM STRUCTURE

To allow for flexibility in programming, the EMTDC solution engine is structured with a main program, which coordinates all activities of input and output and network solution, as well as interfacing to user defined Component Definitions.

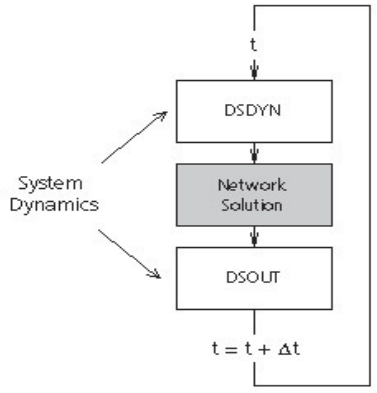

Figure 2-1 - EMTDC Main Program Structure

The EMTDC main program consists of three main parts: The System Dynamics section, which includes the master dynamics subroutine (DSDYN) and the output definition subroutine (DSOUT), and the

Electric Network Solution. These two components are called every time step sequentially, as illustrated in Figure 2-1,

Where,

t = Time

Δt = Time step

User defined code can be added to the System Dynamics section (either DSDYN and DSOUT) through the use of a combination of Fortran statements and PSCAD Script, or by a direct subroutine call from a Component Definition. In this way, the user has access to most EMTDC features and routines, creating a very flexible simulation environment.

Of course, the complete EMTDC solution sequence is much more complicated than that shown in Figure 2-1. There are many other features included, to ensure solution accuracy and speed, which will be discussed in the following sections and chapters.

SYSTEM DYNAMICS

As mentioned earlier, the primary purpose of the System Dynamics section is to provide the user with the ability to insert his or her code directly into the EMTDC main program. At the same time, the System Dynamics section offers the additional advantage in that electrical elements need not be present in a simulation. That is, EMTDC can be run without the presence of any electrical components or networks.

For example, a linear control system can be analyzed by using classic control theory. If non-linearities, such as limits, dead-bands, saturation functions, switches, etc. are part of the controller, then time domain analysis of such a model can be built into the System Dynamics, providing a practical solution method with or without electric circuits.

DSDYN and DSOUT Subroutines

The Systems Dynamics section is split-up into two sub-sections, effectively encompassing the network solution (see Figure 2-1). These are the DSDYN and DSOUT subroutines. The fact that Systems Dynamics can be utilized both before and after the electric network is solved, offers a great advantage in program flexibility. If judicially

selected, the user can avoid time step delays between control and electrical systems, for instance.

Although there are no major differences between DSDYN and DSOUT (i.e. they can be called interchangeably), there are specific uses for each. In general, most control dynamics code should be placed in DSDYN. DSOUT is primarily used to output quantities after the network solution is calculated. The combination of these two subroutines allows the user to decide whether certain control dynamics should be calculated using pre or post solution quantities.

Time Delays and Code Placement

In some instances, it is important to establish the proper use of DSDYN and DSOUT to avoid unnecessary time step delays in the control dynamics.

Meters used for the measurement of voltage and currents for example, are primarily defined in DSOUT. This is the logical subroutine to choose, as DSOUT will supply the measured quantities directly following the network solution. However, measured quantities are often used as inputs to control systems, where the individual control components are defined in DSDYN. Therefore, the control system may be basing outputs on quantities defined in the previous time step.

EXAMPLE 2-1:

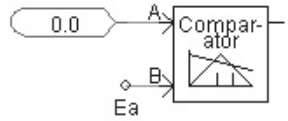

Figure 2-2 - Simple Comparator Circuit created in PSCAD

Consider the simple circuit in Figure 2-2. A comparator is being used to compare a real constant with a signal ‘Ea,’ which is a measured voltage from a voltmeter. The comparator is adjusted so as to give a high output when ‘Ea’ becomes greater than 0.0, and a low output otherwise. The comparator dynamics are defined by default in DSDYN, whereas the measured voltage ‘Ea’ is defined in DSOUT as an output quantity.

With the above configuration, the comparator output signal will contain a single time step delay. This is due to the fact that the comparator dynamics in DSDYN are using a measured voltage 'Ea,' defined in DSOUT from the previous step (see Figure 2-1).

This time step delay can be removed by simply moving the comparator dynamics code from DSDYN to DSOUT. This will force the comparator to use the measured voltage from the present time step, before determining its output.

Assembly of System Dynamics Code (Fortran File)

When a Case Project is compiled in PSCAD, the System Dynamics is automatically assembled into formatted Fortran code. The information required to construct the Fortran Code is extracted from the DSDYN and/or DSOUT segments of each Component Definition present in the case, whether defined in the Master Library or in a user defined library.

Figure 2-3 illustrates how PSCAD prepares and constructs the Fortran code for EMTDC. The PSCAD Case Project shown contains a default main page with a Component Instance and a Page Module. When the case is compiled, an equivalent Fortran file called 'main.f' is created, which acts as the main routine for the case. Within it are placed the DSDYN and DSOUT subroutines (System Dynamics), which contain component code and/or calls to user-defined subroutines, specified in the Component Definitions.

Each sub-page is treated as a subroutine and receives its own Fortran file. As shown in Figure 2-3, the sub-page called 'abc' receives a corresponding file called 'abc.f.' Within this file are placed local DSDYN and DSOUT subroutines, which are renamed to abcDYN and abcOUT (i.e. Page Module name replaces DS). abcDYN and abcOUT are then called from DSDYN and DSOUT in 'main.f'.

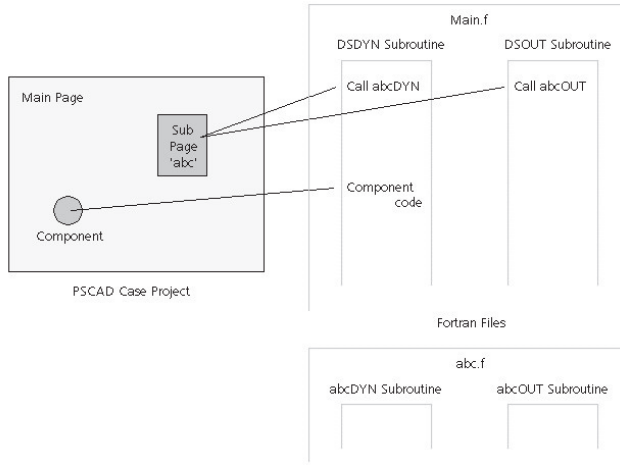

Figure 2-3 - Assembly of DSDYN and DSOUT Subroutines

This method of 'nested' routines continues on if further levels of sub-pages exist. For instance, if another sub-page exists within 'abc,' then the DSDYN and DSOUT subroutines, local to this new sub-page, will be called from abcDYN and abcOUT in 'abc.f.'

> **NOTE**: Sub-pages (or Page Modules) should not be confused with a subsystem. For more on subsystems in EMTDC, see Subsystems in Electric Networks.

EXAMPLE 2-2:

As an illustration as to how the component code is inserted into DSDYN, consider the simple Multiplier component, shown in Figure 2-4:

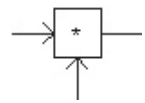

Figure 2-4 - Multiplier Component in PSCAD

The code for this device exists as PSCAD Script in the DSDYN section of the Component Definition. It is shown below for easy reference:

```
!
      $OUT = $IN1 * $IN2
!
```

When the PSCAD Project is compiled, the above script is extracted from the Multiplier Component Definition and converted into standard formatted Fortran, and then inserted as part of the DSDYN subroutine by PSCAD.

Assuming that this component resides on the main page of the Case Project, then the above script would appear in the 'main.f' as shown below, with variables renamed for consistency in EMTDC:

```
!
      RT_1 = RT_2 * RT_3
!
```

NETWORK SOLUTION

EMTDC can be classified as a 'hybrid simulator,' in that both control dynamics code, and electric networks can be included simultaneously in the main program.

As discussed in the previous section, systems dynamics code is compiled into a formatted Fortran file, which contains both the DSDYN and DSOUT subroutines. The electric network (if it exists) however, is defined as part of the EMTDC main program, from which DSDYN and DSOUT are called.

The electric network is not accessed in the same manner as the System Dynamics section. That is, the user does not add script code directly into it. Electric network parameters, such as node and subsystem numbering, are defined automatically by PSCAD, according to how the network is graphically constructed. This information is compiled and summarized into files by PSCAD for use by EMTDC.

Data File
The Data file is to the electric network solution, what DSDYN and DSOUT are to the System Dynamics section. Information regarding which nodes branch elements are connected between, the type of branch elements used, what their values are, etc., is included in the

Data file. In addition, distributed transmission line and transformer matrix information is also summarized here.

Each Page Module existing in a PSCAD Project File will receive its own Data File. Therefore, only node numbers local to that Page Module are used to reference the branches and branch values.

The Data File is automatically built by PSCAD according to the user's graphical representation of the circuit, including some information specified in the Model-Data segment of existing Component Definitions.

> **NOTE:** The Data File is only read by PSCAD if the simulation is initialized from time t = 0.0.

Local Node Voltages

The purpose of the Local Node Voltage section is to specify pre-defined initial voltages at the nodes indicated within the section. Although EMTDC is capable of accepting initial node voltages, PSCAD has not yet been given this functionality, and so it is currently not operational. An example of how the Local Node Voltages section appears in the Data File is shown below:

```
!---------------------------------------
! Local Node Voltages
!---------------------------------------
VOLTAGES:
   1             0.0  / NT_2
   2             0.0  / NT_4
```

The example above indicates that there are two nodes in this particular page of the PSCAD Project File. Both initial node voltages are set to 0.0. NT_2 and NT_4 indicate the default EMTDC Node Names, which may also be defined through the use of Node Label in PSCAD.

> **NOTE:** A 'Local Branch Currents' section may also be added in a future upgrade, to provide the ability to define initial branch currents.

Local Branch Data

The Local Branch Data section is used to define what exists in a branch between two particular nodes.

The following example Local Branch Data section on the next page indicates that there are four branches in this particular Page Module of the PSCAD Project File. For example, the first branch is shown to be between local nodes 1 and 0 (ground), and contains R, L and C elements. The values of these elements are shown to be 10 Ω, 0.0265 H, and 1.0 µF respectively. As before, the default EMTDC Node Names are at the far right.

The second and third branches are defined as being switching branches (RS) with an OFF resistance of 1 MΩ. The last branch is an ideal voltage source (RE), indicated by the 0.0 Ω resistance.

```
!---------------------------------------
! Local Branch Data
!---------------------------------------
BRANCHES:
   1   0   RLC    10.0         0.0265   1.0     / NT_2  GND
   2   1   RS     1000000.0                     / NT_4    NT_2
   1   2   RS     1000000.0                     / NT_2    NT_4
   0   2   RE     0.0                           / GND     NT_4
```

The following table summarizes the symbol definitions used in the Local Branch Data section:

Branch Symbol	Definition
R	Resistance
L	Inductance
C	Capacitance
S	Switching branch
E	Source branch

Table 2-1 - Symbols Used in the Local Branch Data Section of the Data File

> **NOTE:** Combinations of the symbols will appear if a particular branch contains more than one element. For example, an inductive source branch would appear as 'LE.'

Local Transformer Data

The Local Transformer Data section is used for the definition of the transformer mutual inductance matrix. Other text comments, regarding certain transformer parameters, are also included.

The following example Local Transformer Data section shows that the existing transformer, is a non-ideal, two-winding transformer as indicated by the '2' in the first non-commented line. If the transformer were ideal, this would appear as '-2'. The next two lines define the R and L values of the mutual inductance matrix in the following format:

```
R11  L11
R21  L21 R22  L22
```

> **NOTE**: The diagonal elements of this matrix are equal, and are therefore not stated.

```
!---------------------------------------
! Local Transformer Data
!---------------------------------------
TRANSFORMERS:
!  3 Phase, 2 Winding Transformer
!*  Name:  T1 Tmva: 100.0 MVA, Freq: 60.0 Hz, V1: 13.8 kV, V2: 230.0 kV
!*  Imag1: 0.01 p.u.,    Imag2: 0.01 p.u.,   Xl: 0.1 p.u.
!*  Sat: 0 ,
  2 / Number of windings...
  4 5    0.0 1.51547336812 /
  3 0    0.0 14.5753579593   0.0 140.321608159 /
 888 /
  5 6 /
  2 0 /
 888 /
  6 4 /
  1 0 /
!
```

The ‘888’ symbol indicates that the following lines will have the same values as those above, with different local node number connections.

DATADSD and DATADSO

The purpose of the DATADSD and DATADSO sections is to allow the user access to the Data File. These sections work in conjunction with the Model-Data segment in the Component Workshop. That is, any information added in the Model-Data segment of a component, will appear here.

For example, the machine models in PSCAD use this section to define variables according to selected parameters. Therefore, if a PSCAD Case file containing a machine is compiled, data will appear in this section when the Data File is viewed.

Map File

The Map File is used to display information common to the entire PSCAD Project File, as well as to act as the ‘glue’ linking each Data file from each Page Module together. Its key role is to provide node ‘look-up table’ information in order to convert the local node number index from each Page Module to a global one.

This feature is critical to allow for incremental builds. That is, without it, EMTDC would require a complete re-build for each circuit change.

> **NOTE**: Incremental builds are only available in EMTDC versions used after PSCAD V2.

Dimensioning Information

The Dimensioning Information section of the Map File simply lists how the PSCAD Project File has been dynamically dimensioned. An explanation of the dimensions listed is given in Table 2-2:

Dimension	Definition
NPAGES	Total number of Page Modules
SUBSYS	Total number of subsystems
NNODES	Total number of electrical nodes
NODES	Maximum number of electrical nodes per subsystem
BRANCHES	Maximum number of electrical branches per subsystem
TRAFOS	Total number of transformers
WINDINGS	Maximum number of windings per transformer
VARS	*Obsolete*
PGBS	Total number of output channels
STOR	Total number of STOR locations used
STORL	Total number of STORL locations used
STORI	Total number of STORI locations used
STORF	Total number of STORF locations used
STORC	Total number of STORC locations used
STOL	Used internally by EMTDC (not accessible)
STOI	Used internally by EMTDC (not accessible)
STOF	Used internally by EMTDC (not accessible)
STOC	Used internally by EMTDC (not accessible)

Table 2-2 - Definitions for Dimensioning Information in the Map File

NOTE: These dimensions are NOT relevant if using the free EGCS/GNU Fortran 77 compiler. This compiler imposes pre-defined, fixed dimensions for every case compiled.

Runtime Parameters

The Runtime Parameters section of the Map File summarizes information regarding the actual simulation, as well as details on the advanced option configuration for the project. The definitions of these are summarized in Table 2-3:

Dimension	Definition
TITLE	The PSCAD case project description
VERSION	The EMTDC version used
START_TIME	The simulation start time
TIME_STEP	The time step used
PRINT_STEP	The plot step used
CHATTER_LEVEL	The threshold by which to detect chatter
SHORT_CIRCUIT	The threshold by which to use ideal branches
DETECT_CHATTER	Detect chatter yes or no
REMOVE_CHATTER	Remove chatter yes or no
INTERPOLATE	Interpolate the solution yes or no
EXTRAPOLATE	Extrapolate sources yes or no
ECHO_DATA	Write data file and map file information to the message tree
PRINT_DIMENSIONS	Write the project dimensions to the message tree

Table 2-3 - Definitions for Runtime Parameters in the Map File

> **NOTE**: Regarding Extrapolate Sources - 'No' means to use a linear extrapolation method. 'Yes' is more accurate. See Extrapolate Sources for more details.

Node Mapping Information

The Node Mapping Information section is the most useful section in the Map File, providing a 'map' of the existing electrical nodes in the PSCAD Project.

The following example Node Mapping Information section illustrates the results of a PSCAD Case, which contains two subsystems located in the main page. As shown, the information from each subsystem (labelled 'SS_1' and 'SS_2') is extracted from each Data File ('SS_1.dta' and 'SS_2.dta'), and contains a total of six electrical nodes each.

```
!==========================================
! Node mapping information
!------------------------------------------------------------------
--
MAP: "Main.dta" ! Main Page
     0       6     1   2   3   1   2   3  /
     0       2     1   1   1   2   2   2
MAP: "SS_1.dta" ! SENDING END
     0       6     4   5   6   1   2   3  /
     1       1     1   1   1   1   1   1
MAP: "SS_2.dta" ! RECEIVING END
     0       6     1   2   3   4   5   6  /
     2       1     2   2   2   2   2   2
```

The first of data, under each 'MAP:' heading, is used for node number information. The second line of data contains subsystem information.

For example, the above indicates that the main page contains a total of six electrical nodes and a total of two subsystems. The electrical nodes are numbered separately according to the subsystem in which they reside (that is, each node number has a corresponding subsystem number directly beneath it).

> **NOTE**: The main page itself is NOT considered a subsystem and is therefore referred to as 'subsystem 0.'

Global Transmission Lines

The Global Transmission Lines section summarizes some information about existing transmission lines in the PSCAD Project.

```
!---------------------------------------
! Global Transmission Lines
!---------------------------------------
GLOBAL_TLINES:
PSCAD Line Constants
  3  0
1 3 2 1
2 1 5 6
TLINE-INPUT-DATA  FLAT230.tli
TLINE-OUTPUT-DATA FLAT230.tlo
```

The above Global Transmission Lines section example indicates that a single transmission line exists in this Project. The first data line includes a '3' and a '0.' The '3' indicates the number of conductors on this line.

The first number, in the second and third data lines, indicates the subsystem number. The remaining numbers represent the sending and receiving end, local node numbers respectively.

Finally, the respective transmission line input and output file names are shown.

PGB Name Information

The PGB Name Information section of the Map File simply summarizes the Output channels used in the Project. Some information on each Output Channel is also listed.

> **NOTE**: A PGB is the old naming convention for an Output Channel.

INITIALIZATION AND INITIAL CONDITIONS

Starting a simulation and bringing it to steady state, can prove to be a valuable exercise in itself. The process can indicate how robust the models included are and, if the simulation fails to reach steady state (i.e. diverges), provides a warning that problems exist that require attention.

There are generally two ways to start a simulation: Start from time t = 0.0 with no initial conditions (i.e. start from the Data File created

when the case is first compiled) or start with pre-calculated initial conditions imposed on some or all elements. In PSCAD, starting a simulation with initial conditions is achieved by using a Snapshot File.

The Snapshot File
A Snapshot File can be created by starting a simulation at time = 0.0, allowing it to settle to a steady-state condition, and then freezing all states and variables to a file by taking a snapshot. A Snapshot is essentially a new Data File with everything initialized - the user may then re-start the simulation from this file.

The Snapshot file method can be used to impose initial conditions on energy storage devices (i.e. capacitors and inductors), or memory functions involving integration when present in a simulation.

EXAMPLE 2-3:

As a simple example, suppose you wish to study the effects of inductance on the diode decay current if the switch in the circuit of Figure 2-5 opens at time t = 0.1 s.

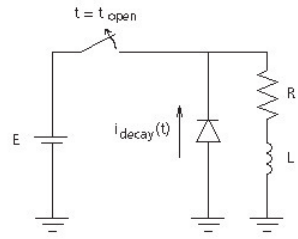

Figure 2-5 - Simple RL Circuit with Freewheeling Diode

Depending on the size of the inductance, it will take a finite amount of time for this simple circuit to reach steady state (if started from time zero). When the switch does actually open, the current flowing in the inductor will decay through the freewheeling diode.

In order to study the effect of the inductor size on the decay time of the current, the simulation may need to be rerun many times, each time changing the inductance value. Since the time to reach steady

state would be of no consequence to the study, then removing this time would be advantageous in reducing the total simulation time.

This can be accomplished in EMTDC by taking a snapshot at a time just previous to the switch opening. Subsequent runs could be started from this snapshot file, which would have stored, among other values, the current flowing in the inductor at the time of the snapshot.

Example 2-3 is a very simple illustration for the use of the Snapshot file, where the time to reach steady state would be in the order of milliseconds, and of not much importance. However, Snapshot Files become advantageous when highly non-linear systems such as DC converters are present in the case, or when saturation is evident in machines and transformers. Initialization calculations in these situations can become horrendous to contemplate.

> **NOTE**: Watch out for model instabilities. If instabilities are present, it may never reach steady state (or at least an excessive amount of computer time may be needed to do so).

MULTIPLE RUNS

EMTDC features the ability to automatically perform multiple simulations on the same case, while changing one or more variables each run.

For example, in non-linear models, such as DC transmission links, it is possible that control system gains and time constants can be sequentially or randomly searched to find an optimum response to a disturbance. Similarly, if a transmission line is being switched, or a transformer is to be energized, a search for peak voltages can be undertaken by varying point on wave switching.

Depending on the amount of variables sequenced, this feature can be time consuming, and may need to be relegated to overnight or over the weekend computations. Nonetheless, it is a powerful technique, especially when peak values or optimum performance of highly non-linear systems is being sought after.

Multiple Runs can be performed by two methods:

1. Using the Multiple Run component

2. Manually defining the multiple run variables

If the Multiple Run component is used, it will automatically set the number of runs depending on the selected variation method. More than one Multiple Run component can be used in the same circuit, as long as only one of them is enabled.

CHANNELLING OUTPUT

EMTDC output signals are obtained solely by the use of Output Channels in PSCAD (any conceivable quantity can be specified). At the end of each plot interval, all signals monitored by Output Channels are written to an output array, along with the current value of time. The user has the option to write this output array to an Output File.

PSCAD automates the process of creating output for plotting by formatting the Output Files. In addition, PSCAD enables any of the selected EMTDC output variables to be plotted on-line. The on-line plotting feature provides a substantial benefit to the user who can observe the simulation results as they are computed. The on-line plots can be manipulated in PSCAD once the simulation is complete, or if EMTDC Output Files have been generated, the output traces can be analyzed using available post-processing graphing software.

Output Files
Output Files are formatted text files, which organize the Output Channel data into columns. Each column, except the first, which is ALWAYS time, represents recorded data from one Output Channel in the PSCAD Project. For example, if two Output Channels exist in the Project (say 'Voltage' and 'Current'), then three columns of data will appear in the Output File.

Output Files are given the extension '*.out' and will be named by default after the PSCAD Project File itself.

Multiple Output Files
The maximum amount of columns per Output file is 11, which includes the time column. Therefore, if more than 10 Output Channels exist in the Project, more than one Output File will be created. For example, if your Project contains 23 Output Channels, a total of three Output Files will be created.

Column Identification and the Information File
EMTDC Output File columns are not labelled. In order to determine which column is what, an Information File ('*.inf') is also created, which contains cross-referencing information. The Information File will be named the same as the Output File.

ENHANCED PROGRAM STRUCTURE

With the addition of initialization, multiple run and channelling output features, the EMTDC main program structure becomes that illustrated in Figure 2-6.

Some of the more important functions of the main program are summarized as follows:

- Read data defining the electric network parameters.
- Assemble networks of resistor, inductor and capacitor components together with Thevenin and Norton sources, mutually coupled windings and distributed transmission line and cable models.
- Interface user models with the electric network (DSDYN Subroutine).
- At the user's definition, any computed output quantity is formulated or processed by modeling statements, into an array each time step and printed into output files at the specified print interval, along with the time of that print step (DSOUT Subroutine).
- During the run or at the end of the run, the complete electric network and/or its associated dynamic model, defined in DSDYN, can be frozen in time for future use in a Snapshot File.
- Multiple runs can also be automatically initiated to search for an optimum value of a parameter.

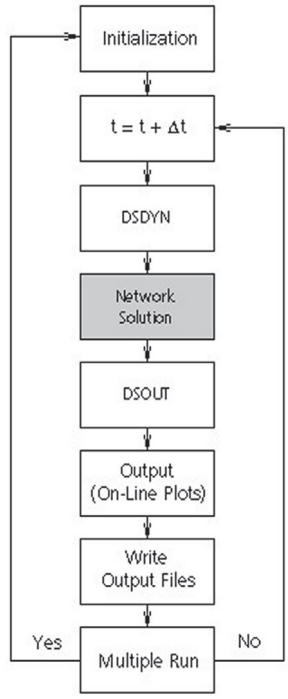

Figure 2-6 - Enhanced Program Structure

Chapter 3:

Electric Network Solutions

REPRESENTATION OF LUMPED R, L AND C ELEMENTS

As described in [1], the principle method for the analysis of lumped inductors and capacitors in

EMTDC is through their representation by a resistance in parallel with a current source as shown below:

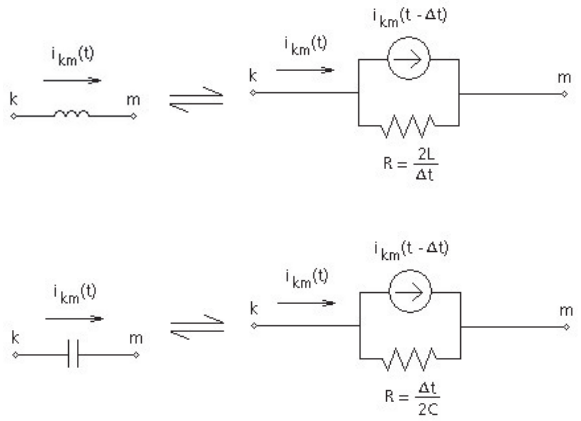

Figure 3-1 - Representation of lumped L and C elements

The equivalent circuits of Figure 3-1 are essentially a numerical representation of the ordinary differential equations, solved for discrete intervals. The trapezoidal rule is used for integrating these equations for lumped inductors and capacitors. It is simple, numerically stable, and accurate enough for practical purposes [1].

The memory function of the integration process is represented by the current source $I_{km}(t - \Delta t)$, which for the inductor is defined as:

$$I_{km}(t - \Delta t) = i_{km}(t - \Delta t) + \frac{\Delta t}{2L} [e_k(t - \Delta t) - e_m(t - \Delta t)] \qquad (3\text{-}1)$$

and for the capacitor as:

$$I_{km}(t - \Delta t) = - i_{km}(t - \Delta t) - \frac{2C}{\Delta t} [e_k(t - \Delta t) - e_m(t - \Delta t)] \qquad (3\text{-}2)$$

Where,

Δt =	Time step
$e_k(t - \Delta t)$ =	Voltage at node k from the previous time step
$e_m(t - \Delta t)$ =	Voltage at node m from the previous time step
$i_{km}(t - \Delta t)$ =	Current through the branch from the previous time step (node k to node m)

NOTE: Lumped resistors are modeled as a simple resistive branch.

Thus, it follows that for a given time step, the current through an inductor or capacitor branch is defined by:

$$i_{km}(t) = \frac{e_k(t) - e_m(t)}{R} + I_{km}(t - \Delta t) \qquad (3\text{-}3)$$

Where,

$R = \frac{2L}{\Delta t}$ For an inductor

$R = \frac{\Delta t}{2C}$ For a capacitor

EQUIVALENT BRANCH REDUCTION

If an electrical branch contains more than one series element, then this branch will be ‘collapsed’ into an equivalent resistance and current source, effectively removing unnecessary nodes (i.e. decreasing the size of the network conductance matrix) and enhancing solution speed.

By the time the network is solved, the branch will be seen as an equivalent conductance. This technique is illustrated below:

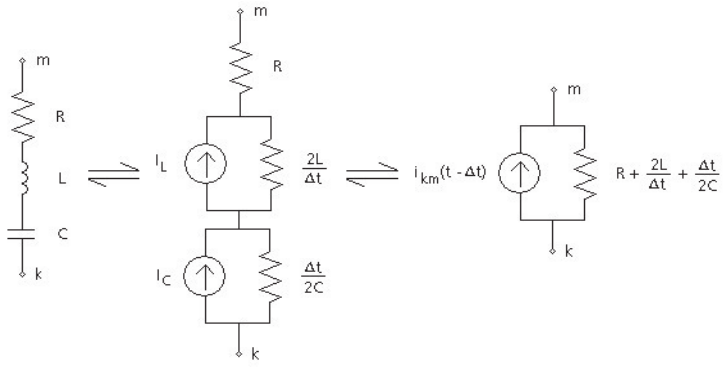

Figure 3-2 - Equivalent Branch Reduction

In this convenient form, other branches of the same type may be paralleled between the same nodes, simply by adding the equivalent branch conductance and current sources.

FORMATION OF SIMPLE NETWORKS

According to that discussed previously, it follows then that a network of lumped R, L and C elements, will be represented in EMTDC as an equivalent circuit of resistive branches and current sources. The resistors are time invariant, except when they are modeled as non-linear or if a specific switching occurs. The equivalent current sources on the other hand, are time and history dependent and must be updated every time step.

Such a structure lends itself to processing by simple matrix methods. Using nodal analysis, a conductance matrix |G| is formed from the inverse resistance value of each branch in the equivalent circuit. |G| is a square matrix, whose size is determined by the number of nodes in the network under study. A column matrix |I| is formed where each element consists of the sum of all current sources at a node.

EXAMPLE 3-1:

Consider a simple R, L, C two node network with its equivalent circuit as shown below:

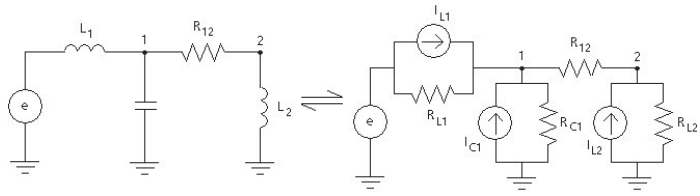

Figure 3-3 - RLC Equivalent Network in EMTDC

The inductors and capacitors are replaced in each case, by an equivalent resistor and current source. The nodal equations are formed as follows:

At node 1:

$$\frac{V_1 - e}{R_{L1}} + \frac{V_1}{R_{C1}} + \frac{V_1 - V_2}{R_{12}} = I_{L1} + I_{C1} \qquad (3\text{-}4)$$

At node 2:

$$\frac{V_2 - V_1}{R_{12}} + \frac{V_2}{R_{L2}} = I_{L2} \qquad (3\text{-}5)$$

These equations are reduced to their matrix form as follows:

$$\begin{bmatrix} \frac{1}{R_{L1}} + \frac{1}{R_{C1}} + \frac{1}{R_{12}} & -\frac{1}{R_{12}} \\ -\frac{1}{R_{12}} & \frac{1}{R_{L2}} + \frac{1}{R_{12}} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} I_{L1} + I_{C1} + \frac{e}{R_{L1}} \\ I_{L2} \end{bmatrix} \qquad (3\text{-}6)$$

Or in short form:

$$[G] \cdot [V] = [I] \qquad (3\text{-}7)$$

The solution to the node voltages defined by column matrix V is then:

$$[V] \cdot [G]^{-1} = [I] \qquad (3\text{-}8)$$

CONDUCTANCE MATRIX INVERSION

EMTDC does not actually calculate the inverse of the conductance matrix |G| directly. Instead it solves $[G]^{-1}$ using forward triangularization and back-substitution - otherwise known as LU decomposition.

The LU decomposition method takes advantage of the sparse nature of the conductance matrix |G| (i.e., entries which are 0.0 are not involved).

SWITCHING AND NON-LINEAR ELEMENTS

Switching and non-linear elements are those that change state during a simulation, according to certain conditions. These devices can range from simple switching elements, such as power electronic devices, (i.e. thyristors, diodes, etc.), breakers and faults, to more complex non-linear elements that can have many states, such as the arrestor.

In addition to those with many states, non-linear devices can also be modeled with an additional compensating current source that can be programmed to represent any non-linearity.

Simple Switches
There are several different methods for representing a simple switching element in time domain simulation programs [5]. The most accurate approach is to represent them as ideal. That is, possessing both a zero resistance in the ON state and an infinite resistance in the OFF state.

Although this approach is very accurate and the resulting equations easier to solve, the drawback is that many possible states are created, which must each be represented by different system equations.

EXAMPLE 3-2:

Consider the network of Figure 3-3 and let the resistance R_{12} represent a simple switch. If R_{12} were considered ideal, then two different networks could result, depending on the state of the switch. This is illustrated in Figure 3-4.

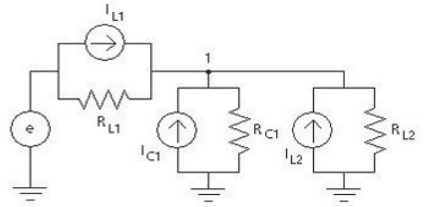

Figure 3-4 (a) - ON State of an Ideal Switch

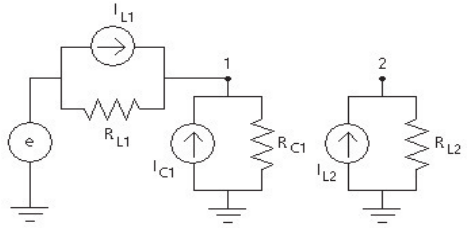

Figure 3-4 (b) - OFF State of an Ideal Switch

Now imagine for instance, a network containing many switches (as in a 48-pulse Graetz bridge STATCOM), a great number of different possible network configurations would result if ideal switching elements were used.

In EMTDC, simple switching devices are represented as a variable resistor, possessing an ON resistance and an OFF resistance. Although this type of representation involves an approximation of both the zero resistance (ON) and an infinite resistance (OFF) of an ideal switch, it is advantageous in that the same circuit structure can be maintained, and the electric network will not need to be split into multiple networks, as a result of each switching event.

Selection of Switching Resistance
In EMTDC, there are provisions to allow for zero resistances (see Ideal Branches). However, while the ideal branch algorithm is very reliable and gives the theoretical result, it does involve extra computations to avoid a division by zero when inverting the conductance matrix. Thus, by inserting a reasonable resistance typical of a closed switch, you can improve the simulation speed. A non-zero value larger than 0.0005 Ω should be used wherever possible.

Selection of the switching resistances is important. If the resistance is too small, its value may dominate the conductance matrix and the other diagonal elements could be overshadowed or lost. This will cause its effective conductance matrix inversion to be inaccurate.

> **NOTE**: The default ON resistance value in PSCAD is 0.001 Ω. The threshold, under which the Ideal Branch algorithm will be invoked, is 0.0005 Ω.

These important factors should be kept in mind when choosing a switch resistance:

- Circuit losses should not be significantly increased.
- Circuit damping should not be reduced significantly. Due to the solution being in steps of discrete time intervals, numerical error may create less damping in the circuit than expected in reality. A small amount of additional resistance from a closed switch, if judiciously selected, may compensate the negative damping effect in the solution method.

Non-Linear Elements
There are two methods for representing non-linear elements in EMTDC; the Piecewise Linear, and the Compensating Current Source methods. Each method possesses its own pros and cons, and the selection of either is simply up to the user.

The Piecewise Linear Method
Although a non-linear device may possess a characteristic that is continuous, controlling the device as continuous is not recommended. Continually changing branch conductance can force a conductance matrix inversion every time step, resulting in a substantially longer run time in large networks.

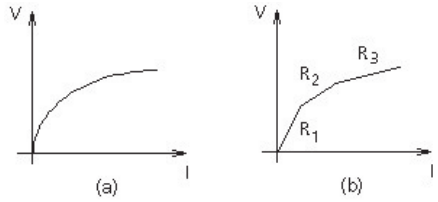

Figure 3-5 - (a) Continuous (b) Piecewise linear conductance properties

In order to minimize run time, a Piecewise Linear approximation method, as illustrated in Figure 3-5 (a) and (b), is used. A piecewise linear curve will introduce several 'state ranges' into the non-linear characteristic, dramatically reducing the amount of matrix inversions per run, yet maintaining reasonable accuracy.

In EMTDC, non-linear devices, such as the Arrestor component, utilize the Piecewise Linear technique.

Compensating Current Source Method

Another method to modeling a non-linear characteristic is through the use of a compensating current source. This technique is accomplished by adding an equivalent Norton current source in parallel with the device itself, thereby allowing for the addition or subtraction of extra current at the device branch nodes.

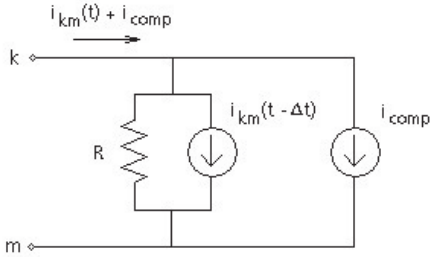

Figure 3-6 - Compensating Current Source Method

Care must be taken when using this method to model device non-linearities. More often than not, the compensating source will be based on values computed in the previous time step, thereby behaving as an open circuit to voltages in the present time step. This can create de-stabilization problems in the simulation [5]. To circumvent this problem, the compensating current source should be used in conjunction with a correction source and terminating

impedance. See Machine Interface to EMTDC and/or [5] for more details on this concept.

In PSCAD, the compensating current source method is used to model core saturation in the Classical Transformer models.

MUTUALLY COUPLED COILS

The representation of mutually coupled coils is an important aspect for analysis of electromagnetic transients. EMTDC provides the ability to model mutually coupled windings through the inclusion of a mutual inductance matrix in a subsystem.

In PSCAD, mutually coupled windings can easily be constructed through the use of transformer components. See the Transformers segment for more details.

SUBSYSTEMS IN ELECTRIC NETWORKS

Best advantage of EMTDC can be made if the electric network to be modeled can be split into discrete subsystems. This is particularly possible when distributed transmission lines or cables separate the electric network.

When EMTDC was first developed, the importance of minimizing the size of the conductance matrix, in order to efficiently represent HVDC systems, was realized. The simulation of these systems involved (and still do) many switching operations. Each time a power electronic device is switched in EMTDC, its resistance value changes and the conductance matrix must be re-triangularized or re-inverted. In larger systems, with matrix dimensions in the thousands, this can substantially decrease simulation speed and efficiency.

EXAMPLE 3-3:

Consider a 10,000 node electric network containing 50 network clusters, with 200 nodes evenly distributed in each cluster. The number of stored elements without splitting into subsystems (i.e. one large non-sparse matrix) would be:

Stored Elements = 10,000 x 10,000 = 100,000,000

The number of stored elements after splitting into subsystems (i.e. 50 non-sparse matrices of 200 x 200 nodes each) would be:

Stored Elements = 200 x 200 x 50 = 2,000,000 (50 times less memory required)

The time for performing an LU matrix decomposition is approximately the same as without any subsystem splitting. However, subsystems create performance advantages when you consider the time required to perform interpolation and switching operations. When an interpolation-switch-interpolation sequence is performed, it will affect only one subsystem, rather than the entire system of equations.

When distributed transmission line or cable models are used to transmit between smaller clusters of electric networks, it is possible to effectively split these clusters and solve them independently. Since distributed line models represent travelling waves, then a switching operation (or source perturbation) at one end of the line, will not impact the electric circuit at the opposite end within the same time step, but at some definable number of time steps following the disturbance. The network clusters at each end can then be considered as de-coupled, discrete subsystems, as no off-diagonal matrix elements will appear between them. Mathematically, this means that a separate conductance matrix can be created for each discrete subsystem and processed independently from other subsystems and their respective conductance matrices. Figure 3-7 represents four conductance matrices representing four de-coupled subsystems.

1	0	0	0
0	2	0	0
0	0	3	0
0	0	0	4

Figure 3-7 - De-Coupled Subsystems

In PSCAD, an electric network can only be split into subsystems by using distributed transmission lines or cables.

Splitting the conductance matrix into subsystems will result, in most cases, in a sparse matrix. That is, a matrix containing zero-elements that are not involved in the system solution (as shown above).

EMTDC does not store in the sparse format but compromises. It stores conductance matrix data in a sequential, non-sparse basis. In other words, some zero elements of the matrix are stored, but are not considered as active subsystems. The addresses of non-zero elements in each subsystem are stored in integer vectors, and are used to access the non-zero elements only. Keeping the storage sequential may not be the most memory efficient method possible, but it has performance advantages; disk/RAM/cache transfers can be streamed more effectively by the Fortran compiler, compared to pure 'random' allocated storage of a sparse matrix vector quantity.

Chapter 4:

Advanced Features

INTERPOLATION AND SWITCHING

As discussed in Chapter 3, transient simulation of an electric network, over a certain period of time, is accomplished by solving the network equations at a series of discrete intervals (time steps) over that period. EMTDC is a fixed time step transient simulation program and therefore, the time step is chosen at the beginning of the simulation, and remains constant thereafter.

Due to the fixed nature of the time step, network events, such as a fault or thyristor switching, can occur only on these discrete instants of time (if not corrected). This means that if a switching event occurs directly after a time step interval, then the actual event will not be represented until the following time step.

This phenomenon can introduce inaccuracies and undesired switching delays. In many situations, such as a breaker trip event, a delay of one time step (say about 50 μs) is of hardly any consequence. However, in power electronic circuit simulation, such a delay can produce very inaccurate results (i.e. 50 μs at 60 Hz is approximately 1 electrical degree). One way to reduce this delay is to reduce the time step. However, this will also increase the computation time proportionately, and still may not give good enough results.

Another method is to use a variable time step solution, where if a switching event is detected, the program will sub-divide the time step into smaller intervals. However, this does not circumvent the problem of spurious voltage and current spikes, due to current and voltage differentials when switching inductive and capacitive circuits.

EMTDC uses an interpolation algorithm to find the exact instant of the event if it occurs between time steps. This is much faster and more accurate than reducing the time step and interpolation allows EMTDC to accurately simulate any switching event, while still allowing the use of a larger time step.

Here is how it works:

1. Each switching device adds its criteria to a polling list when called by the DSDYN subroutine. The main program then solves for the voltages and currents at the end of the time step, while storing the switching device condition at the beginning of the time step. These devices may specify a switching instant by time directly, or by voltage or current crossing levels.

2. The main program determines the switching device, whose criteria for switching has been met first, and then interpolates all voltages and currents in this subsystem to that instant in time. The branch is then switched, requiring a re-triagularization of the conductance matrix.

3. EMTDC then solves for all history terms, increments forward by one time step past the interpolated point, and solves for the node voltages. All devices are polled to see if more interpolated switching is required before the end of the original time step.

4. If no further switching is required, one final interpolation is executed to return the solution to the original time step sequence.

These steps are illustrated in Figure 4-1:

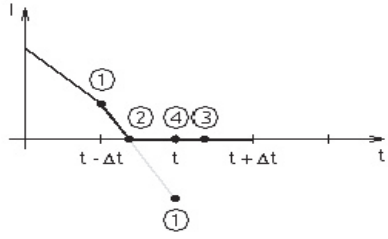

Figure 4-1 - Illustration of Interpolation Algorithm

> **NOTE**: If there is more switching in this particular time step, then steps 1 to 3 are repeated.

EXAMPLE 4-1:

Referring to Figure 4-2, let us consider a diode that is conducting, but should turn off when the current reaches zero. When the diode subroutine is called from DSDYN at time step 1, the current is still positive, so no switching occurs.

If interpolation is not available (or turned off in EMTDC), a solution at time step 2 would be generated. The diode subroutine would then recognize that its current is negative, and subsequently switch itself off for time step 3 - thus allowing a negative current to flow through the device.

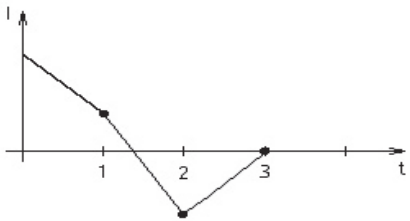

Figure 4-2 - Non-Interpolated Diode Current

In EMTDC (with interpolation turned on), when the diode subroutine is called from DSDYN at time = 1, it still, of course, would not switch the device off because the current is positive. However, because this is a switchable branch, it would be part of a list indicating to the main program that if the current through this branch should go through zero, it should switch the branch off before the end of the time step.

The main program would generate a solution at time = 2 (as it did above), but would then check its list for interpolation requirements. Since the new diode current is negative at time = 2, the main program would calculate when the current actually crossed zero. It would interpolate all voltages and currents to this time (say time = 1.2), and then switch the diode off.

Assuming that there is no further switching in this time step, the main program would appropriately calculate the voltages at time = 1.2 and 2.2 (1.2 + Δt), and then interpolate the voltage back to time = 2 to bring the simulation back on track with integral time steps.

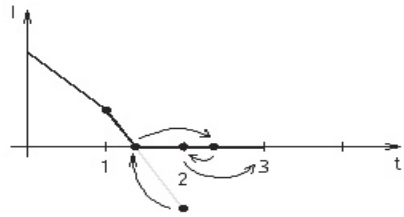

Figure 4-3 - Interpolated Diode Current

> **NOTE**: DSDYN and DSOUT are still only called at times 1, 2 and 3, yet the diode is still turned off at 1.2, therefore no negative current is observed.

The main program would then call DSOUT so that the voltages and currents at time = 2 can be output. It would then call DSDYN at time = 2, and continue the normal solution to time = 3.

There is one additional complication to the above procedure: A chatter removal flag (see Chatter Detection and Removal) is automatically set any time a switch occurs. The flag is cleared as soon as an uninterrupted half time step interpolation is achieved. In the example above, this means that an additional interpolation would be performed to 1.7 (half way between 1.2 and 2.2), a solution at 2.7, and then the final interpolation would return the solution to 2.0 as before.

To prevent an excessive number of switches in one time step, the solution will always proceed forward by at least 0.01% of the time step. In addition, any two (or more) devices, which require switching within 0.01% of each other, will be switched at the same instant.

As an example of the application of interpolation, is a simple HVDC system, where the differences in measured alpha (at the rectifier) for a constant alpha order is illustrated in Figures 4-4 (a) and (b), with a 50 μs simulation time step. While the interpolated firing produces less than 0.001° fluctuation, the non-interpolated firing results in about 1° fluctuation. Such large fluctuations (of one or more degrees) in firing will introduce non-characteristic harmonics and will prevent fine adjustments in firing angles. In these two examples, EMTDC automatically interpolates the thyristor turn off to the zero crossing (negative) of the thyristor c

urrent.

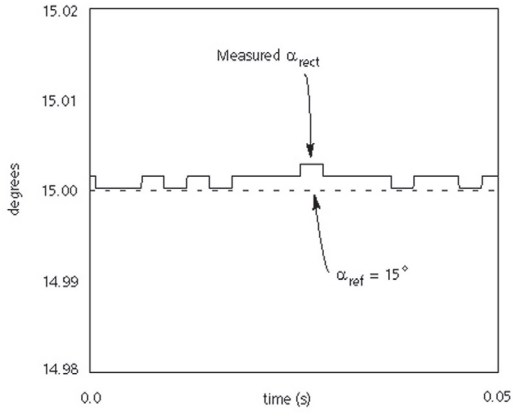

Figure 4-4 (a) - Example of Interpolation Effect: Interpolated

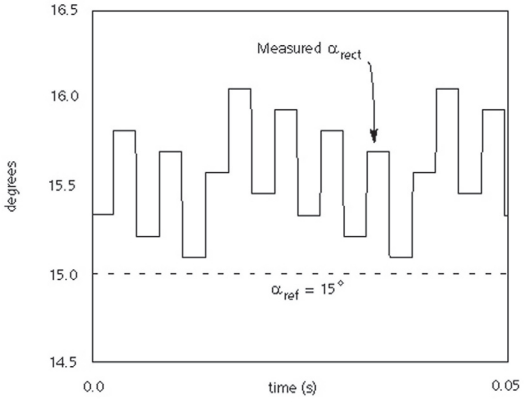

Figure 4-4 (b) - Example of Interpolation Effect: Non-interpolated

Example applications where interpolation is advantageous:

- Circuits with a large number of fast switching devices.
- Circuits with surge arresters in conjunction with power electronic devices.
- HVDC systems with synchronous machines which are prone to sub-synchronous resonance.

- Analysis of AC/DC systems using small signal perturbation technique where fine control of firing angle is essential.
- Force commutated converters using GTOs and back diodes.
- PWM circuits and STATCOM systems.
- Synthesizing open loop transfer functions of complex circuits with power electronic devices.

For more details on interpolation, please see [6], [7] and [8].

CHATTER DETECTION AND REMOVAL

Chatter is a time step to time step, symmetrical oscillation phenomenon inherent in the trapezoidal integration method used in the Dommel algorithm for transient simulation of electrical networks [1].

Chatter is usually initiated by the closing of a switch in a branch containing inductors. It does not matter if the switching occurs between time steps, or at a natural current zero [8]. Figure 4-5 illustrates the presence of voltage chatter, due to a natural turn-off of a series thyristor/inductor, series branch.

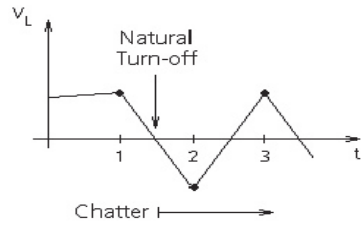

Figure 4-5 - Voltage Chatter Across an Inductor

Since chatter does not represent any electrical network behaviour, it must be suppressed. EMTDC includes a chatter detection algorithm to continuously detect such spurious oscillations and remove them, if so required. Chatter is detected by continuously monitoring every node voltage and branch current and is assumed to be present if these quantities change direction successively for five consecutive time steps. For example: 1.0, -0.9, 0.8, -0.7 and 0.6. In addition, the chatter detection algorithm continually monitors for branch switching events. In this way, chatter introduced by any sudden changes in the electric network (even those not initiated by switching events) is detected.

NOTE: If chatter detection is disabled and chatter removal is enabled, only chatter due to branch switching will be removed. This is sufficient in most situations. The default Chatter Detection Level (CDL) is set to 0.001 p.u. in the PSCAD Project properties.

Either when chatter is detected or when a switching event takes place, a chatter removal algorithm is invoked. Chatter is removed using a half time step interpolation. The user has the option to enable or disable the chatter algorithms in PSCAD, however it is a good practice to keep them enabled for all circuits.

For more detailed information on Chatter and its effects, please see [6] and [8].

EXTRAPOLATE SOURCES

Another feature related to the interpolation algorithm is the Extrapolate Sources algorithm. This relatively simple feature is used only during Step #3 of the interpolation sequence (see Interpolation and Switching), and involves an approximation of the voltage source values at the time $t = t + \delta t$. This is illustrated in Figure 4-6.

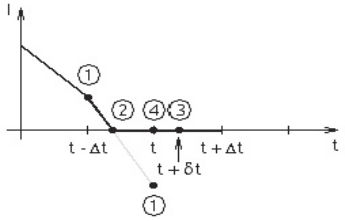

Figure 4-6 - Source Extrapolation

If the Extrapolate Sources algorithm is enabled, then the voltage at point #3 in Figure 4-6 will be calculated as:

$$V' = V \cdot \sin(\omega(t + \delta t + \phi)) \qquad (4\text{-}1)$$

Where,

δt = Interpolated increment

ω = Source frequency [rad/s]

ϕ = Source phase angle

If the Extrapolate Sources algorithm is disabled, then the voltage at point #3 in Figure 4-6 will be approximated with a linear extrapolation.

It is obvious from the above discussion that a more accurate solution, as given by Equation 4-1, will result if the Extrapolate Sources algorithm is left enabled.

> **NOTE**: The Extrapolate Sources algorithm is used ONLY with the Single-Phase Voltage Source Model 2 and the Three-Phase Voltage Source Model 2.

IDEAL BRANCHES

Ideal branches are those with zero impedance. Examples of such branches are infinite voltage sources, ideal short circuits and an ideal switch in closed state. Standard electromagnetic transient solution algorithms using a nodal admittance matrix require every branch to possess a finite impedance. A zero impedance branch would yield an infinite admittance and would thereby lead to numerical problems.

In EMTDC, a provision has been made to allow for zero resistances and true infinite bus voltage sources. The algorithm used permits any combination of ideal branches, including loops. An exception to this is when two or more ideal branches, where one of these is a voltage source. This would create an infinite current in the other parallel branches.

The default threshold value for ideal branches is set to 0.0005 Ω. Therefore, to create an infinite bus, you can either enter 0 or a value less than the threshold for the source resistance. Similarly, for a zero resistance branch, add 0 or a value less than the threshold for the ON resistance of a diode, close resistance of a breaker, etc.

NOTE: The ideal branch algorithm involves extra computations. Thus, a non-zero value of at least 0.0005 Ω (more than the ideal threshold) should be used wherever possible. See Switching and Non-Linear Elements for more.

OPTIMIZATION AND MULTIPLE RUNS

You can use EMTDC in its multiple run mode to determine an optimal set of parameters. Vary a set of parameters either sequentially or randomly for each run and plot the set of values against an objective function, such as peak over voltage, integral square error, etc. The objective function can be built in PSCAD using standard blocks available in the master library or you can custom write one using the standard interface.

This feature allows you to run an EMTDC simulation many times - each time with a different set of parameters. For example, you can run the case several times, each time with different fault or a different gain for a controller. This batch feature is ideal for parametric optimization, determining worst point on wave fault over voltage, determining optimal relay settings, etc.

DYNAMIC DIMENSIONING

EMTDC is available in two different flavours - dynamic and fixed. The fixed version, used exclusively with the free EGCS/GNU Fortran 77 compiler, comes with non-adjustable Fortran dimensions. The fixed version is pre-configured to handle a certain system size and cannot adapt if the system size grows larger (see Dimension Limits).

The dynamic version, used with one of the commercially available Fortran 90 compilers supported by PSCAD, automatically determines the array requirements and allocates memory for optimal performance. The dynamic version is infinitely sizable; the only limitation is the available computer resources/memory.

The dynamic version is slightly slower than the fixed version because of the differences in the memory management model. In most situations, this is not an issue and you are better off using dynamic version whenever possible.

Chapter 5:

Writing Your Own Models

INTRODUCTION TO WRITING YOUR OWN MODELS

As discussed previously, the EMTDC program structure is designed in such a way as to allow the user to introduce his or her code into the main solution algorithm. In most cases, this code will represent models that are not available in the PSCAD Master Library. They could be simple models, which need just a few lines of code, or complex models represented by a combination of several functions and subroutines.

User subroutines can be inserted as part of either DSDYN or DSOUT. In this way, the model can accept input arguments, process them and then either pass the computed responses to an output array, and/or return them to the electric network interface.

There are two methods for introducing user code into EMTDC:

1. PSCAD Script is included in the Component Definition, in order to define either control type or electric branches.

2. The user organizes code into subroutines/functions with a set of call statements included in the Component Definition.

User subroutines or functions can be automatically compiled by PSCAD, or can be pre-compiled and linked with a compiled PSCAD case and the EMTDC library. The user can write his or her code in Fortran or C. For more information on designing components, see Designing Components.

FORTRAN GUIDELINES FOR EMTDC

In order to use both the fixed and dynamic versions of EMTDC, it is a good idea to develop Fortran code so that it is compatible with both Fortran 77 and Fortran 90 standards. The following discusses a set of rules to help the user construct his or her code to be portable, both Fortran 77 and Fortran 90 compatible, robust and easily maintainable.

One Code Source

There should only be one code source for a model, which works on all platforms and Fortran compilers. The Fortran 90 version of EMTDC performs dynamic array allocation, which is a standard Fortran 90 feature. EMTDC receives the exact dimensions of all arrays necessary to simulate the given circuit from PSCAD (stored in the Map File or Snapshot File) and allocates them accordingly. This means that the memory is optimally utilized and there is no limitation on the size of the system that can be simulated.

Guidelines for Compatibility

To make models compatible with both Fortran 77 and Fortran 90 (and as portable as built-in models), the user should adhere to the following general guidelines:

- **Use Only Standard Fortran 77 Features**: Many of the Fortran 77 extensions may not be supported in Fortran 90 compilers, even though they are from the same manufacturer.

- **Continuation Lines**: Always use an "&" in 6th column of a continued line. Also, use an "&" in 73rd column of the continuing (previous) line. Only such continuation lines are compatible in both Fortran 77 and Fortran 90.

- **Comment Lines**: Always use an "!" character for comment lines. This is the common standard between Fortran 77 and Fortran 90. Exclamation characters can appear anywhere on the line except for column 6. Everything after an "!" character in a line is considered to be a comment and ignored. Do not use "!MS$" and "!DEC$" as the first characters of your comment line. These are keywords for setting attributes in the Compaq Fortran and Digital Fortran compilers respectively. Hopefully, user-written routines never have to use these special attributes.

- **Names Convention**: To avoid conflicts between user-written procedure (subroutine/function) names and EMTDC procedure names (and to make the name more meaningful), start the name with 'U_' which stands for User-written, and do not choose a name from the list of EMTDC variables.

- **Carriage Returns**: In order to avoid conflicts with the EGCS/GNU Fortran 77 compiler, you must insert a carriage

return directly following the last line of code in your subroutine.

C SUBROUTINES

If preferred, the user can use the C coding language to write subroutines or functions. These can then be easily called from the DSDYN or DSOUT sections of the Component Definition, when encased in a Fortran 'wrapper.' See Interfacing to C Code for more details.

INTERNAL GLOBAL VARIABLES

In order to maintain program consistency and simplicity, EMTDC uses a variety of internal, globally recognized variable declarations to represent internal features and functions. Some of these variables are accessible to the user, and can be used in user code through standard Fortran INCLUDE statements: COMMON blocks for the Fortran 77 (fixed version), and in MODULEs for Fortran 90 (dynamic version).

The following sections describe the most common Internal Variables. For a summary of which Internal Variables are included in each Include File, see Include Files.

STORx Arrays

In cases where the value of a certain user variable needs to be stored and made available in the following time step, internal variables called STORx arrays should be used. The STORx arrays are multiple-row, single-column storage vectors (or stacks), where the user can store data according to a sequential pointer.

Table 5-1 summarizes the available STORx Arrays and pointers:

Array Name	Pointer	Description
STORL(*)	NSTORL	An array of logical variables, with index NSTORL, used to govern accessibility to the STORL array
STORI(*)	NSTORI	An array of integer variables, with index NSTORI, used to govern accessibility to the STORI array

STORF(*)	NSTORF	An array of floating point variables, with an index pointer NSTORF, used to govern accessibility to the STORF array
STORC(*)	NSTORC	An array of complex variables, with index NSTORC, used to govern accessibility to the STORC array

Table 5-1 - Available STORx Arrays and Pointers

Figure 5-1 illustrates a typical STORx array. Data can be stored at individual address locations, using the corresponding pointer (NSTORx). Proper use of the NSTORx pointer is essential for accurate simulation results. If, for instance, the pointer is not properly addressed, data stored in previously called subroutines, may be over-written.

Figure 5-1 - Typical STORx Array Vector

NOTE: All STORx array pointers are reset to one (1) each time step.

Each time step, the main program is of course sequenced from top to bottom. Using the STORx array and pointer, each individual subroutine can access and store data to and from the STORx array, during the order in which they appear in the main program. In order to avoid stored data being over-written, each subroutine must increment the respective pointer, by the amount of address locations used in the subroutine, before returning to the main program.

EXAMPLE 5-1:

Consider a user-written subroutine, which is to be added to DSDYN. The model requires that two variables be stored for use in the next time step - two REAL (X and Y) and one INTEGER (Z). In addition, the data from the previous time step must be retrieved from the STORx arrays. The code should then appear similar to what is shown below:

```
      SUBROUTINE U_USERSUB(...)
!
! Retrieving the variable values from the STORx array indexes
! before calculations:
!
      X_OLD = STORF(NSTORF)
      Y_OLD = STORF(NSTORF + 1)
      Z_OLD = STORI(NSTORI)
!
   ... (Main body of subroutine)
!
! Saving the values of variables to the STORx arrays (near the end
! of the subroutine):
!
      STORF(NSTORF) = X
      STORF(NSTORF + 1) = Y
      STORI(NSTORI) = Z
!
! Increment the respective pointers (very important):
!
      NSTORF = NSTORF + 2
      NSTORI = NSTORI + 1
!
      RETURN
       END
!
```

In the above example, it can be assumed that previously called subroutines have incremented the pointers correctly.

A good habit to get into is to make a copy of the STORx array pointer, and update the original one. The copy is used only in the local subroutine; therefore, if other nested functions or subroutines exist, storage pointer confusion can be avoided. Using this approach, the above example would become:

```
      SUBROUTINE U_USERSUB(...)
!
! Copying the respective pointers:
!
      MY_NSTORF = NSTORF
      MY_NSTORI = NSTORI
      NSTORF = NSTORF + 2
      NSTORI = NSTORI + 1
!
! Retrieving the variable values from the STORx array indexes
! before calculations:
!
      X_OLD = STORF(MY_NSTORF)
      Y_OLD = STORF(MY_NSTORF + 1)
      Z_OLD = STORI(MY_NSTORI)
!
   ... (Main body of subroutine)
!
! Saving the values of variables to the STORx arrays (near the end
! of the subroutine):
!
      STORF(MY_NSTORF) = X
      STORF(MY_NSTORF + 1) = Y
      STORI(MY_NSTORI) = Z
!
      RETURN
      END
```

If using the dynamic (Fortran 90) version of EMTDC, a special directive statement, called the #STORAGE Directive, is needed in the DSDYN or DSOUT sections of the Component Definition. This directive will inform PSCAD of the total number of storage elements required for this particular component. PSCAD will then use this information to instruct EMTDC as to how many elements, in an array, should be allocated for memory. For instance in Example 5-1, the user would need to add the following Storage Directives before the actual subroutine call:

```
#STORAGE REAL:2 INTEGER:1
      CALL U_USERSUB(...)
```

Accessing Network Quantities

Using Internal Variables, the user can access both branch currents and node voltages in an electric network. EMTDC is a 'branch oriented' program, so usually information about a given branch is accessible through its branch number.

> **NOTE**: In this and the following tables, BRN and SS stand for branch number and subsystem number respectively. NN stands for node number.

Node Numbers

Branch Node Numbers may be accessed with the following Internal Variables:

Internal Variable	Description
IEF(BRN,SS)	Gives the number of the FROM node
IET(BRN,SS)	Gives the number of the TO node

Table 5-2 - Internal Variables to Access Node Numbers

Branch Current

The current flowing in a given branch can be monitored using the Internal Variable CBR:

Internal Variable	Description
CBR(BRN,SS)	Gives the value of current in a particular branch with the direction of positive current being from the FROM node to the TO node. See Internal Output Variables for more details.

Table 5-3 - Internal Variable to Access Branch Current

Node Voltage

The voltage at a given node can be monitored using the Internal Variable VDC:

Internal Variable	Description
VDC(NN,SS)	Gives the value of voltage at node NN. See Internal Output Variables for more details.

Table 5-4 - Internal Variable to Access Node Voltage

Electric Network Interface Variables

EMTDC offers Internal Variables for the purpose of defining the key features of an electric model interface branch.

Internal Variable	Description
EBR(BRN,SS)	Sets the value of branch voltage
CCBR(BRN,SS)	Current source representing the history current when inductors and/or capacitors are used in the interface branch
GEQ(BRN,SS)	Sets the value of the branch equivalent conductance. This conductance will represent an equivalent resistor only if resistive

Table 5-5 - Internal Variables to Define the Electric Interface Branch

> **NOTE**: See Branch Based Electric Interface for more details.

INCLUDE FILES

PSCAD automatically selects these files depending on which Fortran compiler option is selected. Since the Include File names are the same (contents are different) for both Fortran versions, user-written code will not need to be changed if only INCLUDE statements, instead of hard coded COMMON statements, are used. As much as possible, avoid creating your own common blocks and use storage arrays for saving values between time steps.

The following tables provide a quick reference for the most commonly used Include Files, along with a brief description of their contents for the fixed version. The dynamic version uses the same variables but the Include File merely contains the module name.

> **NOTE**: This is also a list of all reserved names, which cannot be used in any user-written models (if the corresponding include file is referenced).

nd.h

This included file contains important dimensioning information for use with the EMTDC Fortran 77 (fixed) version, and should always be the first file included in all models.

emstor.h

Variable Name	Type	Description
STORL(*)	LOGICAL	Array of logical variables
STORI(*)	INTEGER	Array of integer variables
STORF(*)	REAL	Array of floating point variables
STORC(*)	COMPLEX	Array of complex variables
NSTORC	INTEGER	Index of STORC array
NSTORF	INTEGER	Index of STORF array
NSTORI	INTEGER	Index of STORI array
NSTORL	INTEGER	Index of STORL array
THIS	INTEGER	A temporary pointer

s0.h

Variable Name	Type	Description
RDC(*,*,*)	REAL	Trangularized [G] matrix
CCIN(*,*)	REAL	Sets the value of current injected into a specified node from ground

VDC(*,*)	REAL	Gives the value of voltage at the specified node
GM(*,*,*)	REAL	Conductance matrix
CCGM(*,*)	REAL	Transformer current
CCLI(*,*)	REAL	Transmission line / cable current
GGIN(*,*)	REAL	Sets the equivalent conductance value of the Norton current source CCIN.
CA(*)	REAL	Current injection vector for tri-angularized [G] matrix
CDCTR(*,*)	REAL	Current flowing through the M^{th} winding of the N^{th} transformer.
MBUS(*)	INTEGER	Number of nodes in a subsystem
IDEALSS(*)	LOGICAL	True if subsystem contains ideal branches

s1.h

Variable Name	Type	Description
TIME	REAL	Current time of simulation (t) in seconds
DELT	REAL	Simulation time step (Δt) in seconds
ICH		*Obsolete*
PRINT	REAL	Plot step interval in seconds
FINTIME	REAL	Simulation finish time in seconds
TIMEZERO	LOGICAL	True when time t = 0.0
FIRSTSTEP	LOGICAL	True for first step starting from the Data File or Snapshot File
LASTSTEP	LOGICAL	True at last time step of the simulation
ONSTEP	LOGICAL	True if calling from the DSDYN or DSOUT subroutines

s2.h

Variable Name	Type	Description
STOR	REAL	*Obsolete*
NEXC	INTEGER	*Obsolete*

branches.h

Variable Name	Type	Description
CBR(*,*)	REAL	Branch current
CCBR(*,*)	REAL	Equivalent history current
CCBRD(*,*)	REAL	CCBR from previous time step
EBR(*,*)	REAL	Branch voltage source magnitude
EBRD(*,*)	REAL	EBR from previous time step
EBRON(*,*)	REAL	ON state resistance
EBROF(*,*)	REAL	OFF state resistance
SWLEVL(*,*)	REAL	Switching level (I, V, or time)
GEQ(*,*)	REAL	Equivalent branch conductance
GEQON(*,*)	REAL	Equivalent ON state conductance.
GEQOF(*,*)	REAL	Equivalent OFF state conductance.
GEQD(*,*)	REAL	Equivalent conductance from last step.
RLG(*,*)	REAL	Factor used in collapsing RLC branch.
RCG(*,*)	REAL	Factor used in collapsing RLC branch.
RCL(*,*)	REAL	Factor used in collapsing RLC branch.

RSC(*,*)	REAL	Factor used in collapsing RLC branch.
RSL(*,*)	REAL	Factor used in collapsing RLC branch.
CCL(*,*)	REAL	Factor used in collapsing RLC branch.
CCLD(*,*)	REAL	Factor used in collapsing RLC branch.
CCC(*,*)	REAL	Factor used in collapsing RLC branch.
CCCD(*,*)	REAL	Factor used in collapsing RLC branch.
G2L(*,*)	REAL	Factor used in collapsing RLC branch.
G2C(*,*)	REAL	Factor used in collapsing RLC branch.
V12L(*,*)	REAL	Factor used in collapsing RLC branch.
V20L(*,*)	REAL	Factor used in collapsing RLC branch.
NSW(*)	INTEGER	Total # of switches in subsystem SS
BRNSW(*,*)	INTEGER	Branch # (switch => branch)
IEF(*,*)	INTEGER	Branch number of 'from node' (positive current flows out)
IET(*,*)	INTEGER	Branch number of 'to node' (positive current flows in)
THISBR(*,*)	INTEGER	Starting location of data storage.
RESISTOR(*,*)	LOGICAL	True if resistance is present in branch
INDUCTOR(*,*)	LOGICAL	True if inductance is present in branch
CAPACITR(*,*)	LOGICAL	True if capacitance is present in branch
SOURCE(*,*)	LOGICAL	True if internal source is present in branch
SWITCH(*,*)	LOGICAL	True if switchable

IDEALBR(*,*)	LOGICAL	True if ideal branch
OPENBR(*,*)	LOGICAL	True if switch is open
DEFRDBR(*,*)	LOGICAL	True if attributes undecided
FLIPIDLBR(*,*)	LOGICAL	Flag if an ideal branch is collapsed in forward or reverse
GEQCHANGE(*)	LOGICAL	True if branch switched without interpolation
EC_DIODE	INTEGER	Diode (= 20)
EC_THYRISTOR	INTEGER	Thyristor (= 21)
EC_GTO	INTEGER	GTO (= 22)
EC_IGBT	INTEGER	IGBT (= 23)
EC_MOSFET	INTEGER	MOSFET (= 24)
EC_TRANSISTOR	INTEGER	Transistor (= 25)
EC_VZNO	INTEGER	Gapless Metal Oxide Arrestor (= 34)
EC_VSRC	INTEGER	Branch voltage source (= 101)
EC_CSRC1P	INTEGER	Single phase current source (= 111)
EC_CSRC3P	INTEGER	Three phase current source (= 113)

emtconst.h

Variable Name	Type	Description
PI_	REAL	π = 3.141592653589793
TWO_PI	REAL	2π = 6.283185307179586
PI_BY3	REAL	π /3 = 1.047197551196598

PI2_BY3	REAL	2π /3 = 2.094395102393195
PI_BY2	REAL	π /2 = 1.570796326794896
PI_BY180	REAL	π /180 = 0.017453292519943
BY180_PI	REAL	180/π = 57.29577951308232
BY_PI	REAL	1/π = 0.318309886183791
BY_2PI	REAL	1/2π = 0.159154943091895
SQRT_2	REAL	$\sqrt{2}$ = 1.414213562373095
SQRT_3	REAL	$\sqrt{3}$ = 1.732050807568877
SQRT_1BY2	REAL	$\sqrt{1/2}$ = 0.707106781186548
SQRT_1BY3	REAL	$\sqrt{1/3}$ = 0.577350269189626

fnames.h

Variable Name	Type	Description
DUMLIN	CHARACTER	Array used for storing current data line read
SECTION	CHARACTER	Name of the section last read
FILENAME	CHARACTER	Data file name
MAPNAME	CHARACTER	Name of the map to be read for this page
NETDATA	LOGICAL	True while reading network data
INAM	CHARACTER	Input file name (Snapshot or Data files)
TNAM	CHARACTER	Table file name (nodes table file)
INAM1	CHARACTER	Input include file name

ONAM	CHARACTER	Output file name (channel information and/or print plots
SNAM	CHARACTER	End of run snapshot file name
RTSNAM	CHARACTER	Same as SNAM but for runtime snapshot files
MNAM	CHARACTER	Multiple run file name
IUNIT	INTEGER	Current input file unit number
IUNITOUT	INTEGER	Starting unit # for output files

MODEL TYPES

The presence of the DSDYN and DSOUT subroutines allows for the incorporation of controls dynamics into an electric network. That is, EMTDC may be considered a hybrid solution engine that is capable of modeling both controls dynamics, and electric circuits in the same simulation.

Control Models
Control models are not involved in the actual electric network solution, except in some instances, to provide external control to the electrical elements themselves. Many examples of these can be found in the CSMF section of the PSCAD Master Library.

Control model data signals may either be passed to and from other control models, or may be used as input to some special electrical elements. Additionally, measured electric network quantities (such as voltages, currents, etc.) can be used as input data for the control model. The user can also monitor generated output control signals.

Control model code must be inserted into either DSDYN or DSOUT, and therefore, must be called from the Fortran, DSDYN or DSOUT segments of the respective Component Definition.

EXAMPLE 5-2:

To illustrate the above concept, consider the following simple integrator circuit constructed in PSCAD:

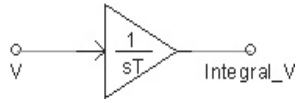

Figure 5-2 - Simple Integrator in PSCAD

The user desires to represent an equivalent flux using the following equation:

$$V = N \cdot \frac{d\phi}{dt} \tag{5-1}$$

Which leads to,

$$\phi(t) = \phi(t - \Delta t) + \frac{1}{N} \cdot \int_{t-\Delta t}^{t} v(t) \cdot dt \tag{5-2}$$

In other words, the measured electric network voltage V, is being input every time step to an integrator control component. This integrated value can then be used to determine the equivalent flux ϕ. If desired, the flux ϕ can then be used as an input control for an electric element, such as a current source.

EXAMPLE 5-3:

The following example code is for a simple integrator function (similar to that in Example 5-2), written in Fortran. Take note of the standardized EMTDC code formatting:

```
!
      SUBROUTINE U_MYINTGL(IN,OUT,LIMITS,ULIMIT,LLIMIT,OUT0)
!
! Purpose  - integration of a real signal
! Language - Fortran 77/90
! Date     -
! Author   -
```

```
!
! Standard Include Files for Control Models
! -------------
      INCLUDE 'nd.h'
      INCLUDE 'emtstor.h'
      INCLUDE 's1.h'
      INCLUDE 'emtconst.h'
!
! Variable Declarations
! ---------------------
      REAL IN, OUT, ULIMIT, LLIMIT, OUT0
      INTEGER LIMITS
      REAL INOLD, OUTOLD
!
! Copy and increment pointers
! ---------------------
      MY_STORF = NSTORF
      NSTORF = NSTORF + 2
!
! Program begins
! ---------------------
      OUT = 0.5*(IN + STORF(MY_STORF))*DELT + STORF(MY_STORF + 1)

!
! Output at time zero
! ---------------------
      IF ( TIMEZERO ) THEN
         OUT = OUT0
      ENDIF
!
! Limit the output if asked
! ---------------------
      IF ( LIMITS .EQ. 1 ) THEN
         IF ( OUT .GE. ULIMIT ) THEN
```

```
            OUT = ULIMIT
         ELSEIF ( OUT .LE. LLIMIT ) THEN
            OUT = LLIMIT
         ENDIF
      ENDIF
!
! Save the data for next time step
! --------------------
      STORF(MY_STORF) = IN
      STORF(MY_STORF + 1) = OUT
!
   RETURN
   END
!
```

In order to include the above code in DSDYN or DSOUT, the subroutine (U_MYINTGL) will need a Component Definition in PSCAD. It will be inserted into DSDYN directly through a call statement in either the Fortran, DSDYN or DSOUT segments of the Component Definition, similar to:

```
#SUBROUTINE U_MYINTGL Integrator Model
#STORAGE REAL:1
      CALL U_MYINTGL($In,$Out,$Limits,$ULimit,$LLimit,$Out0)
!
```

So that PSCAD can find the file where the subroutine exists, either a File Reference component will need to be added to the PSCAD Project, or '*.obj' or '*.lib' file pointers can be added to the PSCAD Case Properties.

Control models usually use a standard set of Internal EMTDC Variables. Below is a quick template, which lists those Include Files commonly used with control type models.

```
      INCLUDE "nd.h"              ! dimensions
      INCLUDE "emtstor.h"      ! storage arrays and indexes
      INCLUDE "s1.h"              ! TIME,DELT,PRINT,FINTIM,...
      INCLUDE "emtconst.h"   ! useful constants
```

Electric Models

All elements that are included in the electric network solution can be classified as electric models. Some examples include simple lumped R, L and C components, transformers and sources.

Writing and interfacing electric models to EMTDC is a bit more complicated than simply inserting control model code into DSDYN or DSOUT. This topic is discussed in more detail in Introduction to Interfacing Electric Models.

Chapter 6:

Interfacing Electric Models

INTRODUCTION TO INTERFACING ELECTRIC MODELS

In the previous chapter, an example was given for the coding of a control type model (integrator), which interfaced to the DSDYN and/or DSOUT subroutines in EMTDC. Interfacing directly to the electric network solution is a bit more involved, but remains an important feature for maintaining programming power and flexibility.

Most of the time, the user can get away with representing electric models by simply combining standard electric elements available in the PSCAD Master Library. However some electric models, such as rotating machines and more complicated FACTS devices, can contain characteristics and features that are impossible to represent simply as a combination of elements.

It is possible to define these more complicated models as a separate, standalone subroutine. This method can create a more flexible programming environment for the user. Many features, such as individual solution styles, can be readily introduced without any impact on the EMTDC main program.

GENERIC ELECTRIC INTERFACE

In EMTDC, electric models written as standalone subroutines can be interfaced directly to the electric network through an equivalent electrical interface branch. This branch effectively represents the model subroutine as a Norton current source, in parallel with an equivalent branch conductance (as discussed in Chapter 3). Figure 6-1 illustrates a generic electrical interface branch:

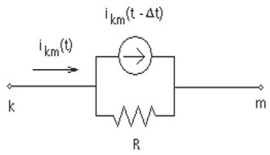

Figure 6-1 - Generic Electrical Interface Branch

In Figure 6-1, k and m are nodes, by which the interface branch is connected to the rest of the electric network. The elements involved in this branch must be defined in the user-written subroutine, either during initialization or every time step depending on the variable and its purpose.

It is important to note however, that information between the interface branch and the main network is exchanged only at each EMTDC time step. This means that the user subroutine, represented by this branch, will always work with electric network quantities from the last calculated time step.

There are two basic interface methods used in EMTDC for the purpose of representing an electric branch. These are:

1. Node-Based Method
2. Branch-Based Method

NODE BASED ELECTRIC INTERFACE

The simplest interface is the node-based electric interface. This interface contains two components: A Norton current source, and an equivalent conductance connected in parallel. These two quantities are represented by the following EMTDC Internal Variables:

Internal Variable	Description
CCIN(NN,SS)	Sets the value of current injected into the network. The current source is inserted between node NN and ground
GGIN(NN,SS)	Sets the conductance value of the Norton current source

Table 6-1 - Internal Variables to Define the Node-Based Electric Interface Branch

Figure 6-2 illustrates the node based electric interface connected to ground. The CCIN current source represents the value of current injected into the specified node (NA) from ground as shown below:

$$CCIN\ (NA,SS) = CCIN\ (NA,SS) + Current \quad (6\text{-}1)$$

$$GGIN\ (NA,SS) = GGIN\ (NA,SS) + Conductance \quad (6\text{-}2)$$

It is important to observe the use of the additional CCIN statement on the right side of Equation 6-1. Although this is not required when modeling a single branch to ground, it will however be required when additional parallel elements are to be connected to this branch. For example, if a second branch is connected between node NA and ground, the total current would be the sum of the CCIN values in each branch. By using the method in Equation 6-1, it will be assured that both currents are included in the total current injected into or out of node NA.

Figure 6-3 shows what is required if the electric interface branch is to be connected to a part of the network that does not involve ground (i.e. connected node-to node). In this case, a second current source must also be defined, equal to the first source, but opposite in sign:

CCIN (NA,SS) = CCIN (NA,SS) + Computed Current

CCIN (NB,SS) = CCIN (NB,SS) - Computed Current

This will ensure that the same amount of current is extracted from node NB (from node), as is injected into node m (to node), creating a 'branch current'. In addition, a second equivalent conductance, equal to the first, must also be added:

GGIN (NA,SS) = GGIN (NA,SS) + Computed Conductance

GGIN (NB,SS) = GGIN (NB,SS) + Computed Conductance

The combined effect of the circuit shown in Figure 6-3 will be the same as that illustrated in Figure 6-1.

NOTE: The values of CCIN and GGIN are reset to zero by the main program at the beginning of each time step, and therefore must be defined each time step.

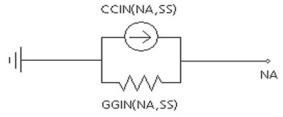

Figure 6-2 - Node-Based Electric Interface Branch (Node to Ground)

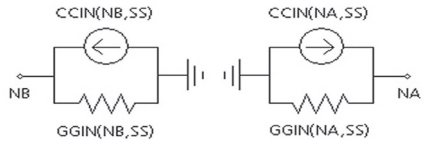

Figure 6-3 - Node-Based Electric Interface Branch (Node to Node)

Enabling CCIN

In order to include the current injection CCIN at the node NN in the calculations for the next time step, the user must initiate the source in the first time step of the simulation. For example, the following logical statement must be included in the user subroutine:

```
IF (TIMEZERO) THEN
   ENABCCIN(NN,SS) = .TRUE.
ENDIF
```

CCIN as a Compensating Current Source

The CCIN current source can also be utilized, without the equivalent conductance GGIN, as a simple ideal current source injection into a specified node. This is particularly useful when modeling a non-linearity with a compensating current source, as discussed in Switching and Non-Linear Elements.

EXAMPLE 6-1:

The following example code shows how to use CCIN and GGIN to represent a simple inductor. Refer to equations 3-1 and 3-3. Also, take note of the standardized EMTDC code formatting:

```
      SUBROUTINE U_SIMPLE_L(SS, NA)
!
! Purpose  - Simple Inductor to Ground using CCIN/GGIN
! Language - Fortran 77/90
! Argument Definitions:
!       SS = Subsystem number
!       NA = Node numbers
!        L = Inductance = 0.001 H
!
! Standard Include Files for Electrical Models
! --------------------
      INCLUDE "nd.h"          ! dimensions
      INCLUDE "emtstor.h"     ! storage arrays and indexes
      INCLUDE "s0.h"          ! VDC,CCIN,GGIN,...
      INCLUDE "s1.h"          ! TIME,DELT,PRINT,FINTIM,..
      INCLUDE "branches.h"    ! CBR,CCBR,EBR,GEQ,IEF,IET,...
!
! Variable Declarations
! --------------------
      REAL L, CURR
      INTEGER SS, NA, MY_STORF
!
! Copy and Increment Pointers:
! --------------------
      MY_STORF = NSTORF
      NSTORF = NSTORF + 2
!
! Initial Branch Definition (t = 0.0):
! --------------------
      IF (TIMEZERO) THEN
        ENABCCIN(NA,SS) = .TRUE.
        L = 0.001
        STORF(MY_STORF) = DELT / (2.0*L)
      ENDIF
!
! Calculate new history current:
! --------------------
      CURR = STORF(MY_STORF)*(-VDC(NA,SS)) + STORF(MY_STORF + 1)
```

```
      STORF(MY_STORF + 1) = CURR + STORF(MY_STORF)*(-VDC(NA,SS))
!
! Define CCIN and GGIN:
! ---------------------
      CCIN(NA,SS) = CCIN(NA,SS) + STORF(MY_STORF + 1)
      GGIN(NA,SS) = GGIN(NA,SS) + STORF(MY_STORF)
!
      RETURN
      END
```

So that PSCAD can find the file where the subroutine exists, either a File Reference component will need to be added to the PSCAD Project, or *.obj' or '*.lib' file pointers can be added to the PSCAD Case Properties.

BRANCH BASED ELECTRIC INTERFACE

The branch-based electric interface is a more sophisticated method, introduced with PSCAD V3.

The most noticeable difference between this and the node-based branch is that all elements of the branch are indexed through branch numbers – not node numbers. Also, these elements are not reset each time step, and therefore do not need to be redefined.

A Norton current source and equivalent conductance still exists as in Figure 6-1, however an additional series Thevenin voltage source is also included. These three quantities are represented by the following EMTDC Internal Variables:

Internal Variable	Description
EBR(BRN,SS)	Sets the value of the optional branch voltage source.
CCBR(BRN,SS)	Sets the value for the equivalent history current source (only if inductors and/or capacitors present in branch).
GEQ(BRN,SS)	Sets the value of the branch equivalent conductance.

Table 6-2 - Internal Variables to Define the Branch-Based Electric Interface Branch

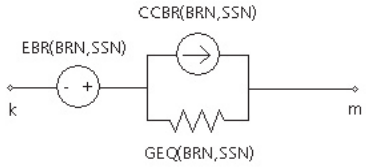

Figure 6-4 - Branch-Based Electric Interface

One of the benefits offered by this method is the simple fact that the current injection is a branch current, and therefore only one current source needs to be defined, instead of two like the node-based interface. Figure 6-4 illustrates the branch-based electric interface.

The Branch-Based Interface and PSCAD
The Branch-Based Electric Interface has been specially designed to allow for manipulation of its parameters directly from the component definition in PSCAD. That is, the elements of the branch interface do not need to be defined through coding, as with CCIN and GGIN. If defining an electric component with code is desired, it is recommended that the node-based interface, described in the previous section, be used.

There are two avenues through which the branch-based electric interface can be accessed in PSCAD, these are:

1. The Branch segment in the Component Definition
2. Calling the EMTDC_VARRLC10 internal subroutine

Branch Section

A segment called 'Branch' in the Component Definition Sections Code, can be included to directly define one or more branches composed of a combination of RLC linear passive elements.

EXAMPLE 6-2:

Consider a 3-phase, Y-connected RLC impedance with R = 1 Ω, L = 0.1 H and C = 1 μF. The PSCAD Script used to define the branch is shown below, as it would appear in the Branch segment of the Component Definition:

```
!
BRANCH1 =  $GND  $NODE1     1.0   0.1   1.0
BRANCH2 =  $GND  $NODE2     1.0   0.1   1.0
BRANCH3 =  $GND  $NODE3     1.0   0.1   1.0
!
```

The above script is used in conjunction with electrical connection nodes 'NODE1,' 'NODE2' and 'NODE3,' and ground node 'GND,' defined in the Graphic section of the Component Definition. PSCAD will automatically construct the three required branch-based interfaces from this information, where the values of CCBR and GEQ are calculated internally.

EMTDC_VARRLC Subroutine

This subroutine is included in EMTDC and can be called from the DSDYN section of any user-written Component Definition. EMTDC_VARRLC10 is also the subroutine called exclusively by the Variable R, L, or C component, which is available in the Master Library.

The power of this tool is that it will allow the user to directly model either a linear or a non-linear R, L or C passive element. There is also a provision to control the internal source EBR.

EXAMPLE 6-3:

The EMTDC_VARRLC10 subroutine is essentially a graphical representation of the branch-based electric interface itself, and can be used as a direct alternative to coding. In the example below, the subroutine is called from the DSDYN section of a user-written Component Definition, with its various arguments predefined as needed.

The DSDYN section of the user component is shown below:

```
#SUBROUTINE VARRLC10 Variable RLC Subroutine
#LOCAL REAL VARR_I
#STORAGE STOR:10
!
      CALL EMTDC_VARRLC10($BR, $SS, $RLC, VARR_I, $L, $E)
```

Here, the EMTDC_VARRLC10 subroutine is configured to model a non-linear inductor, where the inductance is controlled by the signal ‘L’. ‘E’ controls the internal voltage source EBR.

The branch number ‘BR’ is defined in Branch segment of the Component Definition as:

```
BR = $GND  $NA   BREAKER    1.0
```

Of course, as an alternative to calling the EMTDC_VARRLC10 subroutine from a user component, you can simply use the Variable R, L, or C component in the Master Library.

INTERFACING IN GENERAL

An interface to the EMTDC electric network solution (no matter how complicated the model is) will always boil down to a representation by a simple equivalent Norton current source injection, as shown in Figure 6-5.

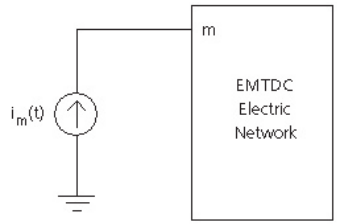

Figure 6-5 - General Interface to the EMTDC Electric Network

Each time step the current source magnitude $i_m(t)$ will be updated and the resulting injection of current will affect the rest of the system.

Care must be exercised when representing electric models in this manner. Due to the finite calculation step, the current source injection $i_m(t)$ will be dependent on the node voltage from the previous time step, thus any sudden change in node voltage would appear as an open circuit to the current source, resulting in spurious voltage spikes and numerical problems at the interface node [5].

To rectify this, a large Norton resistance (small conductance) can be placed between the interface nodes, in order to ensure that a finite impedance is always present, as in Figure 6-6.

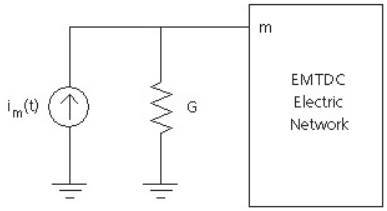

Figure 6-6 - General Interface to the EMTDC Electric Network with Conductance

The addition of the conductance G will introduce a small error into the interface, which should be corrected. This is accomplished through the insertion of a compensating current source, as shown in Figure 6-7.

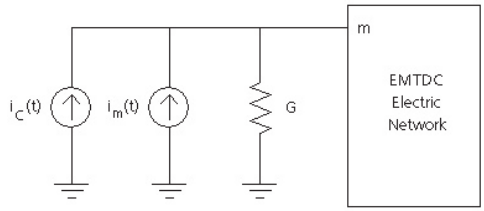

Figure 6-7 - General Interface to the EMTDC Electric Network with Compensating Current Source

Where,

$$i_c(t) = v_m(t - \Delta t) \cdot G$$

Instead of injecting just the calculated current $i_m(t)$, a total compensated current $i_m(t) + i_C(t)$ is injected, where $v(t - \Delta t)$ is the interface voltage at the previous time-step.

This method has proven effective in maintaining the stability of electric models over the years. For instance, this concept allowed for the simulation of multiple rotating machine interfaces on the same bus. See Machine Interface to EMTDC for more.

Chapter 7:

Transformers

INTRODUCTION TO TRANSFORMERS

Transformers are represented in EMTDC through one of two fundamental methods: The classical approach and the unified magnetic equivalent circuit (UMEC) approach.

The classical approach should be used to model windings placed on the same transformer leg. That is, each phase is a separate, single-phase transformer with no interaction between phases. The UMEC method takes inter-phase interactions into account. Thus, 3-phase, 3-limb and 3-phase, 5-limb transformer configurations can be accurately modeled.

Representation of core non-linearities is fundamentally different in each model type. Core saturation in the classical model is controlled through the use of a compensating current source injection across selected winding terminals. The UMEC approach uses a fully interpolated, piecewise linear φ-I curve to represent saturation.

THE CLASSICAL APPROACH

The theory of mutual coupling can be easily demonstrated using the coupling of two coils as an example. This process can be extended to N mutually coupled windings as shown in References [1], [3] and [4]. For our purpose, consider the two mutually coupled windings as shown below:

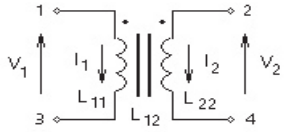

Figure 7-1 - Two Mutually Coupled Windings

Where,

L_{11} = Self inductance of winding 1

L_{22} = Self inductance of winding 2

L_{12} = Mutual inductance between windings 1 & 2

The voltage across the first winding is V_1 and the voltage across the second winding is V_2. The following equation describes the voltage-current relationship for the two, coupled coils:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} L_{11} & L_{12} \\ L_{12} & L_{22} \end{bmatrix} \cdot \frac{d}{dt} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \quad (7\text{-}1)$$

In order to solve for the winding currents, the inductance matrix needs to be inverted:

$$\frac{d}{dt} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \frac{1}{\Delta} \cdot \begin{bmatrix} L_{22} & -L_{12} \\ -L_{12} & L_{11} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \quad (7\text{-}2)$$

Where,

$$\Delta = L_{11} \cdot L_{22} - L_{12}^2 = L_{11} \cdot L_{22} - \left(1 - K_{12}^2\right)$$

$$K_{12} = \frac{L_{12}}{\sqrt{L_{11} \cdot L_{22}}} \quad \text{Coupling coefficient}$$

For ‘tightly’ coupled coils, wound on the same leg of a transformer core, the turns-ratio is defined as the ratio of the number of turns in the two coils. In an ‘ideal’ transformer, this is also the ratio of the primary and secondary voltages. With voltages E_1 and E_2 on two sides of an ideal transformer, we have:

$$\frac{E_1}{E_2} = a \quad (7\text{-}3)$$

And

$$\frac{I_2}{I_1} = a \quad (7\text{-}4)$$

Making use of this turns-ratio, ‘a’ Equation 7-1 may be rewritten as:

$$\begin{bmatrix} V_1 \\ a \cdot V_2 \end{bmatrix} = \begin{bmatrix} L_{11} & a \cdot L_{12} \\ a \cdot L_{12} & a^2 \cdot L_{22} \end{bmatrix} \cdot \frac{d}{dt} \begin{bmatrix} I_1 \\ I_2/a \end{bmatrix} \quad (7\text{-}5)$$

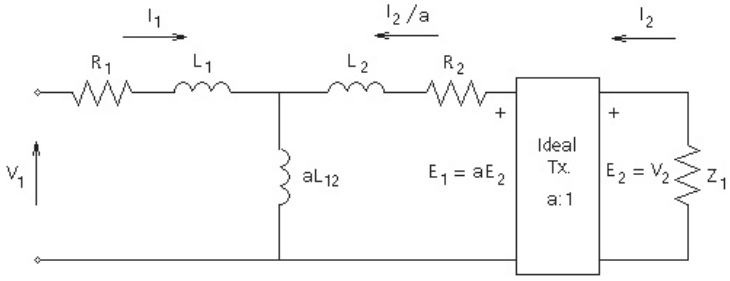

Figure 7-2 - Equivalent Circuit of Two Mutually Coupled Windings

Where,

$$L_1 = L_{11} - a \cdot L_{12}$$

$$L_2 = a^2 \cdot L_{22} - a \cdot L_{12}$$

Now the inductance matrix parameters of Equation 7-1 can be determined from standard transformer tests, assuming sinusoidal currents. The self inductance of any winding ‘x’ is determined by applying a rated RMS voltage V_x to that winding and measuring the RMS current I_x flowing in the winding (with all other windings open-circuited). This is known as the open-circuit test and the current I_x is the magnetizing current. The self-inductance L_{xx} is given as:

$$L_{xx} = \frac{V_x}{\omega \cdot I_x} \quad (7\text{-}6)$$

Where,

ω = The radian frequency at which the test was performed

Similarly, the mutual-inductance between any two coils ‘x’ and ‘y’ can be determined by energizing coil ‘y’ with all other coils open-circuited. The mutual inductance L_{xy} is then:

$$L_{xy} = \frac{V_x}{\omega \cdot I_y} \qquad (7\text{-}7)$$

Transformer data is often not available in this format. Most often, an equivalent circuit, as shown in Figure 7-2, is assumed for the transformer and the parameters L_1, L_2 and aL_{12} are determined from open and short-circuit tests.

For example if we neglect the resistance in the winding, a short circuit on the secondary side (i.e. $V_2 = 0$) causes a current $V_1/\omega \cdot (L_1 + L_2)$ to flow (assuming $aL_{12} >> L_2$). By measuring this current we may calculate the total leakage reactance $L_1 + L_2$. Similarly, with winding 2 open-circuited the current flowing is $V_1/\omega \cdot (L_1 + a \cdot L_{12})$, from which we readily obtain a value for $L_1 + aL_{12}$.

Conducting a test with winding 2 energized and winding 1 open-circuited, $I_2 = a^2 \cdot V_2/L_2 + a \cdot L_{12}$. The nominal turns-ratio ‘a’ is also determined from the open circuit tests.

PSCAD computes the inductances based on the open-circuit magnetizing current, the leakage reactance and the rated winding voltages.

Derivation of Parameters

To demonstrate how the necessary parameters are derived for use by EMTDC, an example of a two winding, single-phase transformer is presented. The data for the transformer is as shown in Table 7-1:

Parameter	Description	Value
T_{MVA}	Transformer single-phase MVA	100 MVA
f	Base frequency	60 Hz
X_1	Leakage reactance	0.1 p.u.
NLL	No load losses	0.0 p.u.
V_1	Primary winding voltage (RMS)	100 kV
I_{m1}	Primary side magnetizing current	1 %
V_2	Secondary winding voltage (RMS)	50 kV
I_{m2}	Secondary side magnetizing current	1 %

Table 7-1 - Transformer Data

If we ignore the resistances in Figure 7-1, we can obtain the (approximate) value for $L_1 + L_2$, from the short circuit test, as:

$$L_1 + L_2 = \frac{0.1 \cdot Z_{base1}}{\omega_{base1}} = 26.525 \text{ mH} \qquad (7\text{-}8)$$

Where,

$$Z_{base1} = \frac{(100 \cdot kV)^2}{(1000 \cdot MVA)} \quad \text{Base impedance}$$

As no other information is available, we assume for the turns ratio 'a' the nominal ratio:

$$a = \frac{100 \cdot kV}{50 \cdot kV} = 2.0 \qquad (7\text{-}9)$$

We also have for the primary and secondary base currents:

$$I_{base1} = \frac{(1000 \cdot MVA)}{(100 \cdot kV)} = 1.0 \text{ kA} \qquad I_{base2} = 0.5 \text{ kA} \qquad (7\text{-}10)$$

Thus, we see that by energizing the primary side with 100 kV, we obtain a magnetizing current:

$$I_{m1} = 1\% \cdot I_{base1} \qquad I_{m2} = 1\% \cdot I_{base2} \qquad (7\text{-}11)$$

But we also have the following expression from the equivalent circuit:

$$\frac{I_{m1}}{I_{m2}} \cdot \frac{V_{base2}}{V_{base1}} = \frac{1}{a^2} \cdot \frac{(L_2 + a \cdot L_{12})}{(L_1 + a \cdot L_{12})} \qquad (7\text{-}12)$$

Where,

$$\frac{V_{base1}}{I_{m1}} = \omega_{base} \cdot (L_1 + a \cdot L_{12})$$

$$\frac{V_{base2}}{I_{m2}} = \frac{\omega_{base}}{a^2} \cdot (L_2 + a \cdot L_{12})$$

Therefore since,

$$\frac{I_{m1}}{I_{m2}} \cdot \frac{V_{base2}}{V_{base1}} = \frac{1}{a^2} \tag{7-13}$$

Then,

$$L_1 = L_2 \tag{7-14}$$

By combining Equations 7-8 and 7-14 we obtain $L_1 = L_2$ =13.263 mH and from Equation 7-12 we obtain aL_{12} = 26.5119 H. The values for the parameters in Equation 7-1 are then obtained as:

$$L_{11} = L_1 + a \cdot L_{12} = 26.5252 \text{ H} \tag{7-15}$$

$$L_{22} = \frac{L_2 + a \cdot L_{12}}{a^2} = 6.6313 \text{ H} \tag{7-16}$$

$$L_{12} = 13.2560 \text{ H} \tag{7-17}$$

Inverting the Mutual Induction Matrix
It was discussed previously that as the coefficient of coupling K approaches unity, the elements of the inverse inductance matrix become large and approach infinity. This makes it impossible to derive the transformer currents in the form given by Equation 7-5.

An excessively small magnetizing current also leads to such ill conditioning. In such cases, it is often advisable to model the transformer with only leakage reactances and no magnetizing branch, as shown in Figure 7-3. Such a transformer is referred to as 'ideal' in this document and also in PSCAD.

For an ideal transformer, the relationship between the derivatives of current (i.e. dI_1/dt,dI_2/dt) and the voltages can be directly expressed as in Equation 7-18; derived by considering the circuit equations for a short-circuit test conducted on one side, with a voltage source applied to the other (keep in mind that $I_2 = -a \cdot I_1$ and either V_1 or V_2 is zero for a given test):

$$\frac{d}{dt}\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \frac{1}{L} \cdot \begin{bmatrix} 1 & -a \\ -a & a^2 \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \qquad (7\text{-}18)$$

Where,

$L = L_1 + a^2 \cdot L_2$ Leakage inductance between windings 1 and 2 as measured from winding 1 terminal

$a = \sqrt{\frac{L_{11}}{L_{22}}} = \frac{V_1}{V_2}$ Turns-ratio

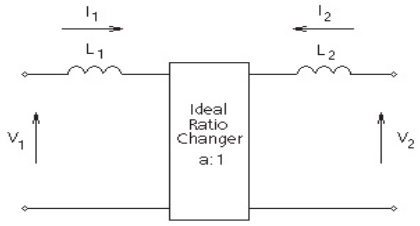

Figure 7-3 - 'Ideal' Transformer Equivalent Circuit

A similar analysis can be used to define the derivatives of the transformer currents in terms of its voltages for an 'ideal' transformer (i.e. zero magnetizing current), when more than two windings are coupled on the same core. Unfortunately, the formulae to calculate these elements are not as simple as they are for a two winding transformer. Therefore, if the ideal transformer option is being used, PSCAD presently allows for a maximum of only three windings per core.

EXAMPLE 7-1

Consider a three winding, 40 MVA transformer with zero magnetizing current. The three-phase winding voltages are 230 kV, 66 kV and 13.8 kV.

An equivalent circuit diagram of the positive sequence leakage reactances is shown in Figure 7-4. The inductances of the equivalent

circuit are all based on the rated voltage of one winding, which for this example is winding 1 (the HV winding), rated at 132.79 kV (230/$\sqrt{3}$). The LV winding rated voltage is 38.1 kV and the tertiary winding rated voltage is 13.8 kV.

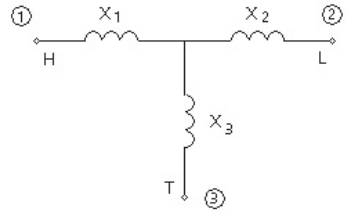

Figure 7-4 - Positive Sequence Leakage Reactance Equivalent Circuit

Where,

$$X_{HL} = 10\%, \quad X_{HT} = 24\%, \quad X_{LT} = 14\%$$

and,

$$X_1 = 10\%, \quad X_2 = 0, \quad X_3 = 14\%$$

For a 60 hertz frequency rating, the inductance of leakage reactance X_1 as shown above, is found to be the expression:

$$L_1 = \frac{V_1^2 \cdot X\% \cdot 0.01}{\omega \cdot MVA} \tag{7-19}$$

The leakage inductance values based on winding 1 voltage are therefore:

$$L_1 = 0.1169 \text{ H}, \quad L_2 = 0.0 \text{ H and } L_3 = 0.1637 \text{ H}$$

As mentioned previously, the inverted inductance matrix of an 'ideal' three winding transformer is not as simple as that of an 'ideal' two winding transformer. The inverted inductance matrix of an 'ideal' three winding transformer is given as follows:

$$\begin{bmatrix} L_{11} & L_{12} & L_{13} \\ L_{21} & L_{22} & L_{23} \\ L_{31} & L_{32} & L_{33} \end{bmatrix} = \frac{1}{LL} \cdot \begin{bmatrix} L_2 + L_3 & -a_{12} \cdot L_3 & -a_{13} \cdot L_2 \\ -a_{12} \cdot L_3 & a_{22} \cdot (L_2 + L_3) & -a_{23} \cdot L_1 \\ -a_{13} \cdot L_2 & -a_{23} \cdot L_1 & a_{33} \cdot (L_1 + L_2) \end{bmatrix} \tag{7-20}$$

Where,

$LL = L_1 \cdot L_2 + L_2 \cdot L_3 + L_3 \cdot L_1$

$a_{12} = \frac{V_1}{V_2}$

$a_{22} = \left(\frac{V_1}{V_2}\right)^2$

$a_{13} = \frac{V_1}{V_3}$

$a_{23} = \frac{V_1^2}{V_2 \cdot V_3}$

$a_{33} = \left(\frac{V_1}{V_3}\right)^2$

V_1 = RMS single-phase voltage rating of winding 1 (HV)

V_2 = RMS single-phase voltage rating of winding 2 (LV)

V_3 = RMS single-phase voltage rating of winding 3 (TV)

Representing Core and Winding Losses
When an 'ideal' transformer is modeled, the magnetizing current is not represented and must be added separately.

Core losses are represented internally with an equivalent shunt resistance across each winding in the transformer. These resistances will vary for each winding in order to maintain a uniform distribution across all windings. The value of this shunt resistance is based on the No Load Losses input parameter.

In most studies, core and winding losses can be neglected because of the little significance to results. Losses in the transmission system external to the transformer tend to dominate.

Core Saturation

Many transformer studies however, do require core saturation to be adequately modeled. Saturation can be represented in one of two ways. First, with a varying inductance across the winding wound closest to the core or second, with a compensating current source across the winding wound closest to the core [3].

In EMTDC, the current source representation is used, since it does not involve change to (and inversion of) the subsystem matrix during saturation. For a two winding, single-phase transformer, saturation using a current source is shown in Figure 7-5.

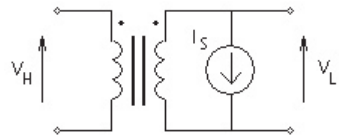

Figure 7-5 - Compensating Current Source for Core Saturation

The current $I_S(t)$ is a function of winding voltage $V_L(t)$. First of all, winding flux $\Phi_S(t)$ is defined by assuming that the current $I_S(t)$ is the current in the equivalent non-linear saturating inductance $L_S(t)$ such that:

$$\Phi_S(t) = L_S(t) \cdot I_S(t) \tag{7-21}$$

The non-linear nature of Equation 7-21 is displayed in Figure 7-6, where flux is plotted as a function of current. The air core inductance L_A is represented by the straight-line characteristic, which bisects the flux axis at Φ_K. The actual saturation characteristic is represented by L_M and is asymptotic to both the vertical flux axis and the air core inductance characteristic. The sharpness of the knee point is defined by Φ_M and I_M, which can represent the peak magnetizing flux and current at rated volts. It is possible to define an asymptotic equation for current in the non-linear saturating inductance L_S if L_A, Φ_K, Φ_M, and I_M are known. Current I_S can be defined as:

$$I_S = \frac{\sqrt{(\Phi_S - \Phi_K)^2 + 4 \cdot D \cdot L_A} + \Phi_S - \Phi_K}{2 \cdot L_A} - \frac{D}{\Phi_K} \tag{7-22}$$

Where,

$$D = \frac{-B - \sqrt{B^2 - 4 \cdot A \cdot C}}{2 \cdot A}$$

$$A = \frac{L_A}{\Phi_K^2}$$

$$B = \frac{L_A \cdot I_M - \Phi_M}{\Phi_K}$$

$$C = I_M \cdot (L_A \cdot I_M - \Phi_M + \Phi_K)$$

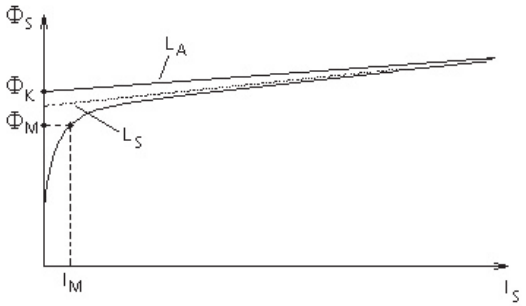

Figure 7-6 - Core Saturation Characteristic of the Classical Transformer

The flux $\Phi_S(t)$ is determined as a function of the integral of the winding voltage $V_L(t)$ as follows:

$$\Phi_S(t) = \int V_L(t) \cdot dt \qquad (7\text{-}23)$$

This method is an approximate way to add saturation to mutually coupled windings. More sophisticated saturation models are reported in literature, but suffer from the disadvantage that in most practical situations, the data is not available to make use of them - the saturation curve is rarely known much beyond the knee. The core and winding dimensions, and other related details, cannot be easily found.

Studies in which the above simple model has been successfully used include:

- Energizing studies for a 1200 MVA, 500 kV, autotransformer for selecting closing resistors. Ranges of inrush current observed in the model compared favourably with the actual system tests.

- DC line AC converter bus fundamental frequency over-voltage studies.

- Core saturation instability where model results agreed closely to actual system responses [9].

In order to appreciate the saturation process described above, Figure 7-7 summarizes the use of Equations 7-22 and 7-23.

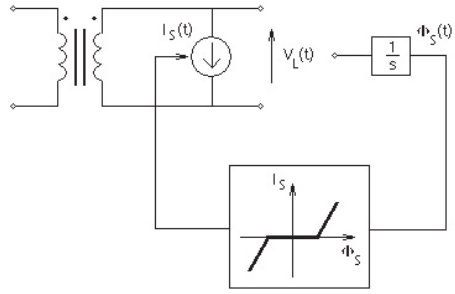

Figure 7-7 - Formulation of Saturation for a Mutually Coupled Winding (TSAT21)

More on Air Core Reactance

The air core reactance L_A in Figure 7-6 may not be known for a transformer under study. One rule of thumb is that air core reactance is approximately twice the leakage reactance.

For example, if the saturation is applied to the tertiary winding of the three winding transformer, a reasonable value to select for air core reactance would be X_{HT}, which is 24%. Thus, as seen from the tertiary, the air core reactance would be 24%; as seen from the low voltage winding, it would be 38%, and as seen from the high voltage winding, it would be 48% or twice the leakage reactance X_{HT}.

The knee point of the saturation curve is sometimes available and is usually expressed in percent or per-unit of the operating point,

defined by rated voltage. Typical ranges in per-unit are 1.15 to 1.25. Referring to Figure 7-6, this would be:

$$\Phi_K = K \cdot \Phi_M \qquad (7\text{-}24)$$

Where,

$$1.15 < K < 1.25$$

If the RMS rated voltage of the winding, across which saturation is applied, is V_M, then Φ_M is:

$$\Phi_M = \frac{V_M}{4.44 \cdot F_R} \qquad (7\text{-}25)$$

Where,

F_R = Rated frequency in Hertz

Equations 7-21 to 7-25 are an approximate means of defining transformer saturation and form the basis upon which the EMTDC subroutine TSAT21 is constructed.

THE UMEC APPROACH

The UMEC (Unified Magnetic Equivalent Circuit) transformer model is based primarily on core geometry. Unlike the classical transformer model, magnetic coupling between windings of different phases, in addition to coupling between windings of the same phase, are taken into account.

In PSCAD, the following transformer core structures can be represented using the UMEC model:

1. Single-phase units with up to four windings.
2. Three-phase, three-limb units.
3. Three-phase, five-limb units.

For more details on the background theory of the UMEC transformer model, see [10].

Basic Modeling Approach

Consider the three-phase, three-limb transformer shown below in Figure 7-8.

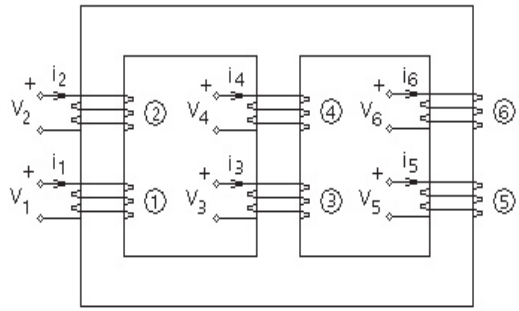

Figure 7-8 - Schematic Representation of the Three-Phase, Three Limb Transformer (2 Windings per Phase)

The voltage-current relationship for the six coils is given by:

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_6 \end{bmatrix} = \begin{bmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_6 \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_6 \end{bmatrix} + \begin{bmatrix} L_1 & M_{12} & \cdots & M_{16} \\ M_{21} & L_2 & \cdots & M_{26} \\ \vdots & \vdots & \ddots & \vdots \\ M_{61} & M_{62} & \cdots & L_6 \end{bmatrix} \cdot \frac{d}{dt} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_6 \end{bmatrix} \qquad (7\text{-}26)$$

Where,

R_i = Winding resistance

L_i = Winding self-inductance

M_{ij} = Mutual inductance between windings i and j

The elements L_i and M_{ij} in Equation 7-26 are dependent on the core dimensions, the magnetic properties of the core material and the number of turns in different windings.

Matrix Element Derivation

Exact core dimensions, along with information about the number of turns and magnetic characteristics, are not generally available to the user. The UMEC model overcomes this drawback by deriving the elements of the inductance matrix, based on the commonly available test data - these include the open and the short circuit tests.

The transformer core, along with windings 1 and 3 are shown in Figure 7-9.

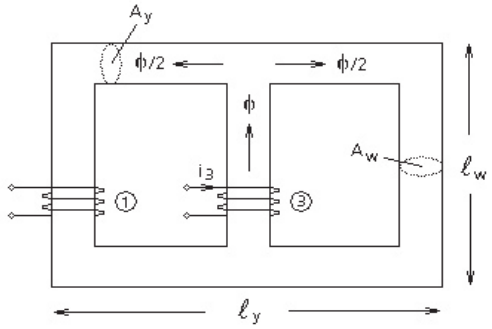

Figure 7-9 - Transformer Core Flux Due To Winding 3 Current

If a current i_3 is passed through winding 3 with all other windings kept open-circuited, the following equations describe the current-flux relationship:

$$N_3 \cdot i_3 = \Re_w \cdot \phi + \Re_y \cdot \frac{\phi}{2} + \Re_w \cdot \frac{\phi}{2} \qquad (7\text{-}27)$$

Where,

$\Re_w$ = Reluctance of the winding limb

$\Re_y$ = Reluctance of the yoke

ℓ_i = Physical length of limb (w) and yoke (y)

A_i = Cross-sectional area of winding limb (w) and yoke (y)

P_i = Permeance of limb (w) and yoke (y)

Therefore for this case,

$$N_3 \cdot i_3 = \left(\frac{3}{P_w} + \frac{1}{P_y} \right) \cdot \phi \qquad (7\text{-}28)$$

The self-inductance of winding 3 is defined as:

$$L_3 = \frac{N_3}{i_3} \cdot \phi = \frac{N_3^{\,2}}{\left(\frac{3}{P_w} + \frac{1}{P_y}\right)} \tag{7-29}$$

The mutual inductance between windings 1 and 3 is defined as:

$$M_{13} = \frac{N_1}{i_3} \cdot \frac{\phi}{2} = \frac{N_1 \cdot N_3 \cdot \mu_0 \cdot \mu_r}{2 \cdot \left(\frac{3}{P_w} + \frac{1}{P_y}\right)} \tag{7-30}$$

Similar expressions can be derived for the other inductance terms in Equation 7-26.

Simple Example Derivation

The following example illustrates the approach adopted by the UMEC models to evaluate the above inductance terms when all the necessary information is not available.

Consider the two simple inductors shown in Figure 7-10:

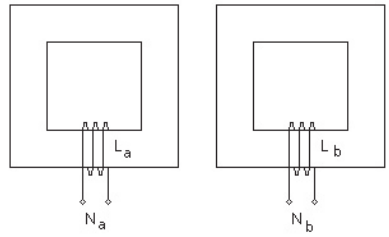

Figure 7-10 - Simple Inductors with Cores

The inductances L_a and L_b take the form:

$$L_j = N_j^2 \cdot P_j \tag{7-31}$$

Where,

P_j = Permeance of the magnetic path

The response of coils ‘a’ and ‘b’ to an electrical signal will depend on the values of L_a and L_b. Even if P_a and P_b are not equal, we can still

select the number of turns in each coil so that $L_a = L_b$. The UMEC approach takes advantage of this fact and assigns the value V_i to the number of turns N_i, so that:

$$N_i = V_i \tag{7-32}$$

Where,

V_i = The rated voltage of the winding in kV

Although the exact dimensions are not known, it is reasonable to assume that the user is provided with a scaled drawing of the transformer. The following aspect ratios can be defined based on such a drawing:

$$r_A = \frac{A_y}{A_w} \tag{7-33}$$

$$r_L = \frac{\ell_y}{\ell_w} \tag{7-34}$$

Thus, substituting these values into Equation 7-29, we get:

$$L_3 = \frac{P_w \cdot N_3^2}{3 + \frac{r_L}{r_A}} \tag{7-35}$$

If winding 3 is then subjected to an open-circuit test (with all other windings kept open-circuited),

$$V_{Rated} = I_{oc} \cdot L_3 \cdot \omega_o \tag{7-36}$$

Where,

V_{Rated} = Rated voltage
I_{oc} = Open-circuit test current in winding 3
ω_0 = Rated frequency

By combining Equations 7-36 and 7-35, the permeance of a winding limb can be found as:

$$P_w = \frac{V_{Rated} \cdot \left(3 + \frac{r_L}{r_A}\right)}{N_3^2 \cdot I_{oc} \cdot \omega_0} \quad (7\text{-}37)$$

Assign the rated voltage value to the number of turns,

$$P_w = \frac{3 + \frac{r_L}{r_A}}{V_{Rated} \cdot I_{oc} \cdot \omega_0} \quad (7\text{-}38)$$

Similarly, the permeance of the yoke can be given as

$$P_y = \frac{1 + 3 \cdot \frac{r_A}{r_L}}{V_{Rated} \cdot I_{oc} \cdot \omega_0} \quad (7\text{-}39)$$

Core Saturation
The UMEC transformer models treat core saturation differently than the classical models. Here, the piecewise linear technique is used to control the model equivalent branch conductance.

The non-linearity of the core is entered directly into the model as a piecewise linear V-I curve, which makes full use of the interpolation algorithm for the calculation of exact instants in changing of state range. See Switching and Non-Linear Elements for more details.

Summary
Once P_w and P_y are calculated in this manner, all the off-diagonal elements in the transformer inductance matrix can be assigned the appropriate value.

The leakage flux is treated in a similar manner. The corresponding inductances are estimated based on short-circuit test results and are added to the self-inductance terms, which forms the diagonal elements.

Thus, in the UMEC models, the transformer inductance matrix is derived based on the nameplate data (V_1, V_2, etc.), the core aspect ratios (r_A and r_L), and short-circuit and the open-circuit test results.

Detailed information of the UMEC approach can be found in [10] and [11]. Interested users are encouraged to refer to these publications.

MODELING AUTOTRANSFORMERS

PSCAD does not currently offer a separate autotransformer model. This section describes how to model an autotransformer, using the existing transformer models with suitable tap changers.

Figure 7-11 illustrates an equivalent circuit for a single-phase autotransformer using a classical single-phase transformer component and a tap changer on the secondary winding (this is one possible method of modeling an autotransformer). The tap can be set up to cover a wide operating range.

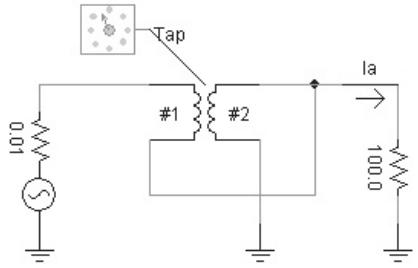

Figure 7-11 - Autotransformer Equivalent Circuit in PSCAD

Some important notes on the use of the configuration in Figure 7-11, as compared with an actual autotransformer model are given as follows:

- The above configuration exactly represents an autotransformer at 100% tap setting.
- The change in tap setting is modeled as a change in the turns-ratio of the transformer. The per unit leakage reactance and magnetizing currents specified for 100% tap are used to calculate admittances for the new voltage rating, corresponding to the tap setting. The magnetizing branch (for a non ideal transformer) is considered to be in the middle of the two winding reactances.

For example, if the magnetizing current is assumed to be negligible, the conductance matrix with tap changer on the secondary winding is calculated as:

$$\frac{1}{L}\cdot\begin{bmatrix} 1 & -\frac{a}{T} \\ -\frac{a}{T} & \frac{a^2}{T^2} \end{bmatrix} \qquad (7\text{-}40)$$

Where,

L = $L_1 + a^2 \cdot L_2$ Leakage inductance between windings 1 and 2 as measured from winding 1 terminal

a = $\sqrt{\frac{L_{11}}{L_{22}}} = \frac{V_1}{V_2}$ Turns-ratio

T = Tap setting on the secondary winding

A similar but more complex expression is used if magnetizing current is considered.

Chapter 8:

Rotating Machines

Introduction to Machines

There are presently four fully developed machine models available in EMTDC: A Synchronous Machine, a Squirrel Cage Induction Machine, a Wound-Rotor Induction Machine and a DC Machine. These models are all programmed in state variable form, using generalized machine theory.

The machine subroutines interface with EMTDC as a compensated current source and special terminating impedance, as well as to other subroutines through mechanical and/or field parameters.

The parameters of the various machines behave differently, so therefore separate subroutines are used. For instance, the Synchronous Machine possesses different parameters on each axis, with only direct-axis mutual saturation of any significance. On the other hand, an induction motor uses the same equivalent circuit on both axes, and experiences both mutual and the leakage reactance saturation.

Other models described in this chapter, which can be used to interface to the machine models, are:

- Exciters
- Governors
- Stabilizers
- Turbines
- Multi-Mass Torsional Shaft Model

Basic Machine Theory

The generalized machine model transforms the stator windings into equivalent commutator windings, using the dq0 transformation as follows:

$$\begin{bmatrix} U_d \\ U_q \\ U_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \cos(\theta - 120^o) & \cos(\theta - 240^o) \\ \sin(\theta) & \sin(\theta - 120^o) & \sin(\theta - 240^o) \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \cdot \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} \qquad (8\text{-}1)$$

The three-phase rotor winding may also be transformed into a two-phase equivalent winding, with additional windings added to each axis to fully represent that particular machine, as is shown in Figure 8-1. For more details on these transformations, please see [12] and [13].

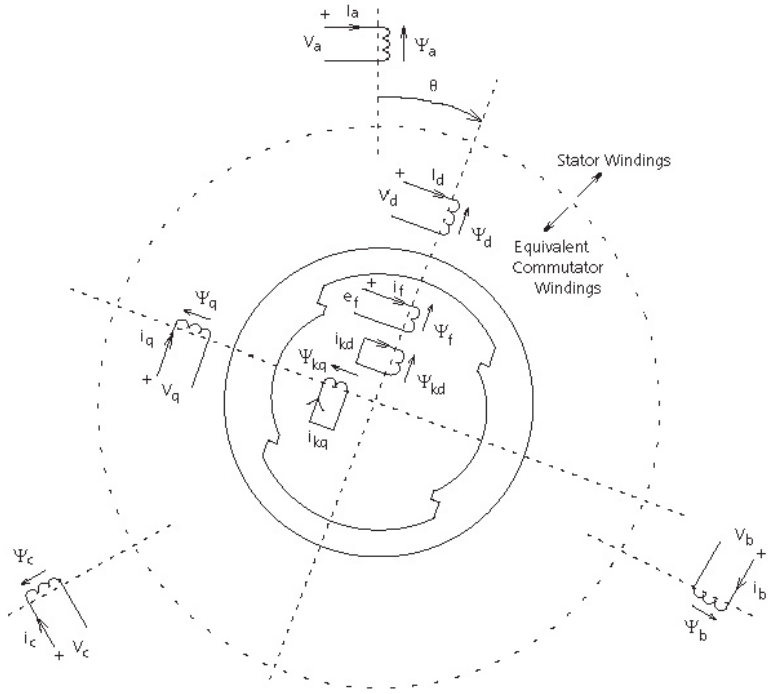

Figure 8-1 - Conceptual Diagram of the Three-Phase and dq Windings

> **NOTE**: All quantities shown in Figure 8-1 are in p.u.

Where,

k =	Amortisseur windings
f =	Field windings
abc =	Stator windings
d =	Direct-Axis (d-axis) windings
q =	Quadrature-Axis (q-axis) windings

Support subroutines are included in the machine model library for calculating the equivalent circuit parameters of a synchronous machine from commonly supplied data. Typical parameters are supplied for small, medium and large squirrel cage motors.

The d-axis equivalent circuit for the generalized machine is shown in Figure 8-2. Figure 8-3 illustrates the flux paths associated with various d-axis inductances:

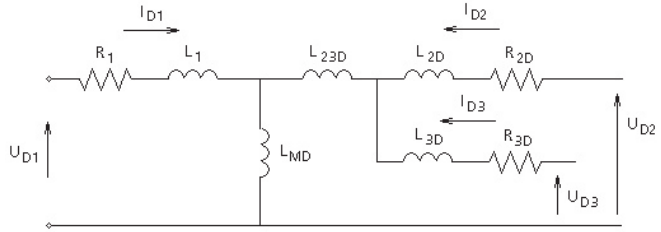

Figure 8-2 - d-axis Equivalent Circuit

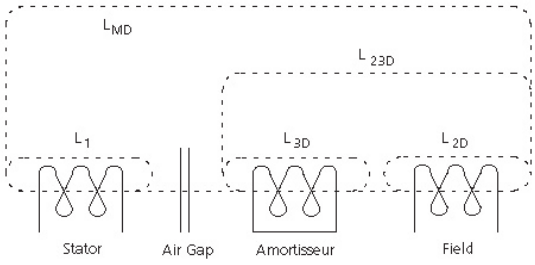

Figure 8-3 - Flux Paths Associated with Various d-axis Inductances

Referring to Figures 8-2 and 8-3, the following equations can be derived:

$$\begin{bmatrix} U_{D1} - \nu \cdot \Psi_q - R_1 \cdot i_{D1} \\ U_{D2} - R_{2D} \cdot i_{D2} \\ U_{D3} - R_{3D} \cdot i_{D3} \end{bmatrix} = L_D \cdot \frac{d}{dt} \begin{bmatrix} i_{D1} \\ i_{D2} \\ i_{D3} \end{bmatrix} \qquad (8\text{-}2)$$

Where,

$$L_D = \begin{bmatrix} L_{MD} + L_1 & L_{MD} & L_{MD} \\ L_{MD} & L_{MD} + L_{23D} + L_{2D} & L_{MD} + L_{23D} \\ L_{MD} & L_{MD} + L_{23D} & L_{MD} + L_{23D} + L_{3D} \end{bmatrix} \quad (8\text{-}3)$$

$$\Psi_q = L_1 \cdot i_{q1} + L_{MQ} \cdot (i_{Q1} + i_{Q2} + i_{Q3}) \quad (8\text{-}4)$$

$$\nu = \frac{d\theta}{dt} \quad (8\text{-}5)$$

Similar equations hold for the q-axis except the speed voltage term $\nu \cdot \Psi_d$, is positive, and:

$$\Psi_d = L_1 \cdot i_{D1} + L_{MD} \cdot (i_{D1} + i_{D2} + i_{D3}) \quad (8\text{-}6)$$

Inversion of Equation 8-2 gives the standard state variable form $\dot{X} = AX + BU$ with state vector X consisting of the currents, and the input vector U, applied voltages. That is:

$$\frac{d}{dt}\begin{bmatrix} i_{D1} \\ i_{D2} \\ i_{D3} \end{bmatrix} = L_D^{-1} \cdot \begin{bmatrix} -\nu \cdot \Psi_q - R_1 \cdot i_{D1} \\ -R_{2D} \cdot i_{D2} \\ -R_{3D} \cdot i_{D3} \end{bmatrix} + L_D^{-1} \cdot \begin{bmatrix} U_{D1} \\ U_{D2} \\ U_{D3} \end{bmatrix} \quad (8\text{-}7)$$

$$\frac{d}{dt}\begin{bmatrix} i_{Q1} \\ i_{Q2} \\ i_{Q3} \end{bmatrix} = L_Q^{-1} \cdot \begin{bmatrix} -\nu \cdot \Psi_d - R_1 \cdot i_{Q1} \\ -R_{2D} \cdot i_{Q2} \\ -R_{3D} \cdot i_{Q3} \end{bmatrix} + L_Q^{-1} \cdot \begin{bmatrix} U_{Q1} \\ U_{Q2} \\ U_{Q3} \end{bmatrix} \quad (8\text{-}8)$$

In the above form, Equations 8-7 and 8-8 are particularly easy to integrate. The equations are solved using trapezoidal integration to obtain the currents. The torque equation is given as:

$$T = \Psi_q - \Psi_d \cdot i_{Q1} \quad (8\text{-}9)$$

and the mechanical dynamic equation for motor operation is:

$$\frac{d\nu}{dt} = \frac{T - T_{MECH} - D \cdot \nu}{J} \quad (8\text{-}10)$$

SALIENT POLE / ROUND ROTOR SYNCHRONOUS MACHINE

The general equivalent circuit for the Synchronous Machine is as shown in Figure 8-1. A second damper winding on the q-axis is included in this model and hence, it can also be used as a round rotor machine to model steam turbine generators. This model is particularly suitable for studying sub-synchronous resonance (SSR) problems as well.

This model operates in the generator mode so that a positive real power indicates electrical power leaving the machine, and a positive mechanical torque indicates mechanical power entering the machine. A positive reactive power indicates the machine is supplying reactive power (i.e. overexcited).

The ability exists to either control the speed directly or to input the mechanical torque, and calculate the speed from Equation 8-10.

Referring to Figures 8-2 and 8-3, the d-axis voltage U_{D2} and current I_{D2} are the field voltage and current, respectively. The damper circuit consists of parameters L_{3D} and R_{3D} with U_{D3} =0. The additional inductance L_{23D} accounts for the mutual flux, which links only the damper and field windings and not the stator winding. The inclusion of such flux, necessary for accurate representation of transient currents in the rotor circuits, is illustrated in [14]. Only the saturation of L_{MD} is considered in this machine and is based on the magnetizing current,

$$i_{MD} = i_{D1} + i_{D2} + i_{D3} \qquad (8\text{-}11)$$

The matrix L_D^{-1} is recalculated each time there is a significant change in the saturation factor.

SQUIRREL-CAGE INDUCTION MOTOR

The Squirrel-Cage Induction Motor is modeled as a double cage machine to account for the deep bar effect of the rotor cage. Both the Direct and Quadrature-Axes are identical and are as shown in Figure 8-1 with $U_{D2} = U_{D3} = U_{Q2} = U_{Q3} = 0$ and $L_{2D} = L_{2Q} = 0$.

Saturation occurs in both the mutual and leakage inductances. The saturation of L_M (both L_{MD} and L_{MQ}) is based on the total magnetizing current,

$$i_M = \sqrt{i_{MD12}^2 + i_{MQ12}^2} \quad (8\text{-}12)$$

while saturation of the leakage inductances L_1 and L_{23} is based on the stator line current,

$$i_L = \sqrt{i_{D12}^2 + i_{Q12}^2} \quad (8\text{-}13)$$

The motor convention is used for the terminal power and shaft torque. That is, positive torque and power indicates motor operation and positive reactive power indicates reactive power into the machine.

WOUND ROTOR INDUCTION MOTOR

In the Wound Rotor Induction Machine, the rotor terminals are accessible to the user, and can be connected to an external resistance or an electrical circuit. In addition to the stator and rotor windings, there is a provision in the model to include up to three additional windings to model the effects of rotor bars (if any).

The d-axis equivalent circuit for the wound rotor induction machine with one squirrel cage in effect is shown in Figure 8-4. This is derived in a manner similar to that for the synchronous machine. Similar equivalent circuits are applicable to the q axis, as well as to the squirrel cage machine.

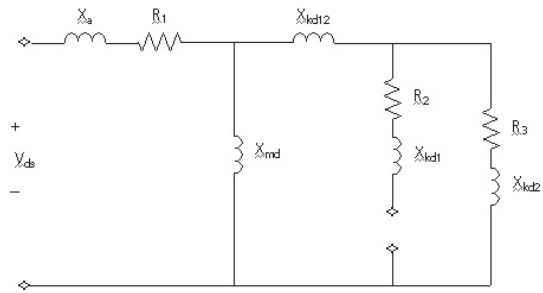

Figure 8-4 - d-axis Equivalent Circuit

NOTE: All reactance and resistance values are those referred to the stator.

Where,

R1= Stator resistance
R2= Wound rotor resistance
R3= First cage resistance
Xa= Stator leakage reactance
Xkd1= Wound rotor leakage reactance
Xkd2= First cage leakage reactance
Xmd= Magnetizing reactance
Xkd12= Mutual inductance - wound rotor - first cage

THE PER-UNIT SYSTEM

The per-unit system is based on the preferred system indicated in [15]. The base values of voltage and current in the three-phase system is the RMS phase voltage V_{ao}, and RMS phase current i_{a0}. The same voltage base is used in the dq0 system but the base current is 3/2 i_{a0}. The dq0 transformation in per unit for the voltage or current is given in Equation 8-1. The inverse transform is:

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 1 \\ \cos(\theta - 120^\circ) & \sin(\theta - 120^\circ) & 1 \\ \cos(\theta - 240^\circ) & \sin(\theta - 240^\circ) & 1 \end{bmatrix} \cdot \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} \qquad (8\text{-}14)$$

Note that unit current in the d-winding produces the same total MMF as unit currents acting in a balanced fashion in the abc-windings. Unit currents in the different circuits should produce the same physical effect. In both systems, base power is:

$$P_0 = 3 \cdot V_{a0} \cdot i_{a0} \qquad (8\text{-}15)$$

The associated bases are:

ω_0 = Rated frequency in radians

t_0 = $1/\omega_0$

L = X

p = p/ω_0

Ψ_0 = $\frac{V_{a0}}{\omega_0}$ Base flux linkage

ν_0 = $\frac{\omega_0}{\text{PolePairs}}$ Base mechanical speed

θ_m = $\frac{\theta}{\text{PolePairs}}$ Mechanical angle

The impedances are given in per unit for both machines. The input voltages are divided by V_{a0} and the incremental time Δt is multiplied by ω_0 to provide a per unit incremental time. The per-unit current is converted to output current by multiplying by i_{a0} after transformation from the two axis-system.

Care should be taken with the following quantities:

- Rated torque is not one per unit for the induction motor but is $\eta \cdot \cos(\theta)/(1 - s)$, which directly relates output mechanical power to input MVA. η is per unit efficiency, $\cos(\theta)$ is power factor and 's' is the rated slip.

- Utilities often specify one per unit field current and voltages as that which produces rated open circuit voltage on the air-gap line (this implies unit power loss in the field circuit). The per unit field current i_{D2}, must be multiplied by XMD0 and then divided by $\sqrt{2}$ in order to convert it to the value of field current used in the utility system. The value of the field voltage is multiplied by R_F/X_{MD0} to give the correct per unit value of U_{D2}.

MACHINE INTERFACE TO EMTDC

The machine models represent the machine as a Norton current source into the EMTDC network. This approach uses the terminal voltages to calculate the currents to be injected.

Figure 8-5 shows a synchronous machine model interfaced to the EMTDC program. The synchronous machine model makes use of the phase voltages calculated by EMTDC to update the injected currents into EMTDC. It is also shown in this figure that multiplying the phase currents by an integer N simulates, from the system point of view, N identical machines operating coherently into the AC system.

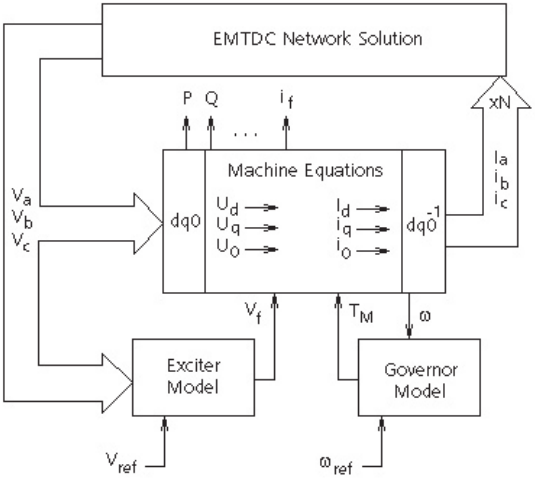

Figure 8-5 - Synchronous Machine Model Interface to EMTDC

Terminating Resistance

It is important to note that representing machines as a Norton current source can have drawbacks. For instance, each machine must be computationally 'far' from other machines for stable operation. In the past, this was usually achieved by separating subsystems containing machines by distributed transmission lines (which are essentially time delays). Since the machine was represented by a simple current source (which was dependent on voltages from the previous time step), any sudden change in voltage would cause a current response only in the next time step. Thus, for the previous time step, the machine looked like an open circuit and spurious spikes appeared on the machine terminal voltage. The cumulative effect of many machines causing this error simultaneously in the same subsystem was proven to be de-stabilizing.

It was found that when the machine neared open circuit conditions, a smaller time step was required to maintain computational

stability. Alternatively, a small capacitance or large resistance could be placed from the machine terminals to ground to prevent the machine from being totally open-circuited. Although the physical meaning of parasitic capacitance or leakage resistance could be applied to these elements, it was not a satisfactory solution.

This idea led to the concept of terminating the machine to the network through a terminating 'characteristic impedance' as shown in Figure 8-6. The effect of this added impedance is then compensated (corrected) by an adjustment to the current injected.

Using this technique, the machine model behaviour has been uniformly good. It essentially combines the compensation-based model and the non-compensated model, while eliminating the restriction of adjacent machines and the necessity of calculating the network Thevenin equivalent circuit.

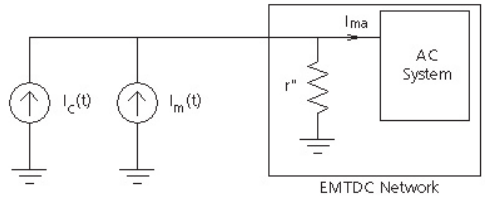

Figure 8-6 - Interface with Terminating Resistance

Where,

$I_c(t) = \frac{V_c(t - \Delta t)}{r''}$ Compensating current

$I_m(t) =$ Calculated machine current

$r'' = \frac{2 \cdot L''}{N \cdot \Delta t}$ Terminating 'characteristic impedance'

The impedance r" is calculated where L" is the 'characteristic inductance' of the machine, N is the number of coherent machines in parallel and Δt is the EMTDC time step. This resistance is placed from each node of the machine terminal to ground within the EMTDC network. Instead of injecting the calculated machine current $I_m(t)$, a compensated current $I_m(t) + I_c(t)$ is injected, where $V(t - \Delta t)$ is the

terminal voltage in the previous time-step. Thus, the actual current injected into the network is,

$$I_{ma}(t) = I_m(t) + \frac{V_c(t - \Delta) - V_c(t)}{r''} \qquad (8\text{-}16)$$

r" is usually quite large, due to the Δt in its denominator. Also, for a small time step $V(t - \Delta t) = V(t)$, and thus $I_{ma}(t) = I_m(t)$, and the error introduced vanishes in the limit with a small Δt. However, for a sudden voltage change, as $I_m(t) + I_c(t)$ is not calculated until the next time step, the network sees the impedance r" for this instant (instead of the open circuit discussed earlier). This is exactly the instantaneous impedance it would have seen had the machine been represented in the EMTDC program main matrix. Therefore, the network current calculated in this instant is more accurate, and the spurious spikes discussed earlier do not arise. Thus, this concept of terminating the machine with its 'characteristic impedance' and then compensating for this in the current injection, is a convenient way for assuring accurate solutions.

MECHANICAL AND ELECTRICAL CONTROL

As shown in Figure 8-4, control blocks used to simulate the excitation and governor systems of the synchronous machine need also to be modeled. These control systems are not included internally in the machine model, so they must be interfaced through external connections.

The exciter and governor systems can either be built using standard control system building blocks (available in the CSMF section of the PSCAD Master Library). Or if preferred, there are standard exciter and governor models available. The following sections briefly describe the general theory behind them.

Exciters

Exciters are externally interfaced to the machine models through external signal connections. Since the exciter models exist as separate components, parameters can be freely selected and adjusted by the user.

Reference [16], published by the IEEE, categorizes exciters into three different types:

1. Type DC excitation systems utilize a DC generator with a commutator as the source of excitation system power.

2. Type AC excitation systems use an alternator, and either stationary or rotating rectifiers, to produce the direct current needed for the synchronous machine field.

3. Type ST excitation systems, where excitation power is supplied through transformers or auxiliary generator windings and rectifiers.

There are many different exciter models classified in one of the above three types, which are representative of commercially available exciters. PSCAD comes complete with each of these standard models.

One of the problems with modeling exciter systems is the inherently large time-constants involved. The user should ensure that the system is started as close to the steady state as possible (unless of course, simulation of start-up transients is required), by properly setting the machine initial conditions. It is also possible, in the case where the exciter transfer function has large time constants, to bypass these during the initial phase. See [16] and [17] for more details on exciters.

Governors

Mechanical transients usually fall in the realm of simulation using stability programs, where the detail of modeling is not as comprehensive as in an electromagnetic transient program. For the duration of most transient simulation runs, the mechanical system can often be considered invariant, and users should make sure that they really need a governor model before they use it.

As mentioned above, most governor transfer functions can be simulated by using control building blocks; interfacing to the machine by either feeding the computed speed or the computed torque.

Governor models are initialized by continuously resetting internal storage locations to produce an output that is exactly equal to the mechanical torque output of the machine.

PSCAD comes complete with both hydro and thermal governor models. These models support variable time step, and have an allowance for initialization at any time (not necessarily at time t =

0.0) through additional control inputs. Refer to [18], [19] and [20] for more on governors.

Stabilizers
Power system stabilizers are used to enhance the damping of power system oscillations, through excitation control. Commonly used inputs to the stabilizers are:

- Shaft speed
- Terminal frequency
- Power

Most stabilizer transfer functions can be simulated through the use of control building blocks. PSCAD comes complete with both single-input and dual-input, power system stabilizers. Refer to [16] for more details.

Turbines
Mechanical power, supplied by turbines, can often be considered invariant for the duration of most transient simulation runs. In some cases however, where provisions are made for fast-valving or discrete control in response to acceleration, prime mover effects can be significant even if the phenomena of interest span only for a few seconds. Also, to ensure the accuracy of longer simulation studies, modeling the turbine dynamics may be considered necessary.

These models support variable time step, and have an allowance for initialization at any time (not necessarily at time t = 0.0) through additional control inputs. Most turbine dynamics can be simulated through the use of control building blocks. PSCAD comes complete with both thermal and hydraulic turbines. Refer to [18] and [19] for more details.

MULTI-MASS TORSIONAL SHAFT MODEL

Some very interesting phenomena can occur when large synchronous machines interact with a power system network. The result can be a sub-synchronous resonance (SSR), which can (and has) literally torn machines shafts apart. The cause of this disaster is the interaction between the mechanical torque placed on turbines and the opposite electrical torque produced by the power system. The resulting torsional stresses on the mechanical connecting shaft, combined with the effect of many masses oscillating back and forth, can be very destructive. A very detailed model of the turbine, generator,

and the mechanical shaft, which couples the mechanical and electrical systems, is required to study such phenomena.

Turbine models have been developed which can accurately represent the dynamics of many masses connected to a single rotating shaft. The models are presently dimensioned to accommodate 6 masses (ex. 5 turbines and 1 generator, or 4 turbines, 1 generator and 1 exciter), but more masses can easily be added if the program is re-dimensioned.

The shaft dynamics and the rotating masses are represented pictorially in Figure 8-7:

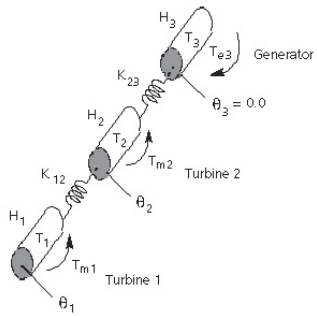

Figure 8-7 - Graphical Model of Multi-Mass Shaft Dynamics

Where,

H =	Inertia constant
K =	Shaft spring constant
T_m =	Mechanical torque on turbines
T_e =	Electrical torque on generators
θ =	Mass angle (reference on generator)

Springs represent the dynamics of the shaft. The torque exerted by the spring is proportional to the relative mechanical angles between adjacent masses.

In addition, two damping coefficients are included. The self-damping coefficient creates a torque on the specified mass, which is proportional to its own speed. Thus, the self-damping feature could be

used to represent friction and windage for each mass. This torque is applied in steady state, as well as in transient conditions.

The mutual damping coefficient creates a torque, which is proportional to the difference in speed from one mass to the next. Thus, this coefficient will not produce torques in steady state, but will damp out oscillations between masses.

Each mass has its own associated inertia constant which reflects the actual size of the mass on the shaft.

The total mechanical torque applied to the shaft (from a governor for example) can be proportioned among each turbine mass. The electrical torque produced by the electrical power system is applied to the generator mass only and opposes the mechanical input torque. In the model, a positive electrical torque corresponds to the generation of electrical power. If an exciter is to be included on the shaft, the input torque on it will be 0.0, but self and mutual damping will still be present.

Multi-Mass Interface
The effects of multiple shaft masses are modeled separately using the Multi-mass Torsional Shaft component. This component interfaces directly with the machine models.

Initialization
The multi-mass output is ignored when the machine rotor is locked, however, the multi-mass model is initialized to the machine conditions.

The multi-mass turbine model may be initialized along with the machine when it changes state freely running machine.

MACHINE INITIALIZATION

Usually a network is started from zero with some simple start-up sequences, such as ramping the source voltage or, if modeling an HVDC system, ramping the power or current order. Usually, most systems can reach steady state in less than 1.0 s. However, if there is a machine in the network, the start-up from zero initial conditions can take longer than usual (sometimes tens of seconds), due to the large electrical and inertial time constants. With multiple machines, the situation can even get worse due to the interaction between

them. Hence, special initialization procedures have been provided to the machine model and to other models interfacing with it.

Synchronous Machine

The Synchronous Machine model can be started from time = 0.0, as a full machine or alternatively, it can be started as a low-resistance voltage source and then converted to a machine after the start-up transients have died away. The latter technique can be used to shorten the time it takes to bring the Synchronous Machine to full steady-running operation.

Initialization for Load Flow

When starting the Synchronous Machine, there is an input argument available that can be used to force the initial machine currents to zero when starting (as a machine) from time = 0.0, regardless of the initial conditions. This feature has a very specific use. Initial conditions for the terminal voltage (i.e. magnitude and angle), as well as real and reactive power, can be entered for initializing the machine. The initial rotor mechanical angle and the initial machine currents will normally be calculated and set. However, if the current is forced to zero, then only the initial rotor mechanical angle will be set and the initial machine currents will be set to zero. This can be useful if the user wishes to bring the synchronous machine up from a de-energized condition, but with the correct rotor angle to fit into a desired load flow.

Starting as a Voltage Source

In order to shorten start-up transients, options are also provided such that the model can start-up as a low-resistance voltage source and subsequently convert to operation as a synchronous machine. In this case, the machine will be initialized at the time of the conversion according to the instantaneous magnitude and angle of the source voltage, and the real and reactive power flowing into the ac system from the source.

For best results, it is recommended that conversion from source to machine operation be delayed until the start-up transient has completely passed. If the source power controller is active, then conversion from source to machine should also be delayed until the output power has stabilized.

Locked Rotor (Rotor Dynamics Disabled) Operation

The conversion from a voltage can either be to a freely running machine, or to a locked rotor state. This additional state is used to

further enhance start-up speed by controlling the rotor mechanical dynamics according to a speed order signal.

The Synchronous Machine provides an additional input argument to transfer from Locked Rotor to Normal (or freely running) mode. When running free, the machine rotor speed varies according to the applied per-unit torque input argument, acting against the electrical torque, friction, windage, and inertia. Operation in this mode uses the inertia constant H and the windage and friction loss specified.

Induction Machines

There are no special initialization considerations for the Induction Machine model. If the motor is used for a constant torque type application, it is good to start up the motor with constant speed mode and then switch to constant torque.

Chapter 9:

Transmission Lines

Introduction to Transmission Lines
Generally in electromagnetic transients simulations, there are two main ways to represent transmission lines. The most familiar method is to use PI sections. The second method is to use a distributed transmission line, which is most suited for transient line response modeling using a digital computer.

A distributed model operates on the principle of traveling waves. A voltage disturbance will travel along a conductor at its propagation velocity (near the speed of light), until it is reflected at the line's end. In a sense, a transmission line or cable is a delay function. Whatever is fed into one end will appear at the other end, perhaps slightly distorted, after some delay. However, there are some other considerations which must be dealt with which include mutual coupling with other conductors, and wave-shape attenuation as it travels along the line.

MODEL SELECTION

There are three basic transmission line or cable modeling techniques in EMTDC: PI sections, the Bergeron Model, and the Frequency-Dependent Line Models. The requirements for your study will determine which of the three models is suitable.

PI Sections
A simple PI section model will give the correct fundamental impedance, but cannot accurately represent other frequencies unless many sections are used (which is inefficient). It also cannot accurately represent the frequency dependent parameters of a line (such as skin effect). It is suitable for very short lines where the traveling wave models cannot be used. For EMTDC, PI sections are not considered a very elegant means of transmission line modeling for the following reasons:

- Greater computational time and increased matrix sizes (if used as a series of PI sections).

- Not practical with a large number of mutually coupled conductors.
- Frequency dependent attenuation of the traveling waves is not easily or accurately incorporated.

The Bergeron Model
The Bergeron Model represents the L and C elements of a PI section in a distributed manner (not using lumped parameters like PI sections). It is roughly equivalent to using an infinite number of PI sections except that the resistance is lumped (i.e. ½ in the middle of the line and ¼ at each end). Like PI sections, it also accurately represents the fundamental frequency. It also represents impedances at other frequencies, except that the losses do not change. This model is suitable for studies where the fundamental frequency load-flow is most important (e.g. relay studies).

The Frequency-Dependent Models
The Frequency Dependent Line Models represent the frequency dependence of all parameters. This model takes longer to run than the Bergeron model, but is necessary for studies requiring a very detailed representation of the line over a wide frequency range. Frequency dependent models can be solved using modal techniques (as in PSCAD V2 models) or using the more advanced phase domain techniques (the best model).

The Transmission Line and Cable Constants routine is used to generate the data required by the EMTDC line models. Only the conductor properties (i.e. resistance, radius, etc.) and tower geometry are required to model the line. The Transmission Line and Cable Constants routine then stores the solved line constants data into output files.

BERGERON LINE MODEL

The Bergeron method is based on a distributed LC parameter traveling wave line model with lumped resistance. This model produces a constant surge impedance and is essentially a single frequency model. The Bergeron method can be used for any general fundamental frequency impedance studies, such as relay testing or matching load-flow results.

Single Conductor Model

In order to appreciate the concept of a distributed transmission line, a single conductor overhead transmission line is considered. Multi-phase and mutually coupled transmission lines follow similar concepts. A distributed transmission line interfaces to the EMTDC electric network as shown in Figure 9-1:

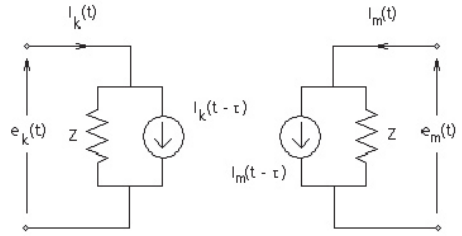

Figure 9-1 – Distributed Transmission Line Interface to EMTDC

The Transmission Line and Cable Constants routines will have generated:

- Z_0 = The Characteristic or surge Impedance [Ω].
- R = The total line resistance [Ω].
- τ = The travel time of the line [s].

The travel time τ and the characteristic impedance Z_0 can be related to inductance and capacitance of the transmission line as follows:

$$\tau = \ell \cdot \sqrt{LC} \qquad (9\text{-}1)$$

$$Z_0 = \sqrt{\frac{L}{C}} \qquad (9\text{-}2)$$

where:

ℓ =	Length of the transmission line
L =	Inductance [H / unit length]
C =	Capacitance [F / unit length]

The Norton interface impedance at each end of the line (shown as Z above) is defined as:

$$Z = Z_0 + \frac{R}{4}$$

The Norton current injected at each end of the line has been determined as in [1] to be:

$$I_k(t - \tau) = \left[\frac{e_m(t - \tau)}{Z} + H \cdot I_m(t - \tau)\right] \quad (9\text{-}3)$$

$$I_m(t - \tau) = \left[\frac{e_k(t - \tau)}{Z} + H \cdot I_k(t - \tau)\right] \quad (9\text{-}4)$$

where,

$$H = \frac{Z_0 - \frac{R}{4}}{Z_0 + \frac{R}{4}} \quad (9\text{-}5)$$

These equations are derived by breaking a loss-less line into two sections, inserting R/2 in the middle and R/4 at each end of the line.

The EMTDC model actually has two frequency paths, a lower and an upper frequency path. This enables the transmission losses to be accurately represented at the fundamental frequency, and to have increased losses at higher frequencies. The line is not exactly modeled at the higher frequency however, because the surge impedance, travel time and transformation matrix are still the steady state values. The low frequency should be the fundamental frequency of the studies system and would generally be then either 50 or 60 Hz for an AC transmission line. For a DC line, a low frequency, such as 5 Hz, should be used. The surge impedance and travel time are calculated at this frequency.

The second, higher frequency is used for calculating the high frequency attenuation of the line. The Transmission Line and Cable Constants routines will create a split at the log average of the two frequencies specified to determine high and low frequency attenuation. A general high frequency to enter would be between 100 Hz and 2 kHz. This high frequency can, if desired, be chosen to accommodate specific resonance conditions the user might be interested in.

A block diagram of a single mode distributed transmission line used in EMTDC is shown in Figure 9-2:

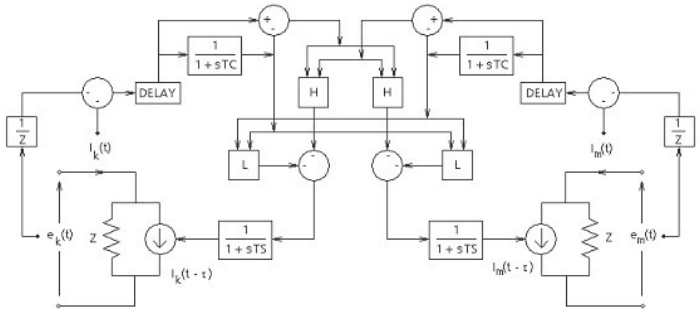

Figure 9-2 – Single Mode of a Distributed Transmission Line

Just before the current is injected into the line terminations, it is processed through a real pole with a shaping time constant of TS seconds, as shown above.

Selection of the wave shaping constant TS can be made in order to shape the steep front of a travelling wave to a known attenuation and slope. If one is concerned about matching the steep front from the EMTDC line model to a known response, adjustment by trial and error of TS will enable this to be done.

Care should be taken when using the wave shaping time constant, because although the real pole will attenuate the high frequency components of a line, it also introduces an additional lag or time delay which affects the overall travel time, and thus the line impedance.

The Bergeron transmission line model should generally be chosen over a PI Section equivalent whenever the lines are sufficiently long, so that the propogation time (approximately the speed of light) is accounted for in the simulation time step. For a general simulation time step of 50 µs, lines over 15 km could be represented by the Bergeron model (assuming travel time as the speed of light).

The Bergeron model also has an important advantage over PI section in that it does not introduce an artificial resonance at high frequencies, as do lines modeled with a limited number of PI sections.

Multi-Conductor Model

So far the mechanism of EMTDC's distributed transmission line has been discussed considering one conductor or mode only. Generally, transmission lines consist of several mutually coupled phases or conductors. These need to be broken-up into modes.

There is one mode for each phase that is mutually coupled. Each mode can be treated as an independent, single-phase transmission line. This method allows unbalanced mutually coupled lines to be modeled. A modal transformation converts the modal response back into the EMTDC network quantities. The modal transformation (and modal quantities) is obtained automatically by the Transmission Line and Cable Constants routines through eigenvalue analysis.

For unbalanced transmission lines, each mode will have different surge impedances and travel times. Usually there are two main types of modes. Firstly, the ground mode or common mode or zero sequence mode. This mode is active whenever ground currents flow. Second, the remaining modes are known as metallic modes, differential modes or positive and negative sequence modes. The ground mode has a longer travel time, higher surge impedance and higher line resistance than the metallic modes. The important aspect of modal analysis is the modal transformation matrix. To summarize this procedure, a matrix diagonalization technique is undertaken so that:

$$[V_{phase}] = [T_e] \cdot [V_{mode}] \tag{9-6}$$

$$[I_{phase}] = [T_i] \cdot [I_{mode}] \tag{9-7}$$

$$[T_e]^T = [T_i]^{-1} \tag{9-8}$$

where,

$[V_{mode}]$ = Line end mode voltage

$[I_{mode}]$ = Line end mode current

$[V_{phase}]$ = Line end phase voltage

$[I_{phase}]$ = Line end phase current

$[T_e]$ = A mode volts to phase volts transformation matrix

$[T_i]$ = A mode current to phase current transformation matrix

NOTE: Eigenvalue analysis is used by the Transmission Line and Cable Constants routine to calculate the transformation matrices.

When there is only a single conductor above a ground plane, there is just a single mode. The path the current can circulate is along the conductor and back through the ground. This is particularly true if one end of the conductor is energized with a voltage source and the other is grounded as shown below:

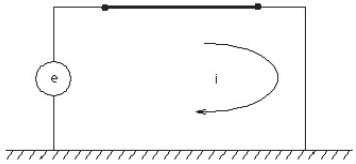

Figure 9-3 – Single Mode for a Single Conductor with Ground Return

For a two-conductor transmission line, such as a bipolar DC line, there are two paths return current can flow, as shown in the following figures:

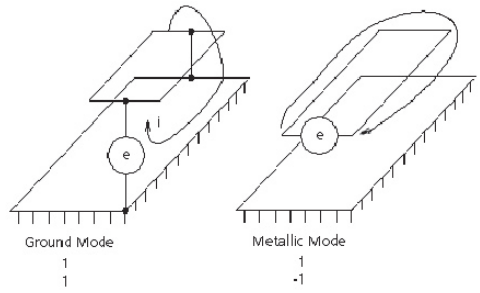

Figure 9-4 – Two Modes of a Two-Conductor Transmission Line

For the ground mode, the current flows down both conductors and back through the ground. The current flows down one conductor and back on the other for the metallic mode. Each of these modes can be differentiated by defining the direction in which current flows as either a +1 or -1. The modal transformation matrix is then:

$$[T_e] = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{9-9}$$

and when normalized it becomes,

$$[T_e] = [T_i] = \begin{bmatrix} 0.707 & 0.707 \\ 0.707 & -0.707 \end{bmatrix} \tag{9-10}$$

It is interesting to note that the transformation matrix for 1 and 2 conductor flat lines does not change with frequency. This is because the conductors are always balanced. If a three-phase line is balanced, then the transformation is also constant with frequency. This is the case for ideally transposed lines, or a delta configured line very high in the air. Under those conditions, the transformation becomes Clark components:

$$[T_e] = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 2 \\ 1 & -1 & -1 \end{bmatrix} \tag{9-11}$$

and when normalized becomes,

$$[T_e] = [T_i] = \begin{bmatrix} 0.5774 & 0.7071 & -0.4082 \\ 0.5774 & 0 & 0.8165 \\ 0.5774 & -0.7071 & -0.4082 \end{bmatrix} \quad (9\text{-}12)$$

The physical realization of the Clark transformation is shown in the following figure:

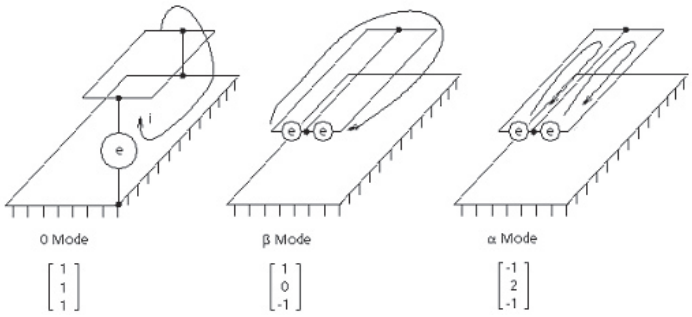

Figure 9-5 – Three Modes of a Three-Conductor Transmission Line using Clark Components

As a three-phase circuit is not usually perfectly balanced, then

$$[T_e] \neq [T_i]$$

and Equation 9-8 determines the relation between the voltage and current transformation matrices. The transformation matrices are represented as real, even though in reality they may be frequency dependent and consist of both real and imaginary components. For overhead conductors, the imaginary components are small compared with the real components and it is a reasonable approximation to neglect the imaginary parts.

FREQUENCY DEPENDENT LINE MODELS

The Frequency-Dependent Line models are basically distributed RLC traveling wave models, which incorporate the frequency dependence of all parameters. Since the Bergeron model is only adequate for studies that essentially require the correct fundamental frequency impedance, the Frequency Dependent Line models should be used for all studies that require frequencies other than the fundamental to be represented accurately (such as, transient over-voltages, harmonic analysis, etc.).

Two frequency dependent models are available: The Frequency Dependent (Phase) model is the most accurate, as it represents the frequency dependence of internal transformation matrices, whereas the Frequency Dependent (Mode) model assumes a constant transformation. For systems of ideally transposed conductors (or 2 conductor horizontal configurations), the 2 models will give identical results (as the transformation is constant anyway).

The Frequency Dependent (Phase) model is numerically robust and more accurate than any other commercially available line/cable model, and thus, is preferred.

Frequency Dependent (Mode) Model

The Frequency Dependent (Mode) model is based on the theory developed in [23]. In order to arrive at the time domain formation of the line equations, it is convenient to first work in the frequency domain, where an exact solution for a given frequency is possible.

Consider the following circuit of a transmission line as seen from the terminations:

Figure 9-6 - Frequency Domain Line Circuit

For a given frequency, the voltages and currents at one end of the line, may be represented in terms of the voltage and current at the other end by:

$$V_k(\omega) = \cosh[\gamma(\omega) \cdot L] \cdot V_m(\omega) - Z_c(\omega) \cdot \sinh[\gamma(\omega) \cdot L] \cdot i_m(\omega) \quad (9\text{-}13)$$

$$i_k(\omega) = \frac{\sinh[\gamma(\omega) \cdot L]}{Z_c(\omega)} \cdot V_m(\omega) - \cosh[\gamma(\omega) \cdot L] \cdot i_m(\omega) \quad (9\text{-}14)$$

Where,

$\gamma(\omega)$ = $\sqrt{Y(\omega) \cdot Z(\omega)}$ propagation constant

$Z_c(\omega)$ = $\sqrt{Z(\omega)/Y(\omega)}$ surge impedance

$Y(\omega)$ = $G + j\omega C$ shunt admittance of the line

$Z(\omega)$ = $R + j\omega L$ series impedance of the line

By introducing the forward and backward traveling wave functions F_k and B_k:

$$F_k(\omega) = V_k(\omega) + Z_c(\omega) \cdot i_k(\omega) \quad (9\text{-}15)$$

$$B_k(\omega) = V_k(\omega) - Z_c(\omega) \cdot i_k(\omega) \quad (9\text{-}16)$$

and similarly at node m:

$$F_m(\omega) = V_m(\omega) + Z_c(\omega) \cdot i_m(\omega) \quad (9\text{-}17)$$

$$B_m(\omega) = V_m(\omega) - Z_c(\omega) \cdot i_m(\omega) \quad (9\text{-}18)$$

Substituting Equation 9-16 into Equation 9-15 gives:

$$F_k(\omega) = 2 \cdot V_k(\omega) - B_k(\omega) \quad (9\text{-}19)$$

and similarly,

$$F_m(\omega) = 2 \cdot V_m(\omega) - B_m(\omega) \quad (9\text{-}20)$$

Equations 9-13 and 9-14 (and their equivalents at node m) can now be expressed in terms of the forward and backward traveling functions:

$$B_k(\omega) = A(\omega) \cdot F_m(\omega) \quad (9\text{-}21)$$

$$B_m(\omega) = A(\omega) \cdot F_k(\omega) \quad (9\text{-}22)$$

where,

$$A(\omega) = \frac{1}{\cosh[\gamma(\omega) \cdot L] \cdot \sinh[\gamma(\omega) \cdot L]} = e^{-\gamma(\omega)L} \qquad (9\text{-}23)$$

$A(\omega)$ is known as the propagation constant and is a complex number. The real part α is the attenuation constant, and the imaginary part β is the phase constant.

Equations 9-16 and 9-18 can be represented with the equivalent circuit as shown below:

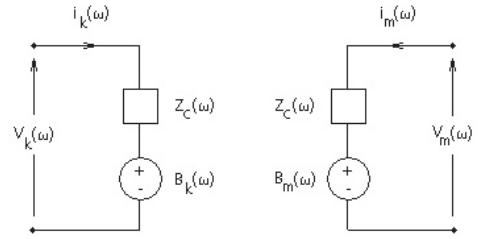

Figure 9-7 - Frequency Domain Equivalent Circuit

Equations 8-20 and 8-21 relate the source $B_k(\omega)$ with the m side quantities $V_m(\omega)$ and $B_m(\omega)$:

$$B_k(\omega) = A(\omega) \cdot [2 \cdot V_m(\omega) - B_m(\omega)] \qquad (9\text{-}24)$$

and similarly,

$$B_m(\omega) = A(\omega) \cdot [2 \cdot V_k(\omega) - B_k(\omega)] \qquad (9\text{-}25)$$

The goal is to represent the circuit in the figure above, and Equations 9-23 and 9-25 in the time domain. The problem however, is the multiplication in Equation 9-25. A multiplication of the frequency domain becomes a convolution in the time domain:

$$A(\omega) \cdot B_m(\omega) \leftrightarrow \int_{\tau}^{t} A(u) \cdot B_m(t - u) \cdot du \qquad (9\text{-}26)$$

Note the lower limit of the integral of Equation 9-26 is the travel time τ, because an impulse on one end of the line will not reach the other

end until τ seconds. The travel time is calculated using the imaginary term β of the propagation constant.

Equation 9-26 is still in a very inconvenient form for a time domain solution, because with each time step, more and more terms of the convolution integral must be evaluated. Fortunately, the convolution can be computed in a recursive form suitable for time domain solution (see [25] for more details).

The recursive convolution algorithm requires $A(\omega)$ to be in the form of a sum of exponentials. This is achieved in the Transmission Line and Cable Constants routines. This program first generates $A(\omega)$ for 100 frequencies equally spaced on a log scale over a wide frequency range. This curve is then approximated with an Nth order rational function A(s), using curve fitting techniques. The number of pole zeroes required depends on the maximum allowed error between the curves. The function A(s) can then be broken down, using partial fraction expansion, into a sum of exponential terms. The more terms there are in A(s), the more terms there are in the recursive convolution, which means the line model takes longer to run.

One point not yet discussed is the representation of the surge impedance in the time domain. The Transmission Line and Cable Constants program generates a curve for $Z_C(\omega)$. The same curve fitting routine used to generate A(s) is used to generate an approximate $Z_C(s)$. This function is now realized by replacing $Z_C(\omega)$ in the last figure with an RC circuit which has the same impedance as $Z_C(s)$. The RC circuit is then solved using the same trapezoidal integration technique as is used in the main program. Finally, the extension of this theory to multi-conductor lines is achieved using the same modal transformations discussed earlier.

It is only necessary to approximate the current transformation matrix (this transforms modal currents to phase currents). It is also necessary to transform the phase voltages to modal quantities, which requires the inverse of the voltage transformation matrix. However, the required matrix is merely the transpose of the current transform.

Frequency Dependent (Phase) Model

In recent years, the need for a transmission line model, which can accurately simulate the undesirable interactions between DC and AC lines in proximity to one another, has become more prevalent. Constant transformation matrix models with frequency dependent modes, such as the Frequency Dependent (Mode) model in EMTDC, have

proven to be unreliable in accurately simulating such situations. In addition, inaccurate representation of unbalanced line geometry has also been a problem.

In 1999, the Frequency Dependent (Phase) model was incorporated into PSCAD to address these shortcomings, thereby providing a general and accurate model for all frequencies. The Frequency Dependent (Phase) model is based on the theory originally proposed in [26], and its actual implementation into EMTDC is outlined in more detail in [27].

The Frequency Dependent (Phase) model transcends the frequency dependent transformation matrix problem by direct formulation in the phase domain. As a result, the matrix elements of the propagation function, $H(\omega) = e^{-\sqrt{Y \cdot Z} \cdot \ell}$, and the characteristic admittance $Y_C(\omega)$ are also fitted directly in the phase domain. Although the $Y_C(\omega)$ elements are relatively smooth and fitting is somewhat straightforward, $H(\omega)$ has proven far more difficult, due to the fact that its elements possess modal contributions with varying time delays.

Frequency Domain Fitting

Fitting is accomplished using a least squares fitting routine called Vector Fitting [28]. First the modes of $H(\omega)$ are calculated through a frequency dependent transformation matrix. Each mode is then fitted as,

$$e^{s\tau_i} \cdot H_i^m(s) = \sum_{m-1}^{N} \frac{c_m}{s - a_m} \qquad (9\text{-}27)$$

$H(\omega)$ is then fitted in the phase domain using,

$$h(s) = \sum_{i-1}^{Ng} \left(\sum_{m-1}^{N} \frac{c_{mj}}{s - a_{mj}} \right) \cdot e^{-s\tau_i} \qquad (9\text{-}28)$$

> **NOTE**: The Vector Fitting routine requires both the mode magnitude and phase angle to be known prior to fitting. In other words, the time constant τ_i must be pre-calculated.

The poles of the different modes may, in some instances, be similar due to the fact that the modes are fitted independently. Instabilities may occur in the time domain if these similarities occur at low frequencies, and so a warning is given if the ratio between phase domain residues and poles is greater than 100. This can be corrected by decreasing the fitting order of magnitude.

Because the characteristic admittance $Y_C(\omega)$ has no time delays, the proper poles can be found by fitting the sum of all modes. However, due to the relation,

$$\sum_{i-1}^{N} \lambda_i = \sum_{i-1}^{N} A_{ij} \qquad (9\text{-}29)$$

where,

A = A square matrix

λ_i = Eigenvalues of A

all that is needed is to sum the diagonal elements of $Y_C(\omega)$. The resulting sum is then approximately fitted by,

$$f(s) = d + \sum_{m-1}^{N} \frac{c_m}{s - a_m} \qquad (9\text{-}30)$$

Lastly, the elements of $Y_C(\omega)$ are fitted in the phase domain using 9-27.

Time Domain Implementation

The interface between the travelling wave equations and EMTDC is illustrated in Figure 9-8 below.

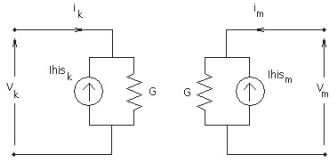

Figure 9-8 - Frequency Dependent (Phase) Model Interface to EMTDC

The currents $Ihis_k$ and $Ihis_m$ are updated in the following sequence:

1. $I_k(n) = G \cdot V_k(n) - Ihis_k(n)$
2. $I_{kr}(n) = I_k(n) - I_{ki}(n)$
3. $I_{ki}(n + 1) = H * I_{mr}(n - \tau)$
4. $Ihis_k(n + 1) = Y_C' * V_k(n) - 2 \cdot I_{ki}(n + 1)$

Where,

i	Denotes incident waves
r	Denotes reflected waves
*	Indicates a convolution integral

For a detailed description of the convolution integration, see [27].

V_k and V_m are calculated in EMTDC and read into the Transmission Line and Cable Constants routine.

User Options
The frequency dependent line model has a number of parameters, which can be selected or adjusted from the model options menu in PSCAD.

The effect of interpolating the travel time is discussed in Time Step Dependencies.

The maximum number of poles/zeroes is usually entered as 20 (the internal maximum dimension is 50). If the curve fitting routine uses all 20 poles, then the approximated curve for the parameter being fitted will be constant for all higher frequencies.

The maximum error will affect the number of poles/zeroes used in the approximation. In general, a smaller error will result in less error in the curve fitting, but will result in a less efficient model due to a larger number of poles. The error is calculated as a percentage of the maximum of the curve. For the attenuation constant and the transform, this is consistent as 1.0. For the surge impedance however, the maximum value will depend on the starting frequency (see below).

The user must select a starting and an end frequency. The upper frequency may be beyond the highest frequency, which can be repre-

sented with the chosen time step, but the program will truncate the curve fitting data to preserve efficiency. A good choice for the end frequency is 1 MHz.

The lower frequency must be selected cautiously, however. Internally in the line constants equations, the shunt conductance G of the line is 1 x 10^{-10}. This means that the surge impedance,

$$Z = \sqrt{\frac{R + j\omega L}{G + j\omega C}} \qquad (9\text{-}31)$$

will tend to become larger as the frequency decreases. The selection of the starting frequency will introduce an effective shunt conductance G' where,

$$Z_{dc} = \sqrt{\frac{R}{G'}} \qquad (9\text{-}32)$$

This means that if the curve fitting is started at 10 Hz, a significant shunt conductance will be introduced. If the starting frequency is reduced, then the shunt conductance is smaller (i.e. less shunt losses).

Care must also be taken, as the choice of the starting frequency will affect the accuracy of the curve fitting of the surge impedance. This is because the maximum error is specified as a percentage of the maximum, and the surge impedance will get larger as the starting frequency is lowered. Finally, choosing too low of a starting frequency means that there may be poles/zeroes at these low frequencies. This can introduce very long time constants in the line models, which means you must run for a long time to reach steady state. A good starting frequency is about 0.5 Hz.

TIME STEP DEPENDENCIES

The Bergeron and Frequency Dependent line models are based on traveling wave theory, and thus have a limitation on the minimum length where they can be used (i.e. the travel time must be >= the time step). For a time step of 50 µs, this length is about 15 km. For lines shorter than this, a PI section model is more suitable.

When the traveling wave line models are used, the calculated travel time of the line/cable will not be an exact integer multiple of the time step. The user is given a choice to interpolate the travel time or not. Not interpolating the travel time can introduce errors by artificially increasing or decreasing the effective line length. Interpolating the travel time will give the correct effective length (and the correct fundamental impedance), but has the disadvantage that it adds additional damping at high frequencies. For example, an open ended loss-less line with a step input should result in the open-end voltage stepping from 0 to 2 forever. If interpolation of the travel time is chosen, the square wave will eventually lose all high frequency components and become more of a sine wave.

If the line is short, then interpolation should be selected. For example, if the line is 40 km, then the travel time (assuming the speed of light) is approximately 133 μs (2.667 time steps @ 50 μs). Modifying the travel time to 3 time steps can introduce a significant error. If the line is long, then interpolating the travel time is not required. The line model will merely take the closest integer multiple. For example, if the line is 1000 km, then the travel time is approximately 3,333 μs (66.67 time steps @ 50 μs). Modifying the travel time to 67 time steps will not introduce a significant error.

TRANSPOSED TRANSMISSION LINES

Many long AC transmission lines have transposed conductors to minimize unbalances in the AC system. There are essentially two methods to model transposed lines in EMTDC. These two methods are valid for both the Bergeron Model and both the Frequency Dependent (Phase) and Frequency Dependent (Mode) models.

1. The first method requires each section of the total line to be represented as individual lines. The circuit interconnections between the line sections are then transposed as in the real system. For example, phase A of the three-phase line may be Conductor 1 for Line section 1, Conductor 3 for the second line, etc.

2. The second method for modeling transpositions is to assume that each circuit is completely balanced or ideally transposed. The line may be represented as a single section from sending end to receiving end using this option. This is a reasonable approximation for relatively short lines, but the

high frequency response for long lines will be affected. The Transmission Line and Cable Constants program will allow the user to select the ideally transposed option.

TRANSMISSION LINE AND CABLE CONSTANTS

Each line model generates the following input data (organized into tables), which is required by the Transmission Line and Cable Constants routines to generate the data required by all of the EMTDC line models.

> **NOTE**: Some additional data (such as conductor names, voltage and current magnitude and phase, AC or DC) are not required in the line constants equations, but are merely for the sake of completeness on the line data summary printout.

Transmission Line Constants Routine:

Conductors	Symbol	**Units**
Number of bundles (or phases)	Nb	
Number of sub-conductors per bundle	Nc	
Radius of a sub-conductor	Rad	m
DC resistance/length of a sub-conductor	Rdc	Ω/m
Conductor shunt conductance	G	S/m
Sub-conductor spacing	Scs	m
Horizontal distance of bundle (any reference may be used)	X	m
Height of a bundle at the tower	Ytower	m
Sag of the bundle at mid-span (maximum height - minimum height)	Sag	m

Table 9-1 - Conductor Data for Transmission Line

Ground Wires	Symbol	**Units**
Number of ground wires	Ngw	
Radius of each ground wire	Rad	m
DC resistance/length	Rdc	Ω/m
Horizontal distance of ground wire (any reference may be used)	X	m
Height of ground wire at the tower	Ytower	m
Sag of the ground wire at mid-span (maximum height - minimum height)	Sag	m

Table 9-2 - Ground Wire Data for Transmission Line

Other Data	Symbol	**Units**
Length of Line	L	m
Ground resistivity	Cg	Ωm

Table 9-3 - Other Data for Transmission Line

Cable Constants Routine:

Conductors	Symbol	**Units**
Number of cables	Nc	
Last layer of cable	LL	
Horizontal position of cable (any reference may be used)	Xi	m
Height of a conductor (negative for below water/ground)	Yi	m
Ground last metallic layer to surrounding medium	LC	

Table 9-4 - Conductor Data for Cable

Conductor Layers	Symbol	**Units**
Inner radius of cable conductor	r1	m
Outer radius of each cable layer	r2…r7	m
Resistivity of conductor, armour and sheath	ρ_C, ρ_A, ρ_S	Ωm
Relative permittivity of insulators 1, 2 and 3	ε_1, ε_2, ε_3	
Relative permeability of all layers	μ_1, μ_2, μ_3 μ_C, μ_A, μ_S	

Table 9-5 - Conductor Layer Data for Cable

Other Data	Symbol	**Units**
Length of Line	L	m
Ground resistivity	Cg	Ωm

Table 9-6 - Other Data for Cable

The following flowchart shows the basic calculations performed by the Transmission Line Constants routine.

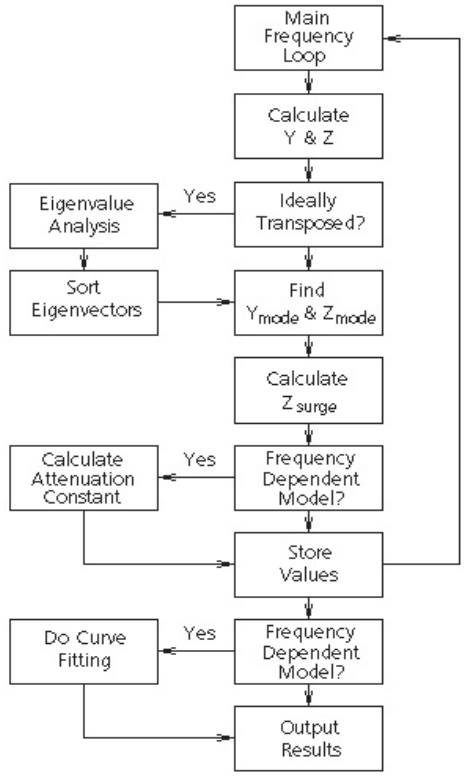

Figure 9-9 - Transmission Line Constants Routine Flowchart

Where,

Y =	Phase shunt admittance matrix
Z =	Phase series impedance matrix
Y_{mode} =	Modal shunt admittance
Z_{mode} =	Modal series impedance

Line Constants Equations

The formula used for the shunt admittance and series impedances [21] will be expressed in terms of the symbols used at the beginning of this chapter.

The shunt admittance generally has a real term G and an imaginary term uC. The shunt conductance G is assumed to be zero. The

curve- fitting algorithm will effectively introduce a shunt conductance term however. The following equation generates the inverse of Y:

$$Y_{ij}^{-1} = \frac{1}{j\omega 2\pi \cdot \varepsilon_0} \cdot \ln\left(\frac{D_{ij}}{d_{ij}}\right) \qquad (9\text{-}33)$$

Where,

$$\varepsilon_0 = \quad 8.854188 \cdot 10^{-12}$$

If i is not equal to j then,

$$D_{ij} = \sqrt{(X_i - X_j)^2 - (Y_i + Y_j)^2} \qquad (9\text{-}34)$$

$$d_{ij} = \sqrt{(X_i - X_j)^2 - (Y_i - Y_j)^2} \qquad (9\text{-}35)$$

If i = j then,

$$D_{ij} = 2 \cdot Y_i \qquad (9\text{-}36)$$

Bundling is accomplished by first calculating the impedances and admittances for the entire system of conductors. Then the matrices are collapsed to eliminate the bundled conductors. This method is more accurate than the GMR formula used in PSCAD V2, particularly when the phase spacing is small compared to the sub-conductor spacing.

The series impedance of an overhead transmission line is considerably more difficult to calculate. The equations for Z must include the effect of an imperfect ground. This can be represented using Carson's integral, but a closed form approximation is available [21].

The mutual impedance between conductors i and j is:

$$Z_{ij} = \frac{j\omega \cdot \mu_0}{2\pi} \cdot \left[\ln\left(\frac{D_{ij}}{d_{ij}}\right) + \frac{1}{2} \cdot \ln\left(1 + \frac{4 \cdot De(Y_i + Y_j + De)}{D_{ij}^{\,2}}\right)\right] \qquad (9\text{-}37)$$

Where,

$$De = \sqrt{\frac{eg}{j\omega \cdot \mu_0}} \text{ depth of penetration}$$

$$\mu_0 = 4\pi \times 10^{-7}$$

$$eg = \text{Ground resistivity } [\Omega m]$$

The self-impedance is:

$$Z_{ii} = \frac{j\omega \cdot \mu_0}{2\pi} \cdot \left[\ln\left(\frac{D_{ij}}{d_{ij}}\right) + \frac{\frac{0.3565 \cdot e_c}{\pi \cdot r^2} + \frac{e_c \cdot M \cdot \coth^{-1}(0.777 \cdot r \cdot M)}{2\pi \cdot r}}{N_c} \right] \quad (9\text{-}38)$$

Where,

$$M = \sqrt{\frac{j\omega \cdot \mu_0}{e_c}}$$

$$e_c = \frac{R_{dc} \cdot L}{A} \text{ conductor resistivity } [\Omega m]$$

$$A = \text{Area of a sub-conductor } [m^2]$$

The expressions for Y and Z are in the phase domain and therefore have diagonal and off-diagonal components. As described earlier, an N-phase line is modeled using N independent modes. The transformation from the phase domain to the mode domain requires an eigenvalue analysis.

Eigenvalue Analysis

The general procedure requires finding the eigenvalues and eigenvectors of the YZ product. A valid eigenvector of the YZ product will also diagonalize Y and Z separately. The eigenvectors are N x N matrices (generally complex), and there are N eigenvalues (also complex). The eigenvectors are the transformation matrices for voltage and current. The square root of an eigenvalue is the propagation constant γ.

This eigenvalue technique was obtained from M. Wedepohl and is called root squaring [21]. To accelerate the root squaring process

and to improve on the numerical accuracy of the results, it is best to find the eigenvalues of:

$$B = \frac{1}{KYZ - U} \tag{9-39}$$

The K factor merely removes any scalars from the Y and Z matrices, and the subtraction of the U is performed (at the formula level) to avoid round-off errors. The unity factor comes from the potential coefficient term which appears in both Y^{-1} and Z. Finally, the inverse of YZ is used instead of YZ as the root squaring process will remove the zero sequence mode last this way. Thus, the highest loss mode will have the greatest round-off error.

The eigenvalues of Equation 9-38 must be inverted, 1 added to them, and then scaled by the factor K before they can be used.

The root squaring eigenvalue algorithm has proven to be extremely reliable and robust. The eigenvectors which result, are simply sorted to achieve continuous eigenvectors versus frequency, which can then be curve fitted for use in the time domain algorithm.

The difficult aspect of eigenvalue analysis, as applied to a frequency dependent line model, is that the eigenvectors must be consistent from one frequency to another so that they may be curve fitted. Another eigenvalue method has been developed which uses a Newton-Raphson algorithm to find eigenvalues/eigenvectors for frequency N + 1 based on matrices from frequency N. This method has been proven to be very robust and efficient and will work for high order matrices.

Other Line Constants Calculations

Once the eigenvalue analysis is complete, we have transformations (the eigenvectors), which decouple the N phases into N independent modes:

$$Z_{mode} = T_v^{-1} \cdot Z \cdot T_I \tag{9-40}$$

$$Y_{mode} = T_I^{-1} \cdot Z \cdot T_v \tag{9-41}$$

Y_{mode} and Z_{mode} are now diagonal matrices.

The Surge Impedance Z_π becomes:

$$Z_{ni} = \sqrt{\frac{Z_{mode}(i,i)}{Y_{mode}(i,i)}} \qquad (9\text{-}42)$$

Finally, the attenuation constant is:

$$A = e^{-L\gamma} \qquad (9\text{-}43)$$

Where,

γ = The propagation constant

Eigenvalue analysis is undertaken at each frequency (as shown earlier in the flowchart), as is the calculation of the surge impedance and the attenuation constant. The results are continuous curves for these quantities versus frequency. The Frequency Dependent Line Models then require these curves to be approximated with Nth order polynomial functions of s.

Curve Fitting

The Frequency-Dependent Line Model requires Z_π, A and T_i to be approximated with Nth order transfer functions, which can then be transformed to the sum of exponential form required for the recursive convolution.

The curve fitting routine will only generate functions with poles and zeroes on the left hand side of the s-plane, thus ensuring the resultant function is minimum phase shift. This means that if the magnitude of a function is known, then its phase is also determined. Thus the magnitudes of Z_π, and A are curve-fitted.

The curve fitting algorithm used automatically places poles and zeroes in the s-plane to obtain a good match to the input function. If the resulting error is too large, then the order of the approximating function is adjusted and a better match is attempted. Both the desired error tolerance (as a percentage of the curve maximum) and the maximum order of the function can be adjusted.

The output is an Nth order approximation to the input function. The error of the approximation and the order of the approximation are available from the Transmission Line and Cable Constant

routines. The magnitude and phase of the original and curve fitted functions can also be written to files if desired.

The travel time for the line is calculated by comparing the phase of the approximate function to the phase of the attenuation constant minus the frequency times the travel time. A simple optimization procedure is used. The optimization ignores all frequencies where the magnitude of the Attenuation constant is below 1×10^{-10} (since any traveling waves at these frequencies would essentially be completely attenuated by the time it reaches the other end).

The user can enter the minimum and maximum frequencies for the line constants equations. The Transmission Line and Cable Constant routines will calculate 100 frequencies spaced evenly on a log scale between these two frequencies.

The choice of the lower frequency will affect the effective shunt conductance of the line. This is because the surge impedance approximated curve levels off at very low frequencies. The lower the minimum frequency, the higher the DC surge impedance becomes, which means a lower effective conductance.

The choice of the higher frequency is not critical. The EMTDC main program will automatically reduce the order of the approximated function by cutting poles and zeroes greater than ten times the Nyquist frequency (which is based on the time step). This prevents needless calculations without affecting results.

Chapter 10:

V2 Conversion Issues

CONVERTING V2 FORTRAN FILES

User-written EMTDC source code must be converted before being used in later versions of PSCAD. This process has been automated through the use of a utility known as the Fortran Filter ('ffilter.exe'). This file is located in a sub-directory called '.../bin/ffilter', which can be found under the PSCAD installation directory.

> **NOTE**: The PSCAD installation script modifies the PATH setting so that access to this directory is possible from other directories.

The Fortran Filter

Essentially, this program is used to convert code to a format, which will be compatible with either Fortran 90 or with the older Fortran 77 standard formats. The Fortran Filter performs the following functions:

- Replaces comment characters (i.e. 'c' or 'C') in the first column, with an exclamation mark '!'
- Ensures code continuation lines are written with a '&' in column 73, as well as column 6 of the following line
- Replaces tabs with spaces
- Replaces the older style EMTDC COMMON statements (such as COMMON /S1/ TIME, DELT, ICH, PRINT, FINTIM) and associated variable declarations with the new style include statements (such as INCLUDE s1.h).

> **NOTE**: The new INCLUDE statements are necessary to allow user-written code to function with either Fortran 77 or the new dynamic dimensioning Fortran 90.

Command Line and Options

The Fortran Filter command line can be used with one of the following formats:

ffilter -[options] <filename>

or

ffilter -d[options] <directoryname>

The Fortran Filter comes with several options, which are described as follows in Table 10-1:

Option	**Description**
t[number]	Converts all tabs to a sequence of characters, where [number] is size of tab. The Default is 8
u	Converts all keywords to uppercase
l	Converts all keywords to lowercase
c	Converts comments
n	Converts continuation characters
i	Replaces inclusion of ‘emt.d’ and ‘emt.e’ with new style Include Files
b	Converts common blocks
v	Converts common blocks even in the case of incomplete declaration. This needs to be specified together with -a or -b options. (eg. -av or -bv)
a	Runs with options: t8ucnhib
d	Run on multiple files in directory
r	Remove lines commented by filter
e	Empty working directory from intermediate files generated by the Fortran Filter
h	Print help

Table 10-1 - Fortran Filter Options

NOTE: This list of options can also be obtained by typing 'ffilter' at a DOS prompt.

Using the Fortran Filter

In order to use the Fortran Filter to convert your V2 Fortran files, you must first open a command prompt and navigate to the directory where your V2 style Fortran file (or the directory containing a set of files) is located. Then, simply type in the command line with the required options.

The Fortran Filter creates a directory called 'temp' inside the directory in which it is run. The original file will be copied into this directory and saved with the postfix 'oryg.' The whole filtering process is conducted in this directory.

EXAMPLE 10-1:

Consider a V2 style Fortran source file located in a directory 'c:\temp\test', called 'file1.f'. To convert the file, open a DOS prompt in 'c:\temp\test' and type:

ffilter -ar file1.f

The new filtered source code file will be placed in the 'c:\temp\test' directory described above (the original files are untouched). Check the '*.log' file for error messages.

EXAMPLE 10-2:

Consider a directory, containing many V2 style Fortran source files, called 'c:\temp\dir1.' To convert the entire directory, open a DOS prompt in 'c:\temp' and type:

ffilter -dar dir1

The new filtered source code file will be placed in the 'temp' described above (the original files are untouched). Check the '*.log' file for error messages.

The Fortran Filter should handle all code, with the exception of code that does not comply with the standard common block or variable names. An example of this is a non-standard STOR name, by specifying the COMMON block with:

```
COMMON /S2/ MYSTOR(ND10), MYNEXC
```

This is a particularly nasty problem, because all code in this subroutine must be manually edited to replace MYSTOR with the standard STOR name.

Another example is if you fail to declare variables used in COMMON blocks:

```
REAL STOR
...
COMMON /S2/ STOR(ND10), NEXC
```

This will fail because NEXC was not declared as an INTEGER.

MANUAL TASKS REQUIRED

There have been many enhancements to EMTDC associated with electrical signals and their interface to user-written components. These changes can cause some compatibility problems with V2 electrical components. Therefore, some manual conversion tasks are required if certain subroutines/functions, or Internal Variables are used in the user-written code.

Obsolete Subroutines/Functions
One important enhancement, for example, is the move from referencing branches by node numbers, to referencing by an actual branch number. The node referencing method was found to create problems with parallel branches, which possess identical connection nodes.

For instance, the branch current output in switching elements, such as faults, breakers, thyristors, diodes, etc. all had an inherent time step delay. This was due to the fact that all parallel switch branches were combined into a single branch for solution in the main program (V2 had a limit of 3 parallel switch branches). This also meant that the branch current could only be output as an argument in the

DSDYN subroutine, and could not be placed in DSOUT. By referencing each branch by a unique branch number, the branch current can now be placed directly in DSOUT.

The move to the new branch number referencing method rendered some function calls obsolete. These are listed in Table 10-2.

Obsolete	Replaced With	**Description**
THYR25	EMTDC_PESWITCH2	Power electronic switch model (Diode, thyristor, GTO, etc.)
SWINT5	EMTDC_BREAKER	Breaker or switch model
G6P200	G6P200	6-Pulse bridge model (arguments only)
VZNINT	VZNO50	Surge arrestor model
ESYS1	1PVSRC or ESYS65_B	Single phase source model
CLSA33	CLSA35	*Obsolete*

Table 10-2 - Obsolete Subroutines/Functions

Obsolete Internal Variables
In addition to subroutines and functions, some EMTDC Internal Global Variables have also been affected. Table 10-3 lists some Internal Variables, which are now considered obsolete:

Obsolete	Replaced With	**Description**
CDC(*,*,*)	CBR(*,*)	Gives the value of branch current
CCDC(*,*,*)	CCBR(*,*)	Gives the value of history current
EDC(*,*,*)	EBR(*,*)	Sets the value of branch source voltage
GDC(*,*,*)	GEQ(*,*)	Sets the value of the branch equivalent conductance (node to node)
GDCS(*,*)	GEQ(*,*)	Sets the value of the branch equivalent conductance (node to ground)

Table 10-3 - Obsolete Internal Global Variables

Table 10-4 lists Internal Variables that are still functional, but their use is in the process of being phased out. Conversion of these variables is not mandatory, but is recommended:

Phasing Out	Replacing With	**Description**
STOR(*)	STORF(*) STORI(*) STORL(*) STORC(*)	Storage array variables
NEXC	NSTORF NSTORI NSTORL NSTORC	Storage array pointers

Table 10-4 - Obsolete But Still Functional Internal Global Variables

NOTE: The old STOR array pointer NEXC was set to 0 each time step. The new pointers are set to 1. The new equivalent conductance GEQ, no longer requires that one branch node be grounded.

EXAMPLE 10-3:

As mentioned above, the variable CDC has been replaced by the variable CBR. Manually replacing these variables should be performed as follows.

Assume that a branch is defined and named ‘BRN’ in the Branch segment of Component Definition:

BRN = $NA $NB 1.0

Then a statement with CDC:

I = CDC($NB, $NA, $SS)

Should be replaced with:

I = CBR($BRN, $SS)

EXAMPLE 10-4:

The variable EDC has been replaced by the variable EBR. Manually replacing these variables should be performed as follows.

Assume that a branch is defined and named ‘BRN’ in the Branch segment of Component Definition:

BRN = $NA $NB SOURCE 1.0

Then statements with EDC:

EDC($NB, $NA, $SS) = EDC($NB, $NA, $SS) + <value>

EDC($NA, $NB, $SS) = EDC($NA, $NB, $SS) - <value>

Should be replaced with:

EBR($BRN, $SS) = <value>

EXAMPLE 10-5:

The variable GDC and GDCS have been replaced by the variable GEQ. Manually replacing these variables should be performed as follows.

Assume that a branch is defined and named ‘BRN’ in the Branch segment of Component Definition:

BRN = $NA $NB BREAKER 1.0

Then statements with GDC:

GDC($NB, $NA, $SS) = GDC($NB, $NA, $SS) + <value>

GDC($NA, $NB, $SS) = GDC($NA, $NB, $SS) + <value>

Should be replaced with:

GEQ($BRN, $SS) = <value>

Also, statements with GDCS:

GDCS($NA, $SS) = GDCS($NA, $SS) + <value>

Should be replaced with:

GEQ($BRN, $SS) = <value>

Storage Issues

As discussed previously, EMTDC contains new storage variables and pointers. A new syntax in the DSDYN and DSOUT segments of the new Component Definitions is available to inform PSCAD of the total number of storage elements required for each component. PSCAD uses this information to tell EMTDC how many elements in the array it should allocate memory for (Fortan 90 version only).

If the user wishes to stick with the STOR and NEXC storage array for now, the proper syntax to use in the Component Definition is:

#STORAGE STOR:200

EMTDC will allocate an additional 10,000 STOR locations only in cases where users, who have converted PSCAD V2 cases, have not yet modified their Component Definitions with this new information. If running PSCAD cases which have just been converted from V2, and custom components are being used, which collectively exceed 10,000 STOR locations, your simulation case will not function until the #STORAGE syntax is added to the Component Definitions.

> **NOTE**: See the STORx Arrays for more details on this storage syntax.

References

1. H. W. Dommel, "*Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks*," IEEE Transactions on Power Apparatus and Systems, PAS-88, #4, pp. 388-399, April 1969.
2. Reference to ANSI Standard X3.9 – 1978 (Fortran 77).
3. H. W. Dommel, "*Transformer Models in the Simulation of Electromagnetic Transients*," Proc. 5th Power Systems Computing Conference, Cambridge, England, September 1-5, 1975, Paper 3.1/4.
4. V. Brandwajn, H. W. Dommel, I. I. Dommel, "*Matrix Representation of Three Phase N-Winding Transformers for Steady State Transient Studies*," IEEE Transactions on Power Apparatus and Systems, PAS-101, #6, pp. 1369-1378, June 1982.
5. A. M. Gole, "*Power Systems Transient Simulation*," Course Notes, University of Manitoba, 1998.
6. P. Kuffel, K. Kent, G. Irwin, "*The Implementation and Effectiveness of Linear Interpolation Within Digital Simulation*," Proceedings, International Conference on Power Systems Transients (IPST '95), pp. 499-504, Lisbon, September 3-7, 1995.
7. A. M. Gole, A. Keri, C. Nwankpa, E. W. Gunther, H. W. Dommel, I. Hassan, J. R. Marti, J. A. Martinez, K. Fehrle, L. Tang, M. F. McGranaghan, O. B. Nayak, P. F. Ribeiro, R. Lasseter, "*Guidelines for Modeling Power Electronics in Electric Power Engineering Applications*," IEEE Transactions on Power Delivery, Vol. 12, No.1, pp. 505-514, January 1997.
8. A. M. Gole, I. T. Fernando, G. D. Irwin, O. B. Nayak, "*Modeling of Power Electronic Apparatus: Additional Interpolation Issues*," International Conference on Power Systems Transients (IPST '97), pp. 23-28, Seattle, June 22-26, 1997.

9. D. A. Woodford, "*Validation of Digital Simulation of DC Links*," IEEE Transactions on Power Apparatus and Systems, PAS-104, #9, pp. 2588-2596, September 1985.
10. W. Enright, O. B. Nayak, G. D. Irwin, J. Arrillaga, "*An Electromagnetic Transients Model of Multi-limb Transformers Using Normalized Core Concept*," IPST '97 Proceedings, Seattle, pp. 93-98, 1997.
11. W. Enright, N. Watson, O. B. Nayak, "*Three-Phase Five-Limb Unified Magnetic Equivalent Circuit Transformer Models for PSCAD V3*," IPST '99 Proceedings, Budapest, pp. 462-467, 1999.
12. B. Adkins, R. G. Harley, "*The General Theory of Alternating Current Machines*," Chapman & Hall, London, 1975.
13. C. V. Jones, "*Unified Theory of Electrical Machines*," Butterworths, London, 1967.
14. I. M. Canay, "*Causes of Discrepancies on Calculation of Rotor Quantities and Exact Equivalent Diagrams of the Synchronous Machine*," IEEE Transactions, Vol. PAS-88, No. F, p. 1114-1120, July 1969.
15. M. R. Harris, P. J. Lawrenson, J. M. Stephenson, "*Per Unit Systems with Special Reference to Electrical Machines*," IEE Monograph, 1970.
16. "*IEEE Recommended Practice for Excitation System Models for Power System Stability Studies*," IEEE Std 421.5-1992.
17. "*Computer Models for Representation of Digital-Based Excitation Systems*," IEEE Transactions 1996.
18. "*Dynamic Models for Fossil Fueled Steam Units on Power System Studies*," by the Working Group on Prime Mover and Energy Supply Models for System Dynamic Performance Studies, Transactions on Power Systems, Vol. 6, No. 2, May 1991.
19. "*Hydraulic Turbine and Turbine Control Models for System Dynamic Studies*," by the Working Group on Prime Mover and Energy Supply Models for System Dynamic Performance Studies, Transactions on Power Systems, Vol. 7, No. 1, February 1992.
20. P. Kundar, "*Power System Stability and Control*," McGraw Hill, 1994.

21. L. M. Wedepohl, "*Theory of Natural Modes in Multi-Conductor Transmission Lines*," Course Notes, May 1989.

22. L. M. Wedepohl, "*Transient Analysis of Underground Transmission Systems*," Proc. IEE, Vol. 120, No. 2, Feb. 1973.

23. J. Marti, "*Accurate Modeling of Frequency Dependent Transmission Lines in Electromagnetic Transients Simulation*," IEEE Transactions on Power Apparatus and Systems, PAS-101, #1, pp. 147-155, Jan. 1982.

24. L. Marti, "*Simulation of Electromagnetic Transients in Underground Cables with Frequency Dependent Modal Transformation Matrices*," Ph.D. Thesis, University of British Columbia, 1986.

25. A. Semlyen, A. Dabuleanu, "*Fast and Accurate Switching Transient Calculations on Transmission Lines and Ground Return Using Recursive Convolutions*," IEEE Transactions on Power Apparatus and Systems, PAS-94, pp. 561-571, March/April 1975.

26. A. Morched, B. Gustavsen, M. Tartibi, "*A Universal Line Model for Accurate Calculation of Electromagnetic Transients on Overhead Lines and Cables*," Paper PE-112-PWRD-0-11-1997.

27. B. Gustavsen, G. Irwin, R. Mangelrod, D. Brandt, K. Kent, "*Transmission Line Models for the Simulation of Interaction Phenomena Between Parallel AC and DC Overhead Lines*," IPST '99 Proceedings, pp. 61-67, 1999.

28. B. Gustavsen, A. Semlyen, "*Rational Approximation of Frequency Domain Responses by Vector Fitting*," IEEE Paper PE-194-PWRD-0-11-1997, presented at IEEE/PES Winter Meeting, Tampa, 1998.

29. G. V. Reklaitis, A. Ravindran, K. M. Ragsdell, "*Engineering Optimization - Methods and Applications*," New York: Wiley-Interscience, 1983.

30. R. L. Haupt, S. E. Haupt, "*Practical Genetic Algorithms*," New York: Wiley-Interscience, 1998.

31. K. Y. Lee, M. A. El-Sharkawi (editors), "*Tutorial on Modern Heuristic Optimization Techniques with Applications to Power Systems*," IEEE Power Engineering Society 02TP160, 2002.

32. J. A. Nelder, R. Mead, "*A Simplex Method for Function Optimization*," The Computer Journal, Vol. 7, No. 4, pp. 308-313, 1965.

33. A. M. Gole, S. Filizadeh, R. W. Menzies, P. L. Wilson, "*Optimization-Enabled Electromagnetic Transient Simulation*," IEEE Trans. on Power Delivery, 2004 (Pending)

34. Krüger, K. H., Lasseter, R. H., "*HVDC Simulation Using NETOMAC*," IEEE Montech '86, Conference on HVDC Power Transmission, Montreal, Canada, Sept. 29 – Oct. 1, 1986.

35. Gole, A. M., Woodford, S. A., Nordstrom, J. E., Irwin, G. D. "*A Fully Interpolated Controls Library for Electromagnetic Transients Simulation of Power Systems*," Proceedings, IPST'01 – International Conference on Power System Transients, Rio de Janeiro, Brazil, June 24-28, 2001.

Index