EMTDC™

Transient Analysis for PSCAD Power System Simulation

USER'S GUIDE

A comprehensive resource for EMTDC

PSCAD

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
- February, 2010 - Version 4.2.1, fifth printing
- April, 2016 - Version 4.6.0
- May 9, 2018 – Version 4.6.0, updated to new MHI Branding Guidelines

Table of Contents

Chapter 1: Introduction to EMTDC

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 (Technical 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 Hydro International Ltd.

Chapter 2: Program Structure

Background

When digital 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 program 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 or her subroutines, if and when 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 may 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.

The EMTDC Solution Process

The EMTDC solution engine consists of two main parts: The *System Dynamics*, which includes the master dynamics subroutine (*DSDYN*), the output definition subroutine (*DSOUT*), and the initialization subroutine (*BEGIN*); and secondly, the Electric Network Solution.
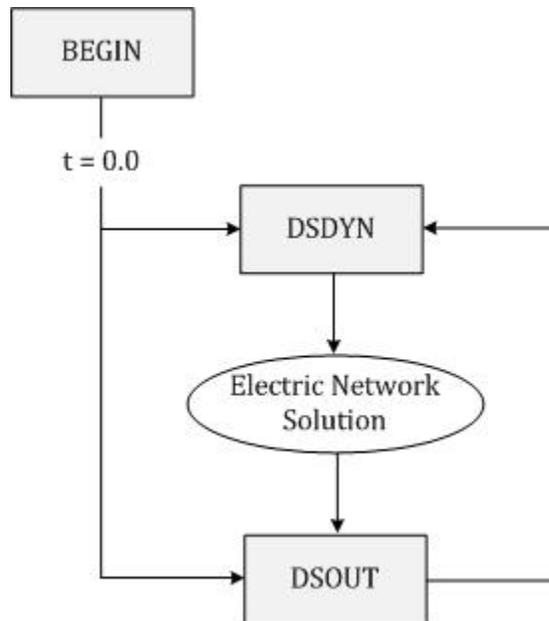
Figure 2-1 - Core EMTDC Solution Process

An EMTDC simulation begins at a specified start time and finishes at a specified end time. Between these two times, the program performs the same sequential process iteratively, each iteration incrementing the time by a specified interval. This interval is referred to as a time step, which remains constant (i.e. fixed) during the course of the simulation.

As illustrated by Figure 2-1, the core solution process begins and ends with the system dynamics. First, variables are initialized and stored in BEGIN. Dynamic functions are performed in DSDYN, which includes the preparation of electric network devices such as sources, etc., as well as the manipulation of control signals. The electric network is solved and the resulting measured quantities are considered, along with any post-solution processing requirements, in DSOUT, before the time step is incremented.

Of course, the complete EMTDC solution sequence is more involved than that shown in Figure 2-1. There are several other features present that ensure solution accuracy and speed; all of which are discussed in the following sections. A more complete program process is illustrated in Figure 2-2.
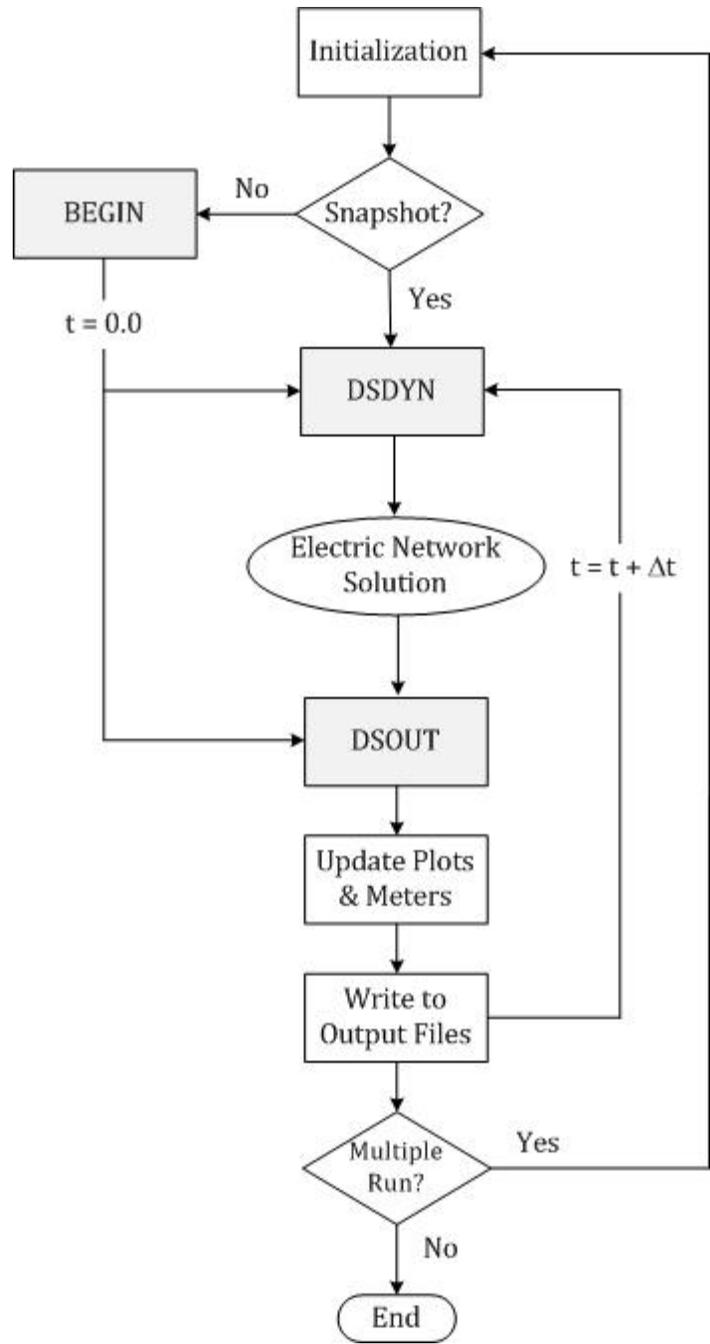
Figure 2-2 - Complete EMTDC Solution Process

System Dynamics

The system dynamics is an essential part of the EMTDC solution process. It serves to encompass the electric network, providing the ability to interface with it on either side: Data may be controlled from one side, while extracted from the other.

Essentially a Fortran program in its own right, the system dynamics is built according to customizable specifications. These specifications are normally provided through PSCAD, in the form of components and modules. Components and modules are graphical representations of how the final program is to be structured, where each module, including the main page, is represented by a unique Fortran file. This file consists of declarations for a DSDYN subroutine, a DSOUT subroutine, and two BEGIN subroutines. Components always exist within modules, so as such they represent inline code that combines to define the contents of each subroutine.

Of course, some components are electrical in nature and hence provide information for the construction of the electric network as well. This information is collected and inserted into a Data file, where each module in a project possesses its own unique Data file. Data files are discussed in more detail later on in the section entitled *Electric Network Solution*.

DSDYN and DSOUT Subroutines

The DSDYN and DSOUT subroutines provide accessibility for control and monitoring of system variables on either side of the electric network. This offers a great advantage in programming flexibility, as it enables both the control of input variables and the monitoring of output variables, all within the same time step. This is an important concept, especially in the design of control systems involving feedback: Judicial selection of code placement can help avoid time delays that are uncharacteristic to the real system being modeled.

When it pertains to source code insertion into the system dynamics, there is no difference between DSDYN and DSOUT, besides their sequence in the solution process. However, there are specific uses for each, and certain code may be more optimally utilized when it is placed in one, rather than the other. For example, DSOUT is primarily used to define output variables directly following the electric network solution. Of course, DSDYN could be used for this purpose as well, but the same output variables would be delayed, due to the time step increment between DSOUT and DSDYN (see Figure 2-1). Likewise, variables controlling electric devices (i.e. network input variables) are best defined in DSDYN, as then updates will occur in the same time step before the network solution.

Component source is inserted in either DSDYN or DSOUT, depending on what the component is connected to. Decisions are made by an involved internal algorithm in PSCAD, which ensures that both feedback and feed-forward signals are properly sequenced, minimizing time delays. However, component source may also be forced into either DSDYN or DSOUT at the user's discretion. To force source script into a specific subroutine, use either the *DSDYN* or *DSOUT* script segments. To utilize the internal algorithm, use the *Fortran* script segment. For more details on these segments, see the section called *Segment Types* in the *Component Design* chapter of the PSCAD Manual.

Example on Time Delays and Code Placement

As mentioned above, it is important to establish the proper use of DSDYN and DSOUT to avoid unnecessary time step delays. 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. Since there is a time step increment between DSOUT and DSDYN, the control system may be basing outputs on quantities defined in the previous time step.

EXAMPLE 2-1:
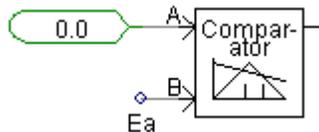
Figure 2-3 - Simple Comparator Circuit created in PSCAD

Consider the simple circuit in Figure 2-3. A comparator is being used to compare a real constant variable equal to *0.0*, with a data 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 the DSDYN subroutine, whereas the measured voltage signal *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 (i.e. the code that determines its output state) are using a measured voltage *Ea*, defined in DSOUT. Since the comparator output state is based on the comparator inputs, the output will appear to have been delayed by one time step.

This delay can be removed simply by forcing the comparator dynamics code from DSDYN to DSOUT. This will force the comparator to consider the measured voltage within the same time step, when determining its state.

The BEGIN Subroutine

The BEGIN subroutine is used for time zero operations, such as variable initialization and storage. Although the use of BEGIN is not mandatory, it is required if a component is to support its use within modules with multiple instances. Components with source specified for the BEGIN subroutine are said to be *Runtime Configurable*.

The primary purpose of BEGIN source is to ensure that variables required for a particular component instance, are initialized and stored for use later during runtime. All components with BEGIN subroutine source will have their variables stored in sequence (according to where the component source appears in the system dynamics), to be later accessed in the same sequence during the main runtime loop. Component parameter values are thereby unique for each instance. This enables components that are defined as part of a module definition, to retain different parameter values in different instances of that module.

For example, Listing 2-1 illustrates snippets of code taken from a module Fortran file. The module contains an instance of the PI Controller component from the master library. Notice that in the BEGIN section, data is stored (in this case the initial output value), and then extracted again in the same sequence during runtime in the DSDYN subroutine.

! 90:[pi_ctlr] PI Controller
RTCF(NRTCF) = 1.57
NRTCF = NRTCF + 1

! 90:[pi_ctlr] PI Controller
RVD1_1 = RTCF(NRTCF)
NRTCF = NRTCF + 1

BEGIN Subroutine   DSDYN Subroutine

Listing 2-1 - Variables Stored Sequentially in BEGIN and then Extracted During Runtime

See *EMTDC Storage Arrays* in the chapter entitled *Custom Model Design* for more details.

System Dynamics Code Assembly

As mentioned above, a module definition is equivalent to a subroutine declaration in the system dynamics structure; where an instance of said module is analogous to a subroutine call. Considering that the system dynamics is partitioned into three separate sections (i.e. DSDYN, DSOUT and BEGIN), where BEGIN is further subdivided into two parts, module representation is not accomplished by a single subroutine, but a group of four – one to represent each dynamics section and subsection. All four subroutines are organized into a single Fortran file, where this file (along with a Data file) forms the definition of the module, in the same way as a definition is the foundation of a regular component.

Components existing within a module are combined to define the subroutine body, by insertion of their script inline, according to their sequence and placement on the module canvas: The actual source code is extracted from components, derived from definition script that exists in either the DSDYN, DSOUT or Fortran segments. Figure 2-4 and Listing 2-2 combine to illustrate this concept.
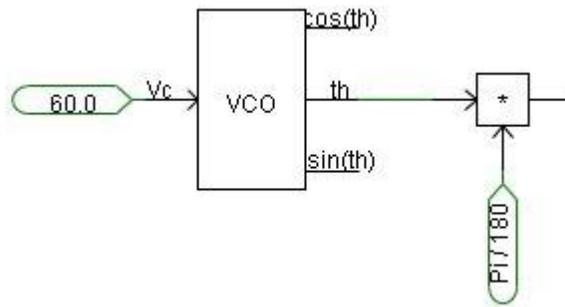
Figure 2-4 – Example Module Circuit Canvas

```
!

    SUBROUTINE DSDyn()

!

! . . .

!

! 10:[const] Real Constant

    RT_2 = 60.0

! 20:[vco] Voltage-Controlled Oscillator

!  Voltage Controlled Oscillator

    RT_1 = STORF(NSTORF)

    IF(RT_1.GE. TWO_PI) RT_1 = RT_1 - TWO_PI

    IF(RT_1.LE.-TWO_PI) RT_1 = RT_1 + TWO_PI

    RT_4 = SIN(RT_1)

    RT_3 = COS(RT_1)

    STORF(NSTORF) = RT_1 + DELT*RT_2*TWO_PI

    RT_1 = RT_1*BY180_PI

    NSTORF = NSTORF + 1

! 30:[emtconst] Commonly Used Constants (pi...)

    RT_6 = PI_BY180

! 40:[mult] Multiplier
```

```
    RT_5 = RT_1 * RT_6

!

! . . .

!

    RETURN

    END
```

Listing 2-2 – Snippet of Inline Code from Module Fortran File (DSDYN Subroutine)

Figure 2-5 and Listing 2-3 illustrate how system dynamics files would be prepared upon building an example project. The project shown consists of the default *Main* module, which harbours two instances of a module called *Module_1* along with a single instance of a component called *Comp_1*. *Module_1* itself contains a single component instance called *Comp_2*.
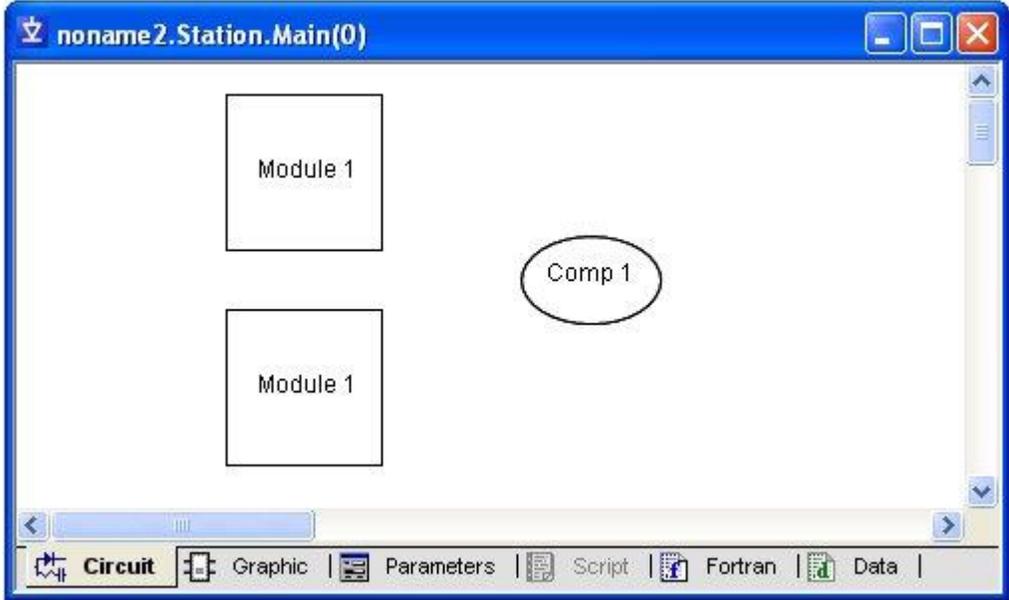
Figure 2-5 – Example Main Module Circuit Canvas

```
!

    SUBROUTINE DSDyn()

!

! . . .

!

! 10:[Module_1]
```

```
!

    SUBROUTINE Module_1Dyn()

!

! . . .

!

! 10:[Comp_2]
```

```
    CALL Module_1Dyn()

! 20:[Module_1]

    CALL Module_1Dyn()

! 30:[Comp_1]

    ...

!

! . . .

!

    RETURN

    END

!

    SUBROUTINE DSOut()

!

! . . .

!

! 10:[Module_1]

    CALL Module_1Out()

! 20:[Module_1]

    CALL Module_1Out()

! 30:[Comp_1]

    ...

!

! . . .

!

    RETURN

    END

!

    SUBROUTINE DSDyn_Begin()

!

! . . .

!

! 10:[Module_1]
```

```
    ...

!

! . . .

!

    RETURN

    END

!

    SUBROUTINE Module_1Out()

!

! . . .

!

! 10:[Comp_2]

    ...

!

! . . .

!

    RETURN

    END

!

    SUBROUTINE Module_1Dyn_Begin()

!

! . . .

!

! 10:[Comp_2]

    ...

!

! . . .

!

    RETURN

    END

!

    SUBROUTINE Module_1Out_Begin()
```

```
    CALL Module_1Dyn_Begin()                                !

! 20:[Module_1]                                             ! . . .

    CALL Module_1Dyn_Begin()                                !

! 30:[Comp_1]                                               ! 10:[Comp_2]

    ...                                                          ...

!                                                           !

! . . .                                                     ! . . .

!                                                           !

    RETURN                                                      RETURN

    END                                                        END

!

    SUBROUTINE DSOut_Begin()

!

! . . .

!

! 10:[Module_1]

    CALL Module_1Out_Begin()

! 20:[Module_1]

    CALL Module_1Out_Begin()

! 30:[Comp_1]

    ...

!

! . . .

!

    RETURN

    END
```

Main.f   Module_1.f

Listing 2-3 – Assembly of System Dynamics Subroutines

Following a build operation, this project will possess two Fortran files: The first represents the definition of the *Main* module (*Main.f*), and the second the definition of *Module_1* (along with their corresponding Data files). Each file contains four, separate subroutines corresponding to the three system dynamics sections. The subroutine names are pre-pended by the name of the module.

Note that although there are two instances of *Module_1*, they are both based on the same definition (i.e. Fortran file). Modules with multiple instances can still be unique however, through the use of its argument list. Subroutine arguments are used to pass both port connections and module input parameter values.

EXAMPLE 2-2:

Consider a user-defined module, which represents a 3-phase, 2-winding Y-⃞ transformer. The module canvas consists simply of a 3 Phase 2 Winding Transformer component from the master library, set as Y-⃞, with a *1 ⃞* resistance connected to the common of the primary winding. It is required that this components *3 Phase Transformer MVA* input parameter be changeable on a per-module instance basis.
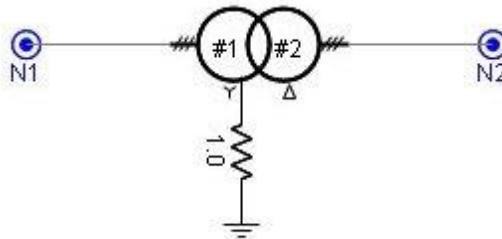
Figure 2-6 – Simple 3-Phase, 2-Winding Transformer Module Canvas

If this module is to be multiply instanced, then all components forming the module must be *Runtime Configurable*. Fortunately, the transformer component used in this module is such. To be runtime configurable means that components will make use of the BEGIN section of the system dynamics if required. However, only those component input parameters defined as *Constant* or *Variable* may be adjusted on a per-module instance basis (*Literal* parameters cannot).

The *3 Phase Transformer MVA* input parameter in the 3 Phase 2 Winding Transformer component is fortunately defined as a *Constant*. Therefore, in order to allow adjustment of this parameter on a per-module instance basis, an input parameter will need to be defined for the module itself, and the parameter value imported onto the canvas. A Constant type module parameter with Symbol name *tx_mva* is created and given a value of *100.0 MVA*. Its corresponding *Import* tag is added to the canvas. The value of the *3 Phase Transformer MVA* input parameter in the 3 Phase 2 Winding Transformer component is also entered as *tx_mva*.
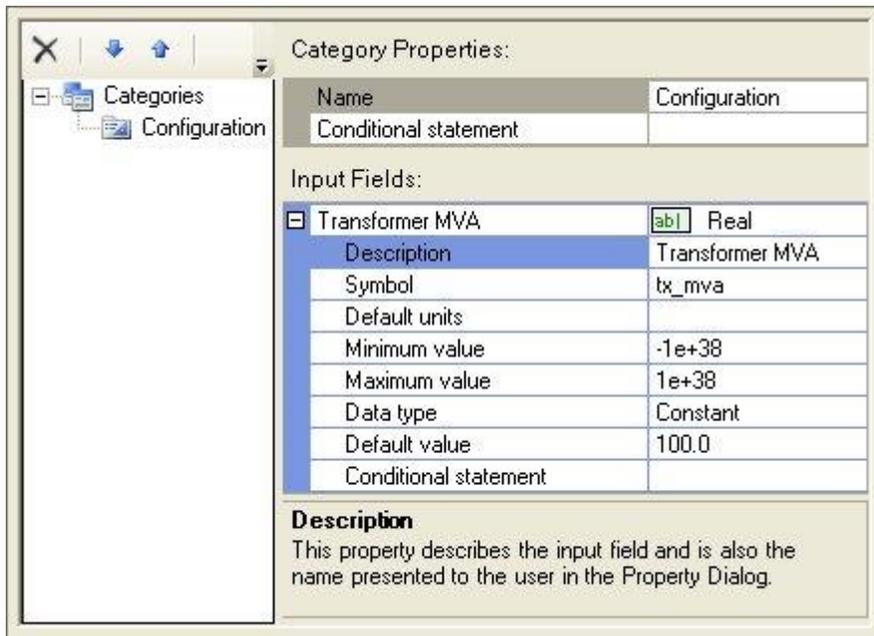
Module Definition Parameters Section
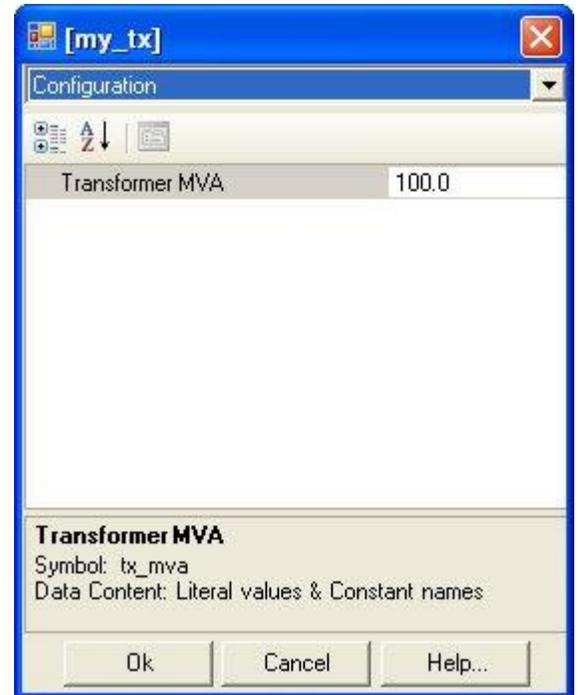
Module Parameters Dialog
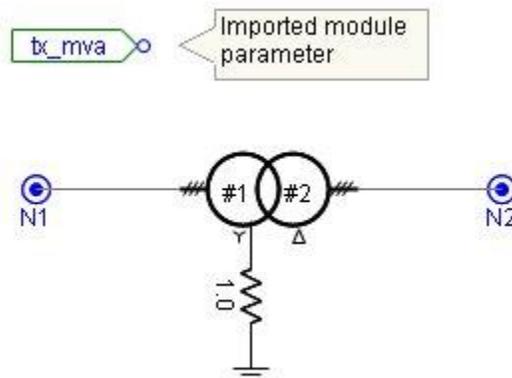
Module Definition Circuit Canvas

Figure 2-7 – Passing an Input Parameter to the Module Canvas

Whatever value is now entered into the module parameter *tx_mva*, will be imported into the module canvas, where it is defined as a data signal, and may be collected by any component with a compatible input parameter. Let's have a look at some bits of the Fortran files generated when this project is built. First of all, the *Main.f* file will contain four subroutines: *DSDyn*, *DSOut*, *DSDyn_Begin* and *DSOut_Begin*. Each of these subroutines will contain a call to the corresponding subroutine declared in the file *my_tx.f*. Notice that an argument containing the value of the module input parameter exists as part of the subroutine call statement. This allows for different values to be passed in to the subroutine, depending on the module instance.

```
! -1:[my_tx]

    CALL my_txDyn(100.0)
```

Listing 2-4 – *my_tx* Module Call in *Main.f* Subroutines

The *my_tx.f* file will also contain four subroutines, representing the *my_tx* module. These are: *my_txDyn*, *my_txOut*, *my_txDyn_Begin* and *my_txOut_Begin*. Making up the body of these subroutines will be the extracted code from the components defining the module (in this case mainly the 3 Phase 2 Winding Transformer).

Now let's consider what happens if a second instance of the *my_tx* module is created, and its input parameter given the value of *200.0 MVA*. Following rebuild, the project will still have only two Fortran files associated with it. This is because there are still only two module definitions. However, the second instance of the *my_tx* module will result in an additional call to the *my_tx* subroutines in the *Main.f* file.

```
! -1:[my_tx]

    CALL my_txDyn(100.0)

! -1:[my_tx]

    CALL my_txDyn(200.0)
```

Listing 2-5 – *my_tx* Module Call in *Main.f* Subroutines (Two Instances of Child Module)

Note this time how the module parameter values differ between calls. This is because of the change in value for the second instance of the *my_tx* module.

The *my_tx.f* file remains unchanged.

Runtime Configuration

In system dynamics terms, each module instance is represented by a call to its corresponding subroutine (one call within each dynamics section). The calls will always be located within the subroutine body of its parent module. In most cases, the *Main* module is the top-most parent. However other modules can act as parents if the project hierarchy is more than two levels deep.

In many cases, it is possible that certain component input parameters may be dependent on the instance of the module in which the component resides. That is to say that one or more parameters may differ depending on the calling point of the module subroutine in the system dynamics. This is where the BEGIN subroutine and its inherent storage arrays come into play.

EXAMPLE 2-3:

Consider a user-defined module, which contains the simple control circuit illustrated below.
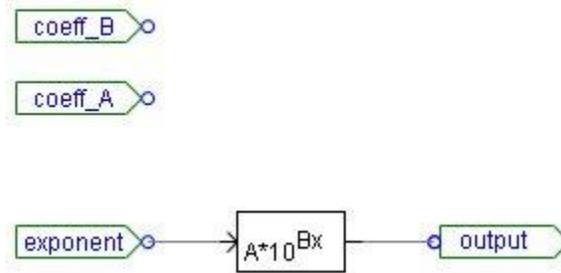
Figure 2-8 – User-Defined Module Circuit Canvas

The canvas contains a single master library component called Exponential Functions, which can be adjusted to represent exponential functions of base *10* or base *e* (in this case it is set to base 10). The component will also accept values for a base coefficient *A*, as well as an exponential coefficient *B*. These two component input parameters are both of type *Constant* and are described as *Coefficient of Base* and *Coefficient of Exponent* respectively.

There are two port connections defined for the module: One is an input connection called exponent, which represents the value of the exponent itself; and *output*, which is the result of base 10 taken to the exponential value. There are also two module input parameters called *Base Coefficient* and *Exponential Coefficient* with Symbol name *coeff_A* and *coeff_B*, which import the parameter values onto the canvas.
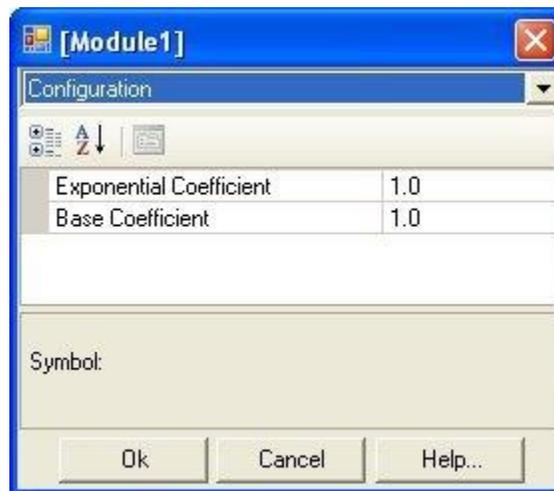
Figure 2-9 – User-Defined Module Input Parameters

The above described system will allow the module itself to be instantiated many times, with each instance able to possess unique values for both the exponent and the two coefficients. For example, let us create a second instance of this module and place them both in a circuit on the main canvas as shown below.

Figure 2-10 – Two Instances of a User-Defined Module on the Main Canvas

The *Fortran* segment in the Exponential Functions component definition contains a *#BEGIN* directive block, which specifies that the script within the block is to be inserted into the BEGIN section of the system dynamics. Since this directive is located in the Fortran segment, PSCAD will decide in which subsection of BEGIN to insert the code (i.e. in this case it will be the *Module1Dyn_Begin* subroutine). Let's look at the *Module1.f* file following a build.

```
!

    SUBROUTINE Module1Dyn_Begin(coeff_B_, coeff_A_)

!

! . . .

!

! Subroutine Parameters
    REAL   coeff_B_, coeff_A_
! Control Signals
    REAL   coeff_B, coeff_A

!

! . . .

!

    coeff_B  = coeff_B_
    coeff_A  = coeff_A_

!

! . . .

!

! -1:[exp] Exponential Functions
    RTCF(NRTCF)   = coeff_A
    RTCF(NRTCF+1) = coeff_B
    NRTCF = NRTCF + 2

!

    RETURN
    END

!
```

```fortran
      SUBROUTINE Module1Dyn(exponent_, output_, coeff_B_, coeff_A_)

!

! . . .

!

! Subroutine Parameters
      REAL   exponent_, coeff_B_, coeff_A_
      REAL   output_
! Control Signals
      REAL   exponent, output, coeff_B, coeff_A

!

! . . .

!

      exponent = exponent_
      coeff_B  = coeff_B_
      coeff_A  = coeff_A_

!

! . . .

!

! -1:[exp] Exponential Functions
!

      output = RTCF(NRTCF) * 10**(RTCF(NRTCF+1) * exponent)
      NRTCF = NRTCF + 2

!

! . . .

!

      RETURN

      END
```

Listing 2-6 – Segments of the *Module.f* File

Within the *Module1Dyn_Begin* subroutine (i.e. the BEGIN section of the system dynamics), the input parameter values for each instance of the module are stored in the *RTCF* storage array and the corresponding *NRTCF* pointer is incremented. During runtime, these same storage locations are accessed within the DSDYN subroutine *Module1Dyn*.

EXAMPLE 2-4:

Consider the circuit described in Example 2-2. The 3 Phase 2 Winding Transformer component is runtime configurable because it comes complete with BEGIN section source. The DSDYN segment in its definition contains a *#BEGIN* directive block, which specifies that the script within the block is to be inserted into the BEGIN section of the system dynamics. Since this directive is located in the DSDYN segment, the script will be inserted specifically into DSDYN subsection of BEGIN (i.e. the *my_txDyn_Begin* subroutine).

```
#BEGIN

 #LOCAL REAL RVD1_1

 #LOCAL REAL RVD1_2

 #LOCAL REAL RVD1_3

 #LOCAL REAL RVD1_4

 #LOCAL REAL RVD1_5

 #LOCAL REAL RVD1_6

 #LOCAL INTEGER IVD1_1

    RVD1_1 = ONE_3RD*$Tmva

    RVD1_2 = $V1~

#CASE YD1 {~*SQRT_1BY3} {~}

    RVD1_3 = $V2~

#CASE YD2 {~*SQRT_1BY3} {~}

    CALL E_TF2W_CFG($#[1],$Ideal,RVD1_1,$f,$Xl,$CuL,RVD1_2,RVD1_3,$Im1)

    CALL E_TF2W_CFG($#[2],$Ideal,RVD1_1,$f,$Xl,$CuL,RVD1_2,RVD1_3,$Im1)

    CALL E_TF2W_CFG($#[3],$Ideal,RVD1_1,$f,$Xl,$CuL,RVD1_2,RVD1_3,$Im1)

    IF ($NLL .LT. 0.001) THEN

      RVD1_5 = 0.0

      RVD1_6 = 0.0

      IVD1_1 = 0

    ELSE

      RVD1_6 = $NLL

      RVD1_4 = 6.0/($Tmva*RVD1_6)
```

```
    RVD1_5 = RVD1_4*RVD1_2*RVD1_2

    RVD1_6 = RVD1_4*RVD1_3*RVD1_3

    IVD1_1 = 1

ENDIF

CALL E_BRANCH_CFG($BR11,$SS,IVD1_1,0,0,RVD1_5,0.0,0.0)

CALL E_BRANCH_CFG($BR12,$SS,IVD1_1,0,0,RVD1_5,0.0,0.0)

CALL E_BRANCH_CFG($BR13,$SS,IVD1_1,0,0,RVD1_5,0.0,0.0)

CALL E_BRANCH_CFG($BR21,$SS,IVD1_1,0,0,RVD1_6,0.0,0.0)

CALL E_BRANCH_CFG($BR22,$SS,IVD1_1,0,0,RVD1_6,0.0,0.0)

CALL E_BRANCH_CFG($BR23,$SS,IVD1_1,0,0,RVD1_6,0.0,0.0)

CALL TSAT1_CFG($BRS1,$BRS2,$BRS3,$SS,RVD1_1,

#IF SAT==1

    +RVD1_2,

#ELSE

    +RVD1_3,

#ENDIF

    +$Xair,$Xknee,$f,$Tdc,$Im1,$Txk)

#ENDBEGIN
```

Listing 2-7 – 3 Phase 2 Winding Transformer Component #BEGIN Directive Block

Now let's look at the *my_tx.f* file following a build.

```
!

    SUBROUTINE my_txDyn_Begin(tx_mva_)
!

! . . .
!

! Subroutine Parameters

    REAL   tx_mva_

! Control Signals

    REAL   tx_mva
```

```
!

! . . .

!

! -1:[xfmr-3p2w] 3 Phase 2 Winding Transformer 'T32'

    RVD1_1 = ONE_3RD*tx_mva

    RVD1_2 = 230.0*SQRT_1BY3

    RVD1_3 = 230.0

    CALL E_TF2W_CFG((IXFMR + 1),0,RVD1_1,60.0,0.1,0.0,RVD1_2,RVD1_3,0.&
    &4)

    CALL E_TF2W_CFG((IXFMR + 2),0,RVD1_1,60.0,0.1,0.0,RVD1_2,RVD1_3,0.&
    &4)

    CALL E_TF2W_CFG((IXFMR + 3),0,RVD1_1,60.0,0.1,0.0,RVD1_2,RVD1_3,0.&
    &4)

    IF (0.0 .LT. 0.001) THEN

     RVD1_5 = 0.0

     RVD1_6 = 0.0

     IVD1_1 = 0

    ELSE

     RVD1_6 = 0.0

     RVD1_4 = 6.0/(tx_mva*RVD1_6)

     RVD1_5 = RVD1_4*RVD1_2*RVD1_2

     RVD1_6 = RVD1_4*RVD1_3*RVD1_3

     IVD1_1 = 1

    ENDIF

    CALL E_BRANCH_CFG( (IBRCH(1)+1),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)

    CALL E_BRANCH_CFG( (IBRCH(1)+2),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)

    CALL E_BRANCH_CFG( (IBRCH(1)+3),SS(1),IVD1_1,0,0,RVD1_5,0.0,0.0)

    CALL E_BRANCH_CFG( (IBRCH(1)+4),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)

    CALL E_BRANCH_CFG( (IBRCH(1)+5),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)

    CALL E_BRANCH_CFG( (IBRCH(1)+6),SS(1),IVD1_1,0,0,RVD1_6,0.0,0.0)

    CALL TSAT1_CFG( (IBRCH(1)+7), (IBRCH(1)+8), (IBRCH(1)+9),SS(1),RVD&
    &1_1,RVD1_2,0.2,1.25,60.0,1.0,0.4,0.1)
```

```
    !

        RETURN

        END



```

<div align="center">

Listing 2-8 – Segment of the *my_tx.f* File

</div>

Within the *my_tx Dyn_Begin* subroutine (i.e. the BEGIN section of the system dynamics), the entire contents of the *#BEGIN* directive block can be found. This chunk of code contains calls to subroutines internal to EMTDC, which perform variable initialization and storage so as to ensure that the 3 Phase 2 Winding Transformer component is indeed runtime configurable.

Network Solution

Unlike the system dynamics, the electric network is not constructed according to module hierarchy or a particular sequence. Fundamentally, the electric network solution is a straightforward number crunch, which solves a vector of currents *I* for a given vector of voltages *V* and a matrix of conductances *G*. In other words:

$$\begin{bmatrix} G_{11} & \cdots & G_{1n} \\ \vdots & \ddots & \vdots \\ G_{n1} & \cdots & G_{nn} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} = \begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} \tag{2-1}$$

Electric network parameters, such as node and subsystem numbers, are defined automatically by PSCAD, according to how the network is graphically constructed. This information is compiled and summarised into files for use by EMTDC.

Data File
The Data file is used solely by the electric network solution for input – each module definition in a project will be represented by a unique Data file. Information regarding node and branch placement, the type of branch elements used, what their values are, etc., is stated in this file. Transmission line and transformer information, as well as information specified in component *Model-Data* segments, is also listed in this file.

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 shown below:

```
  !-------------------------------------

  ! Local Node Voltages

  !-------------------------------------

  VOLTAGES:

     1         0.0  // NT_2

     2         0.0  // NT_4
```

<div align="center">

Listing 2-9 – Local Node Voltage Section of Data File

</div>

The example above indicates that there are two nodes in this particular module. Both initial node voltages are set to *0.0*. *NT_2* and *NT_4* indicate the node names given by the compiler, which may also be defined through the use of a Node Label component.

Local Branch Data

The Local Branch Data section is used to define the contents of branches. Listing 2-10 indicates that there are four branches in this particular module. 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 ΩF* respectively. As before, the default node names, generated during project compilation, are shown at the far right.

```
!-------------------------------------

! 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
```

Listing 2-10 – Local Branch Data Section of Data File

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.

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

Branch Symbol	Description
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

Combinations of the symbols in Table 2-1 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 from a Data file shows that a non-ideal, two-winding transformer exists as indicated by the *2* in the first non-commented line. If the transformer were ideal, this would appear as *-2*.

```
!------------------------------------

! Local Transformer Data

!------------------------------------

TRANSFORMERS:

!  3 Phase, 2 Winding Transformer

 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 /

!
```

Listing 2-11 – Local Transformer Data Section of Data File

The next two un-commented lines define the R and L values of the mutual inductance matrix in the following format:

```
R11  L11

R21  L21 R22  L22
```

Listing 2-12 – Local Transformer RL Data Format

The 888 symbol signifies 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 definition. That is, any information added in the Model-Data segment, will appear here.

For example, the machine models in PSCAD use this section to define variables according to selected parameters. When a project containing a machine is built, 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 project, as well as to act as the link to map each Data file together. Its key role is to provide node look-up table information so as to convert the local node number index from each 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.

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	Description
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
PGBS	*Obsolete*
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)
CX	Controls table size
CXMAP	Controls map size
TX	Transmitter table size
TXRX	Transmitter map size
RTCL	Total number of RTCL locations used
RTCI	Total number of RTCI locations used

| | RTCF | Total number of RTCF locations used |
| | RTCC | Total number of RTCC locations used |

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

Runtime Parameters

The Runtime Parameters section 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	Description
TITLE	The PSCAD case project description
VERSION	The EMTDC version used
START_TIME	The simulation start time
FINISH_TIME	The simulation end 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
USE_SUBSYSTEM	Split electric network into subsystems

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

Subsystem and Node Mapping Information

The Subsystem and Node Mapping Information section provides a global map of all electrical nodes in the project. The following example illustrates the build results of a project containing 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 6 electrical nodes each.

```
!=====================================================================

! Sub-system and node mapping

!----------------------------------------------------------------

SUBSYSDIM:

 2 9 6

SUBS:

 4  1 2 1 2

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" !

 0  9   1 2 3 4 5 6 7 8 9 /

 1  1   1 1 1 1 1 1 1 1 1

MAP: "SS_2.dta" !

 0  6   1 2 3 4 5 6 /

 2  1   2 2 2 2 2 2
```

Listing 2-13 – Subsystem and Node Mapping Information Section of Map File

The *SUBSYSDIM* heading lists the total subsystems in the project, along with the dimensions of each. The *MAP* heading lists all node numbers and their corresponding subsystem directly beneath.

For example, the above indicates that the main page contains a total of 6 electrical nodes and a total of 2 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 that the main page itself is not considered a subsystem, and is therefore listed as subsystem *0*.

Global Transmission Lines
The Global Transmission Lines section summarizes some information about existing transmission line 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
```

Listing 2-14 – Global Transmission Lines Section of Map File

Listing 2-14 indicates that a single global transmission line (i.e. lines whose endpoints span multiple modules) 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.

Recorder Channel Information
The Recorder Channel Information section of the Map File simply summarizes the Output Channels used in the project.

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 zero with no initial conditions (i.e. start from the Data file created when the project is 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 zero, 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 2-5:

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 2-11 opens at time *t = 0.1 s*.
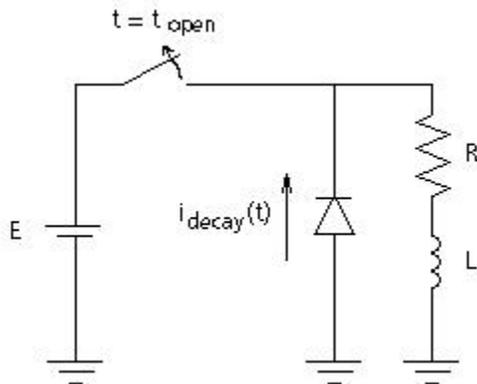
Figure 2-11 – Simple RL Circuit with Freewheeling Diode

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. 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 2-5 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, the project 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 three methods:

1. Using the Multiple Run component

2. Using the Optimum Run component

3. Manually defining the multiple run variables

If one of the components is used, it will automatically set the number of runs depending on the selected variation method. More than one these components can exist 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 Channel components 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 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 a single output channel. For example, if two output channels exist (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 project 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, 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
Output file columns are not labelled. In order to determine which column is what, an Information file (*.inf) is also created, which can contains cross-referencing information. The Information file will be named the same as the Output file.

Chapter 3: Electric Network Solution

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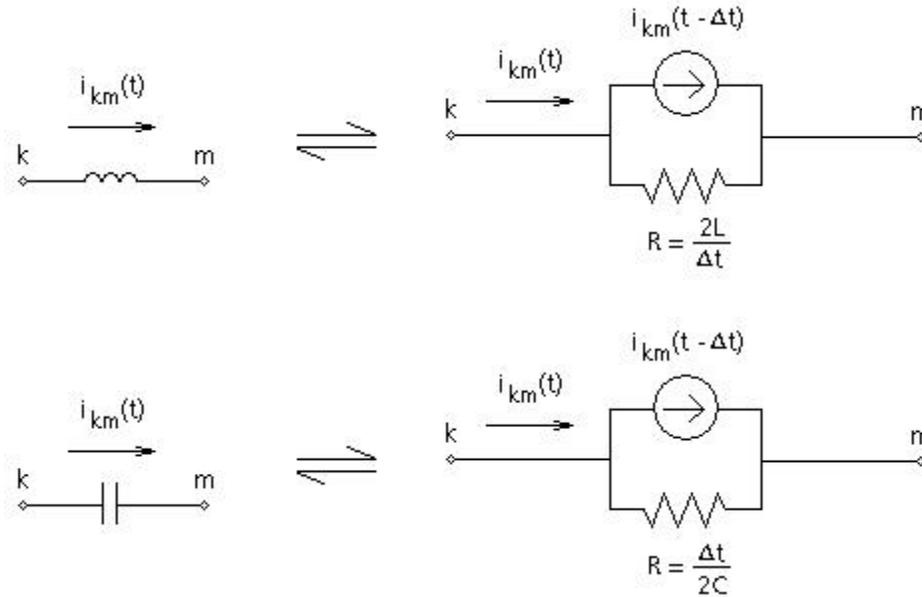
Figure 3-1 - Representation of lumped L and C elements

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:

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

and for the capacitor as:

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

Where,

$\Delta t =$ 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:

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

Where,

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

$R = \dfrac{\Delta t}{2C}$ 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:
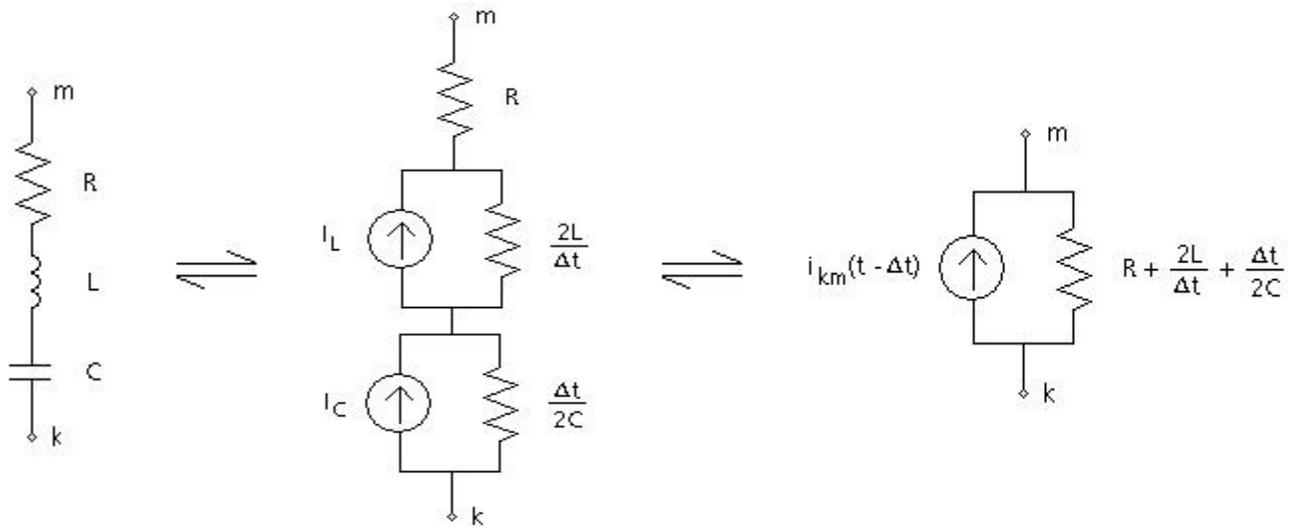
Figure 3-2 - Equivalent Branch Reduction

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:
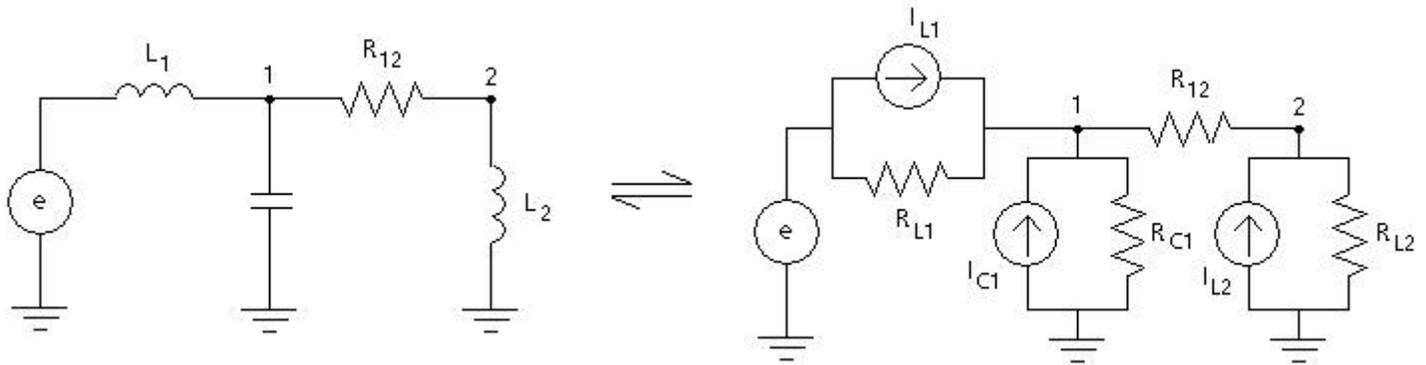
Figure 3-3 - RLC Equivalent Network in EMTDC

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:

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

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:

$$[V] = [G]^{-1} \cdot [I] \qquad (3\text{-}4)$$

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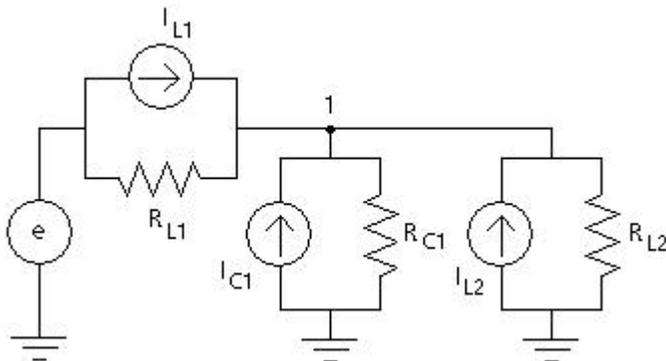
Figure 3-4 (a) - ON State of an Ideal Switch
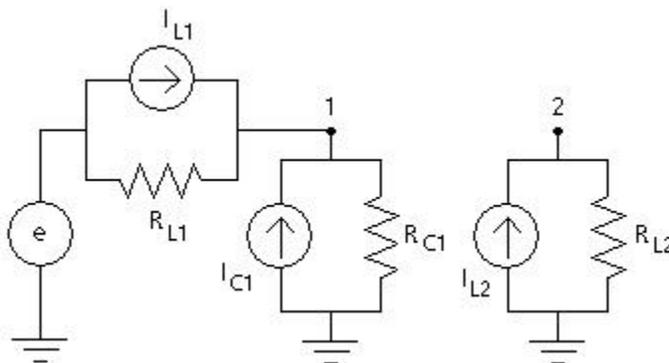
Figure 3-4 (b) - OFF State of an Ideal Switch

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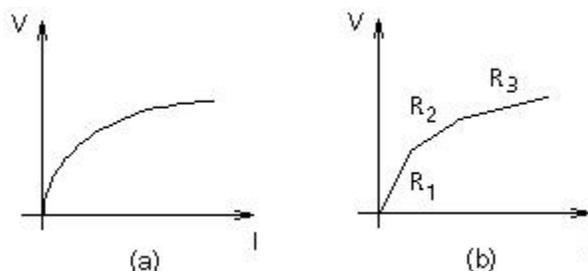
Figure 3-5 - (a) Continuous (b) Piecewise linear conductance properties

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.
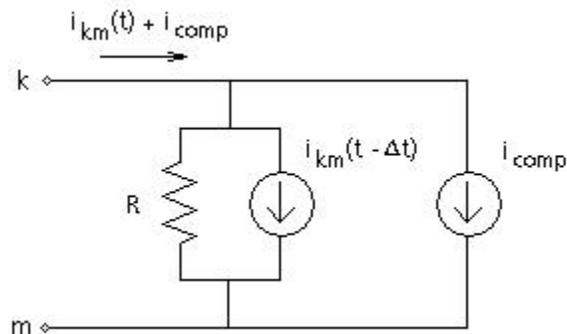
Figure 3-6 - Compensating Current Source Method

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 3-4:

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.

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 ⏃s) is of hardly any consequence. However, in power electronic circuit simulation, such a delay can produce very inaccurate results (i.e. 50 ⏃s 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-triangularization 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:
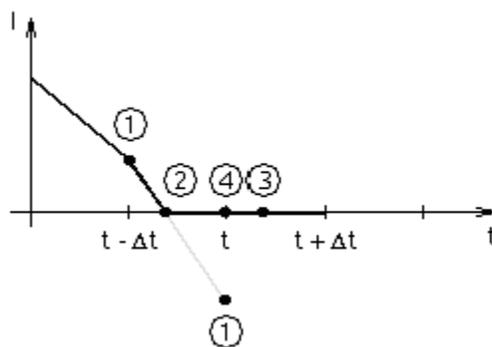
Figure 4-1 - Illustration of Interpolation Algorithm

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.
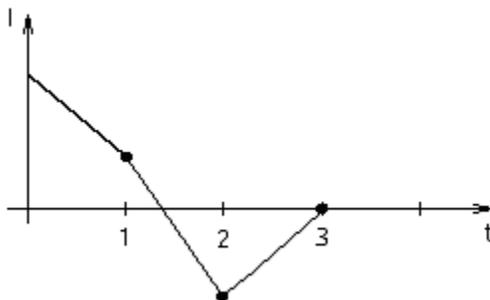
Figure 4-2 - Non-Interpolated Diode Current

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 switch-able 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 + ⸋t), and then interpolate the voltage back to time = 2 to bring the simulation back on track with integral time steps.
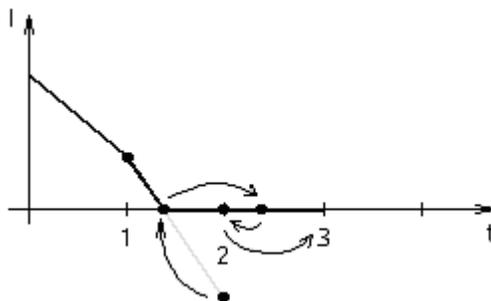
Figure 4-3 - Interpolated Diode Current

NOTE: DSDYN and DSOUT are still only called at times, 1, 2 and 3, yet the diode is still turned off at 1.2, so 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, 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 ⏲s 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 current.
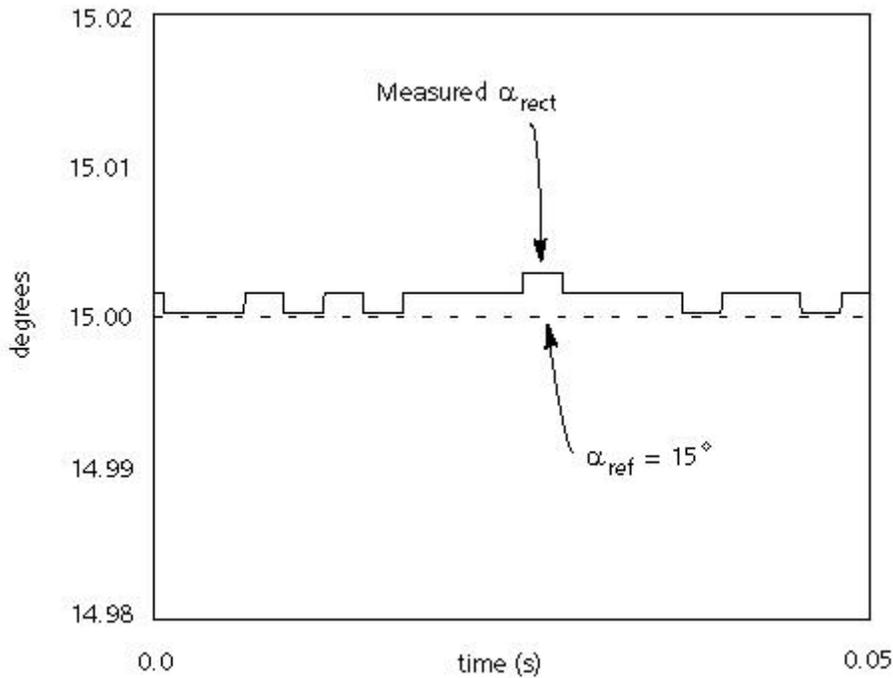
Figure 4-4 (a) - Example of Interpolation Effect: Interpolated
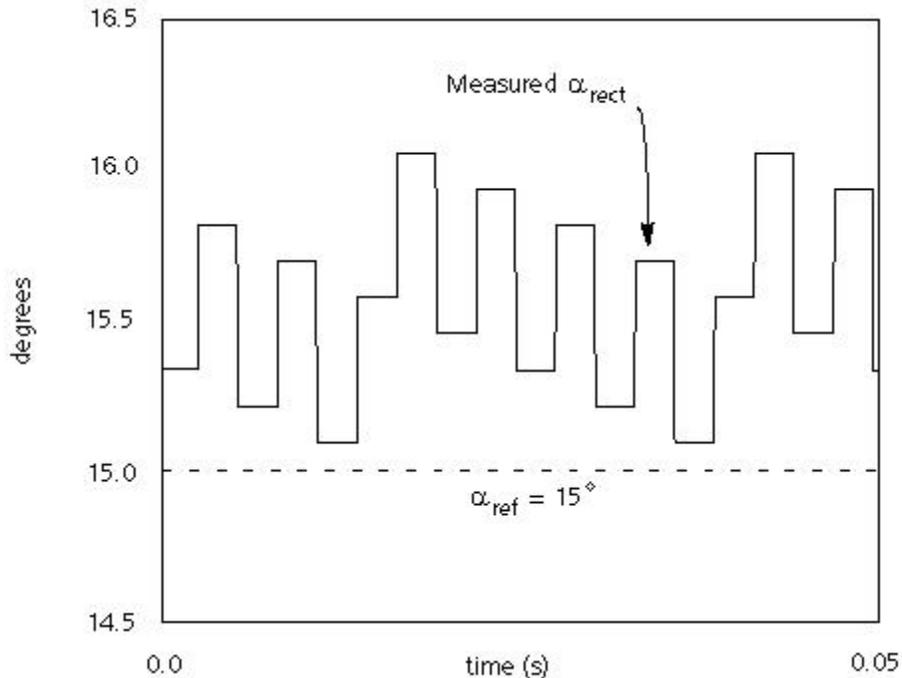
Figure 4-4 (b) - Example of Interpolation Effect: Non-interpolated

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 GTO's 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.
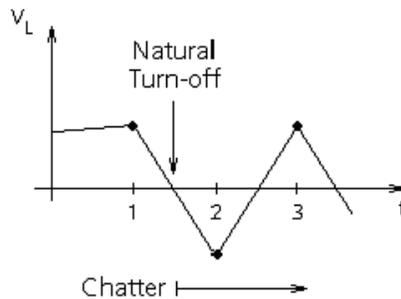
Figure 4-5 - Voltage Chatter Across and Inductor

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 pu in the PSCAD Project properties.

Either when chatter is detected or when a switching event takes place, a chatter removal algorithm in 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 interpolation is the *Extrapolate Sources* algorithm. During the process of switching and chatter removal, it is necessary to extrapolate beyond the current time step. At an interpolated time $t = t + \delta t$, PSCAD will use the exact value if using a sinusoidal voltage source (point A), rather than linearly extrapolating (point B). This is illustrated in Figure 4-6.
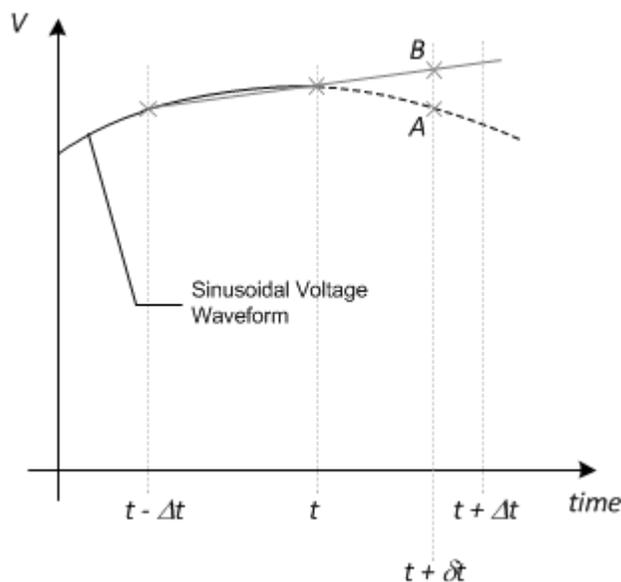
Figure 4-6 - Source Extrapolation

The *Extrapolate Sources* algorithm calculates the voltage at point A in Figure 4-6 as:

$$V_A = V \cdot sin\big(\omega(t + \delta t + \varphi)\big)$$ (4-1)

Where,

$\delta t =$ Time of switching instant between time steps [s]

$\omega =$ Source frequency [rad/s]

$\varphi =$ Source phase angle

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 ⏃. 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 ⏃ (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 a 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, that is 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: Custom Model Design

The EMTDC program is structured to accept user-defined, external source code. This is accomplished by either linking to pre-compiled source, such as object or static library files, or by simply appending the source directly. Whichever means is chosen, the external source will be combined with all other project source during the compilation process; resulting in a customized executable program for running the simulation.

External source is incorporated through the use of component objects. Components are employed in PSCAD to represent system models; in fact, the entire master library project is composed of components. Some of these components are designed to insert source directly into the system dynamics (i.e. BEGIN, DSDYN or DSOUT), and/or provide information for constructing the electric network. Others link to subroutines imbedded within the EMTDC body of source, where intrinsic network and storage variables are manipulated directly to represent complex electrical devices, such as machines and FACTS. Components are fully customizable and may range from the very simple, which require just a few lines of code, to the very complex, represented by a combination of several functions and subroutines.

The component concept affords flexibility in simulation design, providing a graphical interface to both EMTDC and the PSCAD project compiler. This interface ensures that all possible aspects of the EMTDC program (customizable parts anyway) are fully accessible to the user. After all, this design environment is the same one used by both PSCAD and EMTDC developers alike.

Fortran Guidelines for EMTDC

The base complier provided with PSCAD, entitled GFortran, supports up to and including the Fortran 95 language standard. In addition, the following commercial compilers are supported:

- **Intel Visual Fortran 9 to 15**: This is the preferred compiler to be used with PSCAD, which supports the Fortran 95 language standard.

Guidelines for Compatibility
To ensure that your model code remains portable over time the user should adhere to following general guidelines:

- **Comment Lines**: Always use an '!' character for comment lines. This is the common standard between all Fortran language standards. Exclamation characters can appear anywhere on the line except for column 6.

- **Names Convention**: To avoid conflicts between user-written procedure (subroutine/function) names and EMTDC procedures, prefix the name of your procedures with an easily identifiable string such as *U_* or *MY_*.

C Methods and Functions

If preferred, the user may code using the C language. C procedures may be called directly from within the component definition, in the exact same manner as Fortran subroutines and functions.   C language code cannot be inserted directly in component definition script, as the PSCAD project compiler, which writes all of the system Fortran files, will not translate from C.

See Interfacing to C Language Source in the PSCAD manual chapter *Component Design* for details, as well as examples on this subject.

EMTDC Intrinsic Variables

The intrinsic variables within EMTDC can be an essential part of interfacing user source. A good understanding of these variables is necessary to exploit the full potential from the tools available. The majority of EMTDC intrinsic variables are accessible to the user, and may be utilized within external source, provided that the proper header files are included. The following sections describe the most common variables: More detail is provided later in this chapter as well. If more information is required on intrinsic variables that are not discussed here, please contact the PSCAD support desk (support@pscad.com).

For a summary of which header file corresponds to a particular variable, see Include Files to Include later in this chapter.

EMTDC Storage Arrays
Storage arrays are required in cases where variables need to be stored and made available in subsequent time steps, or when passing information from the BEGIN section of the system dynamics to DSDYN and/or DSOUT. The storage arrays are of single-dimension (sometimes referred to as a stack), where a different array exists for each variable type.

	Type	EMTDC Storage Array	Description
Inter Time Step Data Transfer	REAL	STORF(NSTORF)	For floating point (REAL) storage only.
	INTEGER	STORI(NSTORI)	For INTEGER storage only.
	LOGICAL	STORL(NSTORL)	For LOGICAL storage only.
	COMPLEX	STORC(NSTORC)	For COMPLEX storage only.
BEGIN to DSDYN/DSOUT Data Transfer	REAL	RTCF(NRTCF)	For floating point (REAL) storage only.
	INTEGER	RTCI(NRTCI)	For INTEGER storage only.
	LOGICAL	RTCL(NRTCL)	For LOGICAL storage only.
	COMPLEX	RTCC(NRTCC)	For COMPLEX storage only.

Table 5-1 - Available Storage Arrays and Pointers

Inter Time Step Data Transfer

Figure 5-1 illustrates a typical storage array used in the transfer of data between time steps. Data may be stored at individual address locations, using the corresponding pointer integer (i.e. NSTORF, NSTORI, etc.). Proper use of the storage 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.
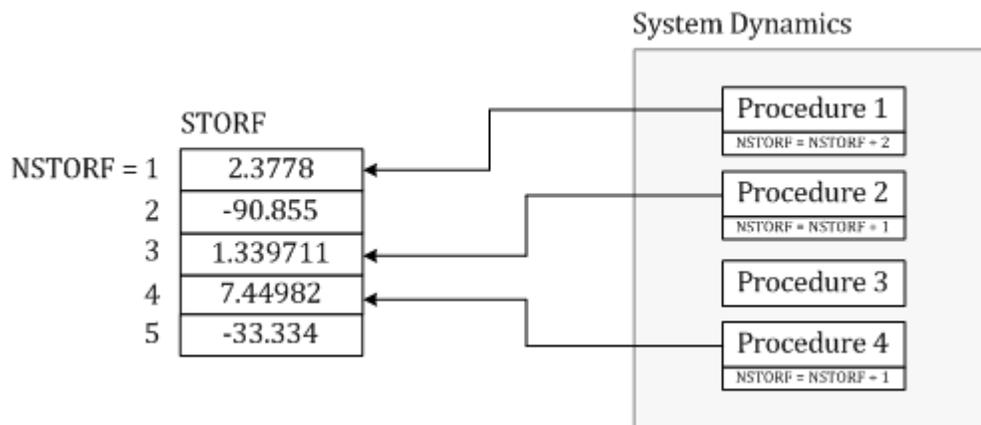
Figure 5-1 - Typical Storage Array (STORF) Usage by System Dynamics Procedures

Each time step, all storage array pointers are reset to one (1) and then the main program is sequenced from top to bottom. Using the storage array and pointer, each subroutine may write data to and read data from an array, consistent with the sequence in which the subroutine appears in the main program. In order to avoid stored data being over-written, underline each subroutine must increment the respective pointer, by the amount of storage locations used in the subroutine, before returning to the main program. This will ensure that the pointers are in their proper positions when the next storage access is performed.

EXAMPLE 5-1:

Consider a user-written subroutine that requires that two variables be stored for retrieval in subsequent time steps: Two REAL variables *X* and *Y* and one INTEGER variable *Z*.   The subroutine should include something similar to what is shown below:

```
!

    SUBROUTINE U_USERSUB(...)

!

    INCLUDE 'nd.h'

    INCLUDE 'emtstor.h'

!

    REAL    X, Y, X_OLD, Y_OLD

    INTEGER  Z, Z_OLD

!

! Retrieve the variables from storage arrays:

!

    X_OLD = STORF(NSTORF)

    Y_OLD = STORF(NSTORF + 1)

    Z_OLD = STORI(NSTORI)

!

! Main body of subroutine:

!

  ...

!

! Save the variables to storage arrays for use in next time step:

!

    STORF(NSTORF)    = X

    STORF(NSTORF + 1) = Y

    STORI(NSTORI)    = Z

!

! Increment the respective pointers before returning to the main

! program (very important):

!
```

```
        NSTORF = NSTORF + 2

        NSTORI = NSTORI + 1

    !

        RETURN

        END

    !
```

Listing 5-1 – Illustration of STORx Array Usage

A good habit to get into is to make a copy of the storage 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 is avoided. Using this approach, the above example would become:

```
    !

        SUBROUTINE U_USERSUB(...)

    !

        INCLUDE 'nd.h'

        INCLUDE 'emtstor.h'

    !

        REAL    X, Y, X_OLD, Y_OLD

        INTEGER  MY_NSTORF, MY_NSTORI, Z, Z_OLD

    !

    ! Copy the pointer values to locally declared variables:

    !

        MY_NSTORF = NSTORF

        MY_NSTORI = NSTORI

    !

    ! Increment the respective pointers before continuing:

    !

        NSTORF = NSTORF + 2

        NSTORI = NSTORI + 1

    !

    ! Retrieve the variables from storage arrays (using local pointers):
```

```
!

    X_OLD = STORF(MY_NSTORF)

    Y_OLD = STORF(MY_NSTORF + 1)

    Z_OLD = STORI(MY_NSTORI)

!

! Main body of subroutine:

!

  ...

!

! Save the variables to storage arrays for use in next time step:

!

    STORF(MY_NSTORF)    = X

    STORF(MY_NSTORF + 1) = Y

    STORI(MY_NSTORI)    = Z

!

    RETURN

    END

!
```

Listing 5-2 – Illustration of STORx Array Usage with Local Pointers

BEGIN to DSDYN/DSOUT Data Transfer

The BEGIN section is used to perform pre-run (or time zero) operations, such as initialization of variables, etc.   Data transfer from BEGIN to both the DSDYN or DSOUT sections occurs at the beginning of each time step.
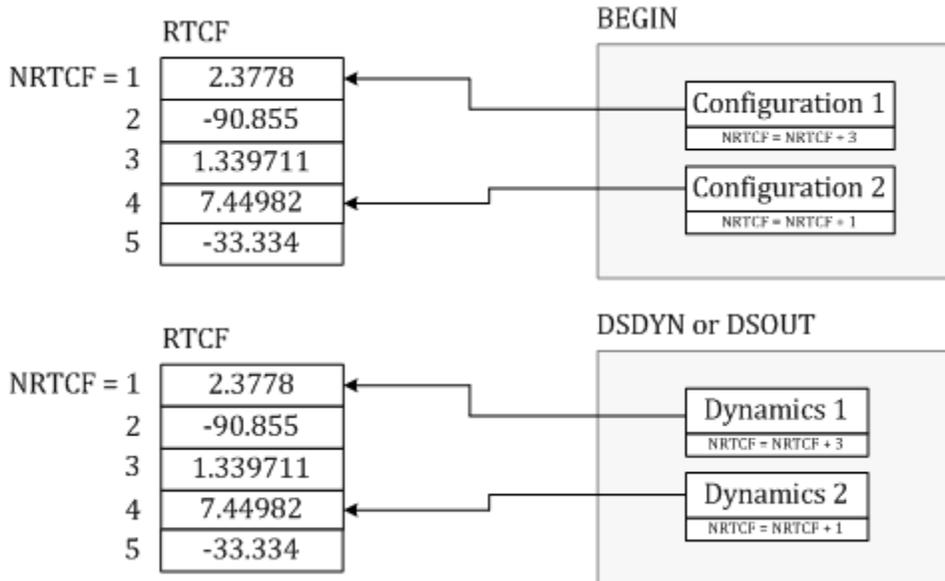
Figure 5-2 - Typical Storage Array Usage by BEGIN Procedures

BEGIN was introduced primarily to provide Runtime Configuration support in EMTDC when components exist within modules with multiple instances. The storage arrays used to transfer data (i.e. the RTCx arrays) are included, so as to not affect the storage operations of legacy user-components that were created using PSCAD versions previous to X4. All master library components use both storage array types when necessary.

To facilitate runtime configuration support (and thereby multiple instance modules support) within user-designed components, it may be necessary to utilize the BEGIN section (and hence the RTCx storage arrays). This is normally the case if the component requires time zero initialization.

EXAMPLE 5-2:

Consider a user-defined component that needs to store a set of two xy data points at time zero, for use later during runtime. The data points are defined as follows:

	X	Y
Point 1	0.7	3.1
Point 2	1.2	3.8

Table 5-2 – Set of XY Data Points for This Example

The storage of these data points will need to occur within the BEGIN section of the system dynamics, so as to ensure that the component is *Runtime Configurable*. The user decides to write a subroutine entitled *U_BGN_XYPOINTS* that will be called within the BEGIN section. Note that the text *BGN* was added to the procedure name – this is to help identify it as BEGIN-type.

```
!

    SUBROUTINE U_BGN_XYPOINTS(X,Y)

!

    INCLUDE 'nd.h'

    INCLUDE 'rtconfig.h'

!

    REAL X(2), Y(2)

!

! Save variables to storage array for use in runtime:

!

    RTCF(NRTCF)    = X(1)

    RTCF(NRTCF + 1) = Y(1)

    RTCF(NRTCF + 2) = X(2)

    RTCF(NRTCF + 3) = Y(2)

!

! Increment the pointers before returning to the main program

! (very important):

!

    NRTCF = NRTCF + 4

!

    RETURN

    END

!
```

Listing 5-3 – Illustration of RTCx Array Usage within BEGIN Subroutine

Now that the subroutine is defined, the user must specify to PSCAD that a call to it is to be inserted in the BEGIN section. This is accomplished by using the #BEGIN/#ENDBEGIN directives within the component definition. See Script Directives in the PSCAD manual chapter *Definition Script* if you are not familiar with these.

```
#BEGIN

    CALL U_BGN_XYPOINTS($X,$Y)

#ENDBEGIN
```

Listing 5-4 –BEGIN Subroutine Call from Component Definition

The *X* and *Y* variables represent pre-defined quantities in the component definition. These could be, for example, component input parameters. Note that instead of defining a subroutine, the data storage may also be coded directly in the component definition as follows:

```
#BEGIN

    RTCF(NRTCF)    = $X(1)

    RTCF(NRTCF + 1) = $Y(1)

    RTCF(NRTCF + 2) = $X(2)

    RTCF(NRTCF + 3) = $Y(2)

!

    NRTCF = NRTCF + 4

#ENDBEGIN
```

Listing 5-5 – Illustration of RTCx Array Usage within Component Definition

Now that the data is being stored in BEGIN, you will need to retrieve it during runtime. The user has written a subroutine that defines the runtime (or the dynamic modeling) operation of the component called *U_DYN_XYPOINTS*. This routine includes RTCx storage retrieval, along with STORx type arrays:

```
!

    SUBROUTINE U_DYN_XYPOINTS(...)

!

    INCLUDE 'nd.h'

    INCLUDE 'emtstor.h'

    INCLUDE 'rtconfig.h'

!

    REAL    X(2), Y(2)

    INTEGER  MY_NSTORI, TEMP

    INTEGER  MY_NRTCF

!

! Copy the pointer values to locally declared variables:

!

    MY_NSTORI = NSTORI

    MY_NRTCF  = NRTCF

!
```

```
! Increment the respective pointers before continuing:

!

    NSTORI = NSTORI + 1

    NRTCF  = NRTCF  + 4  ! Number of RTCF storage elements used

!

! Retrieve the variables from storage arrays (using local pointers):

!

    TEMP = STORI(MY_NSTORI)

    X(1) = RTCF(MY_NRTCF)

    Y(1) = RTCF(MY_NRTCF + 1)

    X(2) = RTCF(MY_NRTCF + 2)

    Y(2) = RTCF(MY_NRTCF + 3)

!

! Main body of subroutine:

!

 ...

!

! Save variables to storage arrays for use in next time step

! (STORx arrays only!):

!

    STORI(MY_NSTORI) = TEMP

!

    RETURN

    END

!
```

Listing 5-6 – Illustration of RTCx Array Usage within DSDYN/DSOUT Subroutine

Common Intrinsic Network Variables

Through intrinsic variables, the user may access electric network data, such as branch and node numbers, as well as monitor branch current and node voltage. Some network variables, such as branch voltage may even be controlled.

Node Numbers

Node numbers may be accessed as follows:

Variable Name	Description
IEF(BRN,SS)	Gives the number of the mapped 'from' node
IET(BRN,SS)	Gives the number of the mapped 'to' node

Table 5-3 – Intrinsic Variables to Access Node Numbers

Branch Current

The current flowing in a given branch can be monitored as follows:

Variable Name	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.

Table 5-4 – Intrinsic Variable to Access Branch Current

The voltage at a given node can be monitored as follows:

Variable Name	Description
VDC(NN,SS)	Gives the value of voltage at node *NN* in subsystem *SS*.

Table 5-5 – Intrinsic Variable to Access Node Voltage

Electric Network Interface Variables

These variables may be used for direct control of the electric network interface:

Variable Name	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 an interface branch
GEQ(BRN,SS)	Sets the value of the branch equivalent conductance.
CCIN(NN,SS)	An ideal current source that sets the value of current injected into the node NN from ground.
GGIN(NN,SS)	Sets a conductance value between node NN and ground

Table 5-6 – Intrinsic Variables for Control of an Interface to the Electric Network

NOTE: See the section entitled Interfacing to the Electric Network later in this chapter for more details.

Sending Model Messages to PSCAD

In order to communicate messages to the user through PSCAD, there are some specific tools to implement in the design of your user component.

The COMPONENT_ID Subroutine

If there are warning messages to be passed from a user component back to PSCAD, then the *COMPONENT_ID* subroutine should be utilized.

Multiple calls to this routine from the same component may be required depending on where the warning messages are generated. For example, if you have warning messages generated inside both the *DSDYN_BEGIN* and *DSDYN* system dynamics routines, then you will need to call the *COMPONENT_ID* subroutine twice, one inside the #BEGIN/#ENDBEGIN directive, as well as another outside in the DSDYN section. *COMPONENT_ID* should be called before the routine that generates the warning message.

```
#BEGIN

    CALL COMPONENT_ID(ICALL_NO,$#Component)

!

! . . .

!

#ENDBEGIN

!

    CALL COMPONENT_ID(ICALL_NO,$#Component)

!

! . . .

!
```

Listing 5-7 – Calling COMPONENT_ID from the Script Section

In Listing 5-7, *ICALL_NO* is used to set the component call number and *$#Component* sets the instance number. Both of these arguments are predetermined by PSCAD.

The EMTDC_WARN Subroutine

The *COMPONENT_ID* subroutine simply sets the EMTDC global variables *COMP_ID1* and *COMP_ID2*, as defined in the include file *warn.h* (see next the section called Include Files in this chapter). These two global variables are used as arguments to the *EMTDC_WARN* subroutine.

EMTDC_WARN is used to generate warning messages from within user component Fortran code.

```
!

    INCLUDE 'warn.h'

!

! . . .

!
```

```
      IF (F .LT. 0.01) THEN

        CALL EMTDC_WARN(COMP_ID1,COMP_ID2,1,"Frequency is below limit")

      ENDIF

!

! . . .

!
```

Listing 5-8 – Calling EMTDC_WARN from within User Component Fortran Code (Single Line)

Note that the *warn.h* file must be declared at the top of the component subroutine. In Listing 5-8, *COMP_ID1* is used to set the component call number and *COMP_ID2* sets the instance number. Both of these arguments are predetermined from within PSCAD using the *COMPONENT_ID* subroutine.

For multiple lines in the same message, use the EMTDC_WARN subroutine as follows:

```
!

    INCLUDE 'warn.h'

!

! . . .

!

    IF (F .LT. 0.01) THEN

      CALL EMTDC_WARN(COMP_ID1,COMP_ID2,3,"This is the first line")

      CALL EMTDC_WARN(COMP_ID1,COMP_ID2,0,"This is an intermediate line")

      CALL EMTDC_WARN(COMP_ID1,COMP_ID2,-1,"This is the last line")

    ENDIF

!

! . . .

!
```

Listing 5-9 – Calling EMTDC_WARN from within User Component Fortran Code (Multiple Lines)

Note that in both Listing 5-8 and 5-9, the 3rd argument is used to indicate to EMTDC how many lines in the message there are, and which is the current line. Indicate the total number of lines in the message in the first call to *EMTDC_WARN*. Following that, a *0* represents an intermediate line, and a *-1* indicates the final line.

Include Files

The previous section described only some of the most commonly used intrinsic variables in EMTDC. There are numerous others as well, but regardless of which are utilized, the appropriate header file must be included within any external source procedure. The inclusion of header files is not required when the component source is situated within the definition itself (i.e. the Script section) – only external subroutines and functions.

The following tables provide a quick reference for the most commonly used include files.

> **NOTE**: This is also a list of all reserved intrinsic variable names, which should not be declared locally in any user-written models.

nd.h

This file contains important network dimensioning information, and must always be the first file included in all external procedures.

emtstor.h

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

rtconfig.h

Variable Name	Type	Description
RTCL(*)	LOGICAL	Storage array for time zero logical variables
RTCI(*)	INTEGER	Storage array for time zero integer variables
RTCF(*)	REAL	Storage array for time zero floating point variables
RTCC(*)	COMPLEX	Storage array for time zero complex variables
NRTCL	INTEGER	Pointer for RTCL array
NRTCI	INTEGER	Pointer for RTCI array
NRTCF	INTEGER	Pointer for RTCF array
NRTCC	INTEGER	Pointer for RTCC array
TFDATA(*,*)	REAL	Used for passing R and L data to the classical transformer model Runtime Configuration procedure
UMECWDGDATA(*,*)	REAL	Used for passing winding specific information to the UMEC transformer model Runtime Configuration procedure
UMECTFDATA(8)	REAL	Used for passing general information (length ratios, areas, etc.) to the UMEC transformer model Runtime Configuration procedure
UMECSATDATA(10,2)	REAL	Used for passing saturation data points to the UMEC transformer model Runtime Configuration procedure

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 triangularized [G] matrix
CDCTR(*,*)	REAL	Current flowing through the Mth winding of the Nth transformer.
MBUS(*)	INTEGER	Number of nodes in a subsystem
IDEALSS(*)	LOGICAL	True if subsystem contains ideal branches
ENABCCIN(*,*)	LOGICAL	True if corresponding CCIN source is enabled

s1.h

Variable Name	Type	Description
TIME	REAL	Current time of simulation (t) in seconds
DELT	REAL	Simulation time step (Δt) in seconds
PRINT	REAL	Plot step interval in seconds
FINTIM	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

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	$\pi/3$ = 1.047197551196598
PI2_BY3	REAL	$2\pi/3$ = 2.094395102393195
PI_BY2	REAL	$\pi/2$ = 1.570796326794896
PI_BY180	REAL	$\pi/180$ = 0.017453292519943
BY180_PI	REAL	$180/\pi$ = 57.29577951308232
BY_PI	REAL	$1/\pi$ = 0.318309886183791
BY_2PI	REAL	$1/2\pi$ = 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

warn.h

Variable Name	Type	Description
COMP_ID1	INTEGER	Component call number
COMP_ID2	INTEGER	Component instance number

Interfacing to the Electric Network

Most of the time, the user can get away with representing electric models by simply combining fundamental electric components from the master library project. However, some electric models may contain characteristics and features that are impossible to represent this way; or perhaps the modeling of such devices is proprietary. Regardless of the reason, interfacing directly to the electric network solution remains an important feature for maintaining flexibility in model development.

Equivalent Conductance (GEQ) Electric Interface

The equivalent conductance (or GEQ) interface provides the user with a straightforward method of interfacing electric models to the greater electric network. Introduced with the release of EMTDC V3, the GEQ interface is an enhancement to the now obsolete, nodal-based GDC and GDCS interfaces used in EMTDC V2. It is also an enhanced alternative to the node-to-node GGIN interface, which is deprecated and will slowly phased out over time.

These older interfaces were hindered by the fact that their parameters were indexed by way of node number, rather than by branch. Problems with these interfaces arose when connecting several of them in parallel. Due to the fact that each parameter was indexed according to node, each paralleled branch would have the exact same index, and so additional steps by the user were required to merge the branches into a single equivalent branch. Also, extracting branch specific quantities, such as branch current was a bit convoluted.

The GEQ interface is virtually identical to the obsolete GDC interface, except that the parameter names and indexing have changed – one of its nodes can be grounded if desired as well. Each parameter is indexed by a branch number directly, thereby enabling the extraction of branch data more easily. The GEQ interface is illustrated below, where *BRN* represents the branch number and SS is the subsystem number:
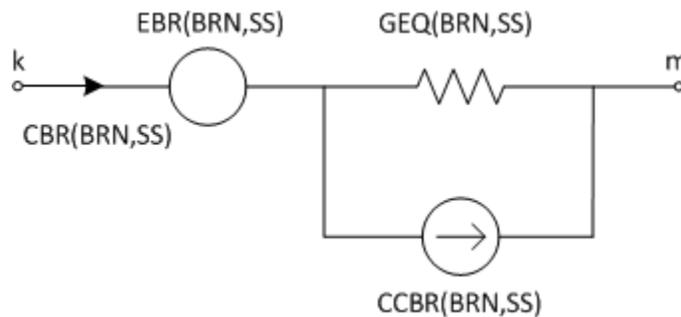
Figure 5-3 – Equivalent Conductance (GEQ) Interface Branch

k and *m* are referred to as the 'from' and 'to' nodes respectively.   The parameters shown in Figure 5-3 represent the electrical functions of the GEQ interface, where *EBR* is an internal, ideal voltage source; GEQ and CCBR are the Dommel equivalent conductance and history current respectively, and CBR contains the calculated branch current. These parameters are described in more detail later on in this section.

One advantage to using the GEQ interface over say, CCIN and GGIN (described later on), is that the branch history current (CCBR) is calculated by EMTDC automatically for LC branches. Also, the user need only provide the conductance (and other branch information) once at the beginning of the simulation (unless of course the branch is a switching branch).

The GEQ interface can be set-up to represent any combination of series RLC passive elements, and along with its internal voltage source, can represent network equivalents.   It is also possible to set-up and control the GEQ interface to model non-linear impedances.

General Considerations

The GEQ interface is primarily used to represent branches consisting of series-connected R, L and/or C elements. Any series combination of such elements is reduced by EMTDC into a single lumped admittance, represented by a Dommel equivalent, Norton conductance and current source (GEQ and CCBR). The design process is somewhat automated where instead of relying on the user to directly calculate quantities each time step, the approach here is to provide EMTDC with information on how the branch is constructed, and what it consists of. This information is given once at the beginning of the simulation (or whenever an R, L or C value changes), and then EMTDC handles the rest, such as internal node voltages and branch history currents.

The GEQ interface relies on the settings of certain parameters in order to effectively reduce a combination of elements to its Dommel equivalent. This reduction is accomplished in two stages: First, each element is reduced to its respective Dommel equivalent, and second, these individual circuits are merged into a single, lumped equivalent.
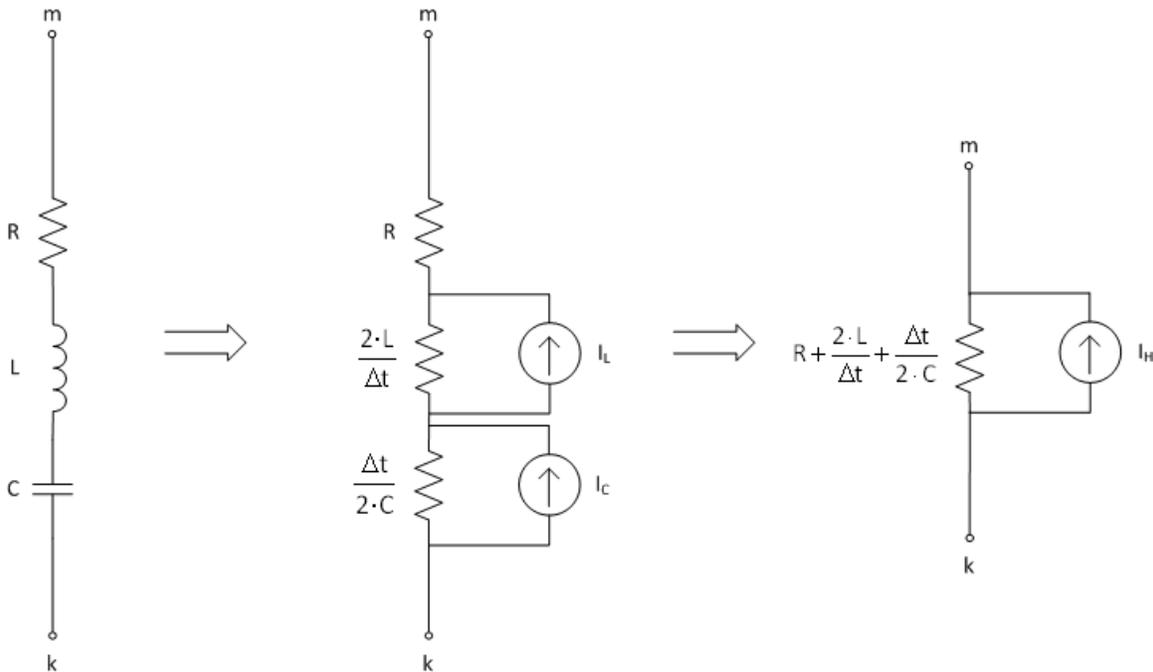
Figure 5-4 – Steps in the Reduction of an RLC Branch Impedance to a Lumped Dommel Equivalent Circuit

Although the process for deriving the reduced branch conductance is relatively straightforward, deriving the equivalent history current I_H is not. The history current derivation depends on combinations of ratios between individual conductance values in the branch, which can be quite elaborate. Nonetheless, users must supply this information to EMTDC and fortunately, the GEQ interface provides an avenue to do this easily.

Custom Current Source and Conductance Interface

A simple interface routine is provided so that users do not have to set global variables related to the branch manually. To use the current source and conductance interface, the user must add the following script to the Branch segment of the user component definition.

```
BR = $A $B BREAKER 1.0
```

Listing 5-10 – Example Branch Section Script for a Custom Current Source and Conductance Branch

BR is the symbolic branch name, *$A* and *$B* are the end nodes of the branch. The default resistance of the branch is set to *1.0 Ω*.

Once a branch has been designated as shown above, the user must enter specific subroutine calls within the component definition DSDYN or Fortran segments (or directly within the user-written subroutine for the model). Within #BEGIN/#ENDBEGIN directives, there should be a call to the *CURRENT_SOURCE2_CFG* routine.

```
CALL CURRENT_SOURCE2_CFG($BR,$SS)
```

Listing 5-11 – Example Code for a Custom Current Source and Conductance Branch (Configuration)

Where *$BR* and *$SS* are the predefined branch number and subsystem number respectively. This subroutine sets the internal EMTDC variable *DEFRDBR($BR,$SS)* to *.TRUE.*, which is required to enable the custom current interface using the *CCBR($BR,$SS)* branch history current. If a call to *CURRENT_SOURCE2_CFG* is not provided, the CCBR variable for this particular branch is ignored. *CURRENT_SOURCE2_CFG* also sets the branch conductance *GEQ($BR,$SS)* to zero, and removes this branch from being used by the harmonic impedance solution.

In the main body of the DSDYN or Fortran segments, users must set the branch *GEQ* and *CCBR* by calling *CURRENT_SOURCE2_EXE* routine.

```
CALL CURRENT_SOURCE2_EXE($BR,$SS,COND,CUR)
```

Listing 5-12 – Example Code for a Custom Current Source and Conductance Branch (Executable)

Where *COND* is the actual conductance value and *CUR* is the injected current value. If the conductance value changes at every time step, then it is advisable to ensure that nodes *A* and *B*, as defined in the component Branch segment above, are switched node types.

Runtime Configurable Passive Branch
If the requirement is a simple passive branch, where the RLC values need be assigned in the BEGIN section at *TIMEZERO*, then the following method can be used. Define a passive branch in the Branch segment and then modify the branch variables in the DSDYN segment within #BEGIN/#ENDBEGIN directive.

```
BR = $A $B 1.0 0.1 1.0
```

Listing 5-13 – Example Branch Section Script for a Runtime Configurable Passive Branch

BR is a symbolic branch name, *$A* and *$B* are the end nodes of the branch. The default resistance is *1.0 Ω*, inductance *0.1 H* and capacitance *1.0 µF*.

Once a branch has been designated as shown above, the user must enter specific subroutine calls within the component definition DSDYN segment (or directly within the user-written subroutine for the model). Within #BEGIN/#ENDBEGIN directives, there should be a call to *E_BRANCH_CFG* routine.

```
CALL E_BRANCH_CFG($BR,$SS,ER,EL,EC,R,L,C)
```

Listing 5-14 – Example Code for a Runtime Configurable Passive Branch (Configuration)

Where,

Argument Name	Type	Description
ER	INTEGER	Resistance:
		• **0**: Disable
		• **1**: Enable
EL	INTEGER	Inductance:
		• **0**: Disable
		• **1**: Enable
EC	INTEGER	Capacitance:
		• **0**: Disable
		• **1**: Enable
R	REAL	Resistance [Ω]
L	REAL	Inductance [H]
C	REAL	Capacitance [μF]

Table 5-7 – Argument List for the *E_BRANCH_CFG* Routine

Care should be taken not to pass a zero or negative value to any of the *R*, *L* or *C* arguments. There should be conditional logic to set the values of *ER*, *EL* and *EC* depending on the values of *R*, *L* and *C*. If *ER* is enabled and *R* value is less than the ideal threshold, then the resistive portion of the branch would be modeled as a short. If *ER*, *EL* and *EC* are all disabled, then the branch is modeled as an ideal open circuit.

Control of the GEQ Interface from PSCAD

Everything discussed thus far regarding the GEQ interface can accomplished directly within the component definition, without the need for calls to external code. That is, it is not necessary for the user to define the GEQ interface parameters through code, if designing a simple RLC branch with or without an internal source.

There are two avenues through which the GEQ interface can be controlled, these are:

1. The Branch segment in the component definition Script section

2. Utilizing the *E_VARRLC1x* internal subroutines.

The Branch segment is normally selected if the user wishes to create a simple, static RLC branch (with or without an internal source). The *E_VARRLC1x* subroutines can be used for this purpose as well, but with the added ability for the online control of a single, non-linear R, L or C element in a branch.

Branch Segment

The Branch segment in the component definition Script section can be used to directly define one or more GEQ interface branches. Here, the user simply needs to create a new component, and enter the R, L and/or C values directly into a defined branch. EMTDC will automatically set-up the required flags, parameters and ratios for you.

EXAMPLE 5-3:

A user wants to create a new component, which will represent a 3-phase, Y-connected RLC constant impedance with *R = 1.2 Ω, L = 0.053 H* and *C = 33.3 μF*. A new component is created with 4 electrical connections named *NA*, *NB*, *NC* (phase nodes) and *GND* (common node).
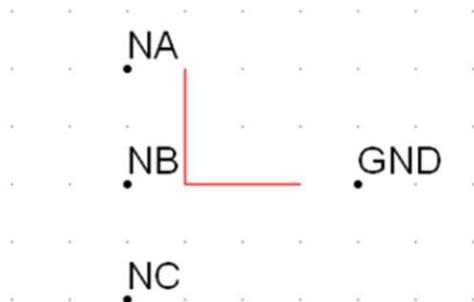
Figure 5-5 – Electrical Port Connections Defined in the Graphics Section of the Component Definition

A Branch segment is added and the three, Y-connected branches are defined as shown below as it would appear in the Branch segment:

```
BRNA = $NA  $GND  1.2  0.053  33.3

BRNB = $NB  $GND  1.2  0.053  33.3

BRNC = $NC  $GND  1.2  0.053  33.3
```

Listing 5-15 – Example Branch Section Script for a Static RLC Branch

The above script will define three separate, series RLC branches, which use the GEQ interface. See the section entitled Segment Types in the chapter Component Design of the PSCAD Manual for more details on defining branches.

If the user wishes to enable and control the internal branch voltage source EBR, then a slight modification is required to Example 5-3:

EXAMPLE 5-4:

A user wants to create the exact same 3-phase, Y-connected load as described in Example 5-3. However an internal branch voltage is also to be included. The component still consists of four electrical connections, but the Branch segment must be modified as shown below:

```
BRNA = $NA  $GND  SOURCE  1.2  0.053  33.3

BRNB = $NB  $GND  SOURCE  1.2  0.053  33.3

BRNC = $NC  $GND  SOURCE  1.2  0.053  33.3
```

Listing 5-16 – Example Branch Section Script for a Static RLC Branch with Internal Voltage Source

The additional *SOURCE* script in each branch effectively enables the EBR internal branch source in the GEQ interface. EMTDC will then expect a value for EBR to be defined by the user each time step.

E_VARRLC1x Subroutines

These subroutines are included within EMTDC and may be called directly from the Fortran, DSDYN or DSOUT segments of any user defined component definition. They allow the user to directly model either a linear or a non-linear R, L or C passive element, as well as provide control of the internal source, EBR. Although this subroutine is, for the most part, a direct link to the GEQ interface, it is limited to the fact that only a single R, L or C element can exist in a single branch.

The subroutine call statements and argument descriptions are given as follows:

CALL E_VARRLC1_CFG(RLC,M,NBR,NBRC)

CALL E_VARRLC1_EXE(RLC,M,NBR,NBRC,Z,E)

Where,

Argument Name	Type	Description
RLC	INTEGER	Branch type:
		• **0**: Resistor
		• **1**: Inductor
		• **2**: Capacitor
		• **3**: Inductor with dL/dt effects
		• **4**: Capacitor with dC/dt effects
M	INTEGER	The subsystem number
NBR	INTEGER	The branch number
NBRC	INTEGER	The branch number of compensating current source for dL/dt or dC/dt effects

---

| Z | REAL | Input branch element magnitude R, L, OR C (Ω, H, μF) |
| E | REAL | Internal branch source voltage (EBR control) |

Table 5-8 – Argument Descriptions for the E_VARRLC1_CFG and E_VARRLC1_EXE Subroutines

Note that the *E_VARRLC1x* subroutines are optimized for the control of non-linear passive elements. It should not be used to represent constant elements, as the branch cannot be collapsed with other series elements in the circuit. This can result in additional nodes, and slower simulation times. See the section entitled Equivalent Branch Reduction in the chapter entitled Electric Network Solution for more details on this.

EXAMPLE 5-5:

In this example, the *E_VARRLC1x* subroutines are called from the Fortran segment of a user defined component definition, with its various arguments pre-defined as needed.   The Fortran segment of the user component is shown below:

```
#SUBROUTINE E_VARRLC1_CFG 'Variable RLC Begin Subroutine'

#SUBROUTINE E_VARRLC1_EXE 'Variable RLC Dynamic Subroutine'

#BEGIN

    CALL E_VARRLC1_CFG(1,$SS,$BRN,$BRN)

#ENDBEGIN

#STORAGE REAL:2

!

    CALL E_VARRLC1_EXE(1,$SS,$BRN,$BRN,$L,$E)

!
```

Listing 5-17 – Calling the E_VARRLC1x Subroutines within the Component Definition

Here, both the *E_VARRLC1_CFG* and *E_VARRLC1_EXE* subroutines are configured to model a non-linear inductor (i.e. input argument *1*), where the inductance is controlled by the input argument *L*. *E* controls the internal voltage source *EBR*. *BRN* and *SS* are the branch number and subsystem number respectively.

The branch number is defined in the Branch segment of the component definition as follows, where *NA* and *GND* are electrical nodes:

```
BRN = $NA  $GND   BREAKER    1.0
```

Listing 5-18 – Branch Segment Script to Coincide with E_VARRLC1x Subroutine Calls

Node Based (GGIN) Electric Interface

The node-based GGIN interface is used primarily for node to ground operation. It may be used as a node-to-node branch interface, but this is not recommended. The GEQ branch interface described above should be used instead, in order to ensure that present and future EMTDC features are supported. The GGIN interface will slowly be phased-out over time.

The GGIN interface is composed of two parts: A Norton current source and an equivalent branch conductance. These two quantities are represented by the following EMTDC intrinsic variables:

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

Table 5-9 – Intrinsic Variables to Define the Node-Based Electric Interface Branch
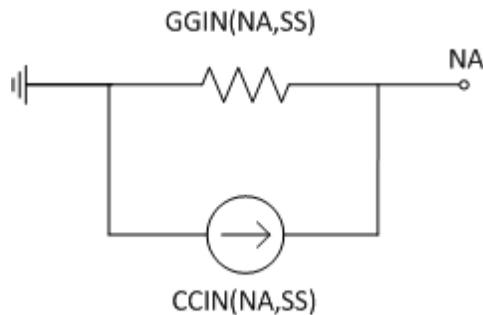
Figure 5-6 - GGIN Electric Interface Branch

The CCIN current source in Figure 5-6 represents the total value of current injected into node NA from ground. It is very important to note that from the perspective of EMTDC, only one CCIN current source or GGIN conductance can exist on a single node. Therefore, the user must ensure that if using a CCIN current source, the value of that particular source be summed with the overall CCIN for that node. This will ensure that if this CCIN source is connected to a node containing other CCIN sources, its value will combine with the existing source values. The same procedure must be performed when providing a GGIN conductance as well. Listing 5-19 below contains a code snippet, which illustrates this concept.

```
!

! Calculate the value of this CCIN source and GGIN conductance:

!

    THIS_CCIN = CURR

    THIS_GGIN = 1.0 / R

!

! Combine this current and conductance with existing CCIN and GGIN

! (may or may not exist).

!

    CCIN(NA,SS) = CCIN(NA,SS) + THIS_CCIN

    GGIN(NA,SS) = GGIN(NA,SS) + THIS_GGIN

!
```

Listing 5-19 – Code Snippet Illustrating Support for Multiple CCIN and GGIN at a Single Node

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.

Enabling CCIN
In order to use the CCIN current source, it must first be enabled at time zero. This is accomplished through the following intrinsic variable:

Variable Name	Type	Description
ENABCCIN(NN,SS)	LOGICAL	Indicates whether or not an CCIN source at node NN is enabled. Set as .TRUE. to enable control of CCIN.

Table 5-10 – Intrinsic Variable to Enable a CCIN Source

CCIN as a Compensating Current Source
The CCIN current source can be used 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 as a compensating current source, as discussed in the section Switching and Non-Linear Elements in the chapter Electric Network Solution.

EXAMPLE 5-6:

The following example code shows how to use CCIN and GGIN to represent a simple inductor.

```fortran
!

    SUBROUTINE U_BGN_SIMPLE_L(SS,NA)

!

!  Common block and module include files:

!

    INCLUDE 'nd.h'

    INCLUDE 's0.h'

    INCLUDE 's1.h'

    INCLUDE 'rtconfig.h'

!

!  Argument List:

!

    INTEGER NA          ! EMTDC node number

    INTEGER SS          ! EMTDC subsystem number

!

!  Local Variables:

!

    INTEGER MY_NRTCF

!

! Set local pointer values and increment:

!

    MY_NRTCF = NRTCF

    NRTCF    = NRTCF + 1

!

! Initial Branch Definition (t = 0.0):

!

    ENABCCIN(NA,SS) = .TRUE.

    RTCF(MY_NRTCF)  = DELT / (2.0*0.001)

!

    RETURN
```

```
        END

!
```

Listing 5-20 – Construction of a CCIN Current Injection within BEGIN Subroutine

```
!

    SUBROUTINE U_DYN_SIMPLE_L(SS,NA)

!

! Common block and module include files:

!

    INCLUDE 'nd.h'

    INCLUDE 's0.h'

    INCLUDE 's1.h'

    INCLUDE 'emtstor.h'

    INCLUDE 'rtconfig.h'

!

! Argument List:

!

    INTEGER NA          ! EMTDC node number

    INTEGER SS          ! EMTDC subsystem number

!

! Local Variables:

!

    REAL    CURR

    INTEGER MY_NSTORF, MY_NRTCF

!

! Set local pointer values and increment:

!

    MY_NSTORF = NSTORF

    MY_NRTCF  = NRTCF

    NSTORF    = NSTORF + 1
```

```
        NRTCF    = NRTCF  + 1

!

! Calculate new history current:

!

     CURR = RTCF(MY_NRTCF)*(-VDC(NA,SS)) + STORF(MY_NSTORF)

!

     STORF(MY_NSTORF) = CURR + RTCF(MY_NRTCF)*(-VDC(NA,SS))

!

! Define CCIN and GGIN:

!

     CCIN(NA,SS) = CCIN(NA,SS) + STORF(MY_NSTORF)

     GGIN(NA,SS) = GGIN(NA,SS) + RTCF(MY_NRTCF)

!

     RETURN

     END

!
```

Listing 5-21 – Construction of a CCIN Current Injection within DSDYN/DSOUT Subroutine

---

Compensating Current Source Injections

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 in parallel with a conductance. Each time step the current source magnitude im(t) is calculated, based on the branch voltage and the current from the previous time step. The result is injected back into the system, affecting nodal voltages. In the next time step, these modified voltages affect the calculation of current injections, and so on.

There are instances however, where a current source may be utilized without a corresponding conductance – to model a non-linearity for example. Care must be exercised when representing electric models in this manner: Due to the finite calculation step, the current source injection im(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) should be placed between the interface nodes in order to ensure that finite impedance is always present.

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 5-7.
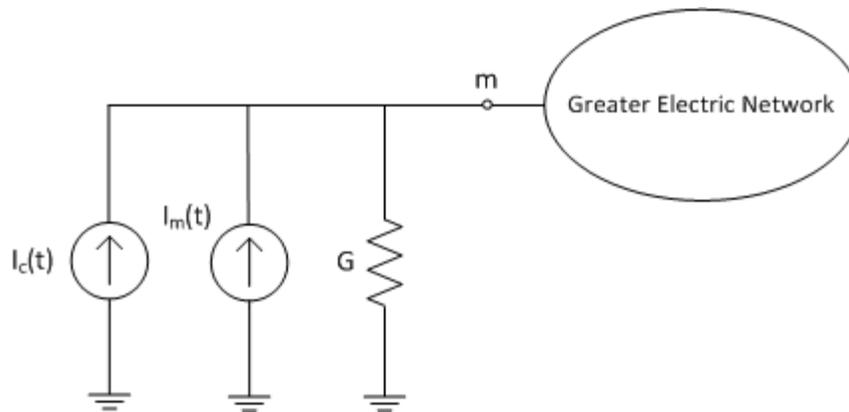
Figure 5-7 – General Interface to the EMTDC Electric Network with Compensating Current Source

Where,

$$i_c(t) = v_m(t - \Delta t) \cdot G \qquad (5\text{-}3)$$

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.

More Custom Model Examples

A picture is worth a thousand words – or at least in this case, an example is worth a thousand words. The following examples attempt to reveal some of the most important aspects of model development in EMTDC. Each example includes additional discussion as situations arise. If you remain unsure about a certain topic, and cannot find it here, please contact the PSCAD Support Desk (support@pscad.com) for help.

Simple Integrator Model
An integrator component is a purely system dynamics model. In other words, at no point does it attempt to interface, nor control the electric network. It operates purely on control signals; modifying an input signal to produce an output.

EXAMPLE 5-7:

To illustrate the above concept, consider the following simple integrator circuit constructed in PSCAD:
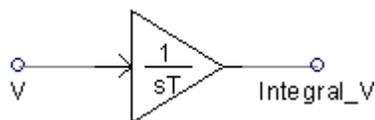
Figure 5-8 - Simple Integrator in PSCAD

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

$$V = N \cdot \frac{d\varphi}{dt}$$

(5-4)

Which leads to,

$$\varphi(t) = \varphi(t - \Delta t) + \frac{1}{N} \cdot \int_{t-\Delta t}^{t} v(t) \cdot dt$$

(5-5)

In other words, the measured electric network voltage *V*, is 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 can then be used as an input control for an electric element, such as a current source. The following example code is for a simple integrator function, written in Fortran:

```fortran
!
      SUBROUTINE U_DYN_MYINTGL(IN,OUT,LIMITS,ULIMIT,LLIMIT)
!
! Purpose  - Integration of a real signal
! Language - Fortran 90
! Date    -
! Author   -
!
! Include Files
!
      INCLUDE 'nd.h'
      INCLUDE 'emtstor.h'
      INCLUDE 'rtconfig.h'
      INCLUDE 's1.h'
      INCLUDE 'emtconst.h'
!
! Variable Declarations
!
      REAL   IN, OUT, ULIMIT, LLIMIT
      INTEGER LIMITS, MY_STORF, MY_RTCF
```

```
      REAL    INOLD, OUTOLD
!
! Copy and increment pointers
!
    MY_STORF = NSTORF
    MY_RTCF  = NRTCF
    NSTORF   = NSTORF + 2
    NRTCF    = NRTCF + 1
!
! Main Program
!
    OUT = 0.5*(IN + STORF(MY_STORF))*DELT + STORF(MY_STORF + 1)
!
! Initial output at time zero
!
    IF ( TIMEZERO )  OUT = RTCF(MY_RTCF)
!
! Limit the output if requested
!
    IF ( LIMITS .EQ. 1 ) THEN
      IF ( OUT .GE. ULIMIT ) THEN
        OUT = ULIMIT
      ELSEIF ( OUT .LE. LLIMIT ) THEN
        OUT = LLIMIT
      ENDIF
    ENDIF
!
! Store data for next time step
!
    STORF(MY_STORF)    = IN
```

```
        STORF(MY_STORF + 1) = OUT

!

    RETURN

    END

!
```

Listing 5-22 – Example Code Defining a Simple Integrator

Something similar to the following script needs to be added to the component definition itself, to ensure that the model subroutine *U_DYN_MYINTGL* is called. Note that a #BEGIN directive has been added to define the initial output quantity for time zero.

```
#BEGIN

    RTCF(NRTCF) = $Out0

    NRTCF      = NRTCF + 1

#ENDBEGIN

!

#SUBROUTINE U_DYN_MYINTGL 'Integrator Model'

#STORAGE REAL:2 RTCF:1

    CALL U_DYN_MYINTGL($In,$Out,$Limits,$ULimit,$LLimit)

!
```

Listing 5-23 – Calling Simple Integrator Subroutine from within the Component Definition

Chapter 6: 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 ⏀-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], [2] and [3]. For our purpose, consider the two mutually coupled windings as shown below:
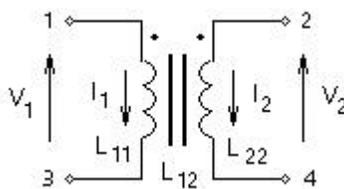
Figure 6-1 - Two Mutually Coupled Windings

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:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} L_{11} & L_{12} \\ L_{12} & L_{22} \end{bmatrix} \cdot \frac{d}{dt} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \tag{6-1}$$

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

$$\frac{d}{dt}\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \frac{1}{\Delta} \cdot \begin{bmatrix} L_{22} & -L_{12} \\ -L_{12} & L_{11} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{6-2}$$

Where,

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

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

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:

$$\frac{E_1}{E_2} = a \tag{6-3}$$

And

$$\frac{I_2}{I_1} = a \tag{6-4}$$

Making use of this turns-ratio 'a' Equation 6-1 may be rewritten as:

$$\begin{bmatrix} V_1 \\ a \cdot V_2 \end{bmatrix} = \begin{bmatrix} L_{11} & a \cdot L_{12} \\ a \cdot L_{12} & a^2 \cdot L_{22} \end{bmatrix} \cdot \frac{d}{dt}\begin{bmatrix} I_1 \\ I_2 / a \end{bmatrix} \tag{6-5}$$
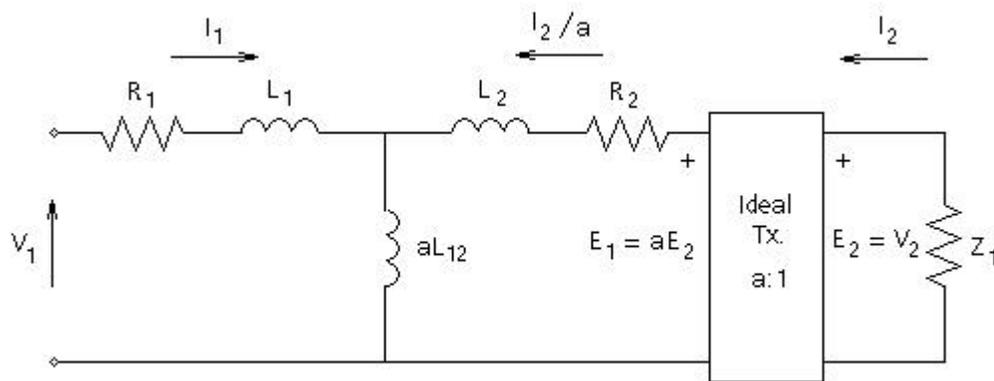
Figure 6-2 - Equivalent Circuit of Two Mutually Coupled Windings

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 6-1 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:

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

Where,

$\omega =$ 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:

$$L_{xy} = \frac{V_x}{\omega \cdot I_y} \qquad (6\text{-}7)$$

Transformer data is often not available in this format. Most often, an equivalent circuit, as shown in Figure 6-2, 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 $aL_{12} \gg L_2$). 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 6-1:

Parameter	Description	Value
T_{MVA}	Transformer single-phase MVA	100 MVA

f	Base frequency	60 Hz
X_1	Leakage reactance	0.1 pu
NLL	No load losses	0.0 pu
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 6-1 - Transformer Data

If we ignore the resistances in Figure 6-1, we can obtain the (approximate) value for $L_1 + L_2$, from the short circuit test, as:

$$L_1 + L_2 = \frac{0.1 \cdot Z_{base1}}{\omega_{base1}} = 26.525 \, mH \tag{6-8}$$

Where,

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

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

$$a = \frac{100 \cdot kV}{50 \cdot kV} = 2.0 \tag{6-9}$$

We also have for the primary and secondary base currents:

$$I_{base1} = \frac{(100 \cdot MVA)}{(100 \cdot kV)} = 1.0 \, kA \qquad I_{base2} = 2.0 \, kA \tag{6-10}$$

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

$$I_{m1} = 1\% \cdot I_{base1} \qquad I_{m2} = 1\% \cdot I_{base2} \tag{6-11}$$

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

$$\frac{I_{m1}}{I_{m2}} \cdot \frac{V_{base2}}{V_{base1}} = \frac{1}{a^2} \cdot \frac{(L_2 + a \cdot L_{12})}{(L_1 + a \cdot L_{12})} \tag{6-12}$$

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,

$$\frac{I_{m1}}{I_{m2}} \cdot \frac{V_{base2}}{V_{base1}} = \frac{1}{a^2} \qquad \text{(6-13)}$$

Then,

$$L_1 = L_2 \qquad \text{(6-14)}$$

By combining Equations 6-8 and 6-14 we obtain $L_1 = L_2 = 13.263$ mH and from Equation 6-12 we obtain $aL_{12} = 26.5119$ H. The values for the parameters in Equation 6-1 are then obtained as:

$$L_{11} = L_1 + a \cdot L_{12} = \quad 26.5252 \text{ H} \qquad \text{(6-15)}$$

$$L_{22} = \frac{L_2 + a \cdot L_{12}}{a^2} = \quad 6.6313 \text{ H} \qquad \text{(6-16)}$$

$$L_{12} = \quad 13.2560 \text{ H} \qquad \text{(6-17)}$$

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 6-5.

An excessively small magnetising 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 6-3. 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 6-18; 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):

$$\frac{d}{dt} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \frac{1}{L} \cdot \begin{bmatrix} 1 & -a \\ -a & a^2 \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \qquad \text{(6-18)}$$

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
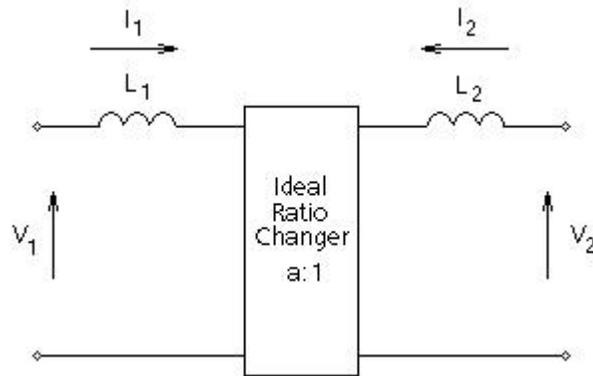
Figure 6-3 - 'Ideal' Transformer Equivalent Circuit

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 magnetising current), when more than two windings are coupled on the same core.

EXAMPLE 6-1

Consider a three winding, 40 MVA transformer with zero magnetising 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 6-4. 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.
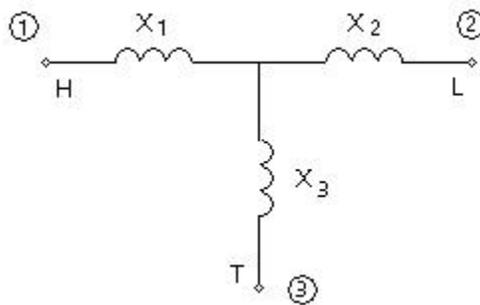
Figure 7-6 - Positive Sequence Leakage Reactance Equivalent Circuit

Where,

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

and,

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

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

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

(6-19)

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

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

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:

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

(6-20)

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. Many transformer studies however, do require core saturation to be adequately modeled.

Core Saturation

An exciting current is required for voltage transformation by a transformer. The excitation characteristic of a transformer is determined completely by the core design, and winding design affects the characteristic only in so far as it determines the flux density in the core. Transformers are constrained in their performance by the magnetic flux limitations of the core. Since ferromagnetic materials cannot support infinite magnetic flux densities and they tend to *saturate* at a certain level. The nonlinear flux-current characteristic of a transformer iron core is illustrated below.
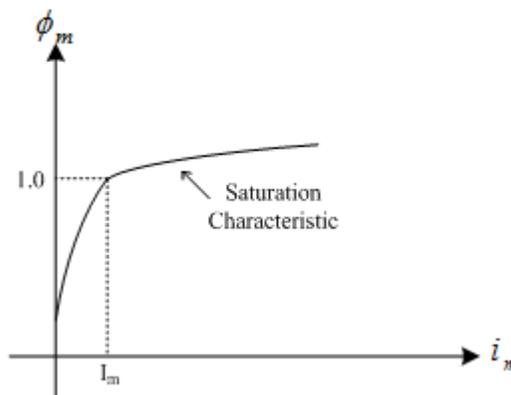
Figure 6-5 - Flux-Current Characteristic of a Saturable Core Transformer

The transformer may experience saturation in any of the following situations:

- When a transformer's primary winding is overloaded from excessive applied voltage, the core flux may reach saturation levels during peak moments of the sinusoidal cycle. If this happens, the voltage induced in the secondary winding will no longer match the wave-shape as the voltage powering the primary coil. In other words, the overloaded transformer will distort the waveshape from primary to secondary windings, creating harmonics in the secondary winding's output.

- When a transformer is initially connected to an AC source, a transient current of up to 10 to 50 times larger than the rated transformer current can flow for several cycles. For large transformers, inrush current can last for several seconds. This happens because the transformer will always possess residual flux, and when the transformer is re-energized, the incoming flux will add to that already existing, which will cause the transformer to move into saturation. Then, only the resistance of the primary side windings and the power line itself limit the current.

Modeling Saturation Across a Specified Winding

In the classical transformer models, core 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 [2]. 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 below:
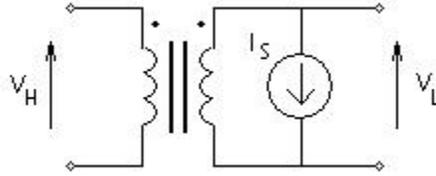
Figure 6-6 - Compensating Current Source for Core Saturation

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

$$\lambda_S(t) = L_S(t) \cdot I_S(t)$$

(6-21)

The non-linear nature of Equation 6-21 is displayed in Figure 6-6, 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 λ_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 λ_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, λ_k, λ_M, and I_M are known. Current I_S can be defined as:

$$I_S = \frac{\sqrt{(\lambda_S - \lambda_K)^2 + 4 \cdot D \cdot L_A} + \lambda_S - \lambda_K}{2 \cdot L_A} - \frac{D}{\lambda_K}$$

(6-22)

Where,

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

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

$$B = \frac{L_A \cdot L_M - \lambda_M}{\lambda_K}$$

$$C = I_M \cdot (L_A \cdot I_M - \lambda_M + \lambda_K)$$

$$\lambda_M = \frac{V_M}{2\pi \cdot f}$$

$$\lambda_K = K \cdot \lambda_M$$
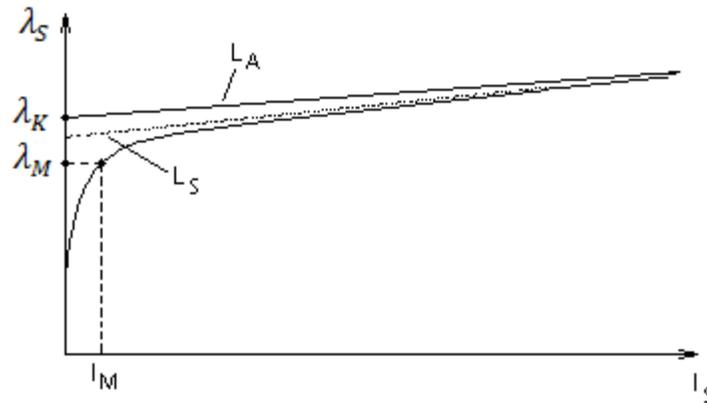
Figure 6-6 - Core Saturation Characteristic of the Classical Transformer

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

$$\lambda_S(t) = \int V_L(t) \cdot dt \qquad \text{(6-23)}$$

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 favorably 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 [4].

In order to appreciate the saturation process described above, Figure 6-7 summarizes the use of Equations 6-22 and 6-23.
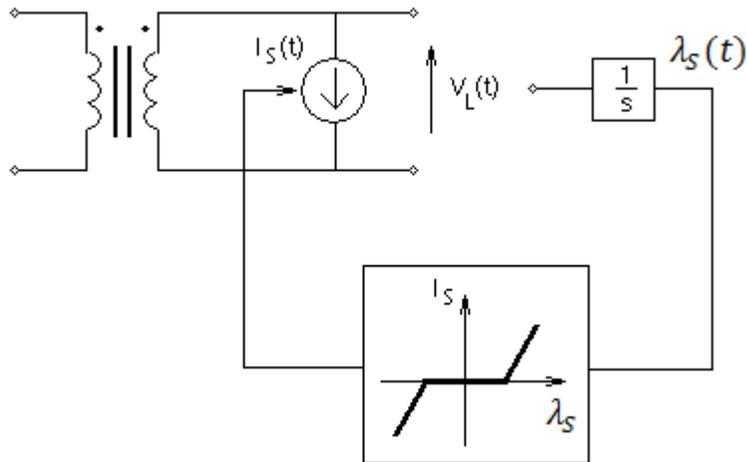
Figure 6-7 - Formulation of Saturation for a Mutually Coupled Winding (TSAT21)

Modeling Saturation Across the Magnetizing Inductance (Middle)

The approach of injecting the current source at the transformer terminal, as discussed above, usually provides acceptable simulation results for many large power system studies. However, the impact of voltage drops across the leakage impedance is significant during the saturation of transformer, since the magnetizing inductance will drop dramatically during the saturation. This phenomenon is not properly included in these models because the saturation current source location has been moved from its true location across the magnetizing impedance, to the new location across transformer terminal. To demonstrate the inaccurate results, two simulation cases that show the transformer inrush current in no load condition have been carried out. The transformer parameters are listed in Table 6-2.

Description	Value
Transformer single-phase MVA	100 MVA
Base frequency	60 Hz
Leakage reactance	0.1 pu
Air core reactance	0.2 pu
Copper losses	0.01 pu
No load losses	0.01 pu
Primary winding voltage (RMS)	230 kV
Secondary winding voltage (RMS)	115 kV
Knee voltage	1.17 pu
Magnetizing current	0.4 %

Table 6-2 - Transformer Data

In the first simulation case the current source is placed at the primary winding terminal and in the second one it is at the secondary winding terminal. The simulation results of transformer magnetizing current and secondary voltage for both scenarios are shown in Figure 6-8a and 6-8b respectively.
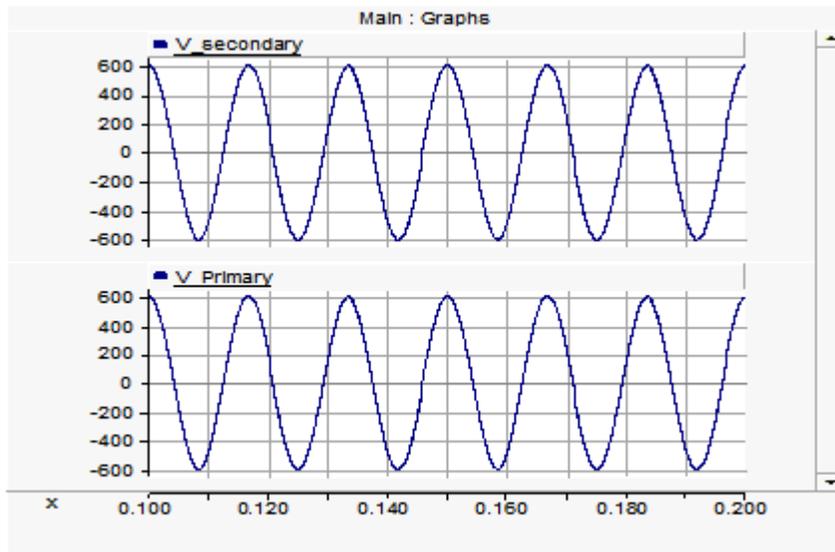
Figure 6-8a - Primary and Secondary Voltages when the Saturation Current is Injected at the Primary Winding
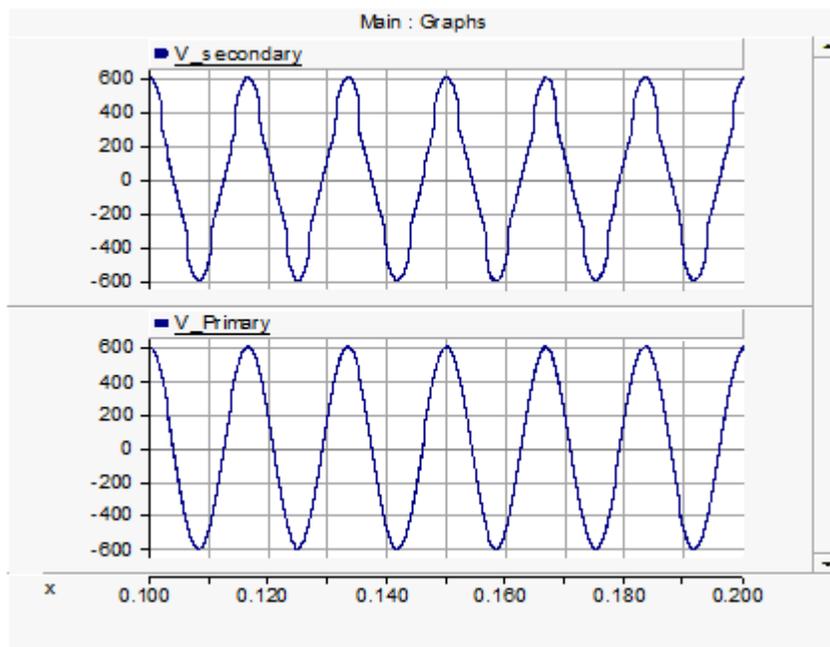
Figure 6-8b - Primary and Secondary Voltages when the Saturation Current is Injected at the Secondary Winding

It can be observed above that when the saturation current is injected at the terminals where the source is connected, the secondary voltage waveform is not distorted since other elements modeling the transformer are linear. On the other hand, injecting the saturation current at the secondary winding increases the distortion in its voltage waveform. This phenomena is also true for three and four winding transformer models.
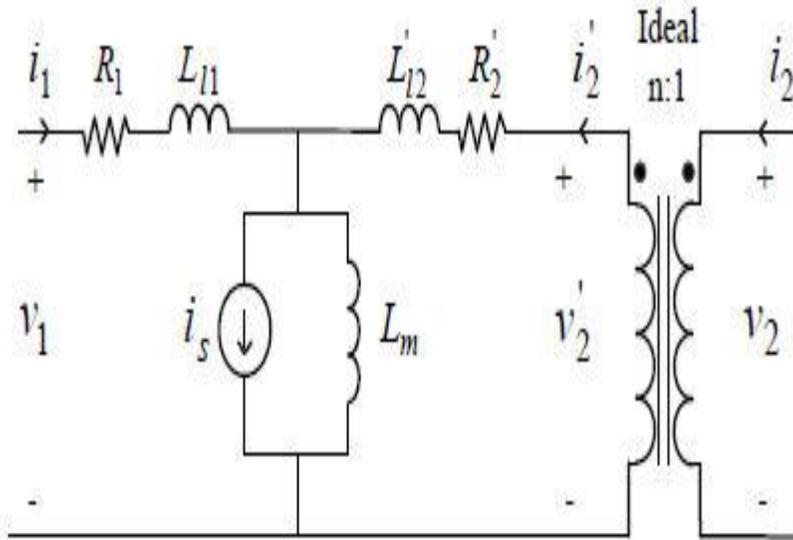
Figure 6-9 - Transformer T-Model Including Saturation Current at the Middle

Figure 6-9 shows the true location of magnetizing branch in the transformer T model. The magnetizing current can be divided into two components; the linear component which is represented as a linear inductance defined as:

$$L_m = \frac{\lambda_M}{I_M}$$

(6-24)

where λM and IM are the peak magnetizing flux and current at rated voltage as shown in Figure 6-6, and the nonlinear component that is the difference between the total magnetizing current and the linear current as illustrated in Figure 6-10. The nonlinear component or saturation current can be calculated as follows:

$$i_s(\lambda_m) = i_m(\lambda_m) - \frac{\lambda_m}{L_m}$$

(6-25)
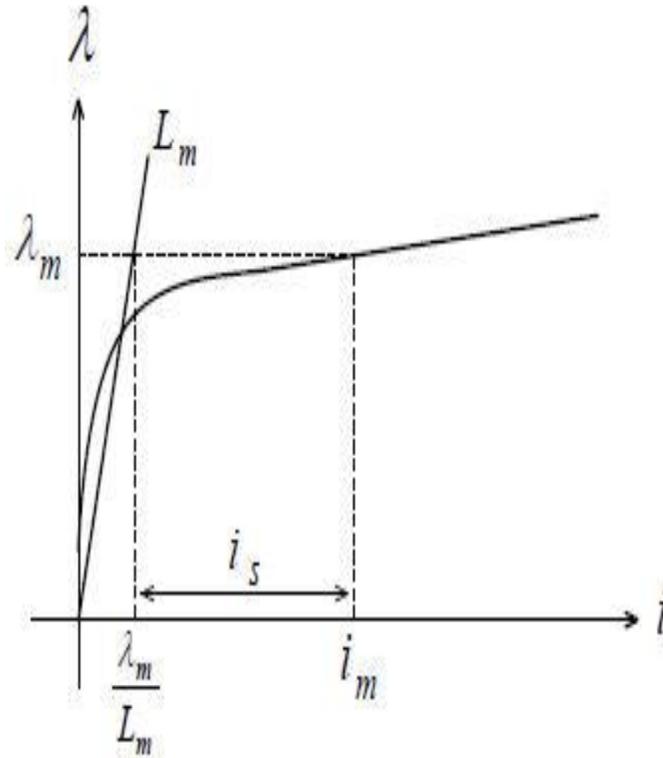
Figure 6-10 - Magnetizing Current Consists of Linear Current and Saturation Current

In Equation 6-25, i_m is the total magnetizing current and λ_m is the magnetizing flux calculated by:

$$\lambda_m = \int \left(v_1 - R_1 i_1 - L_{l1} \frac{di_1}{dt}\right) dt = \int (v_1 - R_1 i_1) dt - L_{l1} i_1 \qquad (6\text{-}26)$$

In this method of saturation modeling, the location of the saturation branch should be maintained at the terminals of the transformer in order to retain the computational efficiency. Therefore the true saturation current is partitioned between the terminal current sources by appropriate ratios as shown in Figure 6-11. To do so, the injected currents at both terminals should be accurately computed to be mathematically equivalent to injection at the magnetizing branch so that the new model behaves exactly the same as the transformer T model.
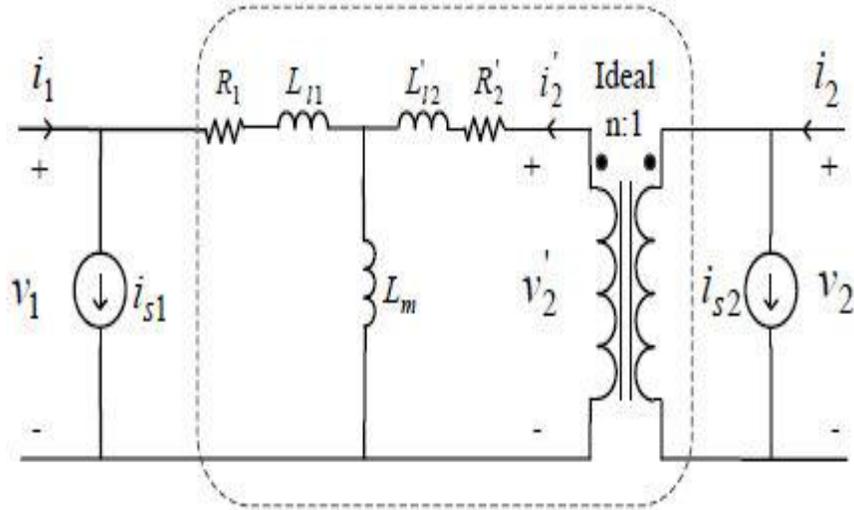
Figure 6-11 - Mathematically Equivalent Model of Transformer Electric Circuit

The simulation results of the secondary voltage of the same case mentioned above in Figure 6-8, using the new model, along with the previous simulations for the placement of the saturation at either ends is presented in Figure 6-12.
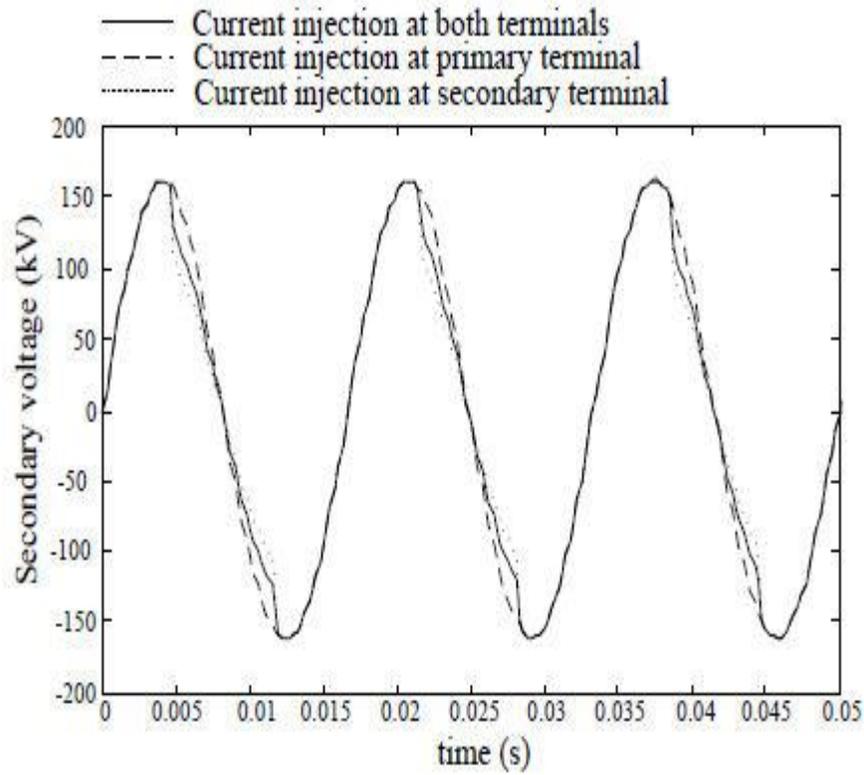
Figure 6-12 - Mathematically Equivalent Model of Transformer Electric Circuit

For more details on this saturation method, please see transformer reference [12].

More on Air Core Reactance

The air core reactance L_A in Figure 6-6 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 is 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 6-6, this would be:

$$\lambda_K = K \cdot \lambda_M \tag{6-24}$$

Where,

$$1.15 < K < 1.25$$

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

$$\lambda_M = \frac{V_M}{4.44 \cdot f} \tag{6-25}$$

Where,

f = Rated frequency in Hertz

Equations 6-21 to 6-25 are an approximate means of defining transformer saturation and form the basis upon which the EMTDC subroutine TSAT21 is constructed.

More on Classical Transformers

The following topics discuss some attributes of the classical transformers in more detail.

On-Line Tap Changer

All transformer components are equipped with an on-line tap changer. When the tap changer is enabled, a diagonal line will appear on the component graphic, labelled **Tap**.

A 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.

EXAMPLE 6-1:

Let us assume for simplicity, that a transformer is configured as Y-Y, with winding voltages rated at 10 kV primary and 100 kV secondary, and a tap changer is placed on the primary winding. The turns-ratio for this transformer is of course 1:10.

An on-line tap input of 1.0 corresponds to a 100% tap (i.e. no tap adjustment). If the tap input value is adjusted to say 1.05, then the turns-ratio would become 1.05:10 or 1:9.52381.

It is possible to have a continuous change of tap, but this would require a re-ordering of the network solution every time step. It is more practical to change tap settings in steps, either through manual adjustment, using a slider or rotary switch, or from a controller with appropriate delays and steps built in.

Adjusting Saturation Properties

Saturation is represented in the classical transformer models with a compensating current source injection across the selected winding, based on a combination of measured winding voltage and component input parameters.

Classic Saturation Method

Properties of the classic core saturation method are described in this section.
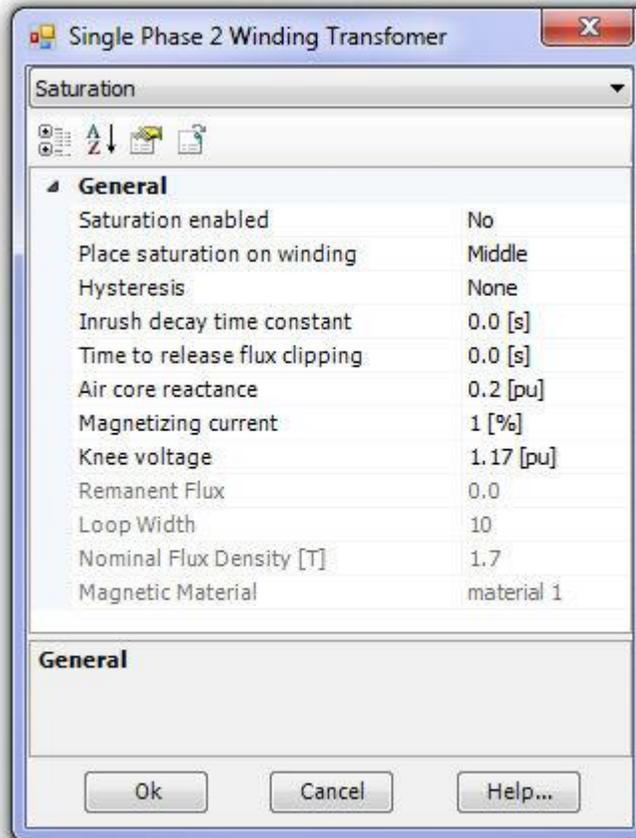
The transformer input parameters directly involved in the adjustment of the classic core saturation characteristic are:

- Air Core Reactance

- Magnetizing Current

- Knee Voltage

These three parameters are shown, as they appear in the Saturation section of the component parameters in the figure above. Air core reactance, knee voltage and magnetizing current provide three degrees of freedom to the shaping of the continuous, core characteristic (knee curve) shown below.
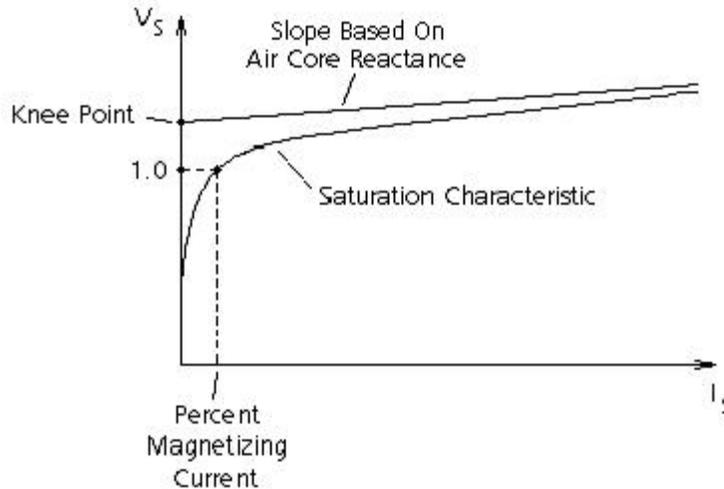
Figure 6-8 -- Typical Classic Core Saturation Characteristic

Air Core Reactance
Adjustment of the air core reactance input affects the slope of the asymptotic line indicated in the above figure.

Knee Voltage
Adjustment of the Knee Voltage vertically shifts the Y-intercept of the air core reactance asymptotic line in the above figure.

Magnetizing Current
Adjustment of the magnetizing current determines the horizontal position, along the V_S = 1.0 pu voltage line, of the effective knee point. That is, an increasing value of magnetizing current will tend to make the saturation characteristic less sharp.

Transformer Magnetic Hysteresis

At present there a few developed theories used to model the magnetization of a transformer core accurately. These models are generally based on the observed behaviour of real materials, and include magnetic hysteresis effects. Two of the best-known and widely used empirical methods to model hysteresis are the *Jiles-Atherton* and Preisach models, which both provide an accurate representation of the hysteresis loop. Jiles and Atherton used a soft magnetic material for their investigation and modeled the saturation characteristic using a modified Langevin function. Preisach theory is based on the fact that a ferromagnetic material is made up of dipoles that can be represented as having a magnetic characteristic with two statistically distributed parameters, a coercivity and a field due to neighboring dipoles [7], [8], [9].

The primary drawback of the above methods is that the core material data (required for magnetization modeling) is not usually available. However, a mathematical technique may be employed to approximate the hysteresis loop behaviour for recoil loops and renascence effects. This technique is general and sufficiently accurate to model the hysteresis phenomena, by using a lookup and interpolation method [10]. A set of curves representing the major (or parent) loop of the Ø-I characteristic are generated. These curves, along with a second set of curves representing flux differences between the two paths of the parent loop, are then used to determine the current to be injected for any given value of flux. In this technique, the minor loop travel, sustained and initial flux remanence, and realistic inrush phenomena are all considered [11].

Basic Hysteresis Model (Based On Loop Width)
In EMTDC, a current source is used to represent the saturation of the core in classical transformers (see Core Saturation), where the saturation current is calculated using Equation 6-22. This equation represents a Ø-I characteristic curve which is asymptotic to the specified air core reactance line, and to the vertical axis, passing through a point defined by the 1 pu. Magnetizing current and flux are shown in Figure 6-6.

In order to generate the major loop of the Ø-I characteristic, first a single-valued saturation curve, between plus and minus values of the specified knee point limits, is obtained as explained earlier. Then, this curve is shifted according to the specified hysteresis loop width in the positive direction of current. The other side of the major loop is then generated as the negative of the negative function described by the positively offset curve. Figure 6-8 represents the major (parent) loop of the Ø-I characteristic [10].
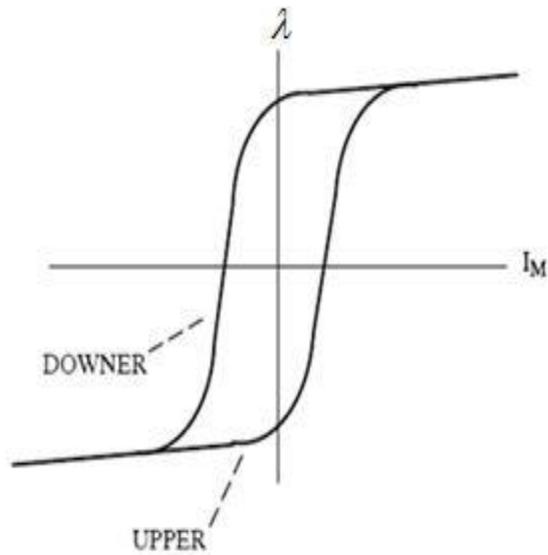
Figure 6-9 - Hysteresis Parent Loop

This maximum loop is the trajectory only when the inductor element is driven from saturation in one direction to saturation in the opposite direction. The UPPER curve or $g(\lambda)$ is for increasing values of flux linkage and the DOWNER curve or $f(\lambda)$ is for decreasing values of the flux linkage. The points of confluence for the two curves making up the major loop are at the defined saturation knee point values. The characteristic is assumed to be single valued beyond the points of confluence (Figure 6-9).
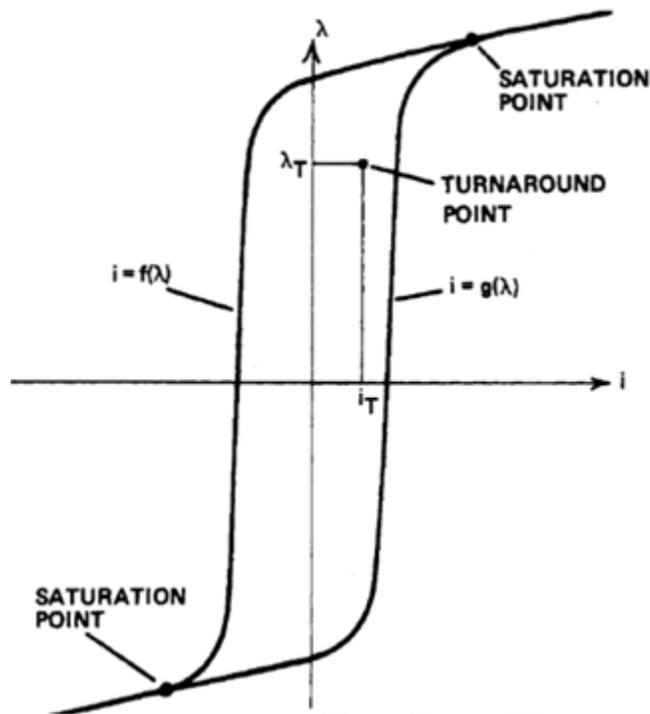
Figure 6-10 - A Typical Hysteresis Loop Including Saturation

A reversal or turnaround point occurs when the flux linkage derivative $d\lambda/dt$ changes sign. The coordinate (λ_T, I_T), represents a generalized flux linkage turnaround point of operation in the non-saturated region, as shown in Figure 6-9. If a flux turn around occurs at any point in the non-saturated region, the path of travel must change along a trajectory, which tends towards the confluence point and travels along the opposite parent curve. In the saturated region, if a flux turn around point occurs, the travel path does not change since only one path exists there. At a turnaround point the trajectory leading from any present point on the correct new point is defined by two quantities, namely [10];

1. The difference in flux between the present point and the point directly above/below the parent curve, toward which it must travel.
2. The point of confluence, toward which it must travel.
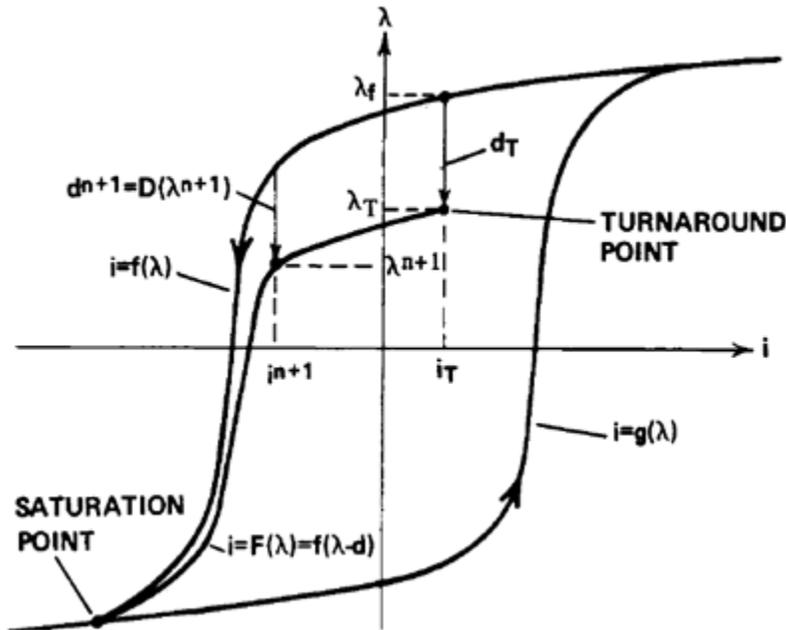
Figure 6-11 - A Mapping of $f(\lambda)$ into $F(\lambda)$ Showing a Typical Turnaround Point and a Generalized Point on the Downer Curve for Discrete Time t^{n+l}.

These trajectory curves can be generated in the non-saturated region by translating a variable displacement $d = D(\lambda)$ from the f or g curves. The simplest relationship for D, and the one chosen for this example of family pair mapping, is to make the displacement d vary linearly from:

$$D(\lambda_T) = d_T \cdot D(\lambda_{SAT})$$

(6-25)

A typical downer curve F resulting from a variable displacement d from curve f is shown in Figure 6-10.

Jiles-Atherton Hysteresis Model

The Jiles-Atherton theory describes the relationship between M, the magnetic moment, and H, the magnetic field intensity [9]. Electrical engineers are more accustomed to dealing with a B–H relationship in the form of a hysteresis loop. Conversion from a B-H and an M-H loop is straightforward using the relationship:

$$B = \mu_0 \cdot (H + M)$$

(6-26)

The magnetization relationship between B and H is replaced by the anhysteretic magnetization curve between H_e and M:

$$M_{an} = M_s \cdot f \cdot (H_e)$$

(6-27)

where $H_e = H + \alpha M$ and α is a parameter that represents inter-domain coupling. M_s is the saturation magnetization. Jiles and Atherton used a modified Langevin function to produce the familiar sigmoid type curve for $f(H_e)$ as expressed below:

$$M_{an} = M_s \left(\frac{1}{\tanh(\frac{H_e}{A})} - \frac{1}{(\frac{H_e}{A})} \right)$$

(6-28)

By differentiating Equation 6-28 with respect to H_e,

$$\frac{dM_{an}}{dH_e} = \frac{M_s}{A} \left(1 - \frac{1}{\tanh^2(\frac{H_e}{A})} - \frac{1}{(\frac{H_e}{A})^2} \right)$$

(6-29)

The Jiles-Atherton theory leads to two components in M as follows:

$$M = M_{irr} + M_{rev}$$

(6-30)

The first component, M_{irr}, is due to pinning of the magnetic domains by discontinuities in the material structure and can be likened to a friction effect. The second term, M_{rev}, is due to domain wall bending in an elastic manner. The authors then build on these fundamental relationships to arrive at the final set of equations to be solved to construct M-H_e and B-H loops [9]:

$$\frac{dM}{dH} = \frac{c\frac{dM_{an}}{dH_e} + \frac{M_{an} - M}{\frac{\delta k}{\mu_0} - \frac{\alpha(M_{an} - M)}{1 - c}}}{1 - \alpha c \frac{dM_{an}}{dM_e}}$$

(6-31)

A, α, c and k are constants for the material being used and δ takes the value 1 or -1 based on the sign of dH/dt.

This equation takes a modified form when $(M_{an} - M)\beta$ is negative as described in [9]:

$$\frac{dM}{dH} = \frac{c\,\dfrac{dM_{an}}{dH_e}}{1 - \alpha c\,\dfrac{dM_{an}}{dH_e}}$$

(6-32)

In power transformers at every time step, the changes in magnetizing current should be calculated based on the changes in magnetizing flux. The magnetizing flux is also calculated by integrating the magnetizing branch voltage.

$$V \xrightarrow{\int} \lambda \xrightarrow[\substack{\lambda = N\phi \\ \phi = BA}]{\times \frac{1}{NA}} B \xrightarrow{\text{JA Model}} H \xrightarrow[Hl = NI]{\times \frac{l}{N}} I_{mag}$$
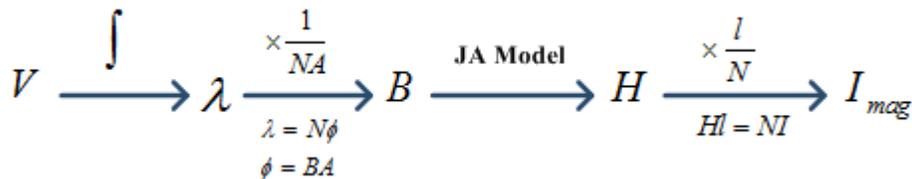
Now, these equations are iteratively solved for the unknown values of ΔH and ΔM in each time step.

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 [5].

Basic Modeling Approach
Consider the three-phase, three-limb transformer shown below:
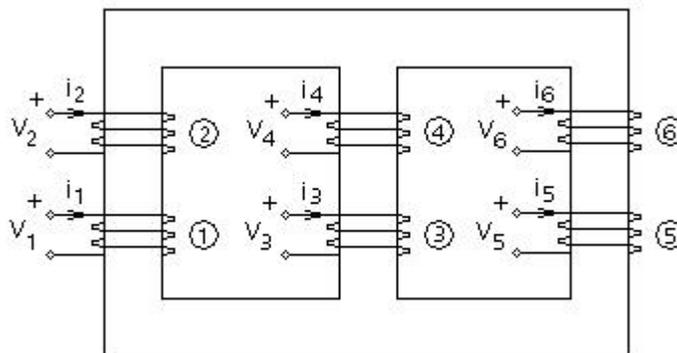
Figure 6-12 - Schematic Representation of the Three-Phase, Three Limb Transformer (2 Windings per Phase)

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

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_6 \end{bmatrix} = \begin{bmatrix} R_1 & 0 & \cdots & 0 \\ 0 & R_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_6 \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_6 \end{bmatrix} + \begin{bmatrix} L_1 & M_{12} & \cdots & M_{16} \\ M_{21} & L_2 & \cdots & M_{26} \\ \vdots & \vdots & \ddots & \vdots \\ M_{61} & M_{62} & \cdots & L_6 \end{bmatrix} \cdot \frac{d}{dt} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_6 \end{bmatrix} \qquad (6\text{-}33)$$

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 6-33 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 below:
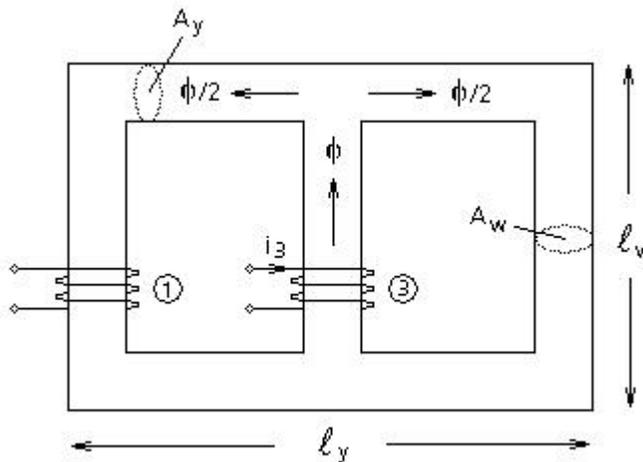
Figure 6-13 - Transformer Core Flux Due To Winding 3 Current

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:

$$N_3 \cdot i_3 = \Re_w \cdot \phi + \Re_y \cdot \frac{\phi}{2} + \Re_w \cdot \frac{\phi}{2} \qquad (6\text{-}34)$$

Where,

$\mathfrak{R}_w$ =   Reluctance of the winding limb

$\mathfrak{R}_y$ =   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,

$$N_3 \cdot i_3 = \left(\frac{3}{P_w} + \frac{1}{P_y} \right) \cdot \frac{\phi}{2} \tag{6-35}$$

The self-inductance of winding 3 is defined as:

$$L_3 = \frac{N_3}{i_3} \cdot \phi = \frac{2 \cdot N_3^2}{\left(\dfrac{3}{P_w} + \dfrac{1}{P_y} \right)} \tag{6-36}$$

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

$$M_{i3} = \frac{N_i}{i_3} \cdot \frac{\phi}{2} = \frac{N_i \cdot N_3}{\left(\dfrac{3}{P_w} + \dfrac{1}{P_y} \right)} \tag{6-37}$$

Similar expressions can be derived for the other inductance terms in Equation 6-33.

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 below:
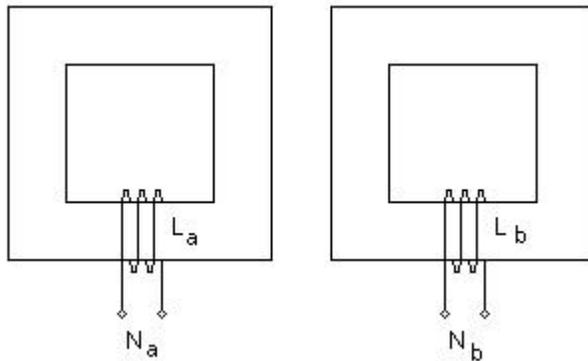
Figure 6-14 - Simple Inductors with Cores

The inductances L_a and L_b take the form:

$$L_j = N_j^2 \cdot P_j \qquad (6\text{-}38)$$

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:

$$N_i = V_i \qquad (6\text{-}39)$$

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:

$$r_A = \frac{A_y}{A_w} \qquad (6\text{-}40)$$

$$r_L = \frac{\ell_y}{\ell_w} \qquad (6\text{-}41)$$

Thus substituting these values into Equation 6-36 for the magnetic configuration in Figure 6-13 (and assuming that $\mu_w = \mu_y$, we get:

$$L_3 = \frac{2 \cdot P_w \cdot N_3{}^2}{3 + \dfrac{r_L}{r_A}}$$

(6-42)

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

$$V_{Rated} = I_{OC} \cdot L_3 \cdot \omega_0$$

(6-43)

Where,

V_{Rated} =   Rated voltage

I_{OC} =   Open-circuit test current in winding 3

ω_0 =   Rated frequency

By combining Equations 6-42 and 6-43, the permeance of a winding limb can be found as:

$$P_w = \frac{V_{Rated} \cdot \left(3 + \dfrac{r_L}{r_A}\right)}{2 \cdot N_3{}^2 \cdot I_{OC} \cdot \omega_0}$$

(6-44)

Assign the rated voltage value to the number of turns,

$$P_w = \frac{3 + \dfrac{r_L}{r_A}}{2 \cdot V_{Rated} \cdot I_{OC} \cdot \omega_0}$$

(6-45)

Similarly, the permeance of the yoke can be given as

$$P_w = \frac{1 + 3 \cdot \dfrac{r_A}{r_L}}{2 \cdot V_{Rated} \cdot I_{OC} \cdot \omega_0}$$

(6-46)

Deriving 5-Limb Core Aspect Ratios

The following illustrates the derivation of 5-limb core aspect ratios:
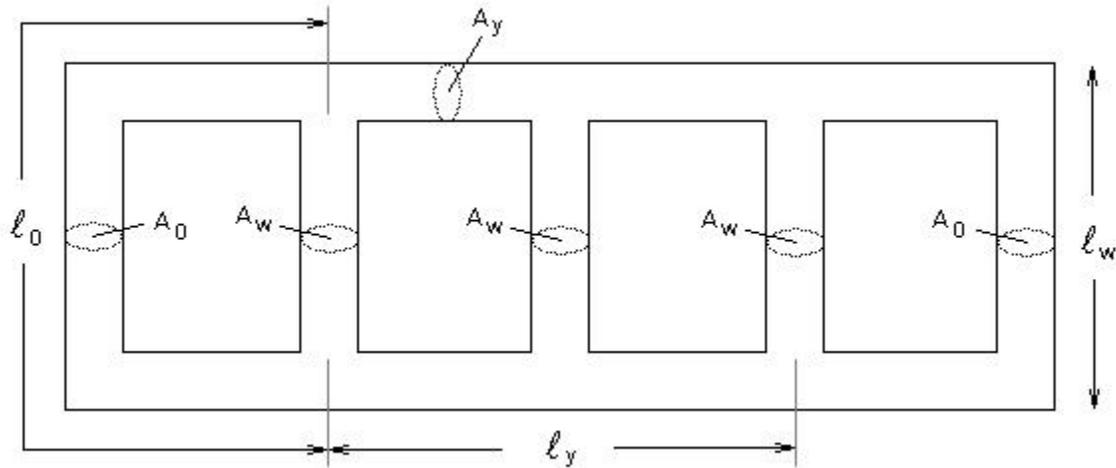
Figure 6-15 - A 5-Limb Transformer Core

Yoke/Winding Limb Ratios:

$$length = \frac{\ell_y}{\ell_w} \qquad (6\text{-}47)$$

$$Area = \frac{A_y}{A_w} \qquad (6\text{-}48)$$

Yoke/Outer Limb Ratios:

$$length = \frac{\ell_y}{\ell_0} \qquad (6\text{-}49)$$

$$Area = \frac{A_y}{A_0} \qquad (6\text{-}50)$$

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 piece-wise 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 [5] and [6]. Interested users are encouraged to refer these publications.

Modeling Autotransformers

The PSCAD Master Library offers limited 1-Phase Auto Transformer and 3-Phase Star-Star Auto Transformer models. If preferred, this section describes how to build an autotransformer directly by utilizing the existing transformer models with suitable tap changers.

Figure 6-16 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.
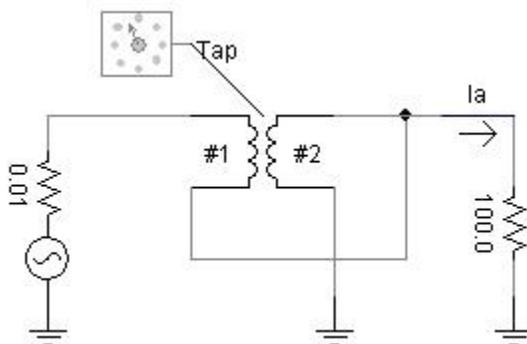
Figure 6-16 - Autotransformer Equivalent Circuit in PSCAD

Some important notes on the use of the configuration in Figure 6-16, 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:

$$\frac{1}{L} \cdot \begin{bmatrix} 1 & -\dfrac{a}{T} \\ -\dfrac{a}{T} & \dfrac{a^2}{T^2} \end{bmatrix} \tag{6-51}$$

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 magnetising current is considered.

Transformer Related 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. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. Rezaei-Zare, A., Iravani, R., Sanaye-Pasand, M., Mohseni, H., Farhangi, S., An Accurate Hysteresis Model for Ferro-resonance Analysis of a Transformer, IEEE Transactions on Power Delivery, Vol. 23, No. 3, pp.1448-1456, July 2008.

8. Chandrasena, W., McLaren, P.G., Annakkage, U.D., Jayasinghe, R.P., An improved low frequency transformer model for use in GIC studies, IEEE Transactions on Power Delivery, Vol. 19, No. 2, pp. 643 - 651, 13-17, April 2004.

9. Annakkage, U. D., McLaren, P. G., Dirks, E., Jayasinghe, R. P., Parker, A. D., A Current Transformer Model Based on the Jiles–Atherton Theory of Ferromagnetic Hysteresis, IEEE Transaction on Power Delivery, Vol. 15, No. 1, January 2000.

10. Talukdar, S.N., Bailey, J.R., Hysteresis models for system studies, IEEE Transactions on Power Apparatus and Systems, Vol.95, No.4, pp. 1429- 1434, July 1976.

11. RTDS® Manual, RTDS Technologies Inc., 100-150 Innovation Drive, Winnipeg, Mantioba, R3T 2E1, Canada.

12. Samimi, M., Gole, A. M., Jayasinghe, R. P., Improvement of Transformer Saturation Modeling for Electromagnetic Transient Programs, IPST 2013, Vancouver, Canada, July 18-20, 2013.

Chapter 7: 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 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:

$$\begin{bmatrix} U_d \\ U_q \\ U_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \cos(\theta - 120°) & \cos(\theta - 240°) \\ \sin(\theta) & \sin(\theta - 120°) & \sin(\theta - 240°) \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \cdot \begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} \tag{7-1}$$

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].
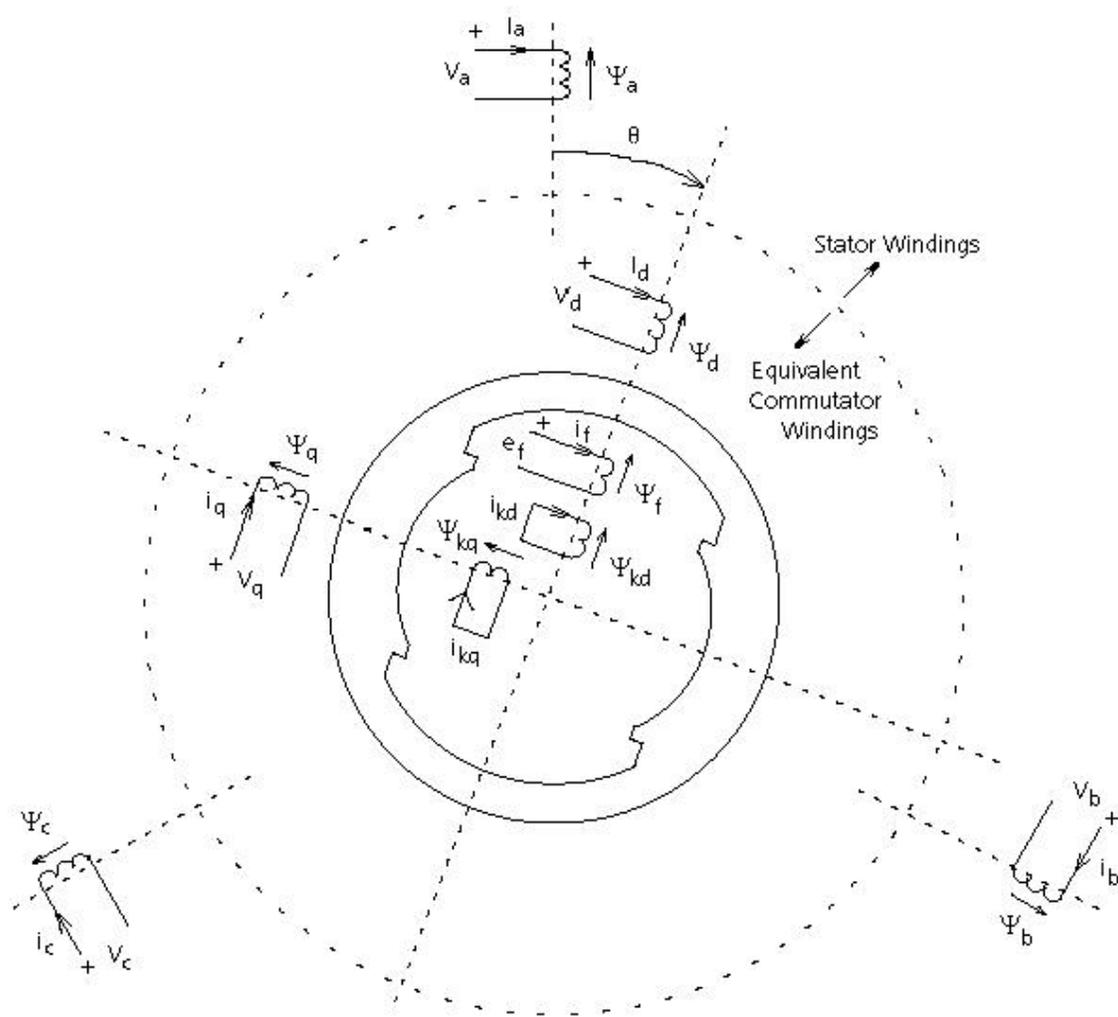
Figure 7-1 - Conceptual Diagram of the Three-Phase and dq Windings

NOTE: All quantities shown in Figure 7-1 are in pu

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 7-2. Figure 7-3 illustrates the flux paths associated with various d-axis inductances:
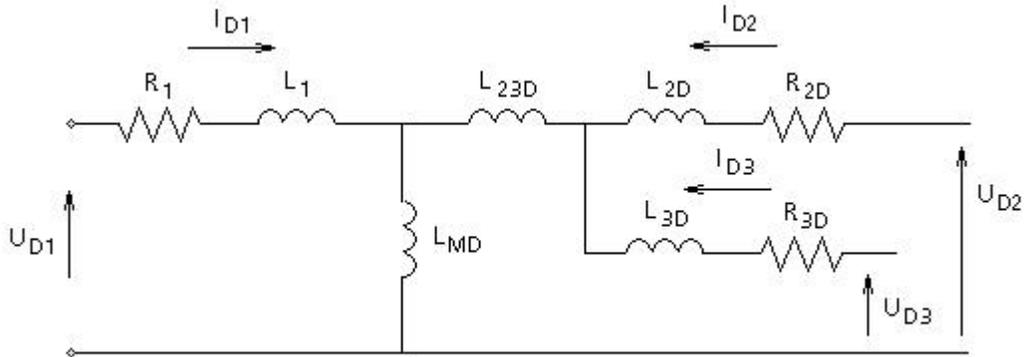
Figure 7-2 - d-axis Equivalent Circuit
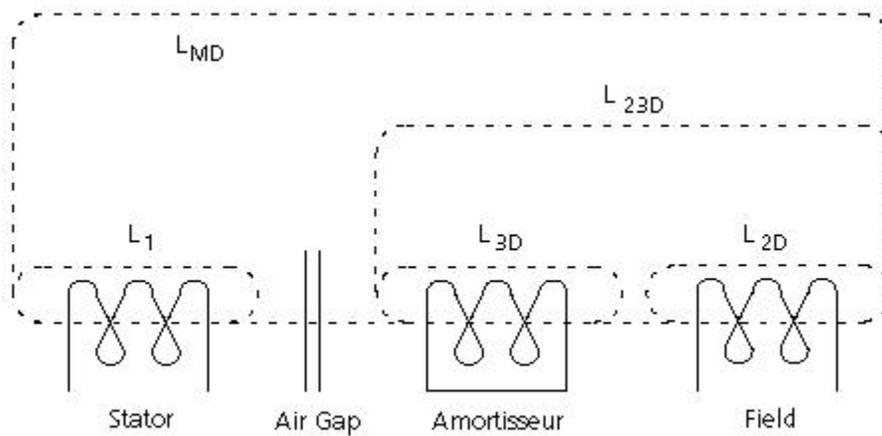
Figure 7-3 - Flux Paths Associated with Various d-axis Inductances

Referring to Figures 7-2 and 7-3, the following equations can be derived:

$$\begin{bmatrix} U_{D1} - v \cdot \Psi_q - R_1 \cdot i_{D1} \\ U_{D2} - R_{2D} \cdot i_{D2} \\ U_{D3} - R_{3D} \cdot i_{D3} \end{bmatrix} = L_D \cdot \frac{d}{dt} \begin{bmatrix} i_{D1} \\ i_{D2} \\ i_{D3} \end{bmatrix} \tag{7-2}$$

Where,

$$L_D = \begin{bmatrix} L_{MD} + L_1 & L_{MD} & L_{MD} \\ L_{MD} & L_{MD} + L_{23D} + L_{2D} & L_{MD} + L_{23D} \\ L_{MD} & L_{MD} + L_{23D} & L_{MD} + L_{23D} + L_{3D} \end{bmatrix} \tag{7-3}$$

$$\Psi_q = L_1 \cdot i_{Q1} + L_{MQ} \cdot \left(i_{Q1} + i_{Q2} + i_{Q3}\right) \tag{7-4}$$

$$\nu = \frac{d\theta}{dt} \tag{7-5}$$

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

$$\Psi_d = L_1 \cdot i_{D1} + L_{MD} \cdot \left(i_{D1} + i_{D2} + i_{D3}\right) \tag{7-6}$$

Inversion of Equation 7-2 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:

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

$$\frac{d}{dt}\begin{bmatrix} i_{Q1} \\ i_{Q2} \\ i_{Q3} \end{bmatrix} = L_Q^{-1} \cdot \begin{bmatrix} \nu \cdot \Psi_d - R_1 \cdot i_{Q1} \\ -R_{2D} \cdot i_{Q2} \\ -R_{3D} \cdot i_{Q3} \end{bmatrix} + L_Q^{-1} \cdot \begin{bmatrix} U_{Q1} \\ U_{Q2} \\ U_{Q3} \end{bmatrix} \tag{7-8}$$

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

$$T = \Psi_q \cdot i_{D1} - \Psi_d \cdot i_{Q1} \tag{7-9}$$

and the mechanical dynamic equation for motor operation is:

$$\frac{d\nu}{dt} = \frac{T - T_{MECH} - D \cdot \nu}{J} \tag{7-10}$$

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 7-10.

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,

$$i_{MD} = i_{D1} + i_{D2} + i_{D3} \qquad (7\text{-}11)$$

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

Tutorial: Synchronous Machine Short-Circuit Test

This tutorial demonstrates the classical short-circuit test on a Synchronous Machine component. The circuit diagram for this example is shown below:
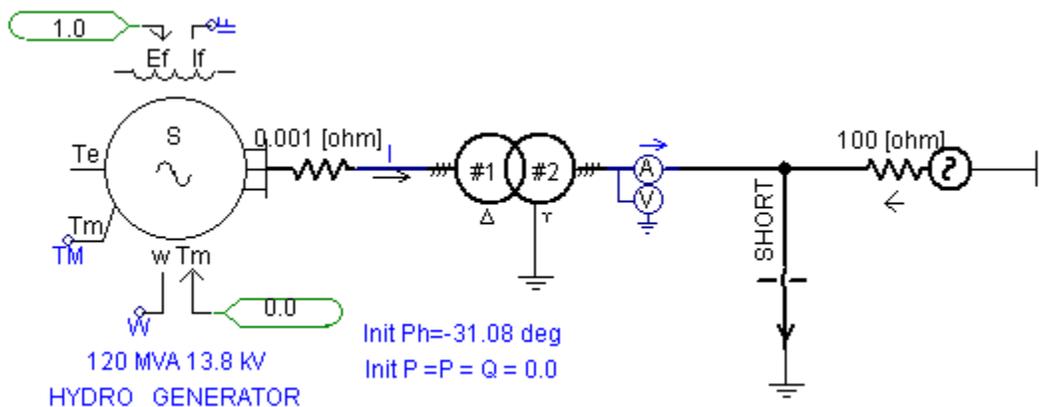
Figure 7-T1: Short-Circuit Test Circuit in PSCAD

The results of this simulation also serve as a validation of the Salient Pole / Round Rotor Synchronous Machine model in PSCAD. The corresponding PSCAD case project is entitled *sync_SCTest.psc* and is available in the *...\Examples\sync_machines* directory within the PSCAD program folder.

Initial Conditions & Simplification
Before proceeding with the short-circuit test, the Synchronous Machine must be running in **steady-state under open-circuit conditions**. The open-circuit condition is achieved by connecting the machine to the source (through a transformer) and adjusting the phase angle and the magnitude of the machine voltage with respect to those of the source voltage, so that the stator current in the machine is zero (negligible) in steady state.

Input Parameters
Voltage magnitude and phase angle of the 'infinite' source (Three-Phase Voltage Source Model 2) is **230.0 kV** and **0.0 degrees**, respectively. The same parameter values are set for the machine at **13.8 kV** and **–31.08 degrees** (includes phase shift by transformer and interface, $\Delta t = 50\ \mu s$) respectively. The field voltage necessary to produce *1.0 pu* terminal voltage on the open-circuit machine is *1.0 pu*. These initial conditions give open-circuit condition for the machine.
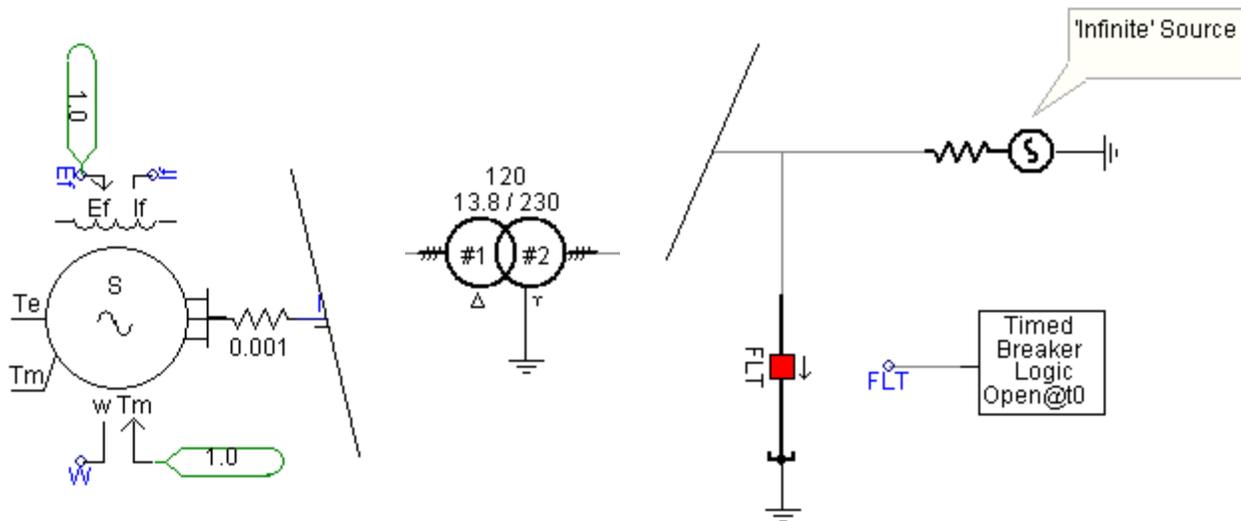
Figure 7-T2: The Synchronous Machine, 'Ideal' Transformer and 'Infinite' Source Components in PSCAD

The Timed Breaker Logic component is set to apply a short-circuit at **0.5056 s**, which is chosen purely for convenience (the A phase current will not have a DC component during the short-circuit test).

The following lists the simplifications and assumptions used in setting-up this test: These simplifications allow us to focus on the machine dynamics better.

Synchronous Machine

- **Mechanical Torque**: The mechanical torque (**Tm**) terminals on the Synchronous Machine component are shorted simply to provide a dummy input to the **Tm** terminal, and serves no other purpose in this example.

- **Prime Mover Dynamics**: The machine is run at constant speed by locking the rotor (**Enab = 0**) at synchronous speed: Thus, there are no prime mover dynamics involved.

- **Exciter Dynamics**: Any exciter dynamics are also eliminated by feeding a constant field voltage (**Ef = 1.0 pu**) to the field winding.

- **Saturation**: Machine saturation is disabled.

The relevant section of the Synchronous Machine component parameters is shown below.
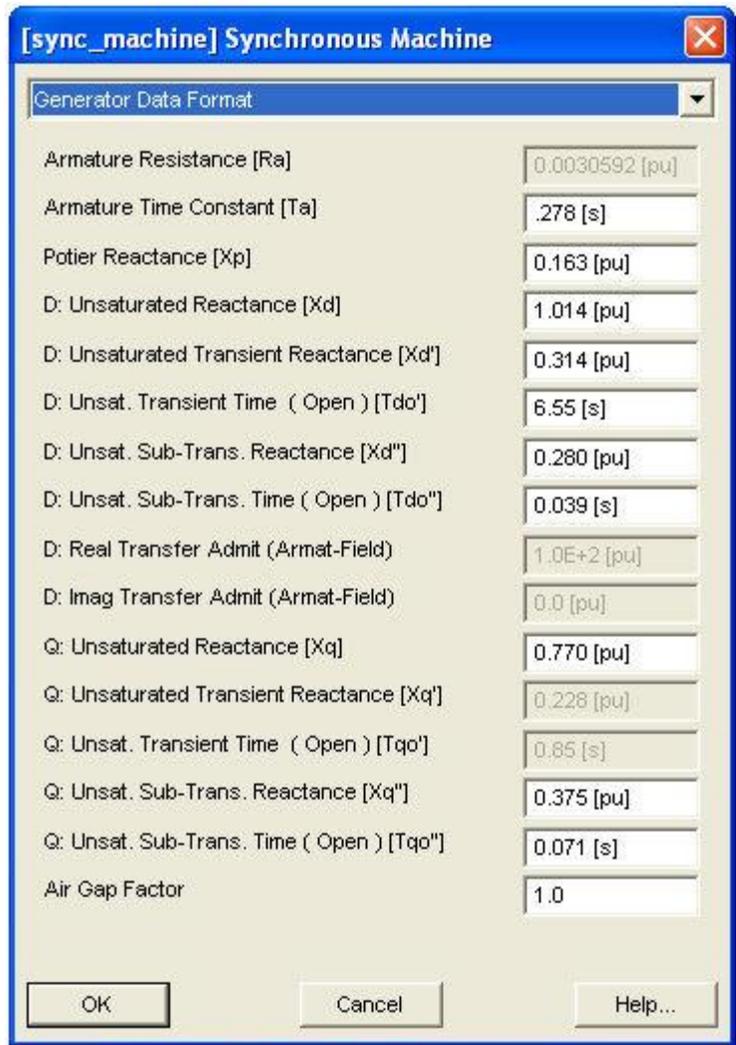
Figure 7-T3: The 13.8 kV, 120 MVA Generator Parameters

Transformer:

- **Leakage Reactance**: The ideal transformer is simply a ratio changer with negligible leakage (**0.005 pu**) reactance.

- **Saturation**: Transformer saturation is disabled.

Simulating the Short-Circuit Test

If not already done so, load the case project entitled ***sync_SCTest.psc*** (in the ***...\Examples\sync_machines*** directory within the PSCAD program folder) and press the run button. Allow the case to run until completion. A short-circuit is applied at **0.5056 s** by the Timed Breaker Logic component. The test results are shown and discussed below:
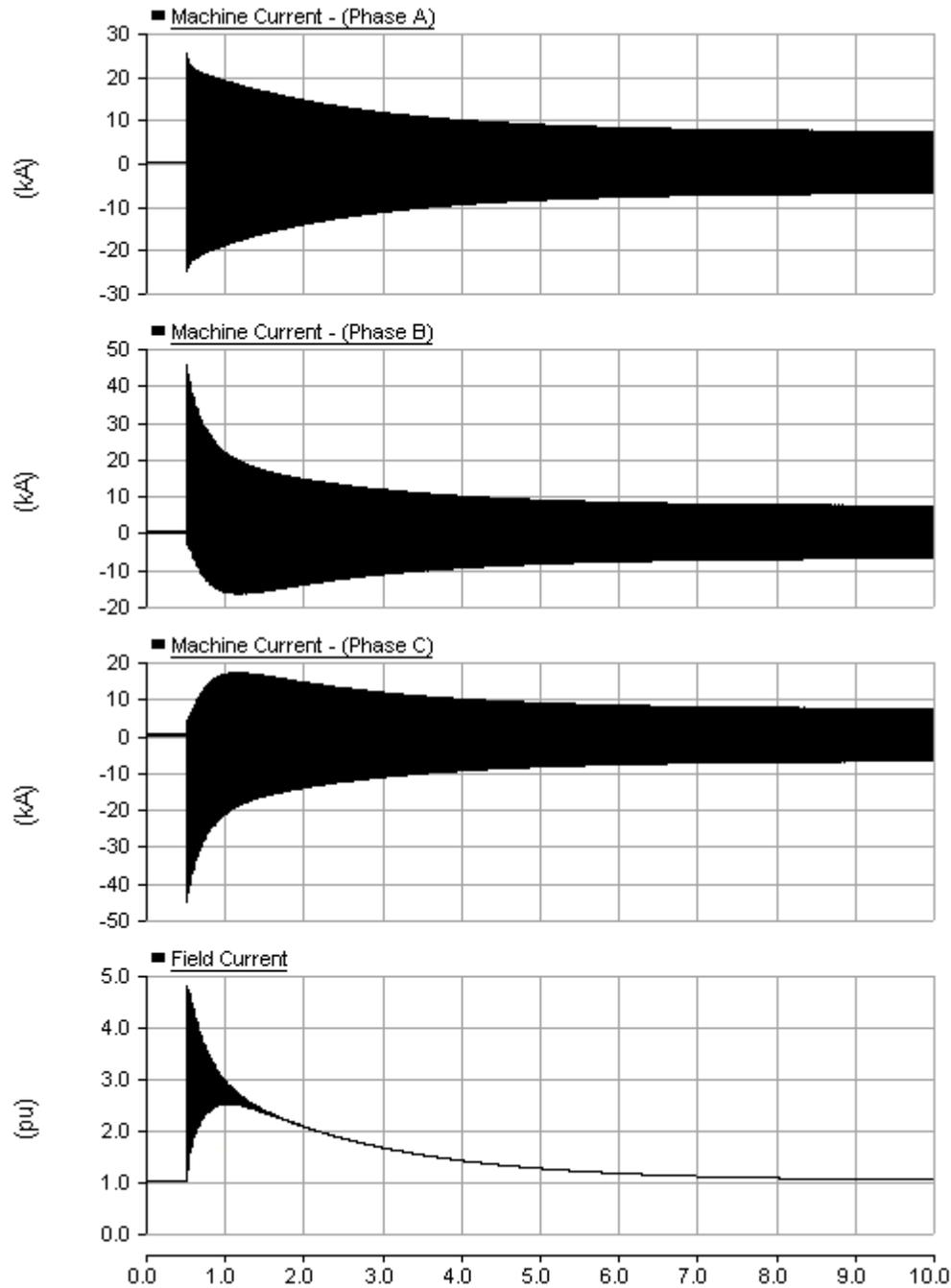
Figure 7-T4: Test Results

Analysing the Results

The following sections discuss the important information that can be derived from the result plots.

Sub-Transient Time Constant

Let us validate the model by comparing the theoretical time constants for the given machine parameters with the time constants demonstrated by the simulation graphs.

The sub-transient component of the short-circuit current should decay with the sub-transient (or damper) time constant (Td'') given by the following equations:

$$Td'' = \left(\frac{Xd''}{Xd'}\right) \cdot Tdo'' \tag{7-T1}$$

$$Td'' = \left(\frac{0.280}{0.314}\right) \cdot 0.039 = 34.7\,ms \tag{7-T2}$$

> **NOTE**: For details on the above, see *C.V. Jones, The Unified Theory of Electrical Machines, Plenum Press, N.Y., 1967, Chapter 20*

Thus, the sub-transient effects will be seen for only about two cycles. This can be seen in an expanded view of the phase A fault current:
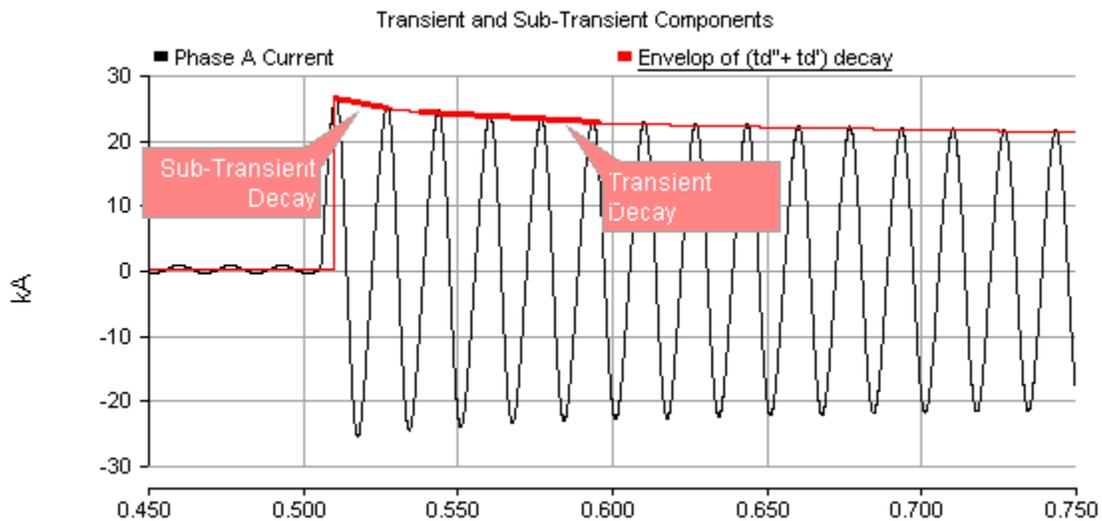
Figure 7-T5: Decay of Sub-Transient and Transient Components of the Phase A Current

Transient Time Constant
The transient component should decay with the transient time constant (Td') given by the following equations:

$$Td' = \left(\frac{Xd'}{Xd}\right) \cdot Tdo' \tag{7-T3}$$

$$Td' = \left(\frac{0.314}{1.014}\right) \cdot 6.55 = 2.03s \tag{7-T4}$$

Field Current Decay Time Constant

The sub-transient time constant is also the time in which it takes the field current to decay to its pre-fault value; that is, if a constant field voltage is applied (as we have done in this case). This can be verified from the field current plot shown below:

Following a time sufficient enough for the sub-transient effects to disappear (but still short enough for the transient component to be unaffected), the magnitude of the field current is given by:

$$i'_{fo} = \left(\frac{Xd'}{Xd''} \right) \cdot i_{fo} \tag{7-T5}$$

$$i'_{fo} = \left(\frac{1.014}{0.314} \right) \cdot 1 = 3.23 \tag{7-T6}$$

The initial DC component of the field current is approximately the mid-point of the first cycle, which agrees with the above calculation at about *3.2 pu*. Assuming a fixed field voltage, the field current will eventually return to its pre-fault, steady-state value. The decay of the field current during the transient period is given by:

$$i'_f = i_{fo} + \left(i'_{fo} - i_{fo} \right) \cdot e^{-t/T'_d} \tag{7-T7}$$

$$i'_f = 1 + \left(3.23 - 1 \right) \cdot e^{-t/T'_d} \tag{7-T8}$$

Thus, after a time period equal to *Td'*, the field current will decay to approximately 37% of its initial value:

$$i'_{f-Td'} = 1 + \left(3.23 - 1 \right) \cdot 0.37 = 1.825 \tag{7-T9}$$

These effects are illustrated in Figure 7-T6.
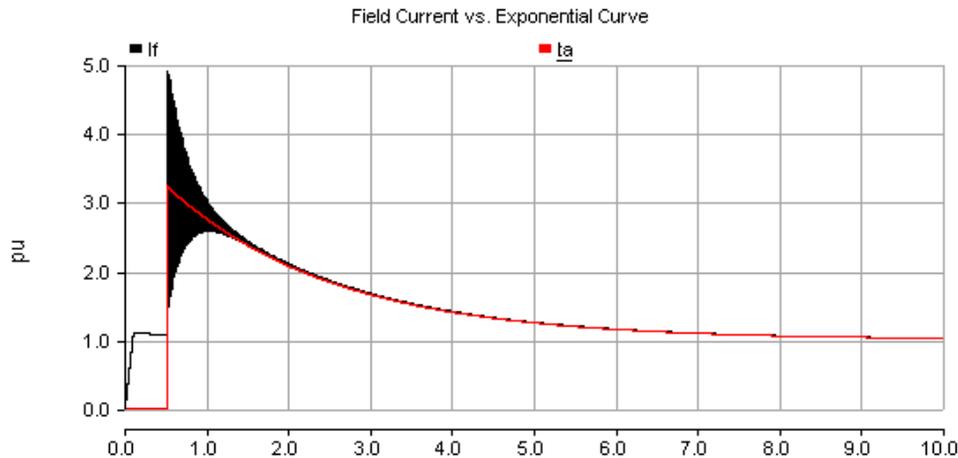
Figure 7-T6: Field Current Response

From Figure 7-T6, it can be seen that the field current reached approximately *1.825 pu* following about *2.0 s* from the fault inception: This agrees with the theoretical calculation of *Td'*. An exponential curve with a time constant of *2.03 s* is superimposed on the field current in Figure 7-T6 to show that the field current indeed decays with this time constant. If the short-circuit is not ideal, but has a resistance (the fault resistance in our case can be considered negligible), this time constant could further be reduced. Moreover, the transient and sub-transient components of current are only different by about *12%*; that is,

$$\frac{I_a{''}}{I_a{'}} = \left(\frac{Xd'}{Xd''}\right) = \left(\frac{0.314}{0.280}\right) = 1.12$$

(7-T10)

With the fast decay rate of $I_a{''}$, this difference is difficult to observe. It is apparent from the results of Figure 7-T5 that the sub-transient and transient currents are almost of the same magnitude.

Another calculation that can be verified is the ratio of the sub-transient component $I_a{''}$ to the steady-state fault current I_a. Note that with constant field excitation we have,

$$\frac{I_a{''}}{I} = \left(\frac{Xd}{Xd''}\right) = \left(\frac{1.014}{0.280}\right) = 3.6$$

(7-T11)

From Figure 7-T1, we obtain a value of *3.57 (= 25.0/7.0)*, which is close to the value calculated using the above equation. The steady-state I_a of *7.0 kA* also exactly matches the calculated value as seen in Figure 7-T7.
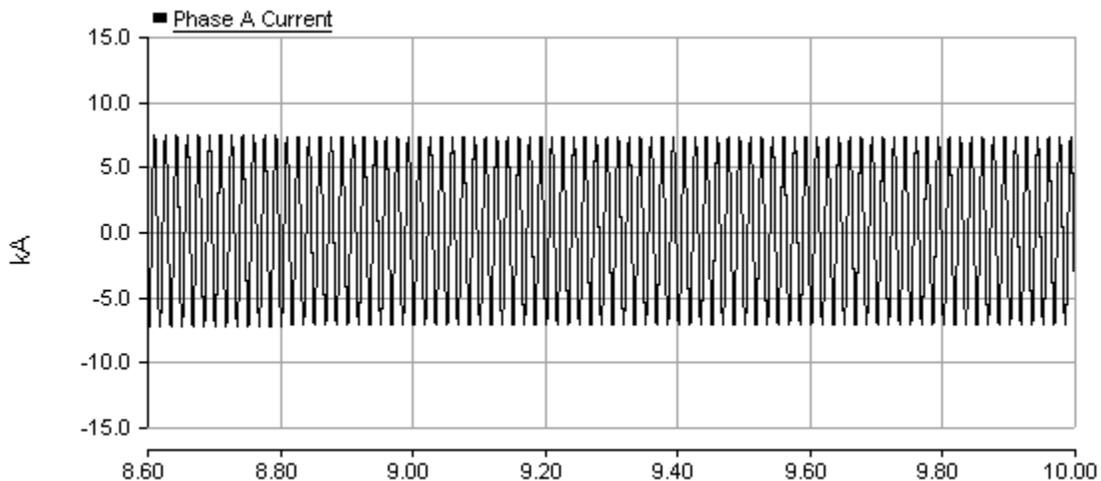
Figure 7-T7: Steady-State Fault Current
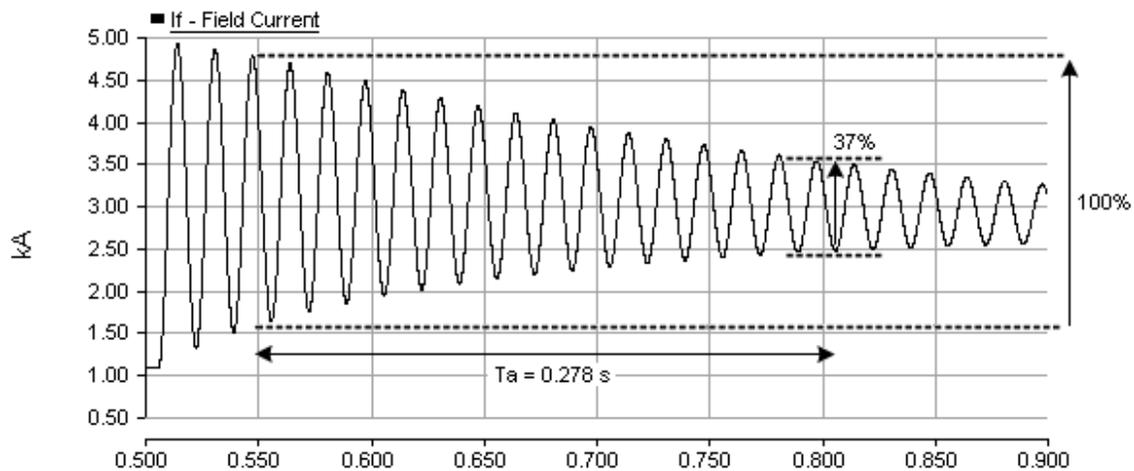
Figure 7-T8: Decay of the Fundamental Frequency Component in the Field Current

The armature time constant *Ta* (*0.278 s*) is the decay time constant of the fundamental frequency component of the field current *I$_f$*: On the stator side this is the time constant, at which the DC component and the second harmonic component of stator current decay. This time constant is estimated in Figure 8-T8. Initial peak-to-peak magnitude of the fundamental frequency component, after the sub-transient influence disappears (a couple of cycles), is about *3.0 pu*. The fundamental frequency component reached *37%* of this value (*1.1 pu*) in about *0.270 s*. This value closely agrees with the given value for parameter *Ta*.

> **NOTE**: This tutorial is based on a popular document section that first appeared in the PSCAD V2 manual. It has been brought up to date, and added to this manual at the request of many users.

	Dr. Om Nayak	Nayak Corporation
Original Authors:		
	Dr. Ani Gole	University of Manitoba

| **Updated By:** | Dr. Dharshana Muthumuni | Manitoba Hydro International Ltd. |

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 7-1 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,

$$i_M = \sqrt{i_{MD12}^2 + i_{MQ12}^2} \tag{7-12}$$

while saturation of the leakage inductances L$_1$ and L$_{23}$ is based on the stator line current,

$$i_L = \sqrt{i_{D12}^2 + i_{Q12}^2} \tag{7-13}$$

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 7-4. 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.
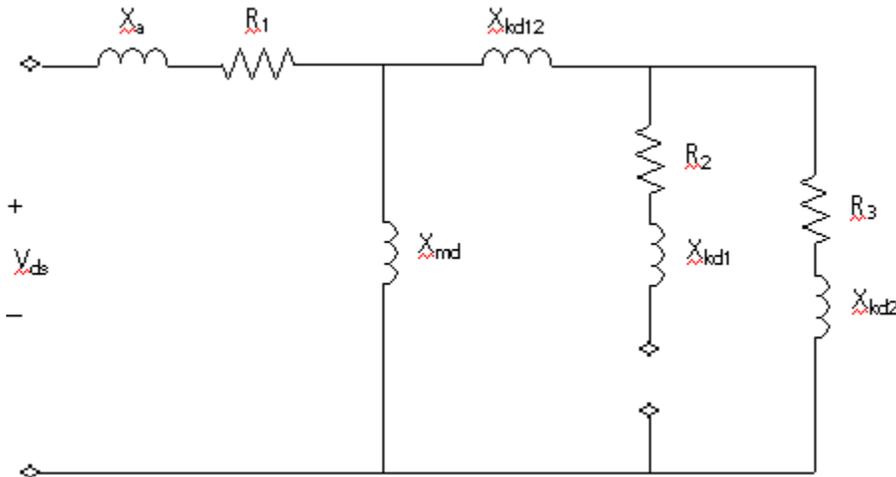
Figure 7-4 - d-axis Equivalent Circuit

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 7-1. The inverse transform is:

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 1 \\ \cos(\theta - 120°) & \sin(\theta - 120°) & 1 \\ \cos(\theta - 240°) & \sin(\theta - 240°) & 1 \end{bmatrix} \cdot \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} \tag{7-14}$$

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:

$$P_0 = 3 \cdot V_{a0} \cdot i_{a0} \tag{7-15}$$

The associated bases are:

ω_0 =   Rated frequency in radians

t_0 = $\dfrac{1}{\omega_0}$

L =   X

p = $\dfrac{p}{\omega_0}$

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

ν_0 = $\dfrac{\omega_0}{PolePairs}$ Base mechanical speed

θ_m = $\dfrac{\theta}{PolePairs}$ Mechanical angle

The impedances are given in per unit for both machines. The input voltages in are divided by V_{a0} and the incremental time ⏲t 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_{MDO} 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 7-5 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.
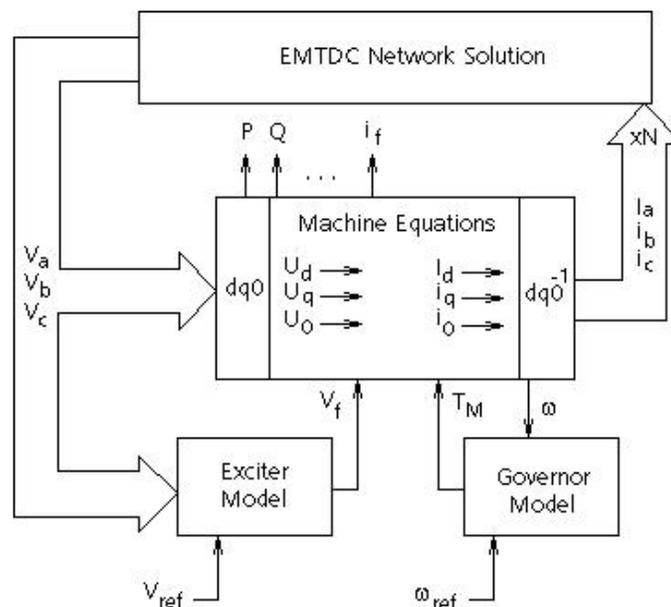
Figure 7-5 - Synchronous Machine Model Interface to EMTDC

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 7-6. 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.
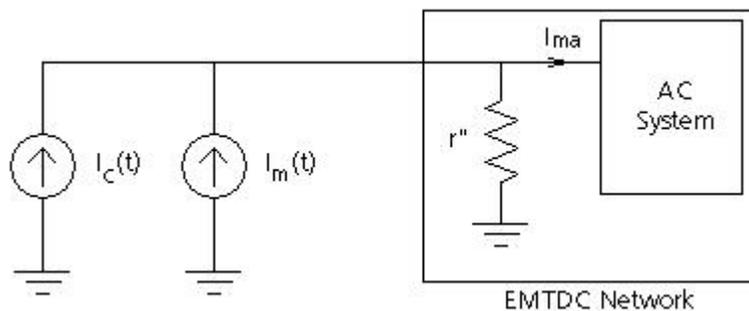
Figure 7-6 - Interface with Terminating Resistance

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

$$I_{ma}(t) = I_m(t) + \frac{V_c(t - \Delta t) - V_c(t)}{r''}$$ (7-16)

r'' is usually quite large, due to the ⎕t in its denominator. Also, for a small time step V(t – ⎕t) = V(t), and thus $I_{ma}(t) = I_m(t)$, and the error introduced vanishes in the limit with a small ⎕t. 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 7-4, 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 7-7:
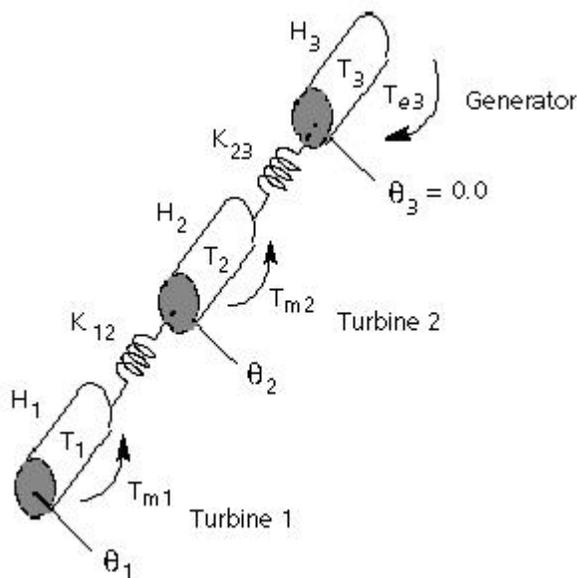

Figure 7-7 - Graphical Model of Multi-Mass Shaft Dynamics

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.

Initialisation
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 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 8: Transmission Lines and Cables

Overview

The major difficulty in modeling aerial transmission lines and underground cables is derived from the fact that they are highly nonlinear in nature; due mainly to the frequency dependency of conductors (skin effect), as well as the ground or earth return path. The ability to represent these systems accurately and efficiently plays an essential part in the electromagnetic transient simulation of power systems as a whole.

Transmission systems are frequency-dependent and so it makes sense to solve their parameters in the frequency domain. This is only suitable for those instances when the frequency domain characteristics are of primary interest. In order to accurately represent a frequency-dependent line when simulating with EMTDC (which of course operates in the time domain), these parameters must be convolved into their equivalent time-domain characteristics. The techniques required for this convolution are quite complex, and are one of the primary attributes that differentiate the various transmission system modeling methods available today.

PSCAD includes a few different modeling techniques, each with their own pros and cons. The most accurate of these, the Frequency Dependent (Phase) model, is one of the most precise in the world.

Basic Overview Of Modeling Techniques

Generally in electromagnetic transient simulations, there are two basic methods to represent transmission systems. The first is the π-section approach, where multi–phase systems can be characterized by a circuit of lumped passive elements. The second and more acknowledged method is a distributed parameter representation. Unlike the lumped element π-section, 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 other end. In an ideal sense, a distributed transmission system is a delay function; whatever is fed into one end will appear at the other end, perhaps slightly distorted, following some delay.

Both �x0-sections and distributed systems can be modeled with EMTDC. Distributed models are further sub-divided into single-frequency, and frequency-dependent representations. The constants required by EMTDC to represent distributed systems are calculated by a separate program called the Line Constants Program or LCP (discussed in the next section), whereas �x0-section representations are executed entirely within EMTDC.
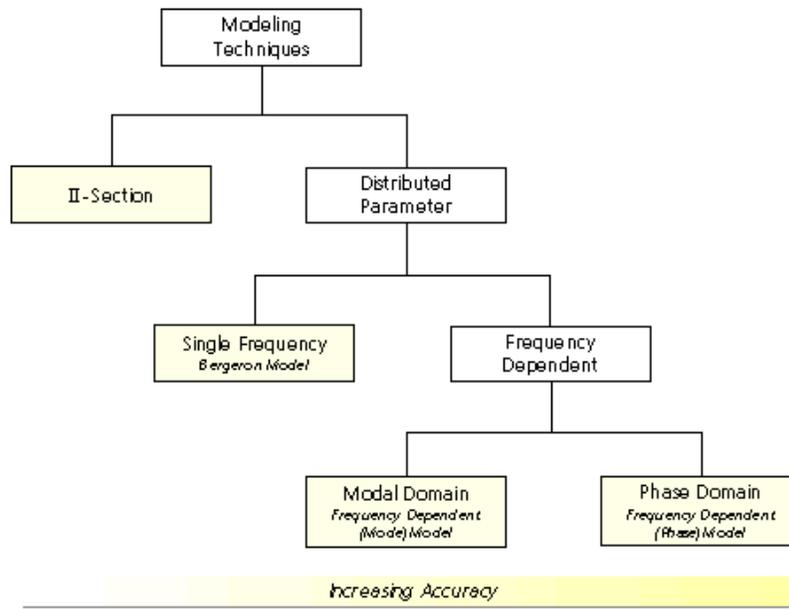

Figure 8-1 –Transmission System Modeling Techniques in EMTDC

P-Sections

A ⯑-section model will give the correct fundamental impedance, but cannot accurately represent other frequencies unless many sections are used (which is inefficient). It is suitable for very short lines where the travelling wave models cannot be used, due to time step constraints. In EMTDC, ⯑-sections are not considered a very elegant means of transmission line modeling for the following reasons:

- Greater computational time and increased EMTDC conductance matrix sizes

- Do not represent propagation delay

- Not practical with a large number of mutually coupled conductors

- Frequency dependent attenuation of the traveling waves is not easily or accurately incorporated

⯑-sections are discussed in detail later in this chapter.

Bergeron Model

The Bergeron model is the simplest, oldest and least accurate of the distributed branch models in PSCAD – mainly due to the fact that it is not frequency-dependent (calculates at a single frequency).

The Bergeron model represents the system *L* and *C* in a distributed manner (as opposed to lumped elements as in ⯑-sections). In fact, it is roughly equivalent to using an infinite number of series-connected ⯑-sections except that the total system resistance *R* is lumped (½ in the middle of the line, ¼ at each end). As with ⯑-sections, the Bergeron model accurately represents system parameters at the fundamental frequency. However, it can also be used to approximate higher frequency attenuation by choosing an additional frequency for calculation

The Bergeron model is suitable for studies where frequencies other than the fundamental are of little or no concern (i.e. many load flow and protection studies). Situations where this model should be chosen over the more accurate frequency dependent models include; when a lack of frequency dependent input data exists (such as when only +, -, and 0 sequence data is known), and/or when computational speed over accuracy is more important.

The Bergeron model is discussed in more detail later in this chapter.

Frequency Dependent Models

The frequency dependent models strive to represent the full frequency dependence of a transmission system. This is accomplished by solving the line parameters at many frequency points within a user-defined scope. As such, the frequency dependent models will take longer to solve than the Bergeron model, but are necessary for studies requiring a very detailed representation of the system over a wide frequency range. Unlike the Bergeron model, these models also represent the total system resistance *R* as a distributed parameter (along with a distributed system *L* and *C*), providing a much more accurate representation of attenuation.

PSCAD offers two frequency dependent models:

- The Frequency Dependent (Mode) model

- The Frequency Dependent (Phase) model

For all new studies involving transmission lines or cables, it is recommended that the *Frequency Dependent (Phase)* model be used. This model is the newest of the two (c. 1998) and was added to PSCAD specifically to replace the *Frequency Dependent (Mode)* model, which has been carried over from PSCAD V2 to allow for compatibility with older projects. The *Frequency Dependent (Phase)* model can represent any type of transmission system (i.e. aerial and underground, symmetrical and non-symmetrical) and is more accurate and more stable than its predecessor. The *Frequency Dependent (Phase)* model should normally be the model of choice for most studies.

The frequency dependent line models are discussed in more detail later in this chapter.

The PSCAD Line Constants Program

Development of the PSCAD Line Constants Program (or LCP) is based on the work of a variety of scientists and engineers in the field of transmission systems over the past few decades, and contains some of the most relevant modeling techniques in the industry today (see References [18], [23] for example). The primary purpose of the LCP is to calculate all frequency domain parameters (or constants) required, so that distributed transmission systems can be convolved into two-port, time domain representations and interfaced with the EMTDC network. Whether modeling the system at a single frequency or performing a full frequency-dependent representation, the LCP will execute the necessary steps to arrive at the required constants.

The present form of the LCP (originally released with PSCAD V3) is actually an amalgamation of two programs that existed in PSCAD V2: *T-LINE* and *CABLE*. It now exists as a separate executable file placed along with the PSCAD executable in the installation directory, and is called by PSCAD whenever a transmission system is encountered. Although the two *T-LINE* and *CABLE* programs are now merged together, the LCP still requires that aerial and underground systems not be combined. That is, underground cables and overhead transmission lines must be designed and solved separately. This is mainly due to the fact that the calculation of ground return impedance in both systems is fundamentally different, and there is presently no means by which to efficiently represent the mutual effects between the two system types.

All input data is read by the LCP from either a transmission line or cable input file (*.tli or *.cli). The input file is generated automatically by PSCAD when a project is compiled, based on the system cross-section in the Transmission Segment Definition Editor (see the next section called Data Input). The LCP output constants are written to either a transmission line or cable output file (*.tlo or *.clo), which is then read in by EMTDC before the simulation begins. The output file contains pre-formatted data to help EMTDC construct a two-port network equivalent for the system. A log file (*.log) and a user output file (*.out) are also generated by the LCP.

The LCP is structured as illustrated in Figure 9-2. All events outlined in this diagram are described in detail later in this chapter.
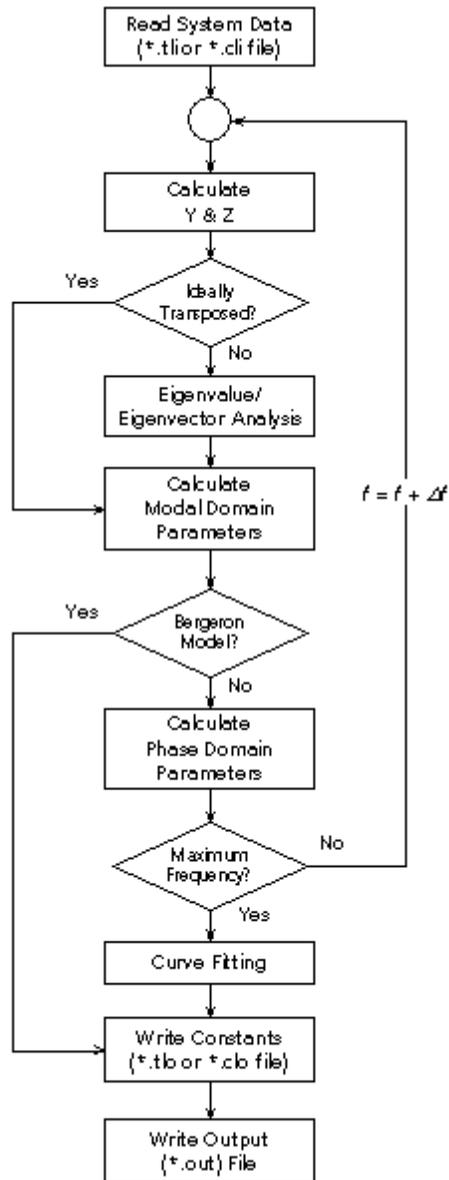
Figure 8-2: Basic PSCAD Line Constants Program Structure

Data Input

The Line Constants Program is controlled through the use of a graphical user interface tool in PSCAD called the Transmission Segment Definition Editor. Within this tool, the user may define the transmission system by constructing a cross-section of the right-of-way, complete with conductor/cable positions, etc. This information is compiled by PSCAD when the constants are solved for that line, and written to a transmission line or cable input file (*.tli or *.cli) as mentioned previously.

The following list is a summary of the input parameters required by the LCP, some are referred to during discussions later in this chapter:

General:

Symbol	Description
ℓ	Line length [km]
f_0	Steady-state frequency [Hz]
n_c	Number of conductors
ρ_g	Ground resistivity [Ωm]
μ_g	Ground relative permeability

Aerial Lines:

Symbol	Description
n_{cct}	Number of circuits
n_b	Number of bundles/phases per circuit
n_{bc}	Number of sub-conductors per bundle
d'_{bc}	Bundled conductor spacing (symmetrical only) [m]
x_t	Horizontal position of tower in right-of-way [m]
x_{bc}	Sub-conductor horizontal position relative to [m]
y_{bc}	Sub-conductor height above the ground plane [m]
x_c	Conductor/bundle horizontal position relative to x_t [m]
y_c	Conductor/bundle height above the ground plane [m]
r_c	Conductor radius (sub-conductor radius if bundled) [m]
R_{cdc}	Conductor DC resistance [Ω/km]
d_{sag}	Conductor sag at midpoint between tower spans [m]
G_s	Conductor shunt conductance [S/m]
x_g	Ground wire horizontal position relative to x_t [m]
y_g	Ground wire height above the ground plane [m]
r_g	Ground wire radius [m]

Symbol	Description
R_{gdc}	Ground wire DC resistance [Ω/km]
d'_{gsag}	Ground wire sag at midpoint between tower spans [m]

Underground Cables:

Symbol	Description
n_c	Number of cables
n_l	Number of conducting layers per cable
r_i	Conductor inner radius [m]
r_o	Conductor outer radius [m]
ρ_c	Conductor resistivity [Ωm]
μ_c	Conductor relative permeability
ε_i	Insulator relative permittivity
μ_i	Insulator relative permeability
x_c	Horizontal position of cable in right-of-way [m]
y_c	Cable depth below surface of ground plane [m]

Conductor/Cable Position

The geometric position of overhead conductors or cable centres in a transmission system cross-section, are ultimately input to the LCP in terms of Cartesian coordinates (i.e. linear *x* and *y* dimensions). These coordinates are relative to a reference point in both the x and y direction. The ground plane is considered the reference level (*y = 0.0*) in the y direction, and the x reference is an arbitrary point that is defined through data entry in the tower or cable cross-section components (*x_t* in Figure 8-3).
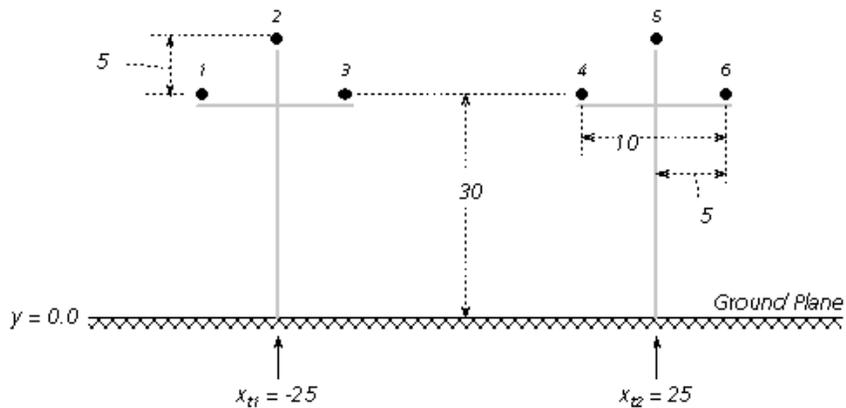
Figure 8-3: *x* and *y* References in a Typical Overhead Line Cross-Section

---

The LCP allows conductor *xy* position data to be entered either directly, or in a relative manner through tower dimensions, such as horizontal and vertical distances between conductors and relative *x* position of the tower. The latter option is provided as transmission line tower data is often given in this format. If the user desires to enter the coordinates directly, then the Universal Tower component is provided in the PSCAD Master Library.

For example, the system in Figure 8-3 consists of two towers and six conductors. A set of *xy* coordinates for each conductor is derived given the tower input data so that (assuming both towers are structurally identical):

$$x_1 = x_{t1} - 5 = -30 \qquad\qquad y_1 = 30$$

$$x_2 = x_{t1} = -25 \qquad\qquad y_2 = 30 + 5 = 35$$

$$x_3 = x_{t1} + 5 = -20 \qquad\qquad y_3 = 30$$

$$x_4 = x_{t2} - 5 = 20 \qquad\qquad y_4 = 30$$

$$x_5 = x_{t2} = 25 \qquad\qquad y_5 = 30 + 5 = 35$$

$$x_6 = x_{t2} + 5 = 30 \qquad\qquad y_6 = 30$$

NOTE: Conductor positions must be considered carefully when entering the data, otherwise erroneous results will be given. The LCP contains some sanity checks to ensure that conductors or cables do not overlap; however there is no way to check for incorrect conductor position!

Conductor Sag

Sag is a phenomena associated with suspended conductors, which tend to droop under their own weight when strung between tower stands. The amount of sag present is dependent not only on conductor weight, but also on tower span length, conductor properties and ambient temperature. Temperatures can vary considerably, depending on environmental conditions and power demand.

The Line Constants Program approximates the affects of conductor sag by simply decreasing the effective conductor height above ground by a factor of 2/3 of the total sag.

$$y' = y - \frac{2}{3} \cdot d_{sag} \tag{8-1}$$

The approximation eliminates sag by assuming a conductor of uniform height *y'*. This effectively makes the calculation of constants, for a particular right of way, independent of tower span.
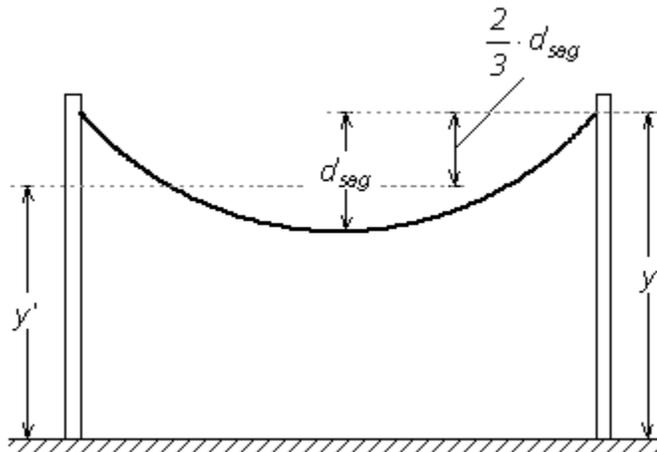
Figure 8-4: Setting Conductor Effective Height Given Sag

Assumptions and Approximations

It is important to realize that the numeric solution of equations involved in transmission line theory is quite involved and complex. From the very beginning, approximations are made in order to simplify the mathematics, indeed even the most fundamental equations are derived based on the assumption of some type of ideal condition.

The following list summarizes some of the more basic assumptions when modeling transmission systems:

- The earth is flat! That is, the curvature of the earth surface (or any change in surface height for that matter) is not considered.

- The ground is homogeneous. The ground resistivity and permeability is constant for the length of the transmission line.

- An aerial conductor is considered to be a perfectly round cylinder. That is, the conductor radius entered is the radius of a perfect cylinder, which is used to calculate cross-sectional area, which in turn is used to derive the conductor resistivity.

It is important to note that no matter how detailed a transmission system is modeled, there will *always* be error. It is the job of PSCAD and the LCP to ensure that any error is minimized to the fullest extent possible, but there are many instances when this responsibility falls in user's lap. For example in some places in the world, ambient temperatures can fluctuate wildly depending on the season, or even the time of day. This temperature variance can affect the impedance properties of the line by way of heating/cooling, as well as conductor sag. Therefore, the user should be diligent in ensuring that all input parameters reflect the real-world conditions.

Deriving System Y and Z Matrices

The analysis of both overhead lines and underground cables begins with two well-known equations, sometimes referred to as the 'fundamental' or 'telegrapher's equations' [5], [7] and [24].

Consider a short transmission segment, whose length is small when compared to the wavelength, and the ground is the voltage reference. The system will possess a series-impedance and shunt-admittance, which is illustrated graphically as shown in Figure 8-5:
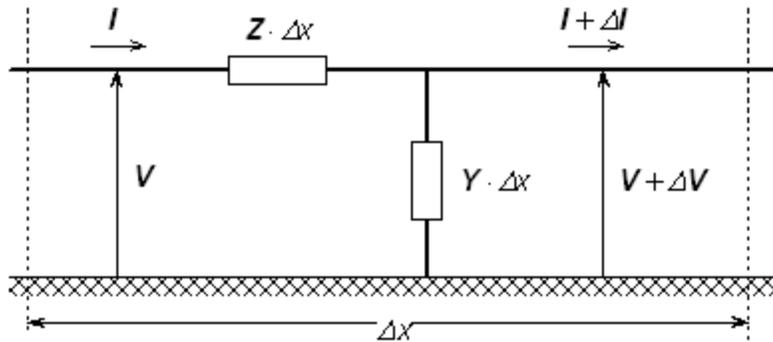
Figure 8-5 – Representation of a Relatively Small Transmission Line Segment

Given that Z and Y are the system impedance and admittance per unit length respectively, V is the conductor voltage and I is the conductor current, then the system in Figure 8-5 is described mathematically by the telegrapher's equations (as the limit $\Delta x \to 0$):

$$\frac{dV}{dx} = -Z \cdot I \qquad (8\text{-}2)$$

$$\frac{dI}{dx} = -Y \cdot V \qquad (8\text{-}3)$$

In multi-conductor systems, Z and Y are square matrices, and V and I are vectors, both with dimension equal to the total equivalent number of conductors.

The derivation of Z and Y differs depending on the system. Specifically whether the system is underground or in air, or whether the conductors are bare (i.e. overhead lines) or bundled in cables.

Underground Cables

Solving the Y and Z matrices for cable systems is a bit more involved than that of aerial transmission lines. In most practical systems, cables will possess multiple conducting layers, and most often run underground.

The LCP includes algorithms to solve single-core (SC) coaxial type cable systems: SC cables will consist of a centre conductor, and can include up to three concentric, conducting layers. Each conducting layer must be separated by an insulating layer, where the outermost insulating layer between the cable and earth is optional. The centre conductor can be either a hollow pipe possessing a finite thickness, or a solid, cylindrical conductor (by entering an inner radius $r_0 = 0.0$). Any insulated conducting layer of each SC cable may be grounded mathematically, where the voltage along the full length of that conductor will be assumed to be 0.0.

A cross-section of an SC cable consisting of core and sheath conductors is illustrated in Figure 8-6.
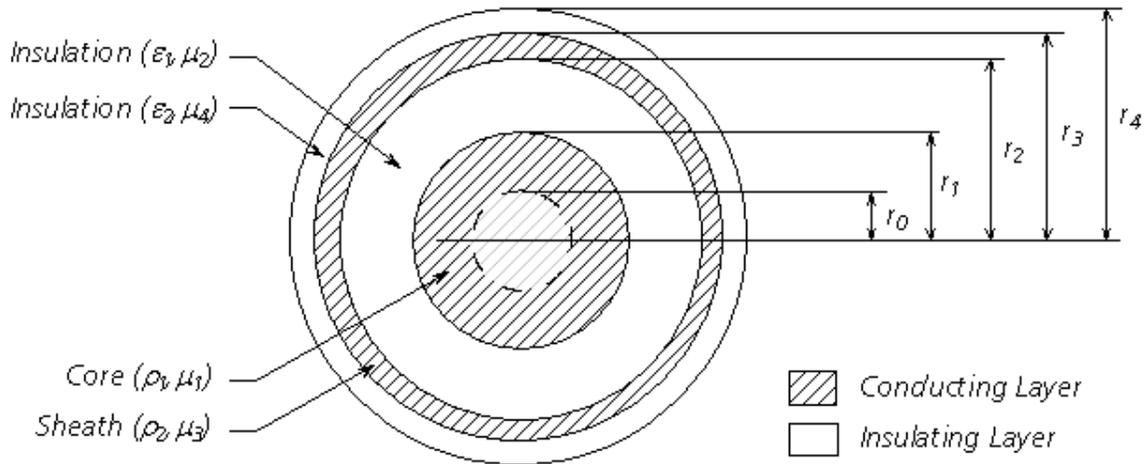
Figure 8-6: Cross-Section of a 2-Conducting Layer SC Cable

Where,

$\mu =$ Relative permeability

$\varepsilon =$ Relative permittivity

$\rho =$ Resistivity [Ωm]

$r =$ Radius of each layer [m]

Series Impedance Z

Apart from the centre conductor, each concentric conducting layer is represented by a combination of inner and outer surface impedances, as well as mutual impedance between it and the adjacent conductors. Each insulating layer is characterized by a single impedance with no mutual components. As such, and with respect to the series impedance, the SC cable shown in Figure 8-6 can be represented by the following equivalent circuit:
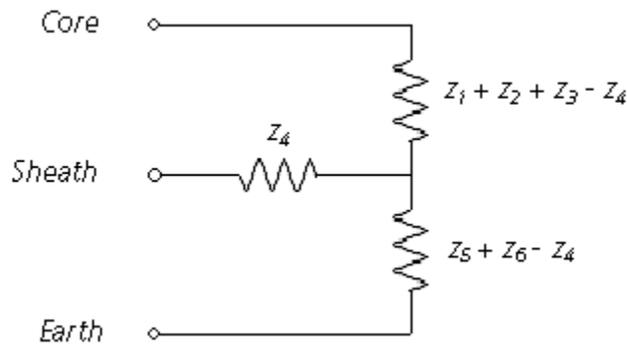
Figure 8-7: Cross-Section of a 2-Layer Coaxial Cable

NOTE: This method of circuit equivalency can be extended to consider any number of coaxial layers. The Line Constants Program extends this method to a maximum of 4 conducting layers plus 4 insulating layers.

This equivalent circuit can be derived by considering the effects of two circulating currents: The first flowing down the core conductor and returning back through the sheath, and the second down the sheath and back through the surrounding media [7].

The series impedance **Z** of *n*, single-core cables is given as follows:

$$Z = Z_i + Z_0 \qquad (8\text{-}4)$$

Where,

$Z_i =$ The SC cable internal impedance matrix

$Z_0 =$ The return media (earth) impedance matrix

The elements of **Z**$_i$ and **Z**$_0$ are themselves sub-matrices of dimension *m* x *m*, where *m* corresponds to the number of conducting layers in each respective cable *n*. If Equation 9-4 is rewritten in matrix format, it would appear as follows:

$$Z = \begin{bmatrix} Z_{i_1} & 0 & 0 & 0 \\ 0 & Z_{i_2} & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & Z_{i_n} \end{bmatrix} + \begin{bmatrix} Z_{0_{ii}} & Z_{0_{i2}} & \cdots & Z_{0_{in}} \\ Z_{0_{i2}} & Z_{0_{22}} & \cdots & Z_{0_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{0_{in}} & Z_{0_{2n}} & \cdots & Z_{0_{nn}} \end{bmatrix} \qquad (8\text{-}5)$$

Where,

$Z_{i_j} =$ Internal impedance sub-matrix of each respective SC cable

$Z_{0_{jk}} =$ Earth return impedance sub-matrix

The SC cable representation in Figure 8-7 (i.e. *m = 2*) corresponds to an internal impedance sub-matrix given as follows:

$$Z_{i_j} = \begin{bmatrix} Z_{cc_j} & Z_{cs_j} \\ Z_{cs_j} & Z_{ss_j} \end{bmatrix} \qquad (8\text{-}6)$$

Where,

$Z_{cc_j} = Z_1 + Z_2 + Z_3 + Z_5 + Z_6 - 2 \cdot Z_4$ Core Self Impedance

$Z_{cs_j} = Z_5 + Z_6 - Z_4$ Mutual Impedance between Core and Sheath

$Z_{ss_j} = Z_5 + Z_6$ Sheath Self Impedance

The earth return impedance sub-matrix in the case of Figure 8-7 is given as:

$$Z_{0_{jk}} = \begin{bmatrix} z_{0_{jk}} & z_{0_{jk}} \\ z_{0_{jk}} & z_{0_{jk}} \end{bmatrix}$$

(8-7)

Where,

$Z_{0_{jk}} =$ Earth return mutual impedance between cables j and k.

> **NOTE**: For more information on the derivation of earth return impedance, see the section entitled Mutual Impedance with Earth Return in this chapter.

There is more than one method by which to calculate the individual internal impedances of an SC cable: One popular method is to solve them directly by using the well-known Bessel function based equations [3]. The LCP instead solves these quantities by way of equivalent analytical approximations as described in [7], which are generally much better suited for digital computation.

In an ideal sense, the Bessel function approach is of course more accurate, as these equations directly describe the theoretical characteristics of the conductor impedances. However when solved numerically in a digital program, the Bessel functions themselves can quickly become mired in precision overflow/underflow problems ($I_n(x)$ tends to infinity and $K_n(x)$ tends to zero), and additional steps must be taken avoid these situations. Also, algorithms available to solve these functions are themselves a combination of approximate formulae [14].

The Wedepohl/Wilcox approximations outlined in [7] are exact at very low and very high frequency. The error between these and the Bessel based equations for impedance of a solid cylinder, peaks at about *0.5%* at mid-frequencies. The approximation error for the annulus impedances is even smaller, and approaches the exact functions with increasing accuracy as the thickness of the annulus becomes small compared with the radius of the annulus. The advantage to using these formulae over the Bessel function based impedances is mainly stability and speed of solution. The error introduced by the approximation equations is negligible in the time domain.

The equations for the individual impedances z_1 to z_6 in Figure 8-7 are given as follows. Both the Bessel function and equivalent approximate formula are presented for comparison:

Internal Impedance of Core Outer Surface (Solid Cylinder):

Approximate:
(8-8)
$$z_i = \frac{m\rho_i}{2\pi \cdot r_i} \cdot \coth(0.733 \cdot mr_i) + \frac{0.3179 \cdot \rho_i}{\pi \cdot r_i^2}$$

Bessel:
$$z_1 = j\omega \cdot \frac{m\rho_1}{2\pi \cdot r_1} \cdot \frac{I_0(mr_1)}{I_1(mr_1)}$$

Internal Impedance of Core Outer Surface (Annulus):

Approximate:
(8-9)
$$z_i = \frac{m\rho_i}{2\pi \cdot r_i} \cdot \coth[m(r_i - r_0)] + \frac{\rho_i}{2\pi \cdot r_i \cdot (r_0 + r_i)}$$

Bessel:
$$z_1 = j\omega \cdot \frac{\mu_0 \mu_1}{2\pi} \cdot \frac{1}{mr_1} \cdot \frac{I_0(mr_1) \cdot K_1(mr_0) + K_0(mr_1) \cdot I_1(mr_0)}{I_1(mr_1) \cdot K_1(mr_0) - K_1(mr_1) \cdot I_1(mr_0)}$$

Core Outer Insulator Impedance:

$$z_2 = j\omega \cdot \frac{\mu_0 \mu_2}{2\pi} \cdot \ln\left(\frac{r_2}{r_i}\right)$$

(8-10)

Internal Impedance of Sheath Inner Surface:

Approximate:

$$z_3 = \frac{m\rho_2}{2\pi \cdot r_2} \cdot \coth\left[m(r_3 - r_2)\right] + \frac{\rho_2}{2\pi \cdot r_2 \cdot (r_3 + r_2)}$$

(8-11)

Bessel:

$$z_3 = j\omega \cdot \frac{\mu_0 \mu_3}{2\pi} \cdot \frac{1}{mr_2} \cdot \frac{I_0(mr_2) \cdot K_i(mr_3) + K_0(mr_2) \cdot I_i(mr_3)}{I_i(mr_3) \cdot K_i(mr_2) - K_i(mr_3) \cdot I_i(mr_2)}$$

Sheath Mutual Impedance:

Approximate:

$$z_4 = \frac{m\rho_2}{\pi \cdot (r_3 + r_2)} \cdot \csc h\left[m(r_3 - r_2)\right]$$

(8-12)

Bessel:

$$z_4 = \frac{\rho_2}{2\pi \cdot r_2 r_3} \cdot \frac{1}{I_i(mr_3) \cdot K_i(mr_2) - K_i(mr_3) \cdot I_i(mr_2)}$$

Internal Impedance of Sheath Outer Surface:

Approximate:

$$z_5 = \frac{m\rho_2}{2\pi \cdot r_3} \cdot \coth\left[m(r_3 - r_2)\right] + \frac{\rho_2}{2\pi \cdot r_3 \cdot (r_3 + r_2)}$$

(8-13)

Bessel:

$$z_5 = j\omega \cdot \frac{\mu_0 \mu_3}{2\pi} \cdot \frac{1}{mr_3} \cdot \frac{I_0(mr_3) \cdot K_i(mr_2) + K_0(mr_3) \cdot I_i(mr_2)}{I_i(mr_3) \cdot K_i(mr_2) - K_i(mr_3) \cdot I_i(mr_2)}$$

Outer Insulator Impedance:

$$z_6 = j\omega \cdot \frac{\mu_0 \mu_4}{2\pi} \cdot \ln\left(\frac{r_4}{r_3}\right)$$

(8-14)

Where,

$I_k(x) =$ Modified Bessel function of the first kind

$K_k(x) =$ Modified Bessel function of the second kind

$$m = \sqrt{j\omega \cdot \frac{\mu_0 \mu_r}{\rho}}$$

Inverse complex depth of penetration

Shunt Admittance Y

Deriving the shunt admittance matrix **Y** for a system of SC cables is relatively simple compared to complexities of the series impedance. **Y** is based on the potential coefficient matrix **P**:

$$P = P_i \qquad\qquad (8\text{-}15)$$

Where,

$$P_i = \qquad \text{The SC cable internal potential coefficient matrix}$$

The elements of P_i are themselves sub-matrices of dimension m x m, where m corresponds to the number of conducting layers in each respective cable n. If Equation 8-15 is rewritten in matrix format, it would appear as follows:

$$P = \begin{bmatrix} P_{i_1} & 0 & 0 & 0 \\ 0 & P_{i_2} & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & P_{i_n} \end{bmatrix} \qquad\qquad (8\text{-}16)$$

Where,

$$P_{i_j} = \qquad \text{Internal potential coefficient sub-matrix of each cable}$$

The SC cable representation in Figure 8-6 (i.e. $m = 2$) corresponds to an internal impedance sub-matrix given as follows:

$$P_{i_j} = \begin{bmatrix} p_{c_j} + p_{s_j} & p_{s_j} \\ p_{s_j} & p_{s_j} \end{bmatrix} \qquad\qquad (8\text{-}17)$$

Where,

Core Potential Coefficient

$$p_{c_j} = \frac{1}{2\pi \cdot \varepsilon_{1_j} \cdot \varepsilon_0} \cdot \ln\left(\frac{r_{3_j}}{r_{2_j}}\right)$$

Sheath Potential Coefficient

$$p_{s_j} = \frac{1}{2\pi \cdot \varepsilon_{2_j} \cdot \varepsilon_0} \cdot \ln\left(\frac{r_{5_j}}{r_{4_j}}\right)$$

The shunt admittance matrix **Y** is then:

$$Y = j\omega \cdot P^{-1}$$

(8-18)

Note that in underground systems, the surrounding earth acts as an electrostatic shield resulting in null off-diagonal elements in **Y** [7].

Semiconducting Layers

The inclusion of semi-conducting layers to the coaxial cable model is based on reference [29]. The semiconductor layers can be easily added by modifying the properties of the main insulation layer (ex. by changing the permittivity of the insulation).
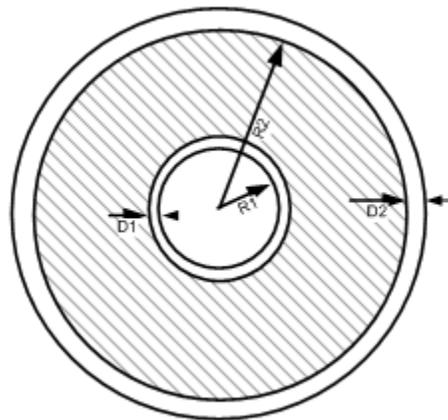
Figure 8-8: Cross-Section of an Coaxial Cable with Semi-Conductor Layers

$$\varepsilon_{new\_ins} = \varepsilon_{inu} \frac{ln\left(\frac{R_4}{R_1}\right)}{ln\left(\frac{B}{A}\right)}$$

(8-19)

Where,

$R_{1=}$ Outer radius of the inner conductor

$R_{2=}$ Outer radius of the insulator

$D_{1=}$ Thickness of inner semiconducting screen

$D_{2=}$ Thickness of outer semiconducting screen

$R_{4=}$ $R_2 + D_2$. Outer radius of inner conductor + thickness of inner semiconducting screen + thickness of insulator layer + thickness of outer semiconducting screen

$A_=$ $R_1 + D_1$. Outer radius of inner conductor + thickness of inner semiconducting screen.

$B_=$ R_2. Outer radius of inner conductor + thickness of inner semiconducting screen + thickness of insulator layer

<u>Assumptions</u>

See reference [30].

The capacitance of the semi-conducting layers is neglected. The permittivity of semi-conducting layers is very high compared to the insulator permittivity (e.g. typical values for relative permittivity of the semi-conducting layers and insulators are 1000 and 3 respectively), hence the capacitance of the semi-conducting layers is much higher than that of insulator. In the admittance equivalent circuit of the cable, two capacitors are in series. The total capacitance is

$$C_{total} = \frac{C_{ins} \cdot C_{semi}}{C_{ins} + C_{semi}}$$

(8-20)

Since $C_{ins} \ll C_{semi}$, then $C_{total} \approx C_{ins}$.

The conductivity of the semi-conducting layer is much lower than that of core or sheath; hence the relative contribution of the longitudinal current in the semi-conducting layers is negligible. Typical values for conductivity of the core and the semi-layers are 1e8 and 10 S/m. Note that the thickness of the first semiconducting layer must be less than the $R_2 - R_1$ and thickness of the second semiconducting layer must be less than the $R_3 - R_2$, where R_3 is the outer radius of the sheath.

Overhead (Aerial) Conductors

Deriving the series impedance **Z** and the shunt admittance **Y** for overhead lines is a relatively straightforward procedure in comparison to cables. This is mainly due to the fact that overhead conductors are generally simple –that is, each medium consists of a single, cylindrical conductor with no additional layers separated by insulation. The Line Constants Program provides the option to model an overhead conductor as both solid or hollow (i.e. as a conducting annulus).
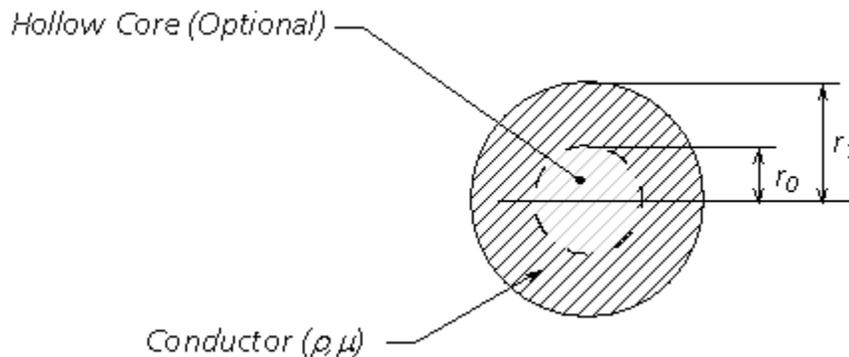
Figure 8-9: Cross-Section of an Overhead Conductor

Where,

$\mu =$ Relative permeability

$$\rho = \quad \text{Resistivity [}\Omega\text{m]}$$

$$r_i = \quad \text{Radius [m]}$$

Series Impedance Z

Each overhead conductor is modeled assuming a perfect cylindrical shape, which may possess a hollow core. As such, the conductor is represented by a combination of inner and outer surface impedances. If the conductor is solid, only the outer surface impedance is considered. The composition of the conductor is assumed to be homogeneous – that is, composite conductors, such as ACSR must be approximated as homogeneous.

The series impedance **Z** of *n*, overhead lines is given as follows:

$$\boldsymbol{Z} = \boldsymbol{Z}_i + \boldsymbol{Z}_0 \tag{8-21}$$

Where,

$$\boldsymbol{Z}_i = \quad \text{The internal impedance matrix}$$

$$\boldsymbol{Z}_0 = \quad \text{The earth return impedance matrix}$$

NOTE: For more information on the derivation of earth return impedance, see the section entitled Mutual Impedance with Earth Return in this chapter

If Equation 8-21 is rewritten in matrix format, it would appear as follows:

$$\boldsymbol{Z} = \begin{bmatrix} Z_{i_1} & 0 & 0 & 0 \\ 0 & Z_{i_2} & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & Z_{i_n} \end{bmatrix} + \begin{bmatrix} Z_{0_{11}} & Z_{0_{12}} & \cdots & Z_{0_{1n}} \\ Z_{0_{12}} & Z_{0_{22}} & \cdots & Z_{0_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{0_{1n}} & Z_{0_{2n}} & \cdots & Z_{0_{nn}} \end{bmatrix} \tag{8-22}$$

Where,

$$Z_{i_j} = \quad \text{Internal impedance of each respective conductor}$$

$$Z_{0_{jk}} = \quad \text{Earth return impedance between conductors } j \text{ and } k$$

The internal impedance for an overhead conductor is given as shown below. The LCP uses an analytical approximation to the well known Bessel based equations. Both the analytical approximation and Bessel equations are given here for easy reference:

NOTE: See the section entitled Underground Cables in this chapter for a detailed explanation why approximations are chosen over the Bessel method.

Internal Impedance of an Overhead Conductor (Solid Cylinder):

Approximate: (8-23)

$$Z_i = \frac{m\rho}{2\pi \cdot r_1} \cdot coth(0.733 \cdot mr_1) + \frac{0.3179 \cdot \rho}{\pi \cdot r_1^2}$$

Bessel:

$$Z_i = j\omega \cdot \frac{m\rho}{2\pi \cdot r_1} \cdot \frac{I_0(mr_1)}{I_1(mr_1)}$$

Internal Impedance of an Overhead Conductor (Hollow Core):

Approximate: (8-24)

$$Z_i = \frac{m\rho}{2\pi \cdot r_1} \cdot coth[m(r_1 - r_0)] + \frac{\rho}{2\pi \cdot r_1 \cdot (r_0 + r_1)}$$

Bessel:

$$Z_i = j\omega \cdot \frac{\mu_0 \mu}{2\pi} \cdot \frac{1}{mr_1} \cdot \frac{I_0(mr_1) \cdot K_1(mr_0) + K_0(mr_1) \cdot I_1(mr_0)}{I_1(mr_1) \cdot K_1(mr_0) - K_1(mr_1) \cdot I_1(mr_0)}$$

Where,

$I_k(x) =$ Modified Bessel function of the first kind

$K_k(x) =$ Modified Bessel function of the second kind

$m = \sqrt{j\omega \cdot \frac{\mu_0 \mu_r}{\rho}}$ Inverse complex depth of penetration

Shunt Admittance Y

The shunt admittance matrix **Y** for aerial transmission systems can be easily derived, due to the fact that the surrounding air may be assumed as loss less, and the ground potential considered zero [24]. Therefore:

$$Y_{ij}^{-1} = \frac{1}{j\omega \cdot 2\pi \cdot \varepsilon_0} \cdot ln\left(\frac{D_{ij}}{d_{ij}}\right)$$ (8-25)

Where,

$\omega =$ Frequency [rad/s]

$\varepsilon_0 =$ Permittivity of free space

$D_{ij}, d_{ij} =$ Distances as described in the next section entitled Mutual Impedance with Earth Return [m].

Mutual Impedance with Earth Return

The previous section dealt with the formulation of the **Y** and **Z** matrices for both overhead lines and underground cables. Although the self impedance and admittance were discussed, the derivation of the mutual impedance between conductors and/or cables was not. This section describes how the mutual impedance between entities, including homogeneous earth return, is calculated in the LCP.

Aerial Lines

One of the most popular methods developed for earth return impedance in overhead lines, was first published by J. R. Carson in 1926 [1]. The equations involved (now known as Carson's Equations) contain infinite integrals with complex arguments that are difficult to evaluate numerically [12]. The equation for mutual impedance between aerial conductors with homogeneous earth return is as given below:

$$
Z_{0_{ij}} = j\omega \cdot \frac{\mu_0}{2\pi} \cdot \left[ln\left(\frac{D_{ij}}{d_{ij}}\right) + 2 \cdot \int_0^\infty \frac{e^{-\alpha \cdot cos(\theta_{ij})} \cdot cos(\alpha \cdot sin(\theta_{ij}))}{\alpha + \sqrt{\alpha^2 + jr_{ij}^2}} \cdot d\alpha \right]
$$

(8-26)

$$
= j\omega \cdot \frac{\mu_0}{2\pi} \cdot \left(Z_{M_{ij}} + Z_{G_{ij}} \right)
$$

Where,

$$
D_{ij} = \begin{cases} \sqrt{(x_i - x_j)^2 + (y_i + y_j)^2}, & i \neq j \\ \\ 2 \cdot h_i, & i = j \end{cases}
$$

$$
d_{ij} = \begin{cases} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, & i \neq j \\ \\ r_i, & i = j \end{cases}
$$

$$
\theta_{ij} = \begin{cases} tan^{-1}\left(\frac{(x_i - x_j)}{(y_i + y_j)}\right), & i \neq j \\ 0, & i = j \end{cases}
$$

$$
r_{ij} = \begin{cases} \sqrt{\omega \cdot \mu_0 \cdot \sigma} \cdot D_{ij} = \frac{D_{ij}}{|d_e|}, & i \neq j \\ \\ \sqrt{\omega \cdot \mu_0 \cdot \sigma} \cdot 2 \cdot h_i = \frac{2 \cdot h_i}{|d_e|}, & i = j \end{cases}
$$

$x_i, x_j =$ Horizontal position of the ith and j^{th} conductor respectively [m]

$$y_i, y_j =$$ Vertical position of the ith and j^{th} conductor respectively [m]

$$h_i =$$ Vertical position of the ith and j^{th} conductor respectively [m]

$$r_i =$$ Radius of the ith conductor [m]

$$d_e = \sqrt{\frac{\rho}{j\omega \cdot \mu}}$$ Depth of penetration

The first term Z_M in Equation 8-24 represents the aerial reactance of the conductor, had the ground return been a perfect conductor. The second term Z_G is referred to as Carson's Integral, which represents the additional impedance due to a lossy ground.
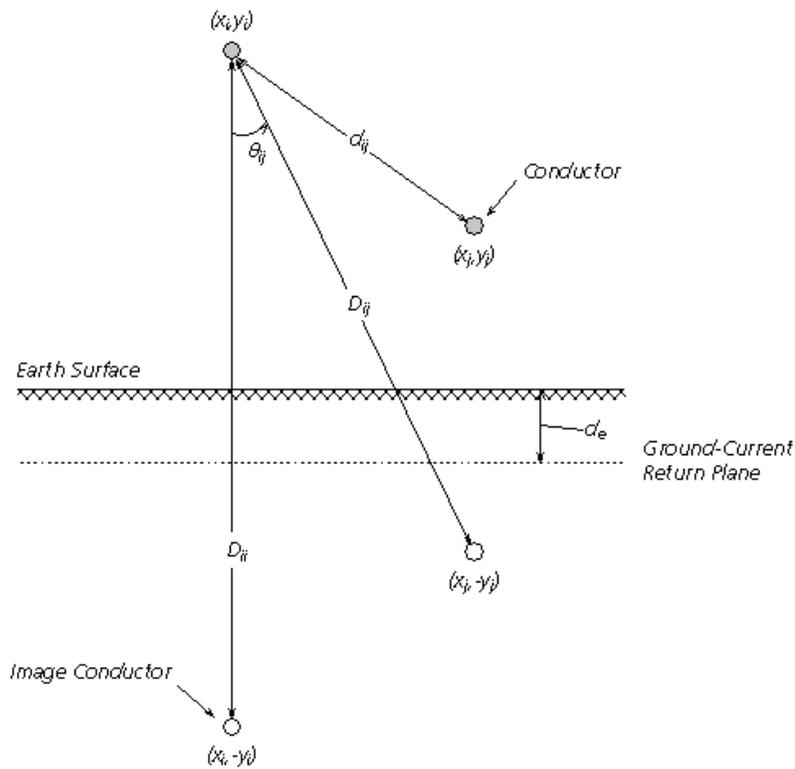
Figure 8-10: Graphical Data used in the Deri-Semlyen Approximation

An ingenious method for dealing with ground return, published by Gary in 1976 [10], introduced the concept of a super conducting, current return plane. This fictitious plane was placed at a complex depth d_e, also known as the depth of penetration, and provided a mirroring surface where the conductor 'reflections' were used to derive very simple equations for ground return impedance. Although published in 1976, mathematical proofs for these equations did not surface until Deri and Semlyen in 1981 [12]. Basically, what is now referred to as the Deri-Semlyen approximation of Carson's Integral is as follows:

$$Z_{G_{ij}} = 2 \cdot \int_0^\infty \frac{e^{-\alpha \cdot \cos(\theta_{ij})} \cdot \cos(\alpha \cdot \sin(\theta_{ij}))}{\alpha + \sqrt{\alpha^2 + jr_{ij}^2}} \cdot d\alpha \approx \ln\left(\frac{D'_{ij}}{D_{ij}}\right) \tag{8-27}$$

Where,

$$D'_{ij} = \sqrt{(y_i + y_j + 2 \cdot d_e)^2 + (x_i - x_j)^2}$$

Combing Equations 8-26 and 8-27 results in the following for mutual impedance between aerial conductors,

$$Z_{0ij} = j\omega \cdot \frac{\mu_0}{2\pi} \cdot \ln\left(\frac{D'_{ij}}{d_{ij}}\right) \tag{8-28}$$

Ground Return Formula Selection

The Line Constants Program provides the user with a choice in representing the ground return impedance in aerial systems. An option is provided to use either the Deri-Semlyen approximation (default), or by direct numerical integration of Carson's integral (both are shown below in Equation 8-31).

Analytical (Deri-Semlyen) Approximation:

Analytical (Deri-Semlyen) Approximation:
$$Z_{Gij} = \ln\left(\frac{D'_{ij}}{D_{ij}}\right)$$
(8-29)

Direct Numerical Integration:
$$Z_{Gij} = 2 \cdot \int_0^\infty \frac{e^{-\alpha \cdot \cos(\theta_{ij})} \cdot \cos(\alpha \cdot \sin(\theta_{ij}))}{\alpha + \sqrt{\alpha^2 + jr_{ij}^2}} \cdot d\alpha$$

Note that although the direct numerical integration method is slightly more accurate, the additional solution time can become quite extensive depending on the complexity of the system if this option is chosen.

Underground Cables

The integrals describing the equations for the earth return impedance in a system consisting of buried cables were originally developed by Pollaczek in 1931 [2]. As with Carson's integral for aerial lines, the Pollaczek integrals are ill-conditioned and are very difficult to evaluate numerically. The Pollaczek integrals define an electric field vector *E* at an arbitrary point within a homogeneous ground, due to electric current flowing in a buried conductor.
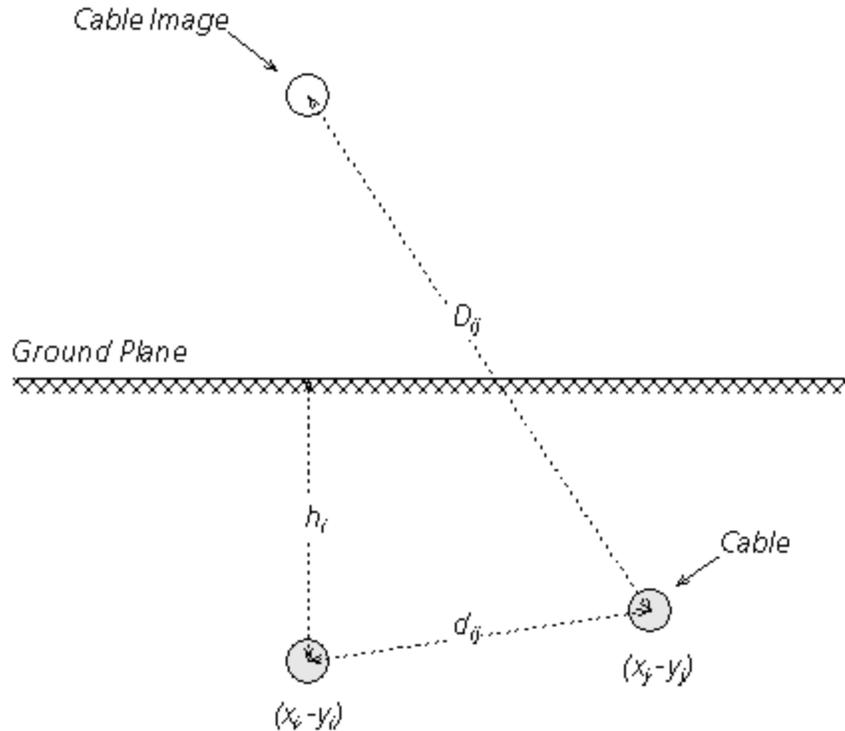
Figure 8-10: Graphical Data used in the Wedepohl-Wilcox Approximation

Equation 8-28 below gives the equation for mutual impedance corresponding to the form given by Pollaczek [7]:

$$Z_0 = \frac{\rho \cdot m^2}{2\pi} \cdot \int_{-\infty}^{\infty} e^{j \cdot \alpha x} \cdot \left[\frac{e^{-(h_i + y_i)\sqrt{\alpha^2 + m^2}}}{|\alpha| + \sqrt{\alpha^2 + m^2}} + \frac{e^{-|y_i - h|\sqrt{\alpha^2 + m^2}} - e^{-|y_i + h|\sqrt{\alpha^2 + m^2}}}{2 \cdot \sqrt{\alpha^2 + m^2}} \right] \cdot d\alpha \qquad (8\text{-}30)$$

In 1973, an analytical approximation to the Pollaczek integral was developed by Wedepohl and Wilcox, who found that up to very high frequencies only a subset of Pollaczek terms need be taken into account. The mathematical proof of their work is quite involved and so only the end result is shown in Equations 8-29 and 8-30 below: Equations for both the self and mutual parts of the earth return impedance is included. For a detailed account of the proof, see [7].

$$Z_0 = j\omega \frac{\mu}{2\pi} \cdot \left(-\ln\left(\frac{\gamma \cdot m \cdot r_i}{2} \right) + \frac{1}{2} - \frac{4}{3} \cdot m \cdot h_i \right) \qquad (8\text{-}31)$$

$$Z_0 = j\omega \frac{\mu}{2\pi} \cdot \left(-\ln\left(\frac{\gamma \cdot m \cdot d_{ij}}{2} \right) + \frac{1}{2} - \frac{2}{3} \cdot m \cdot \Sigma h \right) \qquad (8\text{-}32)$$

Where,

$\mu =$ Ground return (earth) relative permeability

$\gamma =$ 1.7811 Euler's constant

$r_i =$ Outer radius of the ith cable [m]

$d'_{ij} =$ Distance between the centre points of the ith and j^{th} cables [m].

$h_i =$ Depth of the ith cable (centre point) below the ground surface [m]

$m =$ $\sqrt{\dfrac{j\omega \cdot \mu}{\rho}}$ Inverse depth of penetration

$\sum h =$ Sum of the depths of ith and j^{th} cables [m].

Ground Return Formula Selection

As in aerial systems, the Line Constants Program provides a choice in representing the ground return impedance in underground systems. An option is provided to use either the Wedepohl approximation (default), or by direct numerical integration of Pollaczek's integral (both are shown below in Equation 8-33).

Analytical (Deri-Semlyen) Approximation:

Analytical (Deri-Semlyen) Approximation:
$$Z_{Gij} = ln\left(\frac{D'_{ij}}{D_{ij}}\right)$$
(8-33)

Direct Numerical Integration:
$$Z_{Gij} = 2 \cdot \int_{0}^{\infty} \frac{e^{-\alpha \cdot cos(\theta_{ij})} \cdot cos(\alpha \cdot sin(\theta_{ij}))}{\alpha + \sqrt{\alpha^2 + jr_{ij}^2}} \cdot d\alpha$$

Note that although the direct numerical integration method is slightly more accurate, the additional solution time can become quite extensive depending on the complexity of the system.

Combined Aerial and Underground Cables

The theory and construction for aerial cable derivations can be seen from reference [32].
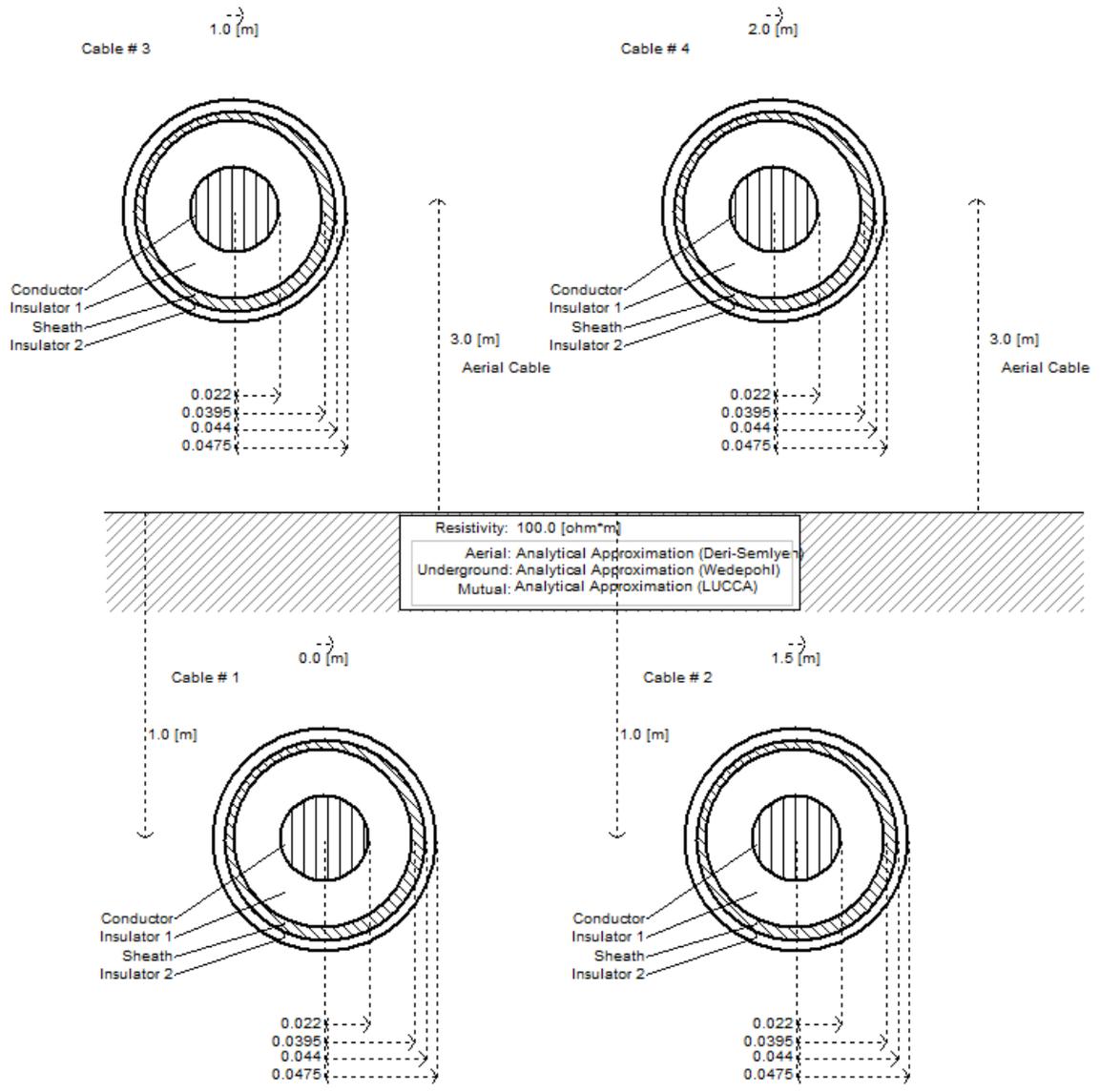
Figure 8-11: Combined Aerial/Underground Cable System in PSCAD

In the above diagram, cables 1 and 2 are underground and cables 3 and 4 are aerial. The Z_{earth} and $P_{external}$ (potential coefficient matrix for external field) matrices are given as follows:

$$Z_{earth} = \begin{bmatrix} Z_{11} & Z_{12} & Z_{13} & Z_{14} \\ Z_{21} & Z_{22} & Z_{23} & Z_{24} \\ Z_{31} & Z_{32} & Z_{33} & Z_{34} \\ Z_{41} & Z_{42} & Z_{43} & Z_{44} \end{bmatrix} \qquad (8\text{-}34)$$

The sub matrices Z_{11}, Z_{12}, Z_{21}, Z_{22} are the self and mutual earth impedance matrices for usual underground cables 1 and 2 (derived from the Pollaczek equations). Z_{33}, Z_{34}, Z_{43}, Z_{44} are derived based on the Carson equations (as for overhead lines) for cables 3 and 4. The other mutual matrices are derived based on mutual impedance between overhead lines and underground cables (another form of Pollaczek equations).

$$P_{external} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix}$$

(8-35)

The sub matrices P_{11}, P_{12}, P_{21}, P_{22} are zero since there is no capacitive coupling between cables. P_{33}, P_{34}, P_{43}, P_{44} are derived as of overhead lines for cable 3 and 4. Other matrices are also zero since there is no capacitive coupling between overhead lines and underground cables [34].
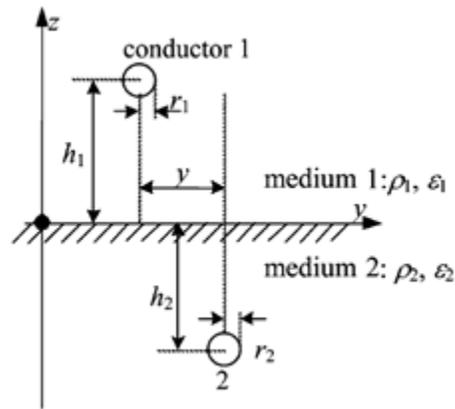
Figure 8-12: Combined Aerial/Underground Cable System

The aerial/underground cable mutual impedance is given as follows.

Assuming:

$$h_1 \geq 0, h_2 \leq 0, i = 1, k = 2$$

(8-36)

$$K_0(m_i d) - K_0(m_i D) = 0, \qquad \text{for } i \neq k$$

Then,

$$Z_m = Z(1, 2) = j\omega \left(\frac{\mu_0}{2\pi}\right) \int_{-\infty}^{\infty} F_c(s) \exp(jys) ds \qquad \text{(8-37)}$$

Where,

$$F_c(s) = \frac{\exp\{-h_1 |s| + h_2 \sqrt{s^2 + m^2}\}}{\sqrt{s^2 + m^2} + |s|}$$

More on the mutual coupling between aerial and underground cables can be found in references [32], [33], [34]. The mutual coupling between overhead line and underground cables is given by [32].

Ground Return Formula Selection

As in both purely aerial or underground systems, the Line Constants Program provides a choice in representing the ground return impedance for combined aerial/underground systems. An option is provided to use either the LUCCA approximation, said to be the most widely used and accurate formula (default), or the Ametani approximation.

Analytical (LUCCA) Approximation: (8-38)

$$Z_L = j\omega \left(\frac{\mu_0}{2\pi}\right) \left[\ln\left(\frac{S}{D}\right) - \left(\frac{2}{3}\right)\left(\frac{h_e}{S^2}\right)^3 H(H^2 - 3y^2) \right]$$

Analytical (Ametani) Approximation:

$$Z_m = j\omega \left(\frac{\mu_0}{2\pi}\right) \exp\left(\frac{-h_2}{h_e}\right) \ln\left(\frac{S}{D}\right)$$

Where,

$$m_i^2 = j\omega\mu_0\sigma_i$$

$$H = h_1 + h_2 + 2h_e$$

$$h_e = 1/m, \; S = \sqrt{H^2 + y^2}, \; D = \sqrt{(h_1 + h_2)^2 + y^2}$$

Conductor Elimination

In many cases, it is possible to reduce the transmission system matrix dimension by eliminating individual conductors. This process can help to increase the solution speed, without greatly affecting the accuracy of results. Conductor elimination is performed within the LCP in the following areas:

- Optional elimination of aerial ground wires in overhead transmission lines

- Optional elimination of conducting layers in single-core cables.

- Reduction of sub-conductor bundles into their equivalent single conductor form (not optional).

The Kron matrix reduction technique is used in all areas.

Kron Reduction

The Kron reduction technique can be illustrated with the following simple example of reducing the system Z matrix by eliminating aerial ground wires:

$$\frac{d}{dx}\begin{bmatrix} V_c \\ V_g \end{bmatrix} = -\begin{bmatrix} Z_{cc} & Z_{cg} \\ Z_{gc} & Z_{gg} \end{bmatrix} \cdot \begin{bmatrix} I_c \\ I_g \end{bmatrix} \tag{8-39}$$

The '*g*' and '*c*' subscripts represent the conductor and ground groups respectively. In the case of eliminating ground wires, it is assumed that $V_g = 0.0$, which leads to:

$$\frac{d}{dx}\begin{bmatrix} V_c \\ 0 \end{bmatrix} = -\begin{bmatrix} Z_{cc} & Z_{cg} \\ Z_{gc} & Z_{gg} \end{bmatrix} \cdot \begin{bmatrix} I_c \\ 0 \end{bmatrix} \tag{8-40}$$

$$\frac{d}{dx}V_c = -Z' \cdot I_c$$

Where,

$$Z' = Z_{cc} - Z_{cg} \cdot Z_{gg}^{-1} \cdot Z_{gc} \tag{8-41}$$

A similar procedure is also applied to the inverse shunt conductance matrix Y^{-1}.

Aerial Ground Wire Elimination

In overhead transmission systems, ground wires are usually strung from tower to tower, above the main phase conductors. The ground wires are bonded to ground at each tower, which normally span approximately 350 m. When considering line parameters below about 350 kHz, it is safe to assume that ground potential is uniform throughout the ground wire span [24]. In other words, the ground wires may be eliminated from the matrix equations, thereby reducing their dimension and speeding up the solving process.

It is important to note that the elimination of ground wires from a transmission line is *not* the same as having no ground wires at all. The mere presence of the ground wires, even if they are eliminated, affects the remaining system *Y* and *Z* matrices.

Conducting Layer Elimination

When defining underground cable systems, an option is provided to allow for mathematical elimination of specified conducting layers in a cable. In a practical sense, this feature is useful in situations where, for example, the outer insulating material on the cable is semi-permeable, thereby permitting contact between the return medium (i.e. sea water or soil) and the outer conducting layer. In this situation, the outer conducting layer effectively becomes part of the return medium, although the overall radius of the cable, which is used in the calculation of earth return impedance, remains unchanged.

Conductor Bundling

The bundling together of two or more sub-conductors to form a single phase conductor is common practice in overhead transmission line design. The sub-conductors are normally held together with spacers in a symmetrical, equidistant pattern (asymmetrical positioning is also used).
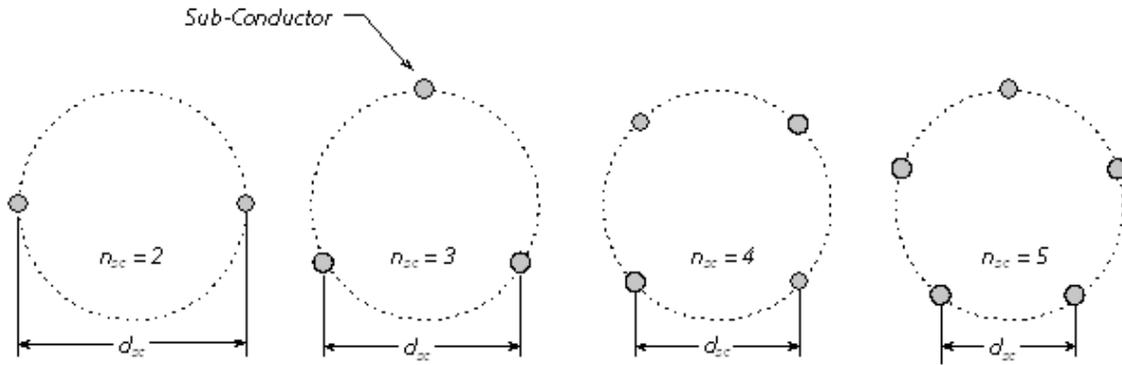
Figure 8-13: Orientations of Symmetrical Bundled Conductors

Based on the assumption that the phase current is divided evenly amongst the sub-conductors, and that each carries the same phase voltage, the conductor bundle may be reduced to a single, equivalent phase conductor. The Line Constants Program performs this equivalency by using the Kron reduction technique as described earlier.

There are two avenues for entering bundled conductor data; symmetrical and asymmetrical. If the bundle is symmetrical (as shown in Figure 8-13), then simple trigonometric techniques are used to calculate the individual XY coordinates of each sub-conductor, given the sub-conductor spacing d_{sc} and the number of sub-conductors per bundle N. If the bundle is asymmetrical, then the XY coordinates of each sub-conductor must be entered directly.
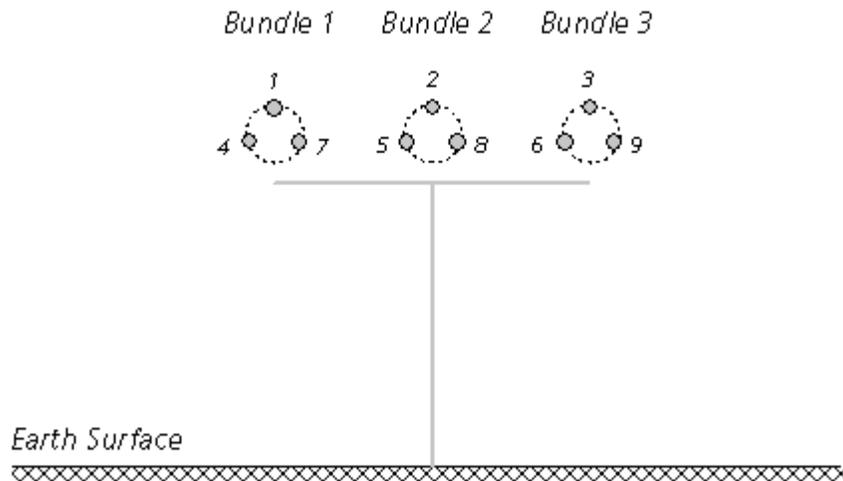
Figure 8-14: Cross-Section of a 3-Phase Transmission System with Three Sub-Conductor Bundles

Each sub-conductor is initially considered a separate entity when the system matrices are constructed. For example, if a transmission system consists of a single three-phase circuit, where each phase conductor consists of three sub-conductors (as shown in Figure 8-14), then the resulting system matrix dimensions would be *9 x 9*, as shown below for the series impedance matrix *Z* (where the individual matrix elements are calculated as described in the section entitled Deriving the System Y and Z in this chapter).

$$Z = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{19} \\ z_{21} & z_{22} & \cdots & z_{29} \\ \vdots & \vdots & \ddots & \vdots \\ z_{91} & z_{92} & \cdots & z_{99} \end{bmatrix} \qquad \text{(8-42)}$$

Once the system matrices are defined, the Kron reduction procedure is invoked, which effectively reduces the system matrices down to their single-conductor, multiple phase equivalent. For example Equation 8-42 is reduced to:

$$Z' = \begin{bmatrix} z'_{11} & z'_{12} & z'_{13} \\ z'_{21} & z'_{22} & z'_{23} \\ z'_{31} & z'_{32} & z'_{33} \end{bmatrix} \qquad \text{(8-43)}$$

In the end, the effective number of conductors in this example is 3. This will need to be considered when you are externally connecting the transmission system to the greater network in PSCAD. See Constructing Overhead Lines or Constructing Underground Cables in the PSCAD Manual.

Conductor Transposition

Many long AC transmission lines employ the concept of conductor transposition in order to minimize system imbalance. In a practical sense, this is achieved by periodically rotating the conductor positions in the circuit, so that each phase spends an equal distance in each position, between the sending and receiving end.
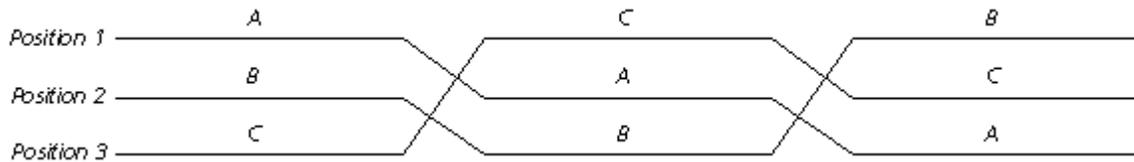
Figure 8-15: Practical Transposition Cycle of a 3-Phase Circuit

There are essentially two methods by which to model practically-connected, transposed lines in PSCAD (valid when using the Bergeron and either of the Frequency Dependent line models). The first method requires that each section of the total line to be represented as an individual line. The transpositions can be implemented in one of two ways for this case: The first is to physically transpose the circuit interconnections between the line sections on the PSCAD Canvas, as shown below:
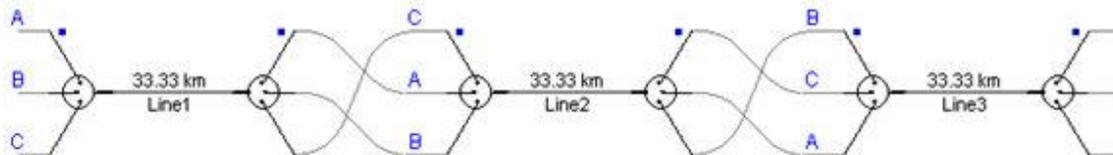
An alternative method is to alter the conductor XY coordinates within the properties editor of each line section. One benefit of this method is that phase A could remain conductor #1 throughout the length of the transmission line.

Ideal Transposition of Aerial Conductors

Practical transposition, like that described above, minimizes system imbalance caused by conductor positioning, but does not completely remove it. The system becomes more and more balanced as the number of transposition cycles increase (and hence the segment lengths decrease), from one end to the other. For studies where it is desirable for line to be completely balanced, the LCP provides an option called 'Ideal Transposition'.

Ideal transposition is a mathematical average of the unbalanced line. This is equivalent to visualizing an infinite number of transposition cycles, where the line segment length approaches zero. The result is a series impedance matrix Z and a shunt admittance matrix Y that are perfectly symmetrical and balanced. For example, if Z_{sc} is the system impedance matrix, then Z'_{sc} is it's ideally transposed equivalent:

$$Z_{sc} = \begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1n} \\ Z_{21} & Z_{22} & \cdots & Z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{n1} & Z_{n2} & \cdots & Z_{nn} \end{bmatrix} \rightarrow Z'_{sc} = \begin{bmatrix} Z_1 & Z_2 & \cdots & Z_2 \\ Z_2 & Z_1 & \cdots & Z_2 \\ \vdots & \vdots & \ddots & \vdots \\ Z_2 & Z_2 & \cdots & Z_1 \end{bmatrix} \tag{8-44}$$

Where,

$$Z_1 = \frac{1}{N_c} \cdot \sum_{n=1}^{N_c} Z_{nn} \tag{8-45}$$

$$Z_2 = \frac{1}{N_c \cdot (N_c - 1)} \cdot \sum_{m=1}^{N_c} \sum_{n=1}^{N_c} Z_{mn} \quad \text{where, } m \neq n \tag{8-46}$$

Ideal Transposition and Multiple-Circuit Towers

The PSCAD Master Library supplies a few towers containing multiple circuits, which include additional options for ideal line transposition. When using these components, users may opt to either transpose each circuit separately, or include all circuits in the transposition calculation. If the latter option is chosen, then the transposition procedure is as described in the previous section. However, if the circuit transposition is performed on a circuit-by-circuit basis, then the procedure is slightly different. For example, it is required that a three-phase, double-circuit tower be separately ideally transposed. Then for the system series impedance,

$$Z_{mc} = \begin{bmatrix} Z_1 & Z_M \\ Z_M & Z_2 \end{bmatrix} \rightarrow Z'_{mc} = \begin{bmatrix} Z'_1 & Z'_M \\ Z'_M & Z'_2 \end{bmatrix} \tag{8-47}$$

Where,

Z_1, Z_2, Z'_1 and Z'_2 are defined separately as given by Equation 8-42 and:

$$Z'_M = \begin{bmatrix} Z_m & Z_m & Z_m \\ Z_m & Z_m & Z_m \\ Z_m & Z_m & Z_m \end{bmatrix}$$

$$Z_m = \frac{1}{N_c^2} \cdot \sum_{m=1}^{N_c} \sum_{n=1}^{N_c} Z_{mn}$$

Practical Cable Cross-Bonding of Underground Cables

Practical transposition can also be performed on underground, single-core coaxial cables called 'Cross-Bonding'. This method is similar to that performed on overhead lines, but in this case the respective conducting layers are switched, while the core conductor remains untouched. Please note that Ideal transposition is not available for underground cables systems in the LCP.
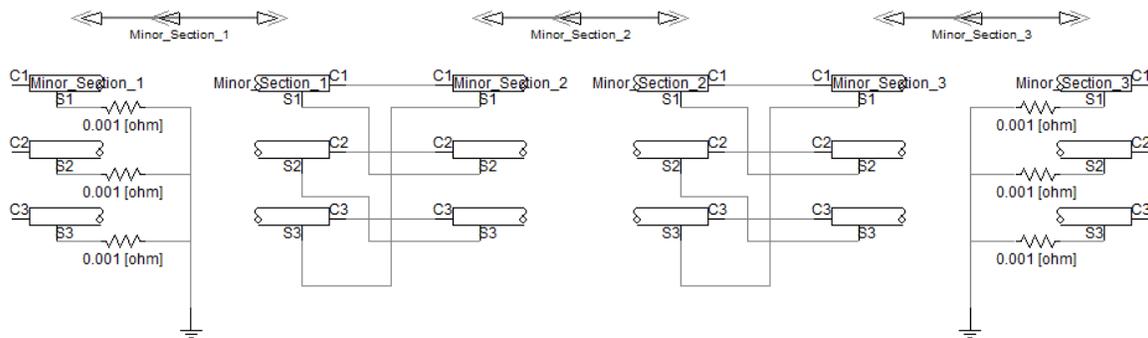

Figure 8-17: Major Section of Cross-Bonded Sheaths for Three Single-Core Cables in PSCAD

Ideal Cross-Bonding of Cables

Many practical underground cable systems employ the cross-bonding of sheath conductors in order to reduce circulating currents. The sheaths are transposed at regular intervals (similar to transposition of phases in the overhead lines in many ways). One way to represent a cross-bonded cable is by splitting the cable into small cable sections (as described above), where the sheaths between two cable sections are interchanged accordingly. However, in many practical situations, the number of cable sections may be very high (ex. A 25 km cross-bonded cable with 400 m cable sections needs 62 unique cable models in the simulation project). The 62 cable models require higher computational effort and memory compared with a single 25 km cable model and also much smaller time step significantly reducing the simulation speed.

Ideal cross-bonding is an alternative way to include cross-bonding to the existing cable line model (see reference [31]). The cable sheath can be assumed as cross-bonded through the entire length of the line (similar to the ideal transposition option in overhead lines). This is a good approximation for situations where there are many cross-bonded sections. This method gives sufficient accuracy for studies including switching transient studies and fault analysis. An additional advantage of this method is that simulation speed is much higher than the conventional approach.

Simulation Results

Figures 8-18a and 8-18b show the simulation results for a three cable system. A step voltage is applied to the conductor of the first cable and receiving-end of the same cable is short-circuited at $t = 0.1\ s$. All other conductors (including sheaths) are connected to ground through 1 Ω resistance. The figures compare the current through the conductor of the first cable and the induced voltage on the conductor of the second cable for three cases, ideal cross-bonded cable system, actual cross-bonded with 9 cross-bonded major sections and a non cross-bonded system. There is a significant mismatch between the cross-bonded and non cross-bonded cable results. Simulation results for ideal cross-bonded system are in a close agreement with cross-bonded case. If there are large number of cross-bonded sections (typical in many practical applications), the ideal cross-bonding is a good approximate for the actual cross-bonded system.
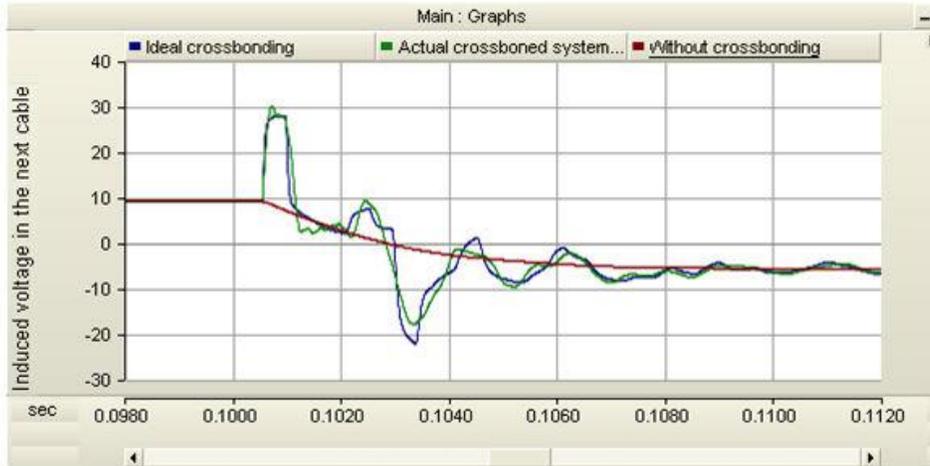
Figure 8-18a: Comparing Induced Voltage in Ideal, Actual and Non Cross-Bonded Systems
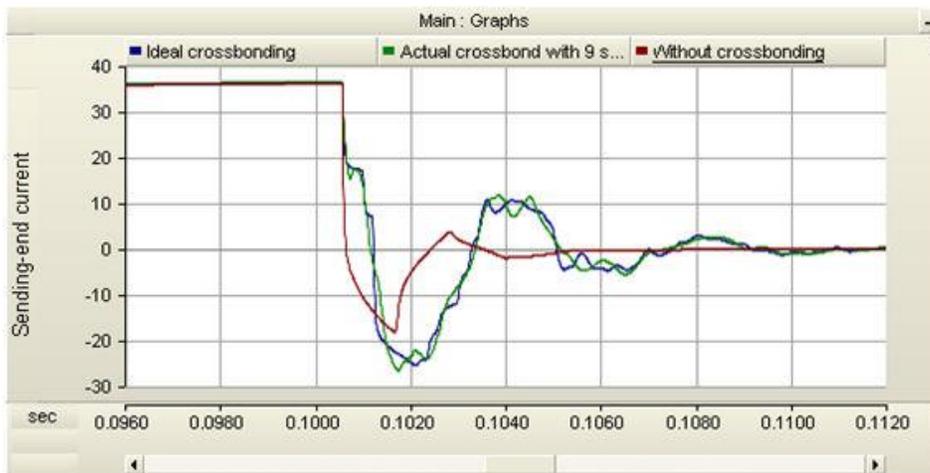
Figure 8-18b: Comparing Sending-End Current in Ideal, Actual and Non Cross-Bonded Systems

The YZ and ZY Matrices

The fundamental equations describing the properties of a transmission system in terms of impedance **Z** and admittance **Y** were defined in previous sections.

Taking the second derivative of these equations results in:

$$\frac{d^2V}{dx^2} = ZY \cdot V \qquad\qquad (8\text{-}48)$$

$$\frac{d^2I}{dx^2} = YZ \cdot I \qquad\qquad (8\text{-}49)$$

The **YZ** matrix is an important parameter as it is a key ingredient within the Line Constants Program for eigenvalue/eigenvector analysis.

Eigenvalue/Eigenvector Analysis

The derivation of system eigenvalues/eigenvector is a very important step in the overall process of solving the line constants. One of the primary purposes of eigenvector matrices for example, is to provide a transformation medium to shift between the modal domain and the phase domain.

The general procedure is to ascertain the eigenvalues and eigenvectors of the **YZ** product matrix. Valid eigenvectors of **YZ** are generally in complex matrix form with dimension *n x n*. For each **YZ** eigenvector matrix T_i, there are *n* eigenvalues λ (also complex). The eigenvector matrix T_i can be derived from the following eigenproblem equation:

$$T_i^{-1} \cdot YZ \cdot T_i = \hat{\lambda}$$

(8-50)

Where,

$\hat{\lambda} =$ The diagonal eigenvalue matrix

Frequency Dependent Analysis

As mentioned in previous discussions, a primary step in representing the full frequency-dependence of a transmission system is to solve the system matrices (i.e. **Y**, **Z** and **YZ**) at a finite number of points in the frequency domain. A key requirement to ensuring that this frequency-dependence is efficiently conveyed from the frequency domain to the time domain is to produce some rational approximations to key properties: In the Frequency Dependent (Phase) model for example, these properties are primarily the characteristic admittance matrix Y_0 and the propagation function matrix **H**.

The **YZ** matrix is frequency dependent, and therefore T_i will also vary with frequency. So as to guarantee a minimum complexity in curve fitting frequency dependent properties that are derived from T_i, such as Y_0 and **H**, the T_i eigenvector matrix should be smooth and free of any abrupt changes or discontinuities in the frequency domain. In more simple systems, this is usually the case. However, in highly asymmetrical overhead lines, or multi-layered, multi-cable underground systems, the elements of T_i may not behave well at all. In fact, eigenvalues may switch places or 'swap' at certain frequency points.

The Line Constants Program employs a Newton-Raphson technique, based on that described in [18], which ensures a smooth and well behaved T_i matrix as a function of frequency - for all types of transmission systems.

There are *n x n* equations contained within Equation 8-50 above, due to the fact that there is a set of *n* equations for each element λ_{kk} of the matrix λ. Each eigenvalue λ_{kk} and corresponding eigenvector matrix column T_i can be solved separately as follows:

$$F(x) = (YZ - \hat{\lambda}_{kk} \cdot U) \cdot T_{I(k)} = 0$$

(8-51)

Where,

$X =$ The column vector of unknowns (i.e. λ_{kk} plus $T_{I(k)}$)

The equation set **F(x)** is combined with its corresponding Jacobian matrix **J(x)** to form the Newton-Raphson iterative equation

$$X_{new} = X_{old} - J(X_{old})^{-1} \cdot F(X_{old})$$

(8-52)

Once the convergence criterion is met (see below), the remaining columns of T_I are solved separately and sequentially in the same manner as above.

Seed Values

The Newton-Raphson procedure will normally converge within a few iterations, provided that the starting (or seed) value is an accurate estimate. The LCP uses a stable, root squaring method to determine reasonable seed values at the starting frequency f_0 [24].

From this point onwards (i.e. frequencies f_1 to f_n), the new vector of unknowns x_{new}, calculated by the Newton-Raphson procedure, is used as a seed value in the subsequent iteration.

Eigenproblem Convergence Condition

The Line Constants Program iterates through Equation 8-52 until the following condition is met.

$$\sum_{j=1}^{n} \sum_{i=1}^{n} F(x_{new})_{i,j} \leq \varepsilon \quad \text{where,} \quad \varepsilon \approx 0 \tag{8-53}$$

ε is a constant parameter, where $\varepsilon = 10^{-8}$).

Modal Analysis

Generally speaking, transmission systems often consist of several mutually coupled phases or conductors. Modal analysis provides a means for separating each mutually coupled phase into a related but independent, single-phase transmission system. In terms of a system matrix, this means that the matrix will become diagonal, where all off-diagonal elements are zero. Each of these independent phases is referred to as a *mode*.

In unbalanced transmission systems, each mode will possess a unique characteristic impedance and travel time. Usually there are two main types of modes: Firstly the ground mode (otherwise known as the *common* mode or *zero-sequence* mode), which is active whenever ground currents flow in the system. And secondly, the remaining modes are known as metallic modes, *differential* modes or *positive* and *negative* sequence modes. The ground mode normally possesses a longer travel time, greater characteristic impedance and larger resistance than the metallic modes.

Phase quantities in the Line Constants Program are transformed into the modal domain by using the **YZ** eigenvector matrix T_I and the **ZY** eigenvector matrix T_V. For example, the modal impedance and admittance matrices are derived as follows:

$$Z^m = T_V^{-1} \cdot Z \cdot T_I \tag{8-54}$$

$$Y^m = T_I^{-1} \cdot Y \cdot T_V \tag{8-55}$$

$$T_I^{-1} = T_V^T \quad \text{and} \quad T_V^{-1} = T_I^T \tag{8-56}$$

Where,

Z^m =   Modal series impedance matrix

Y^m =   Modal shunt admittance matrix

T_I^T =   Transpose of the **YZ** eigenvector matrix

T_V^T =   Transpose of the **ZY** eigenvector matrix

When there is only a single conductor above a ground plane, there is just one mode. The only 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:
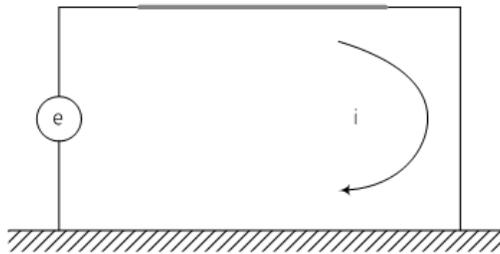
Figure 8-18 – Single Conductor (Single Mode) with Ground Return

For a two-conductor transmission system, such as a bipolar DC line, there are two paths the return current can flow in as shown in below:
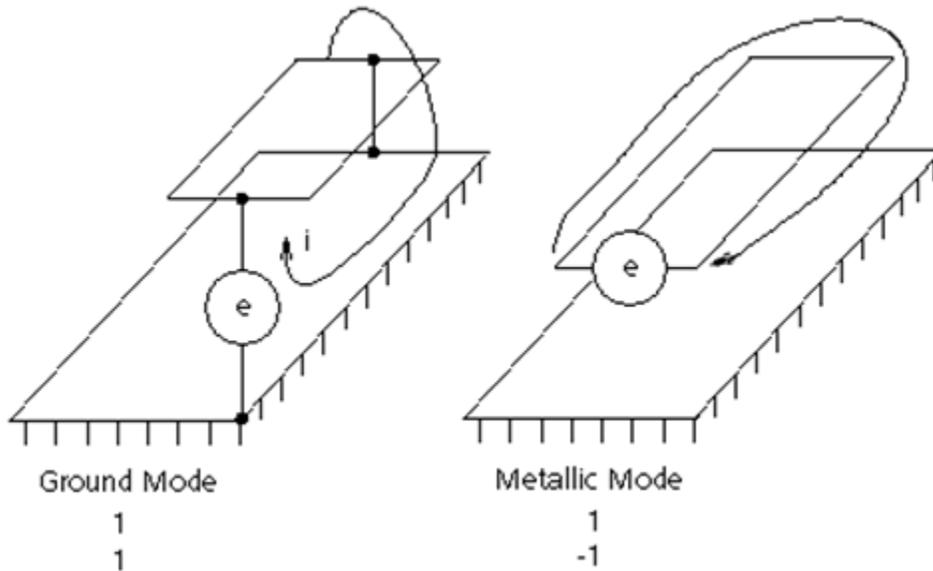
Figure 8-19 – Two Modes of a Two-Conductor Transmission Line

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:

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

(8-57)

and when normalized it becomes,

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

(8-58)

It is interesting to note that the transformation matrix for a single and double-conductor flat configuration 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 these conditions, the transformation matrix becomes that consisting of Clark components:

$$T_i = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 2 \\ 1 & -1 & -1 \end{bmatrix}$$

(8-59)

and when normalized becomes,

$$T_i = \begin{bmatrix} 0.5774 & 0.7071 & -0.4082 \\ 0.5774 & 0 & 0.8165 \\ 0.5774 & -0.7071 & -0.4082 \end{bmatrix}$$

(8-60)

The physical realization of the Clark transformation is shown in the following figure:
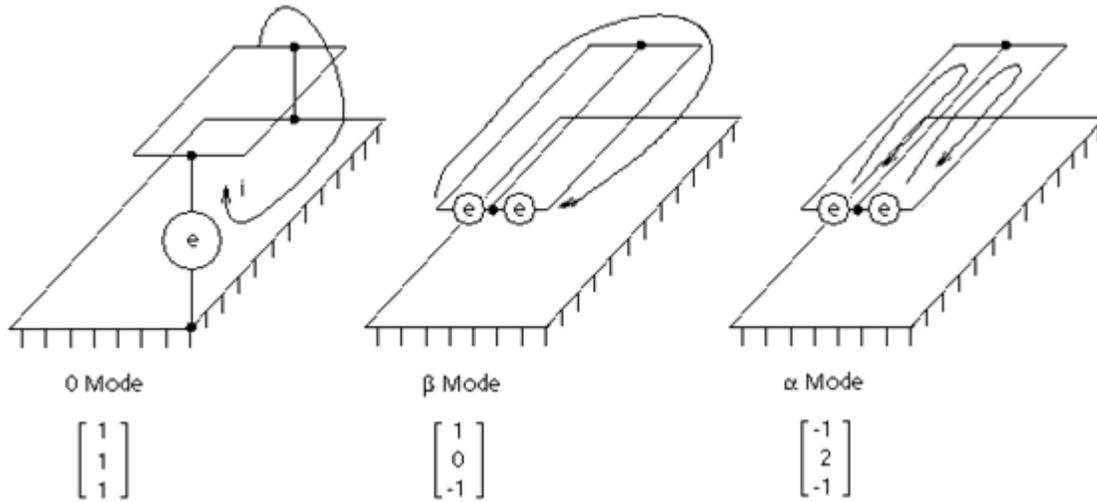
Figure 8-20 – Three Modes of a Three-Conductor Transmission Line using Clark Components

As a three-phase circuit is not usually perfectly balanced, $T_I \neq T_V$ and it is left for the eigenvector/eigenvalue solver to determine a set of frequency-dependent transformation matrices.

Curve Fitting

In the Line Constants Program, curve fitting is the final step in the process of solving a frequency-dependent transmission system. The primary purpose of the fitting routine is to consider a set of frequency domain response points, and fit this data with a low order, rational function approximation.
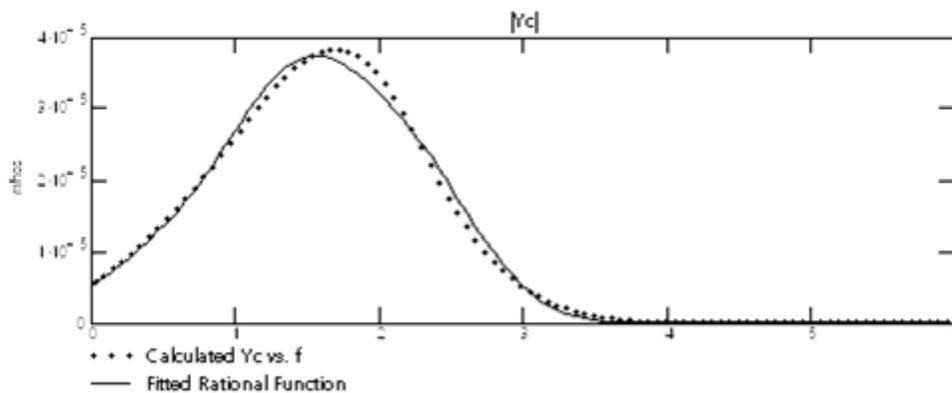
Figure 8-21: Characteristic Admittance Magnitude: Calculated vs. Fitted

NOTE: Figure 8-21 illustrates this concept for the characteristic admittance $Y_C(s)$ verses *log(f)*. Here the average RMS fitting error is about 0.2%.

This linear expression is then provided to EMTDC, so that it may be convolved into the time domain and used to produce an equivalent, two-port interface to the EMTDC electric network. The particulars of this circuit formulation are model dependent, and are described in the section EMTDC Distributed Branch Interface later in this chapter.

Both of the frequency-dependent models in PSCAD utilize a method for rational fitting of frequency-domain responses called *Vector Fitting*.

Vector Fitting

The vector fitting algorithm was first developed by Bjørn Gustavsen and Adam Semlyen in 1996, and the source was made available to the public soon after. It is the core of the Line Constants Program curve fitting algorithm and is now used for both the Frequency Dependent (Mode) and the Frequency Dependent (Phase) models.

> **NOTE**: A quick overview of the vector fitting process is provided here: If the reader would like a more detailed account, please see Reference [21], or visit www.sintef.no.

Generally speaking, the vector fitting algorithm takes a nonlinear, rational approximation of the form,

$$f(s) = d + s \cdot h + \sum_{n=1}^{N} \frac{c_n}{s - a_n} \tag{8-61}$$

Where,

$c_n =$ Residues (can be complex)

$a_n =$ Poles (can be complex)

$d, h =$ Real constants

And rewrites it as a linear problem of type **Ax** = **b**, and then determines the poles and zeros.

Given a reasonable set of starting poles $\overline{a}_n$, *f(s)* is multiplied by an unknown function *g(s)*, and a rational approximation equation for *g(s)* is introduced:

$$\begin{bmatrix} g(s) \cdot f(s) \\ g(s) \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} \frac{c_n}{s - \overline{a}_n} + d + s \cdot h \\ \sum_{n=1}^{N} \frac{\tilde{c}_n}{s - \overline{a}_n} + 1 \end{bmatrix} \tag{8-62}$$

Equation 8-62 can be rewritten as:

$$\left(\sum_{n=1}^{N} \frac{c_n}{s - \bar{a}_n} + d + s \cdot h \right) \approx \left(\sum_{n=1}^{N} \frac{\tilde{c}_n}{s - \bar{a}_n} + 1 \right) \cdot f(s)$$

(8-63)

$$(gf)_{fit}(s) \approx g_{fit}(s) \cdot f(s)$$

Equation 8-63 is linear with unknowns c_n, d, h and $\tilde{c}_n$, and may be written in the form $\boldsymbol{A}_k \boldsymbol{x} = \boldsymbol{b}_k$ for a range of frequency points, where:

$$\boldsymbol{A}_k = \left[\frac{1}{s_k - a_1} \quad \cdots \quad \frac{1}{s_k - a_N} \quad 1 \quad s_k \quad \frac{-f(s_k)}{s_k - a_1} \quad \cdots \quad \frac{-f(s_k)}{s_k - a_N} \right]$$

$$\boldsymbol{x} = \left[c_1 \quad \cdots \quad c_N \quad d \quad h \quad \tilde{c}_1 \quad \cdots \quad \tilde{c}_N \right]$$

(8-64)

$$\boldsymbol{b}_k = f(s_k)$$

The unknown quantities are then solved as a least squares problem.

With the unknown quantities solved, a rational approximation of *f(s)* can be derived from Equation 8-63. This is clear if rewritten as follows:

$$(gf)_{fit}(s) = h \cdot \frac{\prod_{n=1}^{N+1} (s - c_n)}{\prod_{n=1}^{N} (s - \bar{a}_n)}$$

(8-65)

$$g_{fit}(s) = h \cdot \frac{\prod_{n=1}^{N} (s - \tilde{c}_n)}{\prod_{n=1}^{N} (s - \bar{a}_n)}$$

(8-66)

$$f(s) = \frac{(gf)_{fit}(s)}{g_{fit}(s)} = h \cdot \frac{\displaystyle\prod_{n=1}^{N+1}(s - c_n)}{\displaystyle\prod_{n=1}^{N}(s - \tilde{c}_n)} \tag{8-67}$$

Equation 9-60 illustrates that the poles of *f(s)* become equal to the zeroes of *g$_{fit}$(s)*. Therefore, the poles of *f(s)* can be determined more economically by instead solving the zeros of *g$_{fit}$(s)*. This same procedure can be used to solve for the residues *c$_n$* of *f(s)*. See reference [21] for more details on Vector Fitting.

The Frequency Dependent (Phase) Model

The Frequency Dependent (Phase) model is based on what is referred to as the *Universal Line Model* concept described in references [20], [22] and [23]. This model requires that two separate parameters be fitted: The characteristic admittance *Y$_C$(s)* and the propagation function *H(s)*.

Propagation Function Fitting (Mode-Based Method)

Fitting of the propagation function *H(s)* is essentially a two-step process. First, the modes of *H(s)* as given by:

$$H = T_I \cdot H^m \cdot T_I^{-1} \tag{8-68}$$

Where,

$T_I =$ The current eigenvector matrix

$H^m =$ The modal propagation matrix

are used to derive an upper frequency limit ⬚, so that a time delay ⬚ for each mode can be calculated:

$$\tau = \frac{\ell}{v(\Omega)} + \frac{\angle h_{min}(\Omega)}{\Omega} \tag{8-69}$$

NOTE: See Reference [22] for more details on calculating the time delay ⬚.

Where,

$\tau =$ Modal time delay [s]

$\angle h_{min} =$ A value derived by Bode [4]

$\ell =$ The transmission system length [m]

$v =$ Modal velocity [m/s]
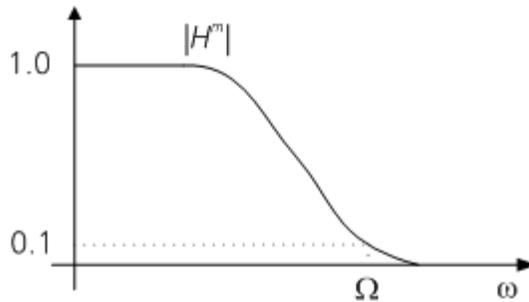
Figure 8-22 – Finding the Upper Frequency Limit based on the Magnitude of H^m

The time delays extracted for each mode are then compared. Modes with very similar time delays are sorted into groups called *delay groups*, where further analysis is performed on a delay-group basis. This grouping of like modes helps to increase the speed of the fitting solution for **H(s)** by effectively reducing the number of modes. This is especially effective in systems with geometric symmetry.

The group time delay is then used to 'back-wind' each respective delay group. The back-wound modal propagation function is then fitted as follows:

$$e^{s\tau} \cdot H_i^m(s) = \sum_{m=1}^{N} \frac{c_m}{s - a_m}$$

(8-70)

The resulting poles from this fit are used as starting poles in fitting **H(s)** in the phase domain:

$$H(s) = \sum_{i=1}^{Ng} \left(\sum_{m=1}^{N} \frac{c_{m,i}}{s - a_{m,i}} \right) \cdot e^{-s\tau}$$

(8-71)

Where,

$Ng =$ The number of delay groups

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 improved by decreasing the fitting order of magnitude (i.e. decreasing the maximum number of poles).

Characteristic Admittance Fitting
Fitting of the characteristic admittance **Y_c(s)** is a relatively straight-forward procedure when compared to that of **H(s)**. Since **Y_c(s)** possesses no time delays, and good starting poles can be found by fitting the sum of all modes. However, due to the relation,

$$\sum_{i=1}^{N} \lambda_i = \sum_{i=1}^{N} A_{ii}$$

(8-77)

Where,

$$A =$$ A square matrix

$$\lambda_i =$$ Eigenvalues of A

all that is needed is to sum the diagonal elements of $Y_C(s)$. The resulting sum as a function of frequency is referred to as the trace of $Y_C(s)$, or:

$$Tr\big[Y_C(s)\big] = \sum_{i=1}^{N} y_{C\,ii} \tag{8-78}$$

This function is approximately fitted by,

$$f(s) = d + \sum_{m=1}^{N} \frac{c_m}{s - a_m} \tag{8-79}$$

NOTE: The proportional term h as shown in Equation 8-61 is set to zero in Equation 8-79 due to the fact that the characteristic admittance $Y_C(s)$ normally converges to a constant value.

The resulting poles from this approximation are then used as starting poles for fitting the actual elements of $Y_C(s)$ in the phase domain, again using 8-79.

The Frequency Dependent (Mode) Model

The Frequency Dependent (Mode) model utilizes methods quite different from the *Frequency Dependent (Phase)* model described above. It relies on a rational function approximation of different parameters; these being the attenuation function $A(s)$ and the characteristic impedance $Z_0(s)$.

Passivity Checking

The stability of a transmission line model can be ensured by verifying a property called *passivity*. The physical meaning of passivity, with regard to electrical circuits, is that passive networks always absorb real power and thus the total energy delivered to the network is positive. Although transmission lines are passive as a matter of physical reality, the transmission line model may be not passive due to the errors introduced during the various approximations (i.e. curve-fitting, etc.).
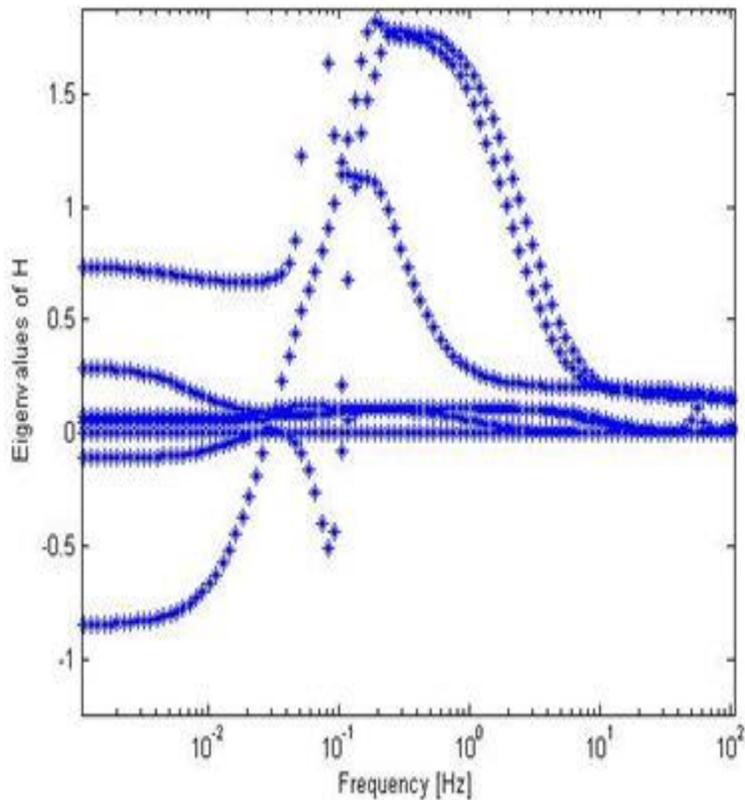
Figure 8-23 – Simple Three Single-Core Cable Configuration

The passivity can be guaranteed by verifying that the eigenvalues of the Hermitian matrix derived from transmission line parameters are positive for the frequency range of interest. The above graph shows the eigenvalues of H for particular cable configuration. The presence of negative eigenvalues at very low frequencies indicates the non-passive cable model which may lead to unstable simulation. In power system studies, the unstable simulation may be frequently observed, especially when simulating HVDC underground cables and overhead lines. The proper fitting parameter selection may help to obtain passive model and hence stable simulation. The following options can be used to verify passivity of the model.

Passivity checking is controlled from within the Frequency Dependent (Phase) Model component.

See chapter references [37] and [38] for more details.

DC Correction

High voltage direct current (or HVDC) transmission over long distances has seen ever increasing application all over the world. Simulation models for such systems are continually being pushed to the limit; requiring accuracy over a wide range of frequencies, including *0 Hz* or DC. Up until recently, transmission properties at DC were based on a best approximation, where the error could vary greatly depending on the system being simulated.

As discussed in previous sections, the use of modern phase domain modeling techniques, coupled with parameter estimation using Vector Fitting, has greatly improved the accuracy of time-domain models for transmission lines and cables. Although the frequency-dependent models simulate the frequency range from a default of *0.5 Hz* to about *1 MHz*, it has been difficult to achieve a good fit in the proximity of *0 Hz* (DC). When dealing with HVDC lines and cables, it is very important to accurately reproduce the response at DC, as this is the nominal frequency of the line. It can be shown that forcibly trying to fit the characteristics at these extremely low frequencies requires high order fitting and sometimes leads to inaccurate results.

A feature exists in the Line Constant Program that modifies the form of the rational function, approximated in the curve fitting procedure, when using the Frequency-Dependent (Phase) model only. The rational function approximations, for both the propagation and the characteristic admittance matrices, can be fitted more accurately without having to substantially increase the number of poles. In fact, with this approach the DC response is exact! This feature is called *DC Correction*.

DC Correction allows for two possible variants of the functional form. In the first approach, the admittance and propagation transfer functions are reformulated so that the DC response is factored out as an additive constant, which can then be directly selected. In the other approach, the transfer function is first fitted over the entire frequency range, which typically results in some fitting error at precisely DC. A low frequency first order pole is then added to the resultant fitted function in order to realize the exact response at DC, without significantly affecting the remainder of the frequency response.

Also see Chapter Reference [27] for more detailed reading.

Issues with Fitting Transfer Matrices at Low Frequency

At very low frequencies, the equations for *A* and *Y$_c$* reduce to,

$$A(s \to 0) \approx 1 - \sqrt{sCR_{dc}\ell} \tag{8-80}$$

$$Y_c(s \to 0) \approx 1 - \sqrt{\frac{sC}{R_{dc}}} \tag{8-81}$$

Where,

$C =$ Capacitance per unit length (F)

$R_{dc} =$ DC resistance of the line per unit length (Ω)

The square root term in Equations 8-80 and 8-81 does not permit a rational function approximation with a low order; and thus a higher order rational function may be needed for the fitting if very low frequencies are considered.

Consider for example, a simple three single-core coaxial cable configuration shown below:
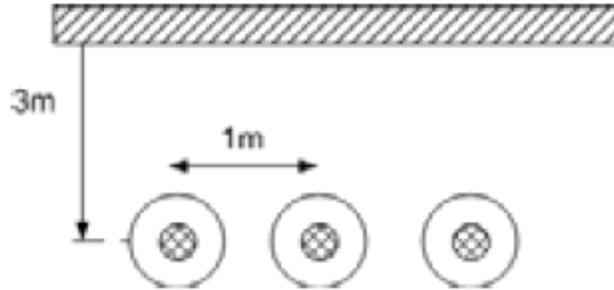
Figure 8-24 – Simple Three Single-Core Cable Configuration

A typical frequency response of the $Y_c(1,1)$ element (i.e. the core admittance of cable 1) is shown in Figure 8-25, which also shows a plot of a rational function approximation obtained by limiting the lower fitting frequency bound to *1 Hz*.
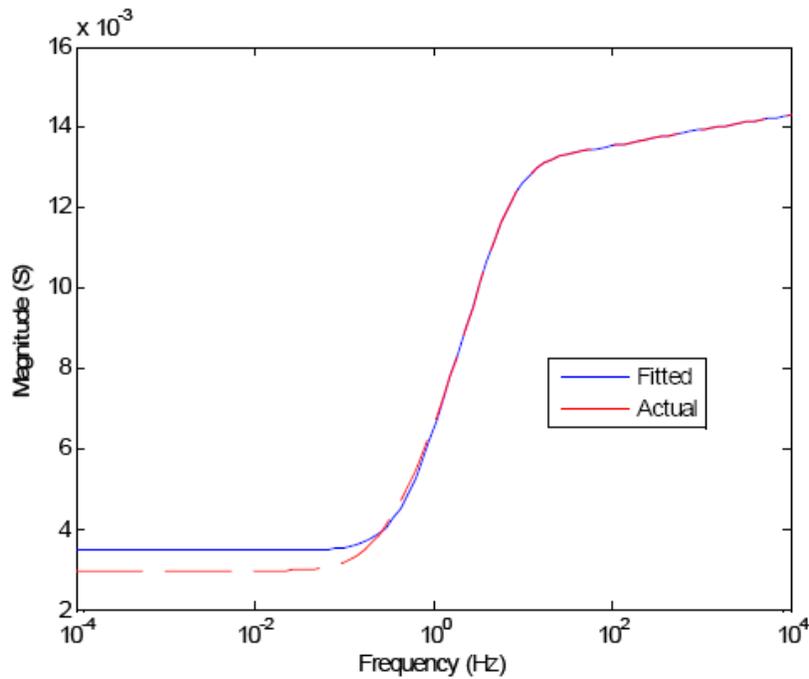
Figure 8-25 – Characteristic Admittance Response of a Simple Three Single-Core Cables

Notice that the fitting at frequencies below the lower bound is poor. The response with a lower bound of *1 Hz*, which is often selected by users when studying DC systems, can produce a significant steady-state error. Reducing the lower bound to *0.1 Hz*, reduces the error, but achieves this with a significant increase in the fitting order as discussed earlier. Note that poor fitting at very low frequencies is a major source of error when modelling DC lines.

DC Correction by Changing the Functional Form

If Equations 8-71 and 8-79 are re-written in an equivalent form (Equations 8-82 and 8-83 respectively), it is readily seen that setting *s = 0* results in a single term, which is the response at DC (i.e. the term $d_{dc,theoretical}$).

$$A_{i,j}^{mod}(s) \approx \sum_{n_1=1}^{N_1} \frac{c_{n_1} \cdot s e^{-s\tau_1}}{s - a_{n_1}} + \sum_{n_2=1}^{N_2} \frac{c_{n_2} \cdot s e^{-s\tau_2}}{s - a_{n_2}} + \cdots + d_{dc,theoretical} \cdot e^{-s\tau_1} \tag{8-82}$$

$$Y_{c_{i,j}}^{mod}(s) = \sum_{m=1}^{M} \frac{c_m \cdot s}{s - a_m} + d_{dc,theoretical} \tag{8-83}$$

By selecting $d_{dc,theoretical}$ to be precisely the known DC value, a perfect fit at DC is guaranteed. This approach has been introduced in the Frequency-Dependent (Phase) model, as discussed in [27]. These modified equations can be re-expressed in a similar form to Equations 8-71 and 8-79, so as to make the formulation amenable to the vector fitting process. Using a proper choice of variables, Equation 8-83 can be converted into a form as shown in Equation 8-84, which is suitable for vector fitting.

$$Y_{c_{i,j}}^{mod}(s) = \sum_{m=1}^{M} \frac{c_m''}{s'' - a_m''} + d_{dc,theoretical} \tag{8-84}$$

Where,

$$c_m'' = -\frac{c_m}{a_m} \quad s'' = \frac{1}{s} \quad a_m'' = \frac{1}{a_m}$$

A minor issue with the above method is that although the DC error is eliminated, the resultant propagation function at very large frequencies deviates marginally from zero; which is contrary to the physical properties of typical propagation functions. Although the error introduced is trivial, the terms in Equation 8-84 can be slightly perturbed, using another least squares fitting, to taper the high frequency response to zero, without altering the correct DC value.

As seen in Figure 8-26, this approach results in an excellent fit over the entire frequency range, without any increase in the order of the fitted function. The corresponding time domain results for a short circuit on the cable in Figure 8-24 is shown in Figure 8-27. Over a *10 second* interval, accurate reproduction of the response is shown when compared to a solution using numerical inverse Laplace transform methods.
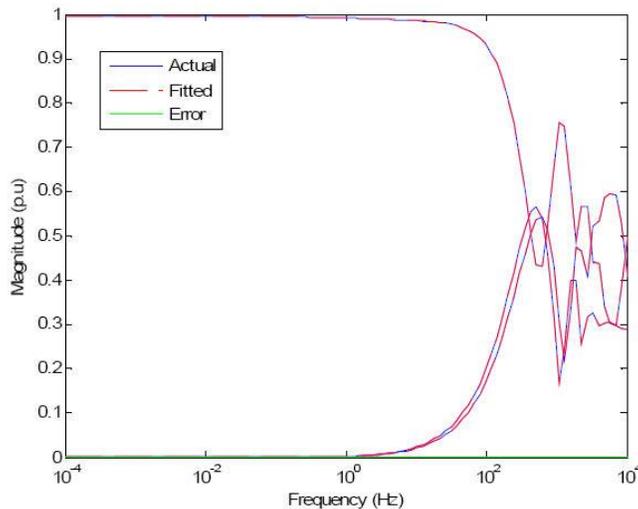
Figure 8-26 – Frequency Response Plots for the Elements of the First Column of the Propagation
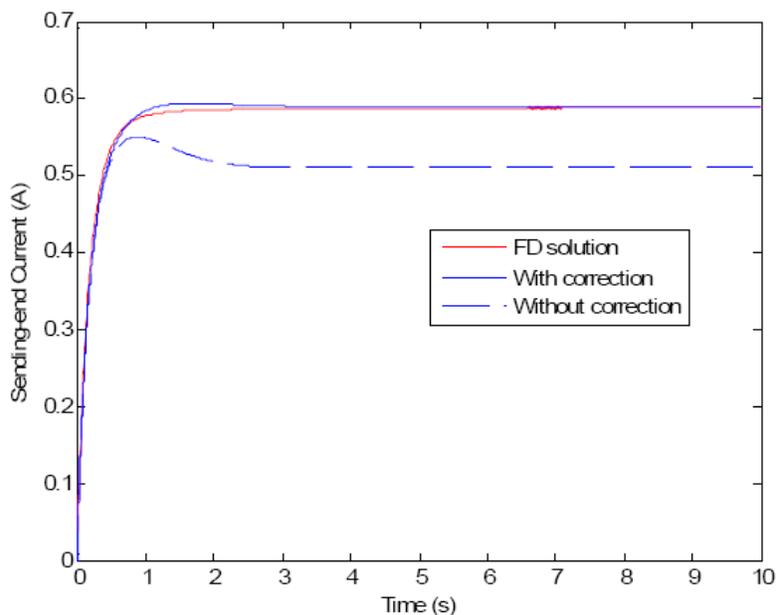
Figure 8-27 – Time domain Results for a Short Circuit Condition

DC Correction by Adding a Pole/Residue

The known characteristics (i.e. elements of the admittance and propagation matrix) are first fitted with a rational polynomial, as is done conventionally for the Frequency-Dependent (Phase) model. First, a real pole $a_0 \in (2\pi \cdot f_{min} \cdot k), k < 1$ with a suitable residue c_0 is added to it, so that the modified function gives the exact value at DC. This modification increases the order of the rational function by one order of magnitude, and does not affect the high frequency asymptote. Also, as the cut-off frequency of the additional term is smaller than the lower fitting bound, this correction is achieved with a very small error to the fitted part.

$$f^{mod}(s) = \frac{c_0}{s - a_0} + f^{fitted}(s)$$

(8-85)
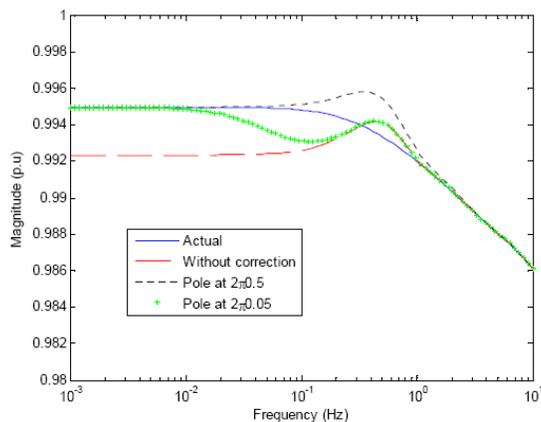
Figure 8-28 – Magnitude of **A(1,1)** Before and After the Addition of Pole/Residue

The choice of a_0 (or k in above paragraph) is selected by another optimization process that minimizes the error between the actual frequency response, and that of f_{mod}. As seen in Figure 8-28, the pole at frequency $2\pi \times 0.5$ Hz gives the most accurate response at frequencies approaching DC, as well as the closest fit over the entire low frequency range. The corresponding time domain simulations for the line current are shown in Figure 8-29, where the superior accuracy of the $2\pi \times 0.5$ Hz pole in the simulation is evident. In the above case, the original function was fitted with a lower frequency bound of *1 Hz*, so as to limit the transfer functions to lower orders.
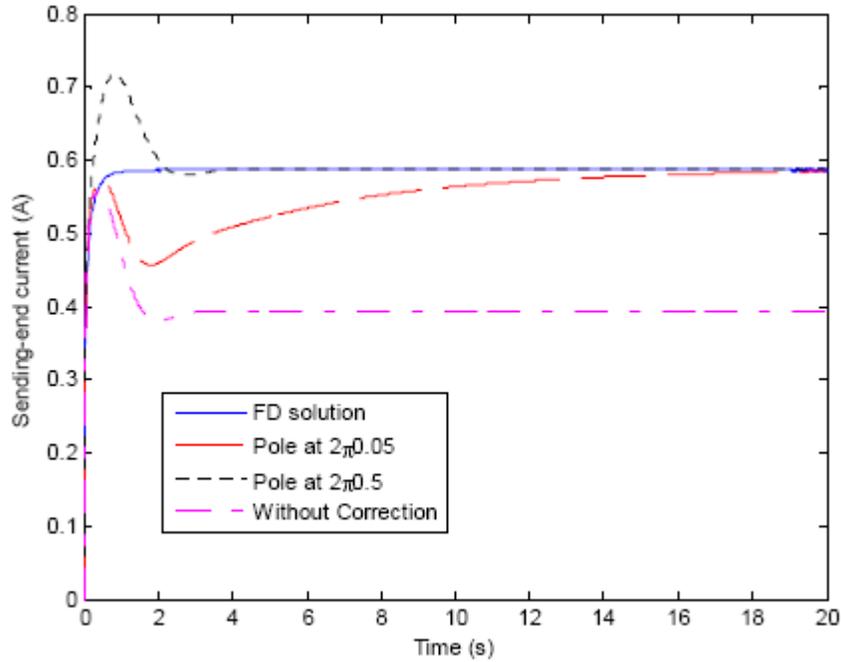
Figure 8-29 – Sending-End Current of First Conductor with Different Poles Selected

Line Constants Program Output

The Line Constants Program can be set to provide an assortment of output data, which can be classified into two types: Frequency domain data and steady-state (single frequency) data.

Output parameters are written to files with the extension *.out, all of which can be found in the PSCAD Temporary Directory (*.emt) for the case project, following a transmission system solve. Each output file will possess a unique name, which consists of the line name prefixed to the name of whatever the quantity exists in the file:

<systemname><quantityname>.out

The only exception to this naming rule is the steady-state output file, which does not have a quantity name.

Steady-State (Single-Frequency) Output
A single file is used by the LCP to display data calculated at a single-frequency. By default, this frequency is the *Steady-State Frequency* entered in the Transmission Line or Cable Configuration component. However, this default may also be changed by using the Additional Options component from the Master Library.

A detailed description of this file is provided in Chapter 8 of the PSCAD Manual under the heading Line Constants Files.

Detailed (Multi-Frequency) Output

When solving frequency-dependent transmission systems, the Line Constants Program can be set to provide output details on some frequency-domain system parameters, the type of which is determined by the model used. When enabled, detailed data is written to a collection of text files in the PSCAD temporary (*.emt) directory, and each file includes data plotted verses both frequency f_k and $log(f_k)$. If the parameter is a matrix or vector, every single element will be included in the file as a distinct scalar quantity. If the data is complex, then a file will be produced separately, both for the magnitude and phase angle of each matrix or vector element. In some cases, the calculated frequency-domain quantities are provided alongside the respective curve fitted quantities. This enables analysis of the curve fitting results and provides feedback for 'tuning' the transmission system.

PSCAD provides a utility called the Detailed Output Viewer, which is programmed specifically for viewing these files.

The parameters written as detailed output depend on the frequency-dependent model being used to solve the system. The following tables summarize the parameters output for each model:

Common to All Frequency-Dependent Models

File Name	Description
*_zm.out	Series Impedance Matrix Magnitudes $\left\|Z(f_k)\right\|$
*_zp.out	Series Impedance Matrix Phase Angles $arg(Z(f_k))$
*_ym.out	Shunt Admittance Matrix Magnitudes $\left\|Y(f_k)\right\|$
*_yp.out	Shunt Admittance Matrix Phase Angles $arg(Y(f_k))$
*_lamdam.out	YZ Eigenvalue Vector Magnitudes $\left\|\lambda(f_k)\right\|$
*_lamdap.out	YZ Eigenvalue Vector Phase Angles $arg(\lambda(f_k))$

Frequency Dependent (Mode) Model

File Name	Description
*_timm.out	YZ Eigenvector (Modal Transformation) Matrix Magnitudes (Calculated/Fitted) $\left\|T_l(f_k)\right\|$
*_tipm.out	YZ Eigenvector (Modal Transformation) Matrix Phase Angles (Calculated/Fitted) $arg(T_l(f_k))$
*_amm.out	Modal Propagation Function Magnitudes (Calculated/Fitted) $\left\|A^m(f_k)\right\|$
*_apm.out	Modal Propagation Function Phase Angles (Calculated/Fitted) $arg(A^m(f_k))$
*_zcmm.out	Modal Characteristic Impedance Vector Magnitudes (Calculated/Fitted) $\left\|Z_C^m(f_k)\right\|$
*_zcpm.out	Modal Characteristic Impedance Vector Phase Angles (Calculated/Fitted) $arg(Z_C^m(f_k))$

Frequency Dependent (Phase) Model

File Name	Description	
*_timp.out	YZ Eigenvector (Modal Transformation) Matrix Magnitudes	$\left|T_l\left(f_k\right)\right|$
*_tipp.out	YZ Eigenvector (Modal Transformation) Matrix Phase Angles	$arg\left(T_l\left(f_k\right)\right)$
*_ycmp.out	Characteristic Admittance Magnitudes (Calculated/Fitted)	$\left|Y_C\left(f_k\right)\right|$
*_ycpp.out	Characteristic Admittance Phase Angles (Calculated/Fitted)	$arg\left(Y_C\left(f_k\right)\right)$
*_hmp.out	Propagation Function Matrix Magnitudes (Calculated/Fitted)	$\left|H\left(f_k\right)\right|$
*_hpp.out	Propagation Function Matrix Phase Angles (Calculated/Fitted)	$arg\left(H\left(f_k\right)\right)$
*_hmm.out	Modal Propagation Function Vector Magnitudes (Calculated/Fitted)	$\left|H^m\left(f_k\right)\right|$
*_hpm.out	Modal Propagation Function Vector Phase Angles (Calculated/Fitted)	$arg\left(H^m\left(f_k\right)\right)$

P-Section Equivalent Circuits

Transmission systems considered to be electrically short (i.e. the propagation delays are negligible) can be represented with reasonable accuracy by a simple circuit consisting of a series-impedance and shunt-admittance. If the total shunt-admittance is halved and placed at each end of the circuit with the series-impedance between the two ends, the resulting equivalent circuit is referred to as a ⷂ-section [16].
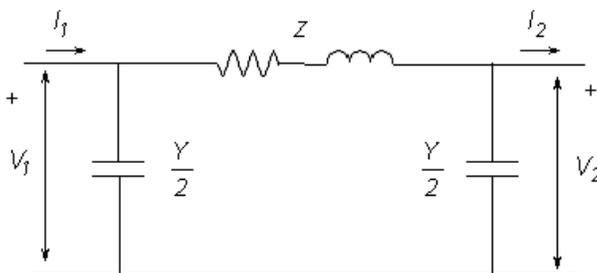

Figure 8-30 – Single-Phase ⷂ-Section Equivalent Circuit

Where,

$$V_1 = \left(V_2 \cdot \frac{Y}{2} + I_2\right) \cdot Z + V_2 \tag{8-86}$$

and,

$$V_1, I_1 \rightarrow$$ The sending end voltage and current respectively

$$V_2, I_2 \rightarrow$$ The receiving end voltage and current respectively

$$Z \rightarrow$$ Total series impedance

$$Y \rightarrow$$ Total shunt admittance (normally purely capacitive)

PSCAD Coupled P-Section Model

The Line Constants Program is not used by EMTDC for the calculation of π-section parameters. This is due to the fact that π-sections are entirely passive, and can be represented directly as part of the EMTDC conductance matrix. The PSCAD Master Library includes a 'coupled' π-section model, which equivalences a single, three-phase, mutually coupled circuit. There is also a double circuit 'coupled' π-section component, where both circuits are three-phase and mutually coupled.
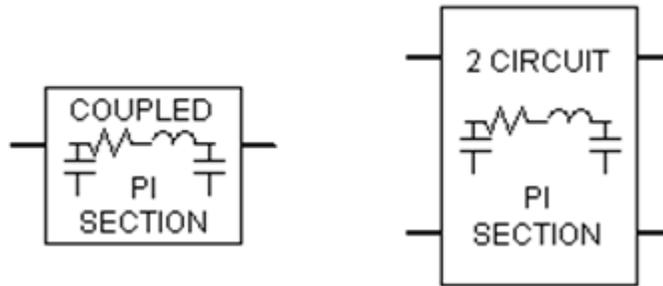
Figure 8-31 – Coupled π-Section Components in PSCAD

For more information on these components, please see the respective topics in the PSCAD On-Line Help.

Long-Line Correction

Long-line correction is required when attempting to accurately equivalence a long transmission line as a coupled π-section circuit. A 'long' transmission line can be defined as any line exceeding approximately *200* to *250* km in length [16], although this is surely open to interpretation.

Regardless of the actual defined length, π-section equivalent circuits will become less and less accurate as the line becomes longer. This is mainly due to the fact that a π-section does not account for the uniform distribution of line parameters. It is possible however, to derive a set of long-line corrected, lumped parameters so that the π-section equivalent circuit does indeed provide a better representation.

The Line Constants Program displays the long-line corrected data directly in the Output File (*.out). This information is provided specifically for users who wish to develop their own, multi-circuit π-section equivalents, using the Line Constants program as a starting point. The long-line correction is performed directly on the elements of the modal impedance Z_M and modal admittance Y_M matrices as follows:

$$Z_{LL} = Z_M \cdot \frac{sinh(\gamma \cdot \ell)}{\gamma \cdot \ell} \tag{8-87}$$

$$Y_{LL} = Y_M \cdot \frac{\tanh(\gamma \cdot \ell/2)}{\gamma \cdot \ell/2} \qquad (8\text{-}88)$$

Where,

$\gamma =$ Modal propagation constant

$\lambda =$ Eigenvalues of the **YZ** matrix

$\ell =$ Line length [m]

The long-line corrected modal parameters are then transformed back to the phase domain before being displayed in the output file.

EMTDC Distributed Branch Interface

In an electromagnetic transients program such as EMTDC (where a fixed time step Δt is assumed), the simulation is started with initial conditions at time *t = 0.0* (or from a snapshot file), and then a system solution is found at times *t = Δt, 2Δt, 3Δt,* etc. until the simulation end time is reached. When considering network branches with distributed parameters such as transmission lines and cables, some historical data must be buffered in order to deal with the finite travel times involved. This is easily taken care of since the system solution at past time steps (i.e. *t = t - Δt, t - 2Δt, t - 3Δt,* etc.) is already known.

EMTDC uses the *Method of Characteristics* (otherwise known as *Bergeron's Method*) to represent distributed parameter branches, which account for travel time delays, in electric networks. The following section provides a brief overview of this method. For more details, see [6].

Method of Characteristics
Consider an ideal (loss-less) transmission system with an inductance *L* and capacitance *C* per unit length. At a point *x* along the system, the voltage and current are represented as follows:

$$\frac{dv}{dx} = -L \cdot \frac{di}{dt}$$

$$\qquad (8\text{-}89)$$

$$\frac{di}{dx} = -C \cdot \frac{dv}{dt}$$

The general solution of for Equation 8-89 can be given in the following form:

$$v(x,t) + Z_0 \cdot i(x,t) = 2Z_0 \cdot f_1(x - \upsilon t)$$

$$\qquad (8\text{-}90)$$

$$v(x,t) - Z_0 \cdot i(x,t) = -2Z_0 \cdot f_2(x + \upsilon t)$$

Where,

$$Z_0 = \quad \text{The characteristic impedance [\Omega]}$$

$$\nu = \quad \text{The phase velocity [m/s]}$$

It can then be shown (see [6]) that for a transmission system of length, Equations 8-89a and 8-89b can be re-written in two-port format as:

$$i_k(t-\tau) = -\frac{1}{Z_0} \cdot v_m(t-\tau) - i_{m,k}(t-\tau)$$

(8-91)

$$i_m(t-\tau) = -\frac{1}{Z_0} \cdot v_k(t-\tau) - i_{k,m}(t-\tau)$$

Where,

$$\tau = \quad \ell \cdot \sqrt{L \cdot C} \quad \text{The transmission system travel time [s]}$$

$$\ell = \quad \text{The transmission system length [m]}$$

Equations 8-91a and 8-91b can be represented schematically as Norton equivalent circuits, representing the sending and receiving ends of the ideal, distributed branch:
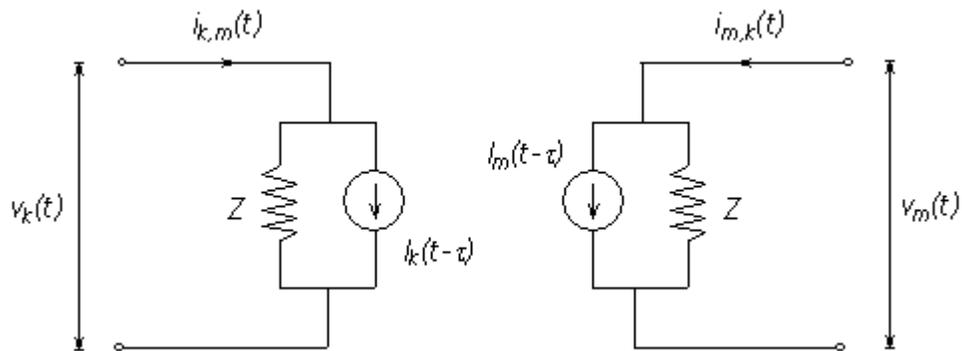

Figure 8-32 – EMTDC Distributed Branch Interface (Single-Phase)

Where, in the case of an ideal, loss-less system $Z = Z_0$.

The Bergeron Model

The Bergeron model is closely related to the Method of Characteristics described in the previous section. That is, it is essentially an ideal (loss-less) model represented by a distributed inductance *L* and a capacitance *C*. However, the Bergeron model goes a step further to include a lumped resistance property to approximate system losses [6].

It is important to note that the Bergeron model is a single-frequency model. That is, all calculated parameters, such as characteristic impedance Z_0, are calculated at a specified frequency: Usually either 50 or 60 Hz for an AC transmission line. Although the Bergeron model can indeed be used for transients simulation in the time domain (which includes all frequencies), only results at the specified steady-state frequency are meaningful. As such, the Bergeron model should only be used for general fundamental frequency impedance studies, such as relay testing or matching load-flow results.

Bergeron Method vs. Multiple P-Sections

The Bergeron model should generally be chosen over a ⬚-section equivalent whenever the system length is sufficient enough to allow the propagation of waves (at approximately the speed of light) to travel the entire length of the line within a single time step. This means that for a typical simulation time step of ⬚t = 50 ⬚s, transmission systems roughly over 15 km in length should be represented by the Bergeron model as opposed to a ⬚-section. The advantages are of course that a real world propagation delay is considered in the Bergeron model, and also that series connected ⬚-sections can introduce artificial resonances at high frequencies [6].

Inclusion of Line Resistance

The Bergeron model approximates losses by adding series lumped resistance elements into the loss-less, distributed parameter branch defined by the Method of Characteristics. Given the total system resistance *R* (as calculated by the PSCAD Line Constants Program), the loss-less line is broken into two segments, each with a ¼*R* resistance at each end. When these segments are combined, a lumped resistance of ½*R* in the middle and ¼*R* at each end results [6, 19].
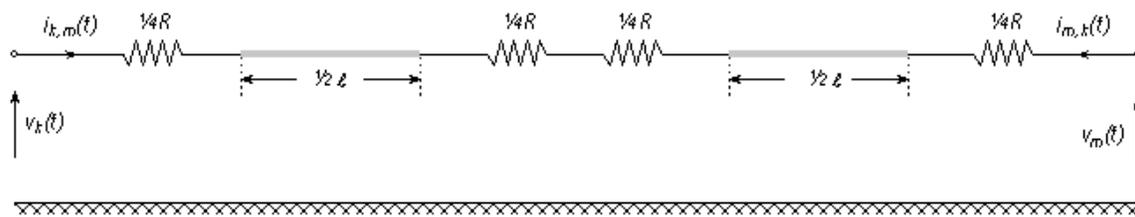
Figure 8-33 – Loss-Less Line with Lumped Resistance Included

Where,

$$R =$$ The total resistance of the transmission system [⬚]

The breaking of the system into two sections, along with the additional resistance elements, results in a change in the Norton representation of the loss-less line shown in Figure 8-33. The equivalent circuit of a loss-less line with lumped line resistance is given below:
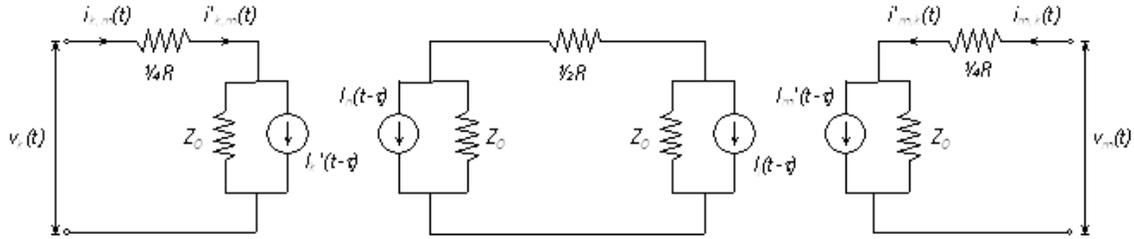
Figure 8-34 – Loss-Less Line with Lumped Resistance Equivalent Circuit

Time Domain Implementation

The circuit of Figure 8-34 can be collapsed into the same two-port format as the Distributed Branch Interface in Figure 8-32 (shown again below):
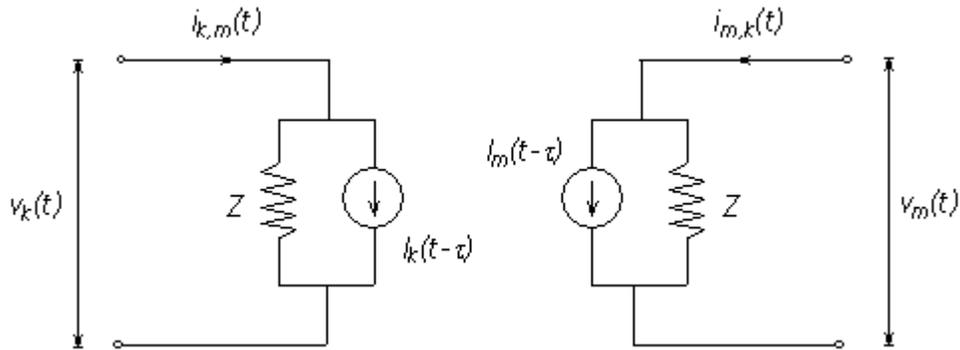
Figure 8-35 – EMTDC Bergeron Model Time Domain Interface (Single-Phase)

The Norton interface impedance Z at each end of the line now becomes:

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

(8-92)

The change in Norton impedance Z is reflected in the definition of the Norton current injections I_k and I_m as follows:

$$I_k(t-\tau) = \left(\frac{1+H}{2}\right) \cdot \left\{-\frac{1}{Z} \cdot v_m(t-\tau) - i_{m,k}(t-\tau)\right\} + \left(\frac{1-H}{2}\right) \cdot \left\{-\frac{1}{Z} \cdot v_k(t-\tau) - i_{k,m}(t-\tau)\right\}$$

(8-93)

$$I_m(t-\tau) = \left(\frac{1+H}{2}\right) \cdot \left\{-\frac{1}{Z} \cdot v_k(t-\tau) - i_{k,m}(t-\tau)\right\} + \left(\frac{1-H}{2}\right) \cdot \left\{-\frac{1}{Z} \cdot v_m(t-\tau) - i_{m,k}(t-\tau)\right\}$$

Where,

$$H = \frac{Z_0 - \dfrac{R}{4}}{Z_0 + \dfrac{R}{4}}$$

(8-94)

Frequency for Loss Approximation

The Bergeron model can actually calculate at two frequency points: The second point being a frequency chosen by the user (greater than the fundamental), for the specific purpose of providing additional attenuation at high frequencies. Note that the transmission system is not modeled exactly at the higher frequency because the characteristic impedance, travel time, etc., are still those calculated at the fundamental frequency.

> **NOTE**: This option can be invoked by choosing the 'Use Damping Approximation?' option in the Bergeron model.

Upon assembly of the of the Bergeron model interface, EMTDC will create a split at the log average ω_m of the two specified frequencies and determine the high and low frequency attenuation paths:

$$\omega_m = 10^{\left(\frac{log(\omega_0) + log(\omega_1)}{2}\right)}$$

(8-95)

Where,

$\omega_0 =$ The fundamental frequency [rad/s]

$\omega_1 =$ The 'Frequency for Loss Approximation' [rad/s]

The frequency for loss approximation ω_1 is normally selected to be in the range between 100 Hz and 2 kHz, which may be chosen to accommodate specific resonance conditions of interest.

The log average frequency ω_m is used to calculate the real pole time constant $\tau_c = \Delta t . \omega_m$, which is applied to the Bergeron model interface.

Shaping Time Constant

If a high frequency loss approximation is indeed performed, then the current is processed through a real pole with a shaping time constant of τ_s just before being injected into the line terminations.

Selection of the wave shaping constant τ_s 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 Bergeron line model to a known response, τ_s must be adjusted by trial and error. Care should be taken when using the wave shaping time constant, because although the real pole will attenuate the high frequency components of the line, it will also introduce an additional lag or time delay, which in turn affects the overall travel time and thus, the effective line impedance.

EXAMPLE 9-1:

Consider a single-phase Bergeron line model connected to an infinite source and terminated with a large resistance.
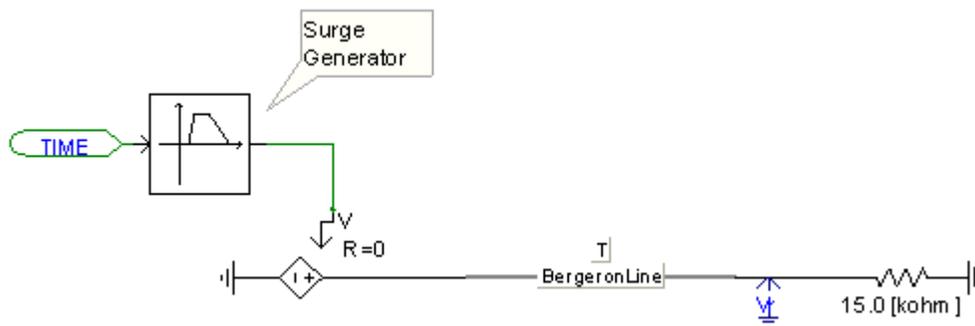
Figure 8-36: Single-Phase Bergeron Transmission Line in PSCAD

A voltage surge is applied at the sending end of the line, and the resultant sending end current is monitored. The waveform is compared to the sending end current given by the Frequency Dependent (Phase) model, which for this example is assumed to very similar to a real world result.

NOTE: This option can be invoked by choosing the 'Use Damping Approximation?' option in the Bergeron model.
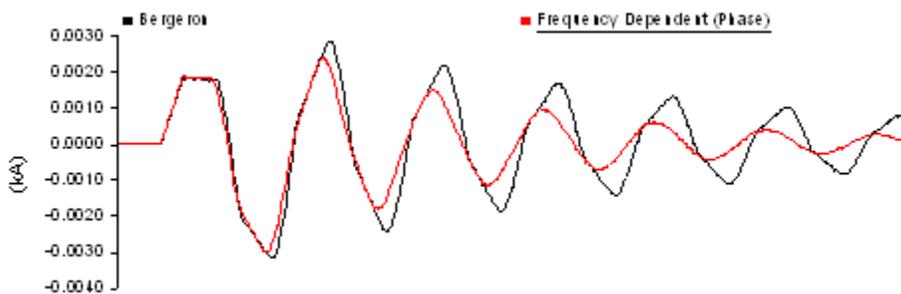
Figure 8-37 – Bergeron Response without High Frequency Attenuation

Due to the lack of high frequency attenuation in the Bergeron model, the receiving end voltage includes frequency components of higher order (as witnessed by the extra 'sharpness' of the reflections).

If the 'Use Damping Approximation' is enabled in the Bergeron model, then a more accurately attenuated current is produced. However, note that a phase shift has been introduced by the wave shaping real pole function.
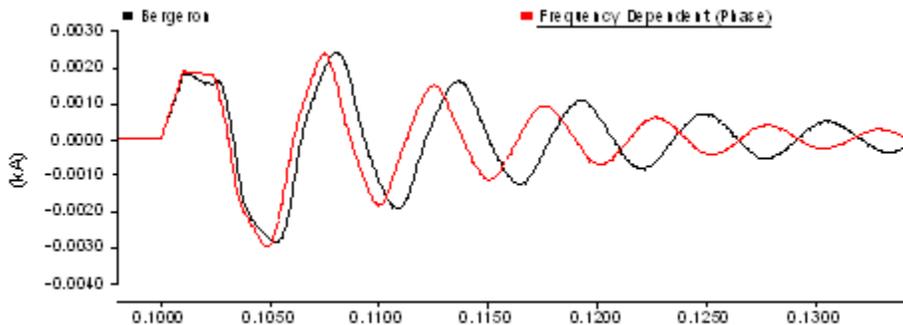
Figure 8-38: Bergeron Response with High Frequency Attenuation

For this example, the parameters were as follows:

$f_0 =$ 60.0   The fundamental frequency [Hz]

$f_1 =$ 1500.0   The 'Frequency for Loss Approximation' [Hz]

$\tau_s =$ 0.05   The 'Shaping Time Constant' [ms]

Frequency Dependent Models

Like the Bergeron model described above, the Frequency-Dependent (FD) models are distributed traveling wave models. However, the system resistance *R* is distributed across the system length (along with *L* and *C*) instead of lumped at the end points. More importantly, the FD models are solved at a number of frequency points, thereby including the frequency dependence of the system.

Two Frequency-Dependent models are available in PSCAD: The Frequency Dependent (Phase) model is the most accurate, as it considers the frequency dependence of internal transformation matrices (thereby accurately representing unbalanced, as well as balanced systems). The older Frequency Dependent (Mode) model assumes a constant transformation and therefore only accurate when modeling balanced systems. For systems consisting of one or two conductors, the two models will give identical results (as the transformation is constant anyway). This is also true for 3-phase, delta configurations (located at a high distance from ground level) and any ideally transposed circuits.

The Frequency Dependent (Phase) model is numerically robust and more accurate than any other commercially available line/cable model, and thus, is the preferred model to use. The FD (Phase) model will of course provide a much more accurate representation of any transmission system than that offered by the Bergeron model.

Frequency Dependent (Mode) Model

The Frequency Dependent (Mode) model is based on methods described in [13]. The model utilizes a constant (or frequency-independent) modal transformation matrix to de-couple multiple-phase systems into separate, mutually exclusive modes. Each mode is thereafter treated as a single-phase circuit. Although classified as 'frequency-dependent', this model is exact in its frequency dependence only for geometrically balanced transmission systems, such as ideally transposed circuits or any other systems where a naturally occurring, constant modal transformation matrix occurs.

In the formulation of a time domain equivalent circuit for the Bergeron model, the expressions were solved in a relatively straightforward manner. However, when considering frequency dependencies, these expressions are next to impossible to formulate directly. It is therefore most convenient to first work in the frequency domain, where an exact solution for a given frequency can be easily derived.

Figure 8-39: Voltages and Currents in a Single-Phase, Overhead Transmission Line

Consider a single-phase of a transmission system as illustrated in Figure 8-39. In the frequency domain, the voltages and currents at one end of the line may be represented in terms of the voltage and current at the other end in the following exact general equation:

$$\begin{bmatrix} V_k \\ I_k \end{bmatrix} = \begin{bmatrix} \cosh(\gamma \cdot \ell) & -Z_0 \cdot \sinh(\gamma \cdot \ell) \\ \dfrac{1}{Z_0} \cdot \sinh(\gamma \cdot \ell) & -\cosh(\gamma \cdot \ell) \end{bmatrix} \cdot \begin{bmatrix} V_m \\ I_m \end{bmatrix}$$

(8-96)

Where,

$Z_0 = \sqrt{Z \cdot Y}$ The characteristic impedance

$\gamma = \sqrt{\dfrac{Z}{Y}}$ The propagation function

$Z, Y = $ The system series impedance and shunt admittance in per-unit length

$\ell = $ The transmission system length

Using a method similar to that described for the Method of Characteristics, forward and backward traveling wave weighting functions F_k, F_m and B_k, B_m (see [8, 13]) are introduced. If the system is assumed to be terminated by an equivalent network whose frequency response is identical to the characteristic impedance $Z_0(\varpi)$, then the forward and backward traveling wave functions can be expressed in the frequency domain as:

$$F_k(\varpi) = V_k(\varpi) + Z_{eq}(\varpi) \cdot I_k(\varpi)$$

(8-97)

$$F_m(\varpi) = V_m(\varpi) + Z_{eq}(\varpi) \cdot I_m(\varpi)$$

(8-98)

And similarly,

$$B_k(\omega) = V_k(\omega) - Z_{eq}(\omega) \cdot I_k(\omega) \qquad (8\text{-}99)$$

$$B_m(\omega) = V_m(\omega) - Z_{eq}(\omega) \cdot I_m(\omega) \qquad (8\text{-}100)$$

Comparing Equations 8-97 and 8-100 with 8-101:

$$B_k(\omega) = A(\omega) \cdot F_m(\omega) \qquad (8\text{-}101)$$

$$B_m(\omega) = A(\omega) \cdot F_k(\omega) \qquad (8\text{-}102)$$

Where,

$$A(\omega) = \frac{1}{\cosh[\gamma(\omega) \cdot \ell] \cdot \sinh[\gamma(\omega) \cdot \ell]} = e^{-\gamma(\omega)\ell}$$

is sometimes referred to as the attenuation function and is a complex number. The real part of $A(\omega)$ is the attenuation constant, and the imaginary part is the phase constant.

The time domain form of Equations 8-101 and 8-102 can be arrived at through the convolution integrals:

$$b_k(t) = \int_{\tau}^{\infty} f_m(t-u) \cdot a(u) \cdot du \qquad (8\text{-}103)$$

$$b_m(t) = \int_{\tau}^{\infty} f_k(t-u) \cdot a(u) \cdot du \qquad (8\text{-}104)$$

Note that the lower limit of the integral in Equations 8-103 and 8-104 is the travel time τ, because the fastest frequency component of an impulse at one end of the transmission system will not reach the other end until this time has elapsed.

Equations 8-103 and 8-104 show that the values of $b_k(t)$ and $b_m(t)$ can be defined entirely by historical values of $f_m(t)$ and $f_k(t)$ (provided that the time step $\Delta t < \tau$). Therefore,

$$b_k(t) = v_k(t - \tau)$$

(8-105)

$$b_m(t) = v_m(t - \tau)$$

And,

$$v_k(t) = Z_{eq} \cdot i_k(t) + v_k(t - \tau)$$

(8-106)

$$v_m(t) = Z_{eq} \cdot i_m(t) + v_m(t - \tau)$$

The equations above can be converted to a modal representation and illustrated schematically as in Figure 8-40.
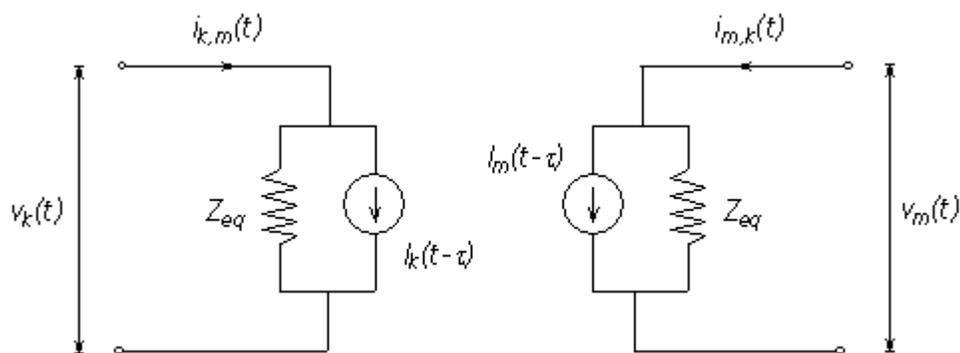
Figure 8-40 – EMTDC Frequency Dependent (Mode) Model Time Domain Interface

Frequency Dependent (Phase) Model

In the 1990's, the need for a transmission line model that could accurately simulate both the undesirable interactions between DC and AC lines, as well as the widely varying modal time delays of underground cables, became more significant. Constant transformation matrix models with frequency dependent modes, such as the Frequency Dependent (Mode) model in PSCAD, dealt with the modal time delay problem in cable systems through modal decomposition techniques, but had proven to be unreliable in accurately simulating systems with unbalanced line geometry (such as AC/DC systems).

In 1999, the *Universal Line Model (ULM)*, based on the theory originally proposed in [22], was incorporated into PSCAD to address these shortcomings, thereby providing a general and accurate frequency dependent model for all underground cables and overhead line geometries. This model (otherwise known as the Frequency Dependent (Phase) model) and its practical implementation into EMTDC is described in [23].

The Frequency Dependent (Phase) model operates on the principle that the full frequency-dependence of a transmission system can be represented by two matrix transfer functions: The propagation function H, and the characteristic admittance Y_c.
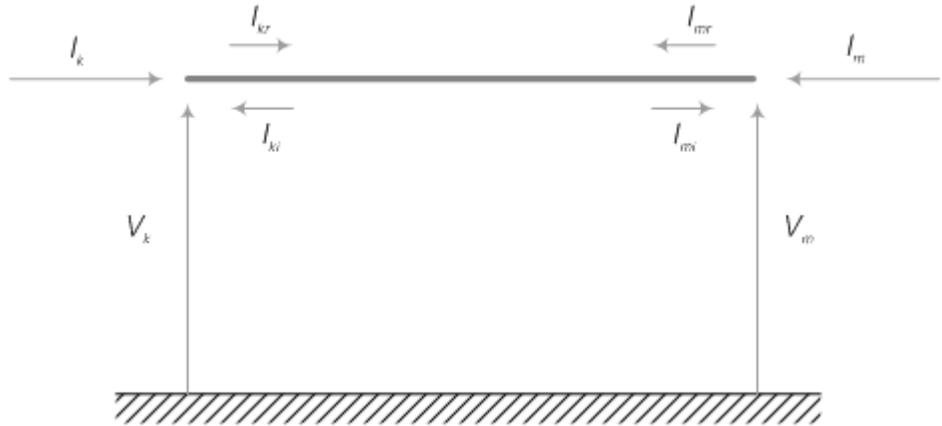
Figure 8-41 – Voltages and Currents in an N-Conductor Transmission Line

Given Figure 8-41, the following can be derived directly from the 'telegrapher's equations' (i.e. equations 8-2 and 8-3) as follows:

$$Y_C \cdot V_k - I_k = 2 \cdot H^T \cdot I_{mr} = 2 \cdot I_{ki} \qquad (8\text{-}107)$$

$$Y_C \cdot V_m - I_m = 2 \cdot H^T \cdot I_{kr} = 2 \cdot I_{mi} \qquad (8\text{-}108)$$

Where,

$$H = e^{-\sqrt{Z \cdot Y} \cdot \ell}$$ Propagation function matrix

$$Y_C = Z^{-1} \cdot \sqrt{Z \cdot Y}$$ Characteristic admittance matrix

And,

$V_k, V_m =$ The node voltage vectors at ends k and m

$I_k, I_m =$ The injected current vectors at ends k and m

$I_{ki}, I_{mi} =$ The incident current vectors at ends k and m

$I_{kr}, I_{mr} =$ The reflected current vectors at ends k and m

H and Y_C are calculated multiple times by the LCP at discrete points in the frequency domain, and then approximated and replaced by equivalent low order rational functions (see Curve Fitting in this chapter for more details). This technique allows for the use of recursive convolution techniques in EMTDC for migration into the time domain, which have proven much more computationally efficient than performing convolution integrals directly [9].

Time Domain Implementation

The *Frequency Dependent (Phase)* model is interfaced to the EMTDC electric network by means of a Norton equivalent circuit, as shown below:
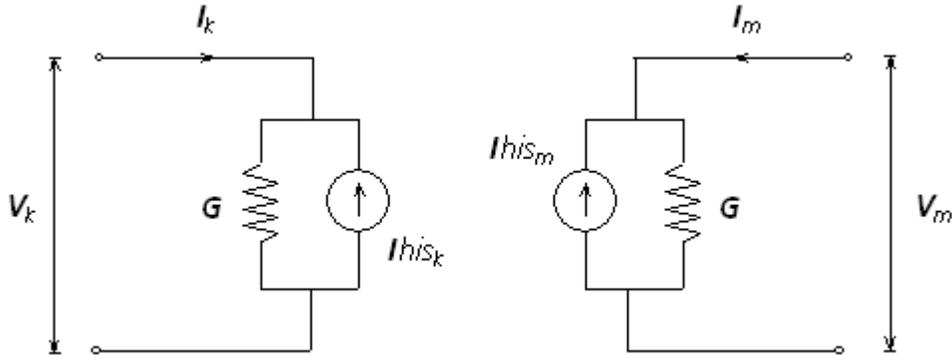
Figure 8-42 – EMTDC Frequency Dependent (Phase) Model Time Domain Interface

The history current source injections I_{hisk} and I_{hism} are updated each time step, given the node voltages V_k and V_m, as calculated by EMTDC. The steps by which this is accomplished by the Frequency Dependent (Phase) Model time-domain interface routine is as given in below:

$$I_k(n) = G \cdot V_k(n) - Ihis_k(n)$$

$$I_{kr}(n) = I_k(n) - I_{ki}(n)$$

(8-109)

$$I_{ki}(n+1) = H * I_{mr}(n-\tau)$$

$$Ihis_k(n+1) = Y_C{'} * V_k(n) - 2 \cdot I_{ki}(n+1)$$

Where,

i	Denotes incident waves
r	Denotes reflected waves
*	Indicates a convolution integral

For a more detailed description of this method including the convolution integration, see [23].

Chapter References

1. J. R. Carson, <u>Wave Propagation in Overhead Wires with Ground Return</u>, Bell Syst. Techn. J., Vol. 5, pp. 539-554, 1926.

2. Pollaczek, F., <u>Sur le champ produit par un conducteur simple infiniment long parcouru par un courant alternatif</u>, Revue Gén, Elec., 1931, 29, pp. 851-867.

3. S. A. Schelkunoff, <u>The electromagnetic theory of coaxial transmission line and cylindrical shields</u>, Bell Syst. Tech. J., vol. 13, pp. 532-579, 1934.

4. H. W. Bode, <u>Network Analysis and Feedback Amplifier Design</u>, D. Van Nostrand, New York, 1945.

5. L. M. Wedepohl, <u>Application of Matrix Methods to the Solution of Travelling-Wave Phenomena in Polyphase Systems</u>, Proc. IEE, Vol. 110, No. 12, pp. 2200-2212, December 1963.

6. H. W. Dommel, <u>Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks</u>, IEEE Transactions on Power Apparatus and Systems, PAS-88, #4, pp. 388-399, April 1969.

7. L. M. Wedepohl, D. J. Wilcox, <u>Transient analysis of underground power-transmission systems. System model and wave-propagation characteristics</u>, Proc. IEE, 120, (2), pp. 253-260, 1973.

8. W. S. Meyer, H. W. Dommel, <u>Numerical Modelling of Frequency-Dependent Transmission-Line Parameters in an Electromagnetic Transients Program</u>, IEEE Transactions on Power Apparatus and Systems, PAS-93, pp. 1401-1409, Sep./Oct. 1974.

9. A. Semlyen, A. Dabuleanu, <u>Fast and Accurate Switching Transient Calculations on Transmission Lines with Ground Return Using Recursive Convolutions</u>, IEEE Transactions on Power Apparatus and Systems, Vol. PAS-94, No. 2, pp. 561-571, March/April. 1975.

10. C. Gary, <u>Approche Complète de la Propagation Multifilaire en Haute Fréquence par Utilisation des Matrices Complexes</u>, EDF Bulletin de la Direction des Études et Recherches-Série B, No. 3/4, pp. 5-20, 1976.

11. A. Ametani, <u>A General Formulation of Impedance and Admittance of Cables</u>, IEEE Transactions on Power Apparatus and Systems, Vol. PAS-99, No. 3, May/June 1980.

12. A. Deri, G. Tevan, A. Semlyen, A. Castanheira, <u>The Complex Ground Return Plane – A Simplified Model for Homogeneous and Multi-Layer Earth Return</u>, IEEE Transactions on Power Apparatus and Systems, Vol. PAS-100, No. 8, pp. 3686-3693, August 1981.

13. J. R. Marti, <u>Accurate Modeling of Frequency Dependent Transmission Lines in Electromagnetic Transients Simulation</u>, IEEE Transactions on Power Apparatus and Systems, PAS-101, #1, pp. 147-155, Jan. 1982.

14. D. E. Amos, <u>A Portable Package For Bessel Functions Of A Complex Argument And Nonnegative Order</u>, Trans. Math. Software, 1986.

15. L. M. Wedepohl, <u>Research Report: Calculation of Impedances of Power Cables</u>, University of British Columbia, Submitted to the Manitoba Hydro International Ltd., File E4.151, 1993.

16. J. J. Grainger, W. D. Stevenson, <u>Power System Analysis</u>, McGraw Hill Inc., 1994.

17. B. Gustavsen, J. Sletbak, T. Henriksen, <u>Calculation of Electromagnetic Transients in Transmission Cables and Lines Taking Frequency Dependent Effects Accurately Into Account</u>, IEEE Transactions on Power Delivery, Vol. 10, No. 2, April 1995.

18. L. M. Wedepohl, H. V. Nguyen, G. D. Irwin, <u>Frequency-Dependent Transformation Matrices for Untransposed Transmission Lines using Newton-Raphson Method</u>, IEEE Transactions on Power Systems, Vol. 11, No. 3, pp. 1538-1546, August 1996.

19. A. M. Gole, <u>Power Systems Transient Simulation</u>, Course Notes, University of Manitoba, 1998.

20. B. Gustavsen, A. Semlyen, <u>Simulation of Transmission Line Transients Using Vector Fitting and Modal Decomposition, IEEE Transactions on Power Delivery</u>, Vol. 13, No. 2, pp. 605-614, April 1998.

21. B. Gustavsen, A. Semlyen, <u>Rational Approximation of Frequency Domain Responses by Vector Fitting</u>, IEEE Trans. on Power Delivery, Vol. 14, No. 3, July 1999.

22. A. Morched, B. Gustavsen, M. Tartibi, <u>A Universal Model for Accurate Calculation of Electromagnetic Transients on Overhead Lines and Underground Cables</u>, IEEE Transactions on Power Delivery, Vol. 14, No. 3, pp. 1032-1038, July 1999.

23. B. Gustavsen, G. Irwin, R. Mangelrød, D. Brandt, K. Kent, <u>Transmission Line Models for the Simulation of Interaction Phenomena Between Parallel AC and DC Overhead Lines</u>, IPST '99 Proceedings, pp. 61-67, 1999.

24. L. M. Wedepohl, <u>The Theory of Natural Modes in Multiconductor Transmission Systems</u>, Course Notes, University of Manitoba, 2002.

25. A. Ramirez, A. Semlyen, and R. Iravani, <u>Order reduction of the dynamic model of a linear weakly periodic system—Part I: General methodology</u>, IEEE Transactions on Power Systems, vol. 19, no. 2, pp. 857–865, May 2004.

26. B. Gustavsen, <u>Time delay identification for transmission line modeling</u>, in Proc. 8th IEEE Workshop Signal Propagation Interconnects, Heidelber, Germany, pp. 103–106, May 9–12, 2004.

27. H.M.J. De Silva, A.M. Gole and L.M. Wedepohl, <u>Accurate Electromagnetic Transient Simulations of HVDC Cables and Overhead Transmission Lines</u>, IPST '07 Proceedings, 2007.

28. B. Gustavsen, J. Nordstrom, <u>Pole Identification for the Universal Line Model Based on Trace Fitting</u>, IEEE Transactions on Power Delivery, vol. 23, no. 1, January 2008.

29. O. Saad, G. Gaba and M. Giroux, <u>A Closed-Form Approximation for Ground Return Impedance of Underground Cables</u>, IEEE Transactions on Power Delivery, vol. 11, no. 3, July 1996, pp. 1536 - 1545.

30. Bjorn Gustavsen, <u>Panel Session on Data for Modeling system Transients in Insulated Cables</u>, Power Engineering Society Winter Meeting, 2001. IEEE, Volume 2, pp. 718-723.

31. L.Marti, R.H. Brierley and T.E. Grainger, <u>Analysis of electromagnetic transients in cross-bonded cable systems using frequency dependent cable models</u>, IPST 1995.

32. Ametani, A., <u>An Investigation of Earth-Return Impedance Between Overhead and Underground Conductors and Its Approximation</u>, IEEE Transactions on Electromagnetic Compatibility, Vol. 51, Issue: 3, pp. 860 - 867, August 2005.

33. Uribe F. A., <u>Calculating Mutual Ground Impedances Between Overhead and Buried Cables</u>, IEEE Transactions on Electromagnetic Compatibility, Vol. 50, Issue: 1, pp. 198 - 203, February 2008.

34. Ametani, A, <u>EMTP Simulations and Theoretical Formulation of Induced Voltages to Pipelines from Power Lines</u>, Session 36, Paper 2, Page 5, 15th PSCC, Liege, 22-26 August 2005.

35. Jun Zou, JunJie Li, J. B. Lee, and S. H. Chang, <u>Fast and Highly Accurate Algorithm for Calculating the Earth-Return Impedance of Underground Conductors</u>, IEEE Transactions on Electromagnetic Compatibility, Vol. 53, Issue: 1, pp. 237 - 240, February 2011.

36. Theodoulidis, T. , <u>Exact Solution of Pollaczek's Integral for Evaluation of Earth-Return Impedance for Underground Conductors</u>, IEEE Transactions on Electromagnetic Compatibility, January 2012.

37. H.M.J.S.P De Silva, Gole, A.M., Nordstrom, J.E., Wedepohl, L.M., <u>Robust Passivity Enforcement Scheme for Time-Domain Simulation of Multi-Conductor Transmission Lines and Cables</u>, IEEE Transactions on Power Delivery, vol. 25, April 2010.

38. C. Chen, E. Gad, R. Achar, <u>Passivity Verification in Delay-Based Macromodels of Multiconductor Electrical Interconnects</u>, IEEE Trans. Circuit and systems, vol. 52, October 2005.

Chapter 9: 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.

> **NOTE**: The Fortran Filter is **no longer provided** with PSCAD versions X4 and onward. To obtain a copy, contact the *PSCAD Support Desk* (support@pscad.com).

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 9-1:

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 'r;emt.d' and 'r;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 9-1 - 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 9-1:

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 9-2:

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 9-2.

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 9-2 - Obsolete Subroutines/Functions

Obsolete Internal Variables

In addition to subroutines and functions, some EMTDC Internal Global Variables have also been affected. Table 9-3 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 9-3 - Obsolete Internal Global Variables

Table 9-4 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(*)	Storage array variables
	STORI(*)	
	STORL(*)	
	STORC(*)	
NEXC	NSTORF	Storage array pointers
	NSTORI	
	NSTORL	
	NSTORC	

Table 9-4 - 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.

EMTDC References

1. H. W. Dommel, <u>Digital Computer Solution of Electromagnetic Transients in Single and Multiphase Networks</u>, 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, <u>Transformer Models in the Simulation of Electromagnetic Transients</u>, 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, <u>Matrix Representation of Three Phase N-Winding Transformers for Steady State Transient Studies</u>, IEEE Transactions on Power Apparatus and Systems, PAS-101, #6, pp. 1369-1378, June 1982.

5. A. M. Gole, <u>Power Systems Transient Simulation</u>, Course Notes, University of Manitoba, 1998.

6. P. Kuffel, K. Kent, G. Irwin, <u>The Implementation and Effectiveness of Linear Interpolation Within Digital Simulation</u>, 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, <u>Guidelines for Modeling Power Electronics in Electric Power Engineering Applications</u>, 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, <u>Modeling of Power Electronic Apparatus: Additional Interpolation Issues</u>, International Conference on Power Systems Transients (IPST '97), pp. 23-28, Seattle, June 22-26, 1997.

9. D. A. Woodford, <u>Validation of Digital Simulation of DC Links</u>, 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, <u>An Electromagnetic Transients Model of Multi-limb Transformers Using Normalized Core Concept</u>, IPST '97 Proceedings, Seattle, pp. 93-98, 1997.

11. W. Enright, N. Watson, O. B. Nayak, <u>Three-Phase Five-Limb Unified Magnetic Equivalent Circuit Transformer models for PSCAD V3</u>, IPST '99 Proceedings, Budapest, pp. 462-467, 1999.

12. B. Adkins, R. G. Harley, <u>The General Theory of Alternating Current Machines</u>, Chapman & Hall, London, 1975.

13. C. V. Jones, <u>Unified Theory of Electrical Machines</u>, Butterworths, London, 1967.

14. I. M. Canay, <u>Causes of Discrepancies on Calculation of Rotor Quantities and Exact Equivalent Diagrams of the Synchronous Machine</u>, IEEE Transactions, Vol. PAS-88, No. F, p. 1114-1120, July 1969.

15. M. R. Harris, P. J. Lawrenson, J. M. Stephenson, <u>Per Unit Systems with Special Reference to Electrical Machines</u>, IEE Monograph, 1970.

16. <u>IEEE Recommended Practice for Excitation System Models for Power System Stability Studies</u>, IEEE Std 421.5-1992.

17. <u>Computer Models for Representation of Digital-Based Excitation Systems</u>, IEEE Transactions 1996.

18. <u>Dynamic Models for Fossil Fueled Steam Units on Power System Studies</u>, 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. <u>Hydraulic Turbine and Turbine Control Models for System Dynamic Studies</u>, 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, <u>Power System Stability and Control</u>, McGraw Hill, 1994.

21. L. M. Wedepohl, <u>Theory of Natural Modes in Multi-Conductor Transmission Lines</u>, Course Notes, May 1989.

22. L. M. Wedepohl, <u>Transient Analysis of Underground Transmission Systems</u>, Proc. IEE, Vol. 120, No. 2, Feb. 1973.

23. J. Marti, <u>Accurate Modeling of Frequency Dependent Transmission Lines in Electromagnetic Transients Simulation</u>, IEEE Transactions on Power Apparatus and Systems, PAS-101, #1, pp. 147-155, Jan. 1982.

24. L. Marti, <u>Simulation of Electromagnetic Transients in Underground Cables with Frequency Dependent Modal Transformation Matrices</u>, Ph.D. Thesis, University of British Columbia, 1986.

25. A. Semlyen, A. Dabuleanu, <u>Fast and Accurate Switching Transient Calculations on Transmission Lines and Ground Return Using Recursive Convolutions</u>, IEEE Transactions on Power Apparatus and Systems, PAS-94, pp. 561-571, March/April 1975.

26. A. Morched, B. Gustavsen, M. Tartibi, <u>A Universal Line Model for Accurate Calculation of Electromagnetic Transients on Overhead Lines and Cables</u>, Paper PE-112-PWRD-0-11-1997.

27. B. Gustavsen, G. Irwin, R. Mangelrod, D. Brandt, K. Kent, <u>Transmission Line Models for the Simulation of Interaction Phenomena Between Parallel AC and DC Overhead Lines</u>, IPST '99 Proceedings, pp. 61-67, 1999.

28. B. Gustavsen, A. Semlyen, <u>Rational Approximation of Frequency Domain Responses by Vector Fitting</u>, 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, <u>Engineering Optimization - Methods and Applications</u>, New York: Wiley-Interscience, 1983.

30. R. L. Haupt, S. E. Haupt, <u>Practical Genetic Algorithms</u>, New York: Wiley-Interscience, 1998.

31. K. Y. Lee, M. A. El-Sharkawi (editors), <u>Tutorial on Modern Heuristic Optimization Techniques with Applications to Power Systems</u>, IEEE Power Engineering Society 02TP160, 2002.

32. J. A. Nelder, R. Mead, <u>A simplex method for function optimization</u>, The Computer Journal, vol. 7, no. 4, pp. 308-313, 1965.

33. A. M. Gole, S. Filizadeh, R. W. Menzies, P. L. Wilson, <u>Optimization-Enabled Electromagnetic Transient Simulation</u>, IEEE Trans. on Power Delivery, 2004.

34. Krüger, K. H., Lasseter, R. H. <u>HVDC Simulation using NETOMAC</u>, 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. <u>A Fully Interpolated Controls Library for Electromagnetic Transients Simulation of Power Systems</u>, Proceedings, IPST'01 – International Conference on Power System Transients, Rio de Janeiro, Brazil, June 24-28, 2001.

36. H. M. J. De Silva, A. M. Gole, L. M. Wedepohl, <u>Accurate Electromagnetic Transient Simulations of HVDC Cables and Overhead Transmission Lines</u>, Proceedings IPST '07, Lyon, France, June 4-7, 2007.

37. B. Gustavsen, J. Nordstrom, <u>Pole Identification for the Universal Line Model Based on Trace Fitting</u>, IEEE Transactions on Power Delivery, Vol. 23, No. 1, January 2008.