

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,600

Open access books available

137,000

International authors and editors

170M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



An Advanced Transmission Line and Cable Model in Matlab for the Simulation of Power-System Transients

Octavio Ramos-Leaños, Jose Luis Naredo and Jose Alberto Gutierrez-Robles

Additional information is available at the end of the chapter

<http://dx.doi.org/10.5772/48530>

1. Introduction

The design and operation of power systems, as well as of power apparatuses, each time depends more on accurate simulations of Electromagnetic Transients (EMTs). Essential to this is to count with advanced models for representing power transmission lines and cables. Electromagnetic Transients Program (EMTP), the most used EMT software, offer various line models. Among these, the most important ones are: 1) the Constant Parameters Line model (CP), 2) the Frequency Dependent or J. Marti Line model (FD) and 3) the Universal Line Model (ULM). The CP Line model is the simplest and most efficient one from the computational point of view. Nevertheless, it tends to overestimate the transient phenomena as it considers that line parameters are constant. Thus, it is recommended only for modeling lines on zones distant to an area where a transient event occurs. The FD Line model (Marti, 1982) evaluates multi-conductor line propagation in the modal domain and takes into account effects due to frequency dependence of the line parameters. Nevertheless, as the transformations between the modal and the phase domains are approximated by real and constant matrices, its accuracy is limited to cases of aerial lines which are symmetric or nearly symmetric. The FD model tends to underestimate the transient phenomena. ULM (Morched et al., 1999) takes into account the full-frequency dependence of line parameters. ULM works directly in phase domain, thus avoiding simplifying assumptions regarding modal-to-phase transformations. So far it is the most general model, capable to accurately represent asymmetric aerial lines as well as underground cables.

The development of ULM is fairly recent and these authors consider that it still is a subject for further research and development. The authors believe also that researchers and power system analysts will benefit considerably from the full understanding of the theoretical basis

of the ULM, as well as from counting with a ULM-type code that is easy to understand and modify. One problem with this is that the theoretical basis of ULM includes various topics and subjects that are scattered through several dozens of highly specialized papers. Another difficulty with this is the high complexity of the code for a ULM-type model. This chapter aims at providing a clear and complete description of the theoretical basis for this model. Although this description is intended for power engineers with an interest in electromagnetic transient phenomena, it can be of interest also to electronic engineers involved in the analysis and design of interconnects. The chapter includes as well the description of Matlab program of a ULM-type model, along with executable code and basic examples.

2. Multi-conductor transmission line analysis

2.1. Telegrapher's Equations

Electromagnetic behavior of transmission lines and cables is described by the Modified Telegrapher Equations, which in frequency domain are expressed as follows:

$$-\frac{d\mathbf{V}}{dx} = \mathbf{Z}\mathbf{I}. \quad (1)$$

$$-\frac{d\mathbf{I}}{dx} = \mathbf{Y}\mathbf{V}. \quad (2)$$

where \mathbf{V} is the vector of voltages, \mathbf{I} is the vector of currents, \mathbf{Z} and \mathbf{Y} are the $(N \times N)$ per unit-length series impedance and shunt admittance matrix of a given line with N conductors, respectively. To solve equations (1) and (2), let equation (1) be first differentiated with respect to x ; then, (2) is used to eliminate the vector of currents at the right hand side. The resulting expression is a second order matrix ODE involving only unknown voltages:

$$\frac{d^2\mathbf{V}}{dx^2} = \mathbf{Z}\mathbf{Y}\mathbf{V}. \quad (3)$$

In the same way, equation (2) can be differentiated with respect to x and (1) can be used to eliminate the right-hand-side voltage term. The resulting expression involves unknown currents only:

$$\frac{d^2\mathbf{I}}{dx^2} = \mathbf{Y}\mathbf{Z}\mathbf{I}. \quad (4)$$

Solution to (4) is:

$$\mathbf{I}(x) = \mathbf{C}_1 e^{-\sqrt{\mathbf{Y}\mathbf{Z}}x} + \mathbf{C}_2 e^{\sqrt{\mathbf{Y}\mathbf{Z}}x}, \quad (5)$$

where \mathbf{C}_1 and \mathbf{C}_2 are vectors of integration constants determined by the line boundary conditions; that is, by the connections at the two line ends. In fact, the term including \mathbf{C}_1 represents a vector of phase currents propagating forward (or in the positive x -direction) along the line, whereas the one with \mathbf{C}_2 represents a backward (or negative x -direction) propagating vector of phase currents. Expression (5) is an extension of the well-known solution for the single-conductor line. Note that this extension involves the concept of matrix functions. This topic is explained at section 2.2.

The solution to (3) takes a form analogous to (5) and it is obtained conveniently from (5) and (2) as follows:

$$\mathbf{V}(x) = -\mathbf{Y}^{-1} \frac{d\mathbf{I}}{dx} = \mathbf{Z}_c \left[\mathbf{C}_1 e^{-\sqrt{\mathbf{Y}\mathbf{Z}}x} - \mathbf{C}_2 e^{\sqrt{\mathbf{Y}\mathbf{Z}}x} \right] \quad (6)$$

where, $\mathbf{Z}_c = \mathbf{Y}^{-1} \sqrt{\mathbf{Y}\mathbf{Z}}$ is the characteristic impedance matrix and its inverse is the characteristic admittance matrix $\mathbf{Y}_c = \sqrt{\mathbf{Y}\mathbf{Z}} \mathbf{Z}^{-1}$.

2.2. Modal analysis and matrix functions

Matrix functions needed for multi-conductor line analysis are extensions of analytic functions of a one-dimensional variable. Consider the following function and its Taylor expansion:

$$f(x) = \sum_{k=0}^{\infty} a_k x^k \quad (7)$$

The application of $f()$ to a square matrix \mathbf{A} of order $N \times N$ as its argument is accomplished as follows:

$$f(\mathbf{A}) = \sum_{k=0}^{\infty} a_k \mathbf{A}^k, \quad (8)$$

where \mathbf{A}^0 is equal to \mathbf{U} , the $N \times N$ unit matrix. Consider now the case of a diagonal matrix:

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_N \end{bmatrix}. \quad (9)$$

Application of (8) to $\mathbf{\Lambda}$ yields:

$$f(\mathbf{\Lambda}) = \sum_{k=0}^{\infty} a_k \mathbf{\Lambda}^k = \begin{bmatrix} \sum_k a_k \lambda_1^k & 0 & \dots & 0 \\ 0 & \sum_k a_k \lambda_2^k & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sum_k a_k \lambda_N^k \end{bmatrix} = \begin{bmatrix} f(\lambda_1) & 0 & \dots & 0 \\ 0 & f(\lambda_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & f(\lambda_N) \end{bmatrix} \quad (10)$$

This expression thus shows that the function of a diagonal matrix is simply obtained applying the one-dimensional form of the function to the matrix nonzero elements. Consider next the function of a diagonalizable matrix A ; that is, a matrix A that is similar to a diagonal one Λ :

$$\mathbf{A} = \mathbf{M} \Lambda \mathbf{M}^{-1}. \quad (11)$$

where \mathbf{M} is the nonsingular matrix whose columns are the eigenvectors of A , while Λ is the matrix whose diagonal elements are the eigenvalues of A (Strang, 1988).

Application of $f()$ as in (10) to A yields:

$$f(\mathbf{A}) = \sum_{k=0}^{\infty} a_k (\mathbf{M} \Lambda \mathbf{M}^{-1})^k = \mathbf{M} \left(\sum_{k=0}^{\infty} a_k \Lambda^k \right) \mathbf{M}^{-1} = \mathbf{M} f(\Lambda) \mathbf{M}^{-1} \quad (12)$$

Therefore, the function of a diagonalizable matrix is conveniently obtained first by factoring A as in (10), then by applying the function to the diagonal elements of Λ and, finally, by performing the triple matrix product as in (11) and (12).

It is clear from subsection 2.1, that multi-conductor line analysis requires evaluating matrix functions of \mathbf{YZ} . To do so, it is generally assumed that \mathbf{YZ} always is diagonalizable (Wedephol, 1965; Dommel, 1992). Although there is a possibility for \mathbf{YZ} not being diagonalizable (Brandao Faria, 1986), occurrences of this can be easily avoided when conducting practical analysis (Naredo, 1986).

3. Line modelling

Figure 1 shows the representation of a multi-conductor transmission line (or cable) of length L , with one of its ends at $x = 0$ and the other at $x = L$. Let \mathbf{I}_0 be the vector of phase currents being injected into the line and \mathbf{V}_0 the vector of phase voltages, both at $x=0$. In the same form, \mathbf{I}_L and \mathbf{V}_L represent the respective vectors of injected phase currents and of phase voltages at $x=L$. Line equation solutions (5) and (6) are applied to the line end at $x=0$:

$$\mathbf{I}_0 = \mathbf{I}(0) = \mathbf{C}_1 + \mathbf{C}_2 \quad (13)$$

$$\mathbf{V}_0 = \mathbf{V}(0) = \mathbf{Z}_C [\mathbf{C}_1 + \mathbf{C}_2]. \quad (14)$$

Then, the value of \mathbf{C}_1 is determined from (13) and (14):

$$\mathbf{C}_1 = \frac{\mathbf{I}_0 + \mathbf{Y}_C \mathbf{V}_0}{2}. \quad (15)$$

Expressions (5) and (6) are applied to the line end conditions at $x = L$:

$$\mathbf{I}_L = -\mathbf{I}(L) = -\mathbf{C}_1 e^{-\sqrt{\mathbf{YZ}}L} - \mathbf{C}_2 e^{\sqrt{\mathbf{YZ}}L} \quad (16)$$

and

$$\mathbf{V}_L = \mathbf{V}(L) = \mathbf{Z}_C \left[\mathbf{C}_1 e^{-\sqrt{\mathbf{Y}\mathbf{Z}}L} - \mathbf{C}_2 e^{\sqrt{\mathbf{Y}\mathbf{Z}}L} \right]. \quad (17)$$

After doing this, (17) is pre-multiplied by \mathbf{Y}_C and subtracted from (16) to obtain:

$$\mathbf{I}_L - \mathbf{Y}_C \mathbf{V}_L = -2\mathbf{C}_1 e^{-\sqrt{\mathbf{Y}\mathbf{Z}}L}. \quad (18)$$

Finally, (15) is introduced in (18) rendering

$$\mathbf{I}_L - \mathbf{Y}_C \mathbf{V}_L = -e^{-\sqrt{\mathbf{Y}\mathbf{Z}}L} [\mathbf{I}_0 + \mathbf{Y}_C \mathbf{V}_0] \quad (19)$$

Expression (19) establishes the relation among voltages and currents at the terminals of a multi-conductor line section. Its physical meaning follows from realizing that the term $\mathbf{I}_0 + \mathbf{Y}_C \mathbf{V}_0$ at its right hand side represents a traveling wave of currents leaving the line end at $x = 0$ and propagating in the positive x -axis direction, whereas $\mathbf{I}_L - \mathbf{Y}_C \mathbf{V}_L$ at the left hand side is the traveling wave of currents leaving the line end at $x = L$.

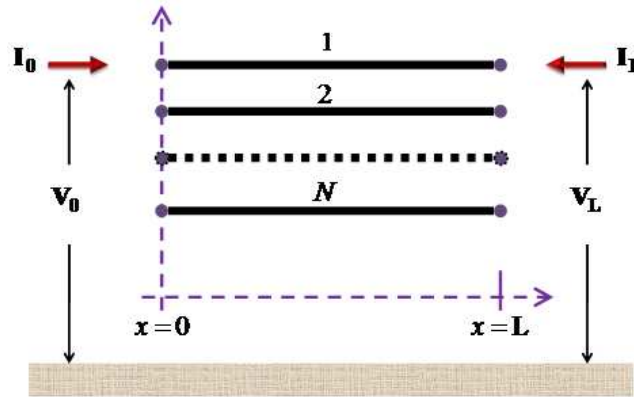


Figure 1. Multi-conductor line segment of length L .

By a similar process as the previous one for deriving (19), it is possible to show also that the following relation holds as line equation solutions (5) and (6) are applied to line end conditions at $x=0$:

$$\mathbf{I}_0 - \mathbf{Y}_C \mathbf{V}_0 = -e^{-\sqrt{\mathbf{Y}\mathbf{Z}}L} [\mathbf{I}_L + \mathbf{Y}_C \mathbf{V}_L] \quad (20)$$

Note however that this relation can also be obtained by simply exchanging at (19) sub-indexes 0 and L . This exchange is justified by the input/output symmetry of the line section. Expressions (19) and (20) provide a very general mathematical model for a multi-conductor transmission line. This is a model based on traveling wave principles. Let (19) and (20) be rewritten as follows:

$$\mathbf{I}_L = \mathbf{I}_{sh,L} - \mathbf{I}_{aux,L} \quad (21)$$

where, $\mathbf{I}_{sh,L} = \mathbf{Y}_C \mathbf{V}_L$ is the shunt currents vector produced at terminal L by injected voltages \mathbf{V}_L . $\mathbf{I}_{aux,L} = \mathbf{H} \mathbf{I}_{rf,0}$ is the auxiliary currents vector consisting of the reflected currents at terminal 0 , $\mathbf{I}_{rf,0} = \mathbf{I}_0 + \mathbf{Y}_C \mathbf{V}_0$ and the transfer functions matrix $\mathbf{H} = e^{-\sqrt{\mathbf{Y}\mathbf{Z}}L}$.

In the same way as it has been previously done for (19), expression (20) is conveniently represented as follows:

$$\mathbf{I}_0 = \mathbf{I}_{sh,0} - \mathbf{I}_{aux,0} \quad (22)$$

with, $\mathbf{I}_{sh,0} = \mathbf{Y}_C \mathbf{V}_0$, $\mathbf{I}_{aux,0} = \mathbf{H} \mathbf{I}_{rfl,L}$, and $\mathbf{I}_{rfl,L} = \mathbf{I}_L + \mathbf{Y}_C \mathbf{V}_L$.

Expressions (21) and (22) constitute a traveling wave line model for the segment of length L depicted in figure 1. The former set of expressions represents end L of the line segment, while the latter set represents end 0 . A schematic representation for the whole model is provided by figure 2. Note that the coupling between the two line ends is through the auxiliary sources $\mathbf{I}_{aux,0}$ and $\mathbf{I}_{aux,L}$.

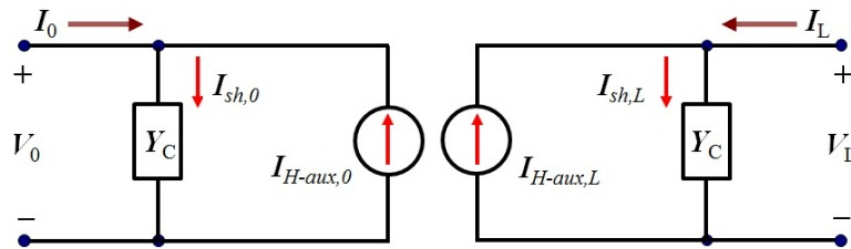


Figure 2. Frequency domain circuit representation of a multi-conductor line.

The line model defined by expressions (21) and (22) is in the frequency domain. Power system transient simulations require this model to be transformed to the time domain. For instance, the transformation of (21) to the time domain yields:

$$\mathbf{i}_0 = \mathbf{i}_{sh,0} - \mathbf{i}_{aux,0} \quad (23)$$

with

$$\mathbf{i}_{sh,0} = \mathbf{y}_C * \mathbf{v}_L \quad (24)$$

and

$$\mathbf{i}_{aux,0} = \mathbf{h} * \mathbf{i}_{rfl,0} \quad (25)$$

Note that at (23), (24), (25) the lowercase variables represent the time domain images of their uppercase counterparts at (22) and that the symbol $*$ represents convolution. Reflected currents can be represented as

$$\mathbf{i}_{rfl,L} = 2\mathbf{i}_{sh,L} - \mathbf{i}_{aux,L} \quad (26)$$

Expressions (23)-(26) constitute a general traveling-wave based time-domain model for line end 0 . The model corresponding to the other end is obtained by interchanging sub-indexes " 0 " and " L " at (23)-(26). Equation (23) essentially provides the interface of the line-end 0 model to the nodal network solver that, for power system transient analysis, usually is the EMTP (Dommel, 1996). Expressions (24) and (25) require the performing of matrix-to-vector

convolutions that are carried out conveniently by means of State–Space methods (Semlyen & Abdel-Rahman, 1982). State–Space equivalents of (23) and (24) arise naturally as Y_c and H are represented by means of fitted rational functions (Semlyen & Dabuleanu, 1975).

4. Phase domain line model

Since rational fitting and model solutions are carried out directly in the phase domain, the model described here is said to be a phase domain line model. Rational fitting for this model is carried out using the Vector fitting (VF) tool (Gustavsen, 2008). In the case of Y_c , the whole fitting process is done in the phase domain, whereas for H initial poles and time delays are first calculated in the modal domain.

4.1. Rational approximation of Y_c

The following rational representation has been proposed for Y_c in (Marti, 1982) and (Morched et al., 1999):

$$Y_c = G_0 + \sum_{i=1}^{N_y} \frac{G_i}{s - q_i} \quad (27)$$

where N_y is the fitting order, q_i represents the i -th fitting pole, G_i is the corresponding matrix of residues and G_0 is a constant matrix obtained at the limit of Y_c when $s = j\omega \rightarrow \infty$. Note in (27) that common poles are used for the fitting of all the elements of Y_c obtained by fitting the matrix trace and finally the fitting of the matrices of residues G_i and of proportional terms G_0 is done in the phase domain. Section 7 provides an overview of the VF procedure and further information is to be found in (Gustavsen & Semlyen, 1999) and (Gustavsen, 2008).

4.2. Rational approximation of H

Rational fitting of transfer matrix H is substantially more involving than the one of Y_c above. To attain an accurate and compact (low order) rational representation for H it is essential to factor out all terms involving time delays (Marti, 1982). The major difficulty here is that its elements could involve a mix of up to N different delay terms due to the multimode propagation on an N -conductor line (Wedepohl, 1965). Separation of matrix H into single-delay terms is obtained from the following modal factorization (Wedepohl, 1965):

$$H = M H_m M^{-1} \quad (28)$$

where H_m is a diagonal matrix of the form

$$H_m = \text{diag}[e^{-\gamma_1 L}, e^{-\gamma_2 L}, \dots, e^{-\gamma_N L}] \quad (29)$$

and $\gamma = \sqrt{YZ}$ is the propagation constant of a conductor line (Wedepohl, 1965). As the triple product in (28) is performed by partitioning M in columns and M^{-1} in rows, the following expression is derived:

$$\mathbf{H} = \sum_{i=1}^N \mathbf{D}_i e^{-\gamma_i L} \quad (30)$$

where \mathbf{D}_i is the rank-1 matrix obtained from pre-multiplying the i -th column of \mathbf{M} by the i -th row of \mathbf{M}^{-1} . Matrix \mathbf{D}_i in fact is an idempotent (Marcano & Marti, 1997). The exponential factors at (30) can be further decomposed as follows:

$$e^{-\gamma_i L} = e^{-\tilde{\gamma}_i L} e^{-s\tau_i}; i = 1, 2, \dots, N \quad (31)$$

where $\exp(-\tilde{\gamma}_i L)$ is a minimum phase-shift function (Bode, 1945) and τ_i is the time delay associated to the velocity of the i -th mode. Hence:

$$\mathbf{H} = \sum_{i=1}^N \mathbf{D}_i e^{-\tilde{\gamma}_i L} e^{-s\tau_i} \quad (32)$$

The time delays in (32) can be initially estimated by applying Bode's relation for minimum phase complex functions (Bode, 1945) to the magnitudes of $\exp(-\gamma_i L)$ in (30). Although (32) provides the desired separation of \mathbf{H} as a sum of terms, each one involving a single delay factor, the following consideration is brought in for computational efficiency (Morched et al., 1999). Modal delays often occur in groups with almost identical values. Suppose that a number N_g of these groups can be formed and that (32) can be modified as follows:

$$\mathbf{H} = \sum_{k=1}^{N_g} \tilde{\mathbf{H}}_k e^{-s\tau_k} \quad (33)$$

where N_g is less or equal to N , and τ_k is the representative delay for the k -th group. Clearly, by comparing (33) and (32):

$$\tilde{\mathbf{H}}_k = \sum_{i=1}^{lk} \mathbf{D}_i e^{-\tilde{\gamma}_i L} e^{-s\tau_i} \quad k = 1, 2, \dots, N_g \quad (34)$$

with lk being the number of modal terms in the k -th group. To determine whether a set of exponential factors can be grouped or not, the maximum phase shifts associated to their time delays are compared. The set is grouped into a single delay group if the phase shift differences are below a pre-established value typically chosen at 10° (Morched et al., 1999). Each term $\tilde{\mathbf{H}}_k$ at (34) can now be considered free of delay factors and can be fitted as follows:

$$\tilde{\mathbf{H}}_k = \sum_{i=1}^{Nh(k)} \frac{\mathbf{R}_{k,i}}{s - p_{k,i}} \quad k = 1, 2, \dots, N_g \quad (35)$$

where $Nh(k)$ is the fitting order for the k -th term $\tilde{\mathbf{H}}_k$, $p_{k,i}$ represents its i -th fitting pole and $\mathbf{R}_{k,i}$ is the corresponding matrix of residues. Note in (35) that common poles are being used to fit all elements at each matrix $\tilde{\mathbf{H}}_k$. As (35) is introduced in (33), the following rational form is obtained for \mathbf{H} :

$$\mathbf{H} = \sum_{k=1}^{N_g} e^{-s\tau_k} \sum_{i=1}^{Nh(k)} \frac{\mathbf{R}_{k,i}}{s - p_{k,i}} \quad (36)$$

Initial estimates for the poles as well as time delays are obtained in the modal domain. The poles result from applying VF to the sum of the modal exponential factors conforming each delay group. The time delays proceed from Bode's formula as it has been said before. With all the poles $p_{k,i}$ and group delays τ_k of (36) being estimated in the modal domain, the overall fitting of H is completed in the phase domain by obtaining the matrices of residues $R_{k,i}$ and recalculating the poles (Gustavsen & Nordstrom, 2008). The fitting parameters so obtained can be taken as final or can be further refined by an iterative process. VF is applied throughout all these fitting tasks and detailed descriptions of these processes can be found in (Gustavsen & Nordstrom, 2008; Gustavsen & Semlyen, 1999).

4.3. State-space analysis

With the rational representation of Y_c and H state-space forms to evaluate $i_{sh,0}$ and $i_{aux,0}$ arise naturally. Consider first the case of $i_{sh,0}$. Taking (22) and introducing (27) yields

$$\mathbf{I}_{sh,0} = \mathbf{G}_0 \mathbf{V}_0 + \sum_{i=1}^{Ny} \mathbf{W}_i \quad (37)$$

where

$$\mathbf{W}_i = \frac{\mathbf{G}_i}{s - q_i} \mathbf{V}_0 \quad i = 1, 2, \dots, Ny \quad (38)$$

Application of the Inverse Laplace Transform to (37) and (38) gives the following continuous-time-state-space (CTSS) form for $i_{sh,0}$:

$$\mathbf{i}_{sh,0} = \mathbf{G}_0 \mathbf{v}_0 + \sum_{i=1}^{Ny} \mathbf{w}_i \quad (39)$$

and

$$\frac{d\mathbf{w}_i}{dt} = q_i \mathbf{w}_i + \mathbf{G}_i \mathbf{v}_0 \quad i = 1, 2, \dots, Ny \quad (40)$$

The CTSS form to evaluate $i_{aux,0}$, is derived now. On replacing the fitted form of H given by (36) into (22):

$$\mathbf{I}_{aux,0} = \sum_{k=1}^{Ng} \sum_{i=1}^{Nh(k)} \mathbf{X}_{k,i} \quad (41)$$

with

$$\mathbf{X}_{k,i} = \frac{R_{k,i}}{s - p_{k,i}} I_{rfl,L} e^{-s\tau_k}; \quad \begin{matrix} i = 1, 2, \dots, Ny \\ k = 1, 2, \dots, Ng \end{matrix} \quad (42)$$

Application of the Inverse Laplace Transform to (41) and (42) renders the following CTSS form:

$$\mathbf{i}_{aux,0} = \sum_{k=1}^{Ng} \sum_{i=1}^{Nh(k)} \mathbf{x}_{k,i} \quad (43)$$

$$\frac{d\mathbf{x}_{k,i}}{dt} = p_{k,i}\mathbf{x}_{k,i} + \mathbf{R}_{k,i}\mathbf{i}_{rfl,L}(t - \tau_k); \quad \begin{matrix} i = 1, 2, \dots, Ny \\ k = 1, 2, \dots, Ng \end{matrix} \quad (44)$$

CTSS forms (39), (40), (43) and (44) provide the basis for a phase domain line model (Morched et al., 1999). Nevertheless, their solution by a digital processor requires the conversion to discrete-time state-space (DTSS). This is accomplished by applying a numerical differentiation rule to the CTSS forms. The one adopted here is the mid-point rule of differentiation, which is equivalent to the trapezoidal integration rule extensively used in EMTP (Dommel, 1969, 1992). Application of this rule to (44) with Δt as the solution time step results in:

$$\mathbf{x}_{k,i} = a_{k,i}\mathbf{x}'_{k,i} + \tilde{\mathbf{R}}_{k,i}[\mathbf{i}_{rfl,L}(t - \tau_k) + \mathbf{i}'_{rfl,L}(t - \tau_k)]; \quad \begin{matrix} i = 1, 2, \dots, Ny \\ k = 1, 2, \dots, Ng \end{matrix} \quad (45)$$

where $a_{k,i} = (2 + \Delta t p_{k,i}) / (2 - \Delta t p_{k,i})$ and $\tilde{\mathbf{R}}_{k,i} = (\Delta t \mathbf{R}_{k,i}) / (2 - \Delta t p_{k,i})$. $\mathbf{x}_{k,i}$ are discrete-state variables and primed variables denote their value at one previous time step $\mathbf{x}'_{k,i} = \mathbf{x}_{k,i}(t - \Delta t)$. The discrete-time version of (43) maintains its original form:

$$\mathbf{i}_{aux,0} = \sum_{k=1}^{Ng} \sum_{i=1}^{Nh(k)} \mathbf{x}_{k,i}$$

Transmission line simulation of EMTs requires the use of time steps Δt smaller than any of the travel times τ_k in the line. Hence, (45) provides the update of state vectors $\mathbf{x}_{k,i}$ using only past values of variables already available, either from initial conditions or from previous simulation time steps.

The differentiation mid-point rule is now applied to (40):

$$\mathbf{w}_i = a_i \mathbf{w}'_i + \tilde{\mathbf{G}}_i(v_0 + v'_0); i = 1, 2, \dots, Ny \quad (46)$$

where $a_i = (2 + \Delta t q_i) / (2 - \Delta t q_i)$ and $\tilde{\mathbf{G}}_i = (\Delta t \mathbf{G}_i) / (2 - \Delta t q_i)$

Expression (46) is not a proper DTSS form, as \mathbf{w}_i depends on the present-time value of v_0 which still is to be determined (Gustavsen & Mahseredjian, 2007). This problem is fixed here with the following redefinition of the state variable vector:

$$\mathbf{y}_i = (\tilde{\mathbf{G}}_i^{-1} \mathbf{w}_i - v_0) / (a_i + 1); i = 1, 2, \dots, Ny \quad (47)$$

Introducing (47) in (46) and (39) the following expressions are obtained:

$$\mathbf{y}_i = a_i \mathbf{y}'_i + v'_0; i = 1, 2, \dots, Ny \quad (48)$$

$$\mathbf{i}_{sh,0} = \mathbf{G} \mathbf{v}_0 + \mathbf{i}_{y-aux,0} \quad (49)$$

where

$$\mathbf{i}_{y-aux,0} = \sum_{i=1}^{N_y} \hat{\mathbf{G}}_i \mathbf{y}_i ; \hat{\mathbf{G}}_i = (a_i + 1) \tilde{\mathbf{G}}_i ; \mathbf{G} = \mathbf{G}_0 + \sum_{i=1}^{N_y} \tilde{\mathbf{G}}_i$$

Expression (48) is a proper DTSS form for the sequential evaluation of $\mathbf{i}_{sh,0}$ at the phase domain line model.

Finally, the introduction of (43) and (49) in (23) results in:

$$\mathbf{i}_0 = \mathbf{G} \mathbf{v}_0 + \mathbf{i}_{hist,0} \quad (50)$$

with

$$\mathbf{i}_{hist,0} = \mathbf{i}_{y-aux,0} - \mathbf{i}_{aux,0} = \sum_{i=1}^{N_y} \hat{\mathbf{G}}_i \mathbf{y}_i - \sum_{k=1}^{N_g} \sum_{i=1}^{N_h(k)} x_{k,i}$$

Expression (50), along with (48) and (49), provides a discrete time-domain model for end 0 of the line segment at figure 1. The expressions for the model at end L are simply obtained by exchanging sub-indexes 0 and L at (48), (49) and (50). Obviously, state variables “ y_i ” and “ $x_{k,i}$ ” of end L model are different from those of end 0. Figure 3 provides a discrete-time circuit-model for the line segment of length $x=L$. This model is based on expression (50) and its companion for line end L . Note that the model consists of parallel arrangements of shunt conductances and auxiliary sources of currents comprising historic terms of ends (or nodes) 0 and L . Figure 3 model is thus in an appropriate form for computer code implementation. In this chapter, the Matlab environment has been chosen for this end.

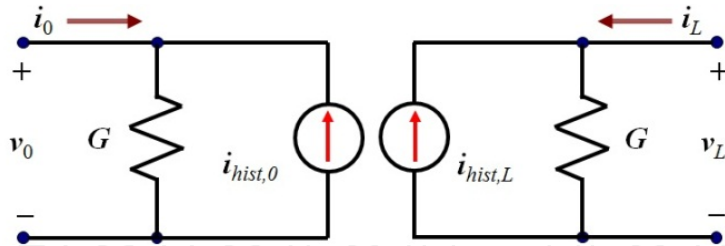


Figure 3. Discrete time domain circuit representation of a multi-conductor line.

5. Line model implementation in Matlab

The discrete-time line model depicted in figure 3 and defined by (50) has been programmed by these authors in Matlab as an M-code function (see Appendix). This function consists of two sub-blocks, one for each multi-conductor line end. This model is to be used with a nodal network solver, a complete explanation on the nodal solver can be found in (Dommel, 1969 & 1992). Expression (50) constitutes essentially the interface between the line model and the nodal solver. Each one of the two sub-blocks in the line model performs iteratively the six tasks that are described next for line-end 0 sub-block. Figure 4 provides the block diagram of the complete line/cable model, along with its interfacing with the nodal solver.

Step 1. State-variable and history-current values are assumed known, either from initial conditions or from previous simulation steps. These values are used by the nodal solver to determine line end (nodal) voltages v_0 and v_L .

Step 2. Shunt current due to the characteristic admittance of the line is calculated by (49) repeated here for convenience:

$$\mathbf{i}_{sh,0} = \mathbf{G}\mathbf{v}_0 + \mathbf{i}_{y-aux,0}$$

Step 3. Auxiliary current source value, due to the reflected traveling waves at the remote line end, is updated by (43):

$$\mathbf{i}_{aux,0} = \sum_{k=1}^{Ng} \sum_{i=1}^{Nh(k)} \mathbf{x}_{k,i}$$

Step 4. Vector of reflected currents at the local line-end (node) “ $i_{rfl,0}$ ” is calculated for the present time by means of (26) being modified to suit line-end 0:

$$\mathbf{i}_{rfl,0} = 2\mathbf{i}_{sh,0} - \mathbf{i}_{aux,0}$$

This vector is delivered to end L sub-block through a delay buffer. Although branch current vector \mathbf{i}_0 usually is not explicitly required, it is conveniently evaluated here by (50):

$$\mathbf{i}_0 = \mathbf{G}\mathbf{v}_0 + \mathbf{i}_{hist,0}$$

Step 5. Internal states inside the line model are updated by (48) and (45):

$$\mathbf{y}_i = a_i \mathbf{y}'_i + v'_0$$

$$\mathbf{x}_{k,i} = a_{k,i} \mathbf{x}'_{k,i} + \tilde{\mathbf{R}}_{k,i} [\mathbf{i}_{rfl,L}(t - \tau_k) + \mathbf{i}'_{rfl,L}(t - \tau_k)]$$

Step 6. The vector of history currents for end (node) 0 is updated by means of (50) and the update is delivered to the nodal-network solver.

Steps 1 to 6 are iterated N_t times until $N_t \Delta t$ spans the total simulation time of interest.

5.1. Handling of line-travel delays

It follows from expressions (43) and (45) that the calculation of $\mathbf{i}_{aux,0}$ requires the reflected currents vector $\mathbf{i}_{rfl,L}$ being evaluated with various time delays $\tau_1, \tau_2, \dots, \tau_{Ng}$. Recall that the delays are due to the travel time needed by a wave to travel from one line end to the other. Past values of $\mathbf{i}_{rfl,L}$ can be obtained either from line initial conditions or from previous simulation steps; nevertheless, these values are given by samples regularly distributed Δt seconds apart. Since the involved travel times (or line delays) usually are not integer multiples of Δt , the required values of $\mathbf{i}_{rfl,L}$ must be obtained by means of interpolations. The standard procedure for this is to interpolate linearly (Dommel, 1992) and this is adopted here.

Evaluation of the delayed values require a memory buffer spanning at least the largest travel time

$$\tau_{\max} = \max\{\tau_1, \tau_2, \dots, \tau_{N_g}\}, \quad (51)$$

and buffer length N_b is calculated as follows:

$$N_b = \left\lceil \frac{\tau_{\max}}{\Delta t} \right\rceil + 1 \quad (52)$$

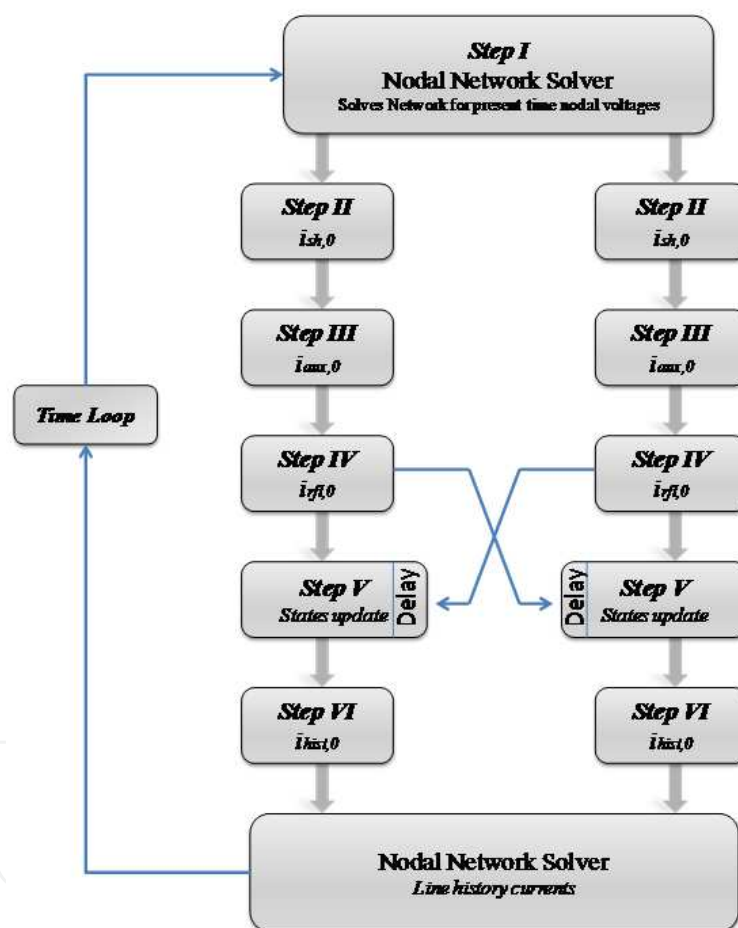


Figure 4. Line/Cable model's complete flow diagram.

If a propagation delay is an integer multiple of Δt , the required value of i_{rl} can be readily retrieved from the memory buffer. This is illustrated by figure 5 where the simulation time step is $\Delta t=0.03$ ms and the travel time is $\tau=0.10$ ms. It can be seen that at simulation time $t=0.24$ ms the required history value at 0.09ms is available from the table.

On a multiphase system, nevertheless, it is highly improbable that all the propagation times can be made integer multiples of a single value of Δt suitable for transient simulations. Thus,

the required values must be obtained through interpolation. Figure 6 illustrates this case, where a simulation time step $\Delta t = 0.04$ ms is assumed instead of the $\Delta t = 0.03$ ms one at figure 5. Notice that now the required history value, for a time delay of 0.09ms, is not readily available.

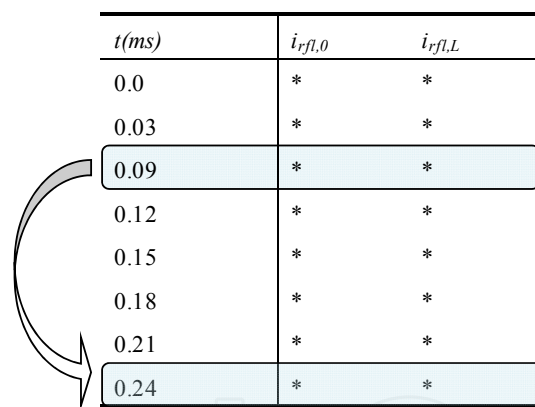
Suppose now that the required value $i_{rfl}(t-\tau)$ is between the k -th and the $(k+1)$ -th stored samples of i_{rfl} . Let ζ be the fractional part of $\tau/\Delta t$, that can be obtained as follows:

$$\zeta = \frac{\tau}{\Delta t} - \left\lfloor \frac{\tau}{\Delta t} \right\rfloor, \quad (53)$$

with, $0 \leq \zeta < 1$. The estimated value of $i_{rfl}(t-\tau)$ by linear interpolation is thus:

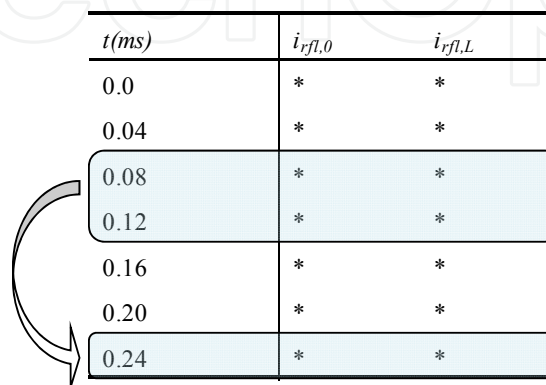
$$i_{rfl,L}(t - \tau_k) \cong i_{rfl,L}(t - r\Delta t) + \zeta[i_{rfl,L}(t - k\Delta t) - i_{rfl,L}(t - k\Delta t - \Delta t)]. \quad (54)$$

Figure 7 illustrates the memory buffer management, either for $i_{rfl,0}$ or for $i_{rfl,L}$. At the first simulation time step corresponding to time $t=0 \times \Delta t$, calculated i_{rfl} is stored at memory 1, and so on until step N_b which is the buffer size limit. Beyond this limit, memory cells 1, 2, 3 and on, are overwritten as figure. 7 shows, since their previously stored values are not needed any longer.



$t(ms)$	$i_{rfl,0}$	$i_{rfl,L}$
0.0	*	*
0.03	*	*
0.09	*	*
0.12	*	*
0.15	*	*
0.18	*	*
0.21	*	*
0.24	*	*

Figure 5. Interpolation scheme: Δt integer multiple.



$t(ms)$	$i_{rfl,0}$	$i_{rfl,L}$
0.0	*	*
0.04	*	*
0.08	*	*
0.12	*	*
0.16	*	*
0.20	*	*
0.24	*	*

Figure 6. Interpolation scheme: Δt non integer multiple.

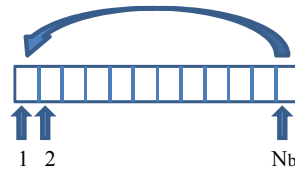


Figure 7. History buffer management.

Linear interpolation is an order 1 numerical procedure and the trapezoidal rule used for the rest of the line model is of order 2. The question arises as to whether or not the order 2 quadratic interpolation should be adopted instead. This has been investigated at (Gutierrez-Robles et al., 2011) and it has been found that the increase in accuracy is marginal.

6. Application examples

The simulation results presented as follows are obtained with the Matlab implementation of the model being described here. These results are compared against those from the phase domain line model in EMTP-RV. Two examples are presented next, first an aerial 9-conductor line and, finally, one for an underground cable. Also a basic m-code for the described phase domain line model is provided at the appendix. The code is given along with the companion routines to perform the first example presented in (Ramos-Leaños & Iracheta, 2010). The reader can readily modify the provided m-code for other applications of interest.

6.1. Aerial line case

The transversal geometry of this test case is shown in figure 8. Phase conductors are 1192.5 ASCR 54/19 and ground wires are 7 No 5 AWLD. This case consists of three coupled three-phase transmission lines. First line (or circuit 1) is composed of conductors 1 to 3, second line (or circuit 2) includes conductors 5 to 6 and the third line (or circuit 3) comprises conductors 7 to 9. The line length is 150 km. The test circuit is shown in figure 9 where the source is 169 kV, Y-grounded, source impedance is determined by its zero and positive sequence data in Ohms: $R_0=2$, $R_1=1$, $X_0=22$, $X_1=15$, and closing times are 0 s for phase a, 0.63 ms for phase b and 0.4 ms for phase c. The simulation time step is 5 μ s.

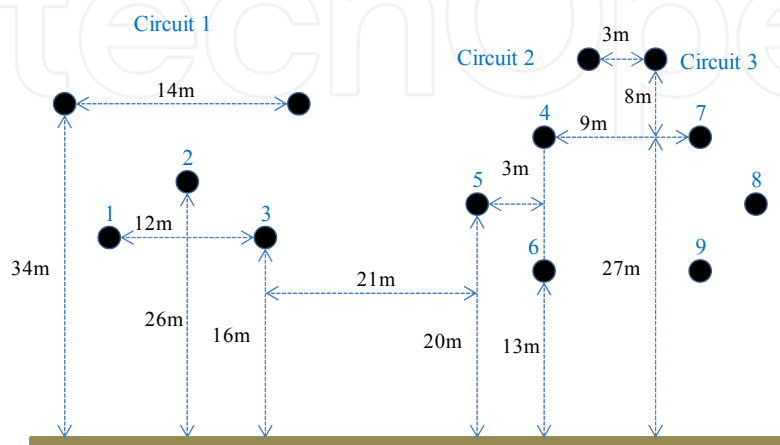


Figure 8. Aerial line transversal geometry.

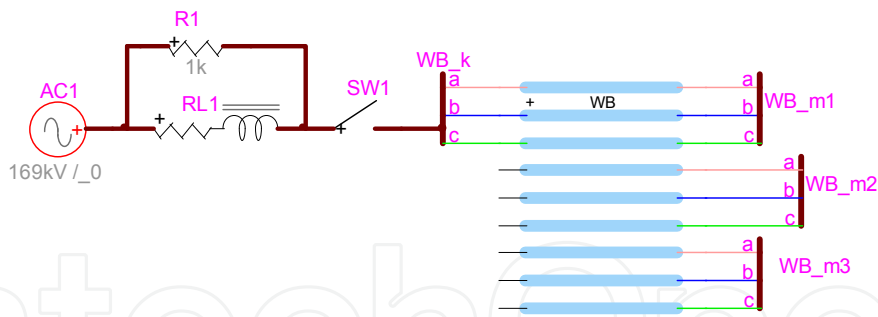


Figure 9. Test circuit for the case of a nine–conductor line.

Simulation results are presented in figure 10 where the receiving end voltage waveforms of circuit 1 are shown, those for phase a are in blue, those for phase b are in green and those for phase c are in red. A dashed line is used for waveforms obtained with EMTP-RV, while a solid line is used for the results with the line model in Matlab. Notice that the two sets of results overlap and not difference can be seen. Figure 10 provides the differences between the two sets of results. Note that the largest difference is around $3\text{e-}9$.

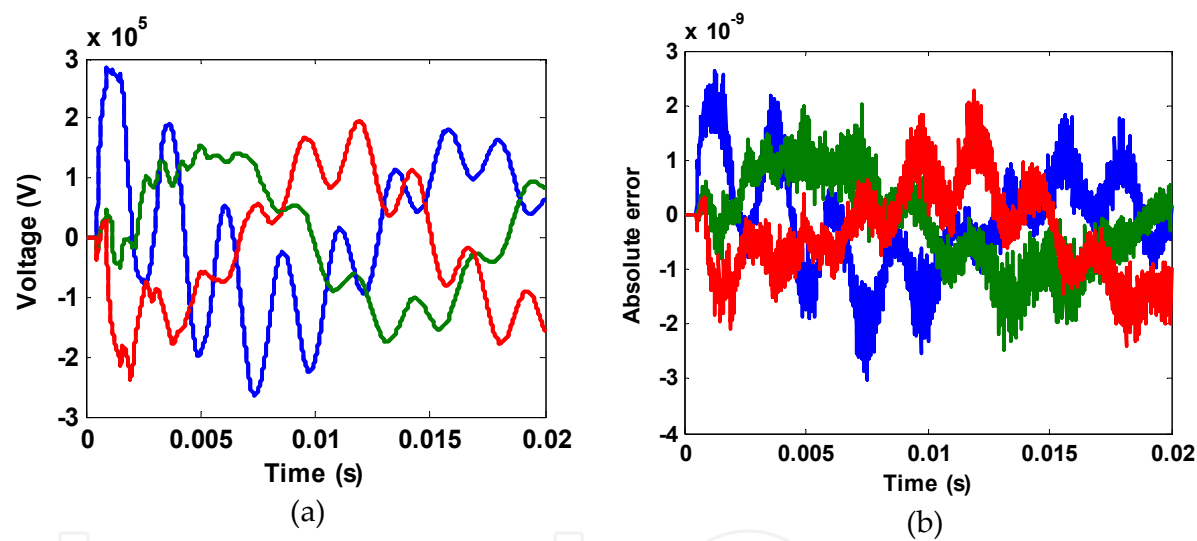


Figure 10. (a) Over voltages at receiving end for conductors 1, 2 and 3, (b) Differences between results with Matlab model and with EMTP–RV.

6.2. Underground cable case

The underground cable system used for this test consists of three single–phase coaxial cables, its transversal layout is shown in figure 11. The Corresponding connection diagram is provided in figure 12. Circuit parameters are given in table 1, the cable length is 6.67km and the time step used for the simulation is $1\text{ }\mu\text{s}$. The applied excitation is by a 3ph 169kV ideal source.

The simulation experiment consists in the simultaneous energizing of the three cable cores. The results presented in figure 13 correspond to the core voltages at the far end. Phase a voltages are in blue, phase b voltages are in green and those for phase c are in red. A dashed line is used for the results obtained with EMTP-RV, while a solid line is used for the Matlab

model results. Notice that both sets of results overlap and that no difference can be seen by eye. Figure 13 also shows the difference between the two sets of results which is around $4\text{e-}9$. Compared to the $1.69\text{e+}5$ amplitude of the excitation source, this difference shows the outstanding accuracy of the Matlab model.

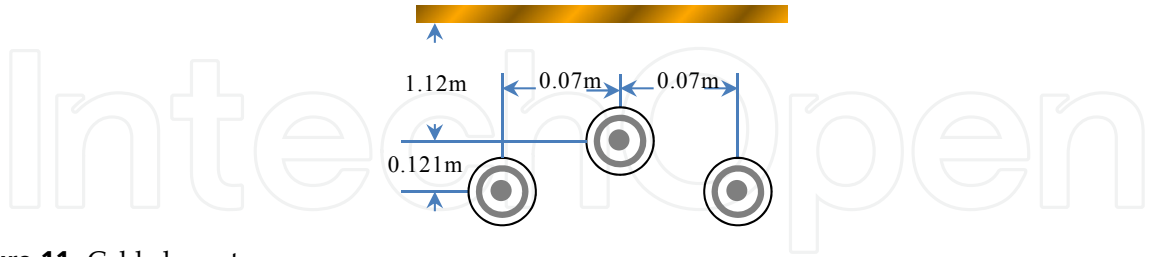


Figure 11. Cable layout.

Radius of inner solid conductor (m)	0.015
Resistivity nuclei/sheath (ohm/m)	$4.25\text{e-}8/2.84\text{e-}8$
Inner/Outer radius of sheath (m)	0.0258/0.0263
Relative permittivity of 1 st & 2nd insulation	2.5

Table 1. Cable data.

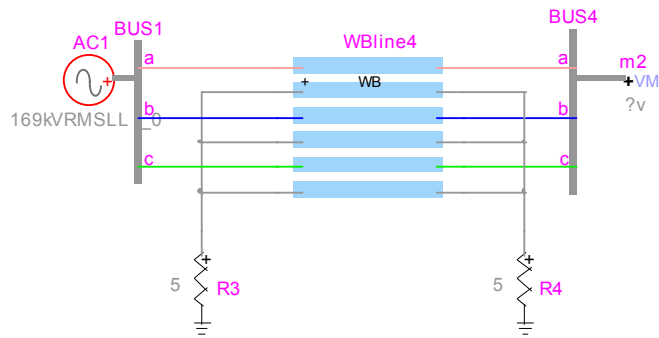


Figure 12. Cable test circuit.

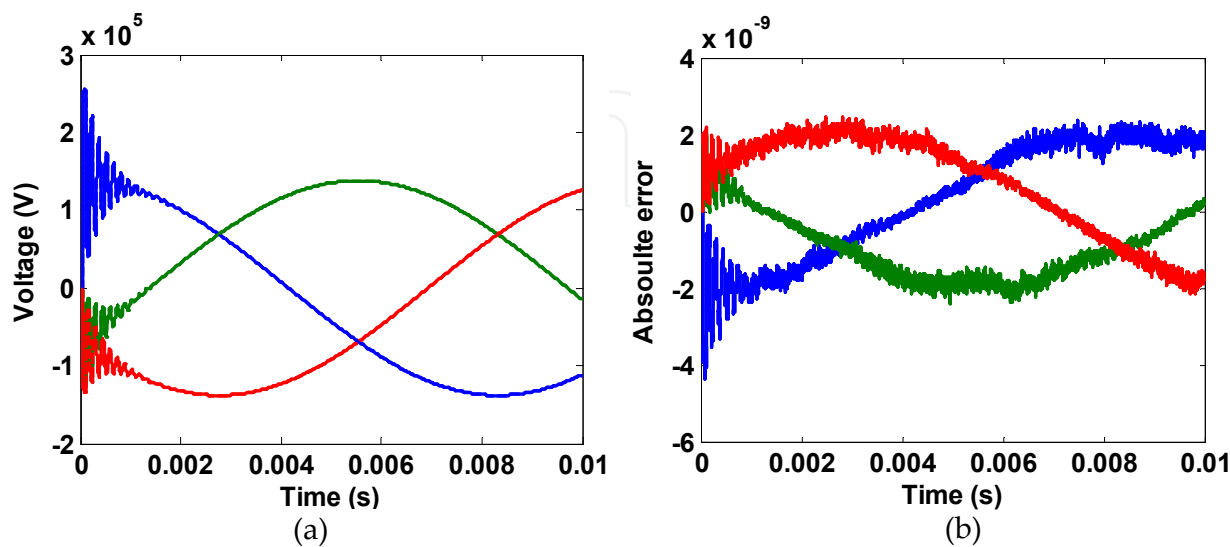


Figure 13. (a) Receiving end core voltages, (b) absolute error.

7. Vector fitting

The goal of VF is to approximate a complex function of frequency by means of a rational function; that is, a quotient of two polynomials of the frequency variable (Gustavsen & Semlyen, 1999). The function to be approximated could be transcendental or could be specified by its values at a number of frequency points. The form of the approximation obtained with VF is that of a partial fraction expansion:

$$f(s) \cong \sum_{n=1}^N \frac{r_n}{s + \bar{p}_n} \quad (55)$$

VF estimates the system parameters by means of a two-stage linear least-squares procedure. First a set of initial poles for the partial fraction basis (55) is selected and relocated iteratively until a prescribed convergence criterion is attained. Then, convergence is tested by means of a second linear least-squares procedure in which the previously obtained poles are fixed and the corresponding residues are taken as the unknown parameters.

Consider the following relation (Gustavsen & Semlyen, 1999):

$$\sum_{n=1}^N \frac{\hat{r}_n}{s + \bar{p}_n} \cong f(s) \left(1 + \sum_{n=1}^N \frac{\tilde{r}_n}{s + \bar{p}_n} \right), \quad (56)$$

where, N is the order of approximation, \bar{p}_n represents the unknown poles and \hat{r}_n and \tilde{r}_n are unknown residues. Poles are initialized by values distributed logarithmically over the frequency range of interest. Expression (56) is now rewritten as follows:

$$\sum_{n=1}^N \frac{\hat{r}_n}{s + \bar{p}_n} - \left(\sum_{n=1}^N \frac{\tilde{r}_n}{s + \bar{p}_n} \right) f(s) \cong f(s). \quad (57)$$

An over-determined least squares equation-system is then obtained by evaluating (57) at a number M of specific frequencies, with $M > 2N$:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad (58)$$

where \mathbf{A} is the $M \times 2N$ matrix whose elements depend on the poles, \mathbf{x} is the $2N$ -dimension vector of unknown residues and \mathbf{b} is the M -dimension vector with the values of the function to be approximated (Gustavsen & Semlyen, 1999). Special care is taken to accommodate next to each other those complex-conjugate pairs of pole-residues that can arise. Expression (58) is solved through an iterative process represented symbolically as follows:

$$\mathbf{A}^{(j-1)}\mathbf{x}^{(j)} = \mathbf{b}, \quad (59)$$

where $(j-1)$ and (j) represent super-indexes and j is the iteration index. $\mathbf{A}^{(0)}$ is obtained from the initial poles with logarithmic distribution over the frequency range of interest

(Gustavsen & Semlyen, 1999). As (59) is solved in the first iteration, a second step is to use the obtained residue values to recalculate new poles for the function to be fitted $f(s)$. This is accomplished by computing the eigenvalues of the following matrix \mathbf{Q} (Gustavsen & Semlyen, 1999):

$$\mathbf{Q} = \mathbf{W} - \mathbf{g}\tilde{\mathbf{x}}^T, \quad (60)$$

where \mathbf{W} is a diagonal matrix containing previously calculated poles \bar{p}_n , \mathbf{g} is a vector of ones and $\tilde{\mathbf{x}}$ is a vector containing the \tilde{r} terms only. The reason for using (60) is explained next. Let (56) be rewritten as follows:

$$\frac{\sum_{n=1}^N \frac{\hat{r}_n}{s + \bar{p}_n}}{\sum_{n=1}^N \frac{\tilde{r}_n}{s + \bar{p}_n} + 1} = \frac{\prod_{n=1}^{N'} (s + \hat{z}_n)}{\prod_{n=1}^N (s + \tilde{z}_n)} \cong f(s). \quad (61)$$

It is clear in (61) that the two polynomials containing the poles \bar{p}_n cancel each other, and that the zeros \tilde{z}_n become the poles of $f(s)$. Notice further that the denominator on the left-hand-side of (61) can be written as follows:

$$\sum_{n=1}^N \frac{\tilde{r}_n}{s + \bar{p}_n} + 1 = \frac{\prod_{n=1}^N (s + \tilde{z}_n)}{\prod_{n=1}^N (s + \bar{p}_n)}. \quad (62)$$

Zeros \tilde{z}_n are then obtained by finding the roots of (Gustavsen & Semlyen, 1999)

$$\sum_{i=1}^N \left(\tilde{r}_i \prod_{n=1, n \neq i}^N (s + \bar{p}_n) \right) + \prod_{n=1}^N (s + \bar{p}_n) = 0, \quad (63)$$

which is equivalent to finding the eigenvalues of \mathbf{Q} in (60) (Gustavsen & Semlyen, 1999).

The newly found set of poles is replaced in (55) to determine a new set of residues r_n . This is again an over-determined linear system. The fitting error is tested at this stage for each available sample of $f(s)$. If the error level is not acceptable, the new poles are used to restart the procedure as with (56). If the desired error limit is not met after a pre-specified number of iterations, then, the order of approximation N is increased and the iterative procedure is restarted (Gustavsen & Semlyen, 1999).

Even in most cases where initial poles are not chosen adequately, VF is capable of finding a solution at the expense of more iterations. In some cases an iteration can produce unstable poles; these poles simply are flipped into the left-hand-side part of the complex plane (i.e., the stable part) and a new solution is searched (Gustavsen & Semlyen, 1999).

8. Conclusions

Proper design and operation of present-day power systems and apparatuses each time require accurate simulations of their electromagnetic transient performance. An important aspect of these simulations is the realistic representation of transmission lines by digital computer models. The ULM is the most general line model available today, mostly with EMTP-type programs. By being of relatively recent creation, this model still is a subject for substantial improvements in accuracy, stability and computational efficiency. It has been postulated in this work that, both, researchers and power system analysts will benefit considerably from the full understanding of the theoretical basis of the ULM, as well as from counting with a ULM-type code that is easy to understand and modify. It has been contended also that the best way to carry out ULM research and development is by providing a model version in an interpretive environment and Matlab has been the platform chosen for this. This chapter provides a comprehensive description of the theoretical basis of ULM, phase domain line model. In addition to this, full description of a ULM prototype in Matlab has been provided here, along with executable code and typical application examples.

Author details

Octavio Ramos-Leaños

École Polytechnique de Montréal, Canada

Jose Luis Naredo

Cinvestav-Guadalajara, Mexico

Jose Alberto Gutierrez-Robles

University of Guadalajara, Mexico

Appendix

CODE EXECUTION

The following code provides the line model described in the paper and it is embedded into an application example. It simulates the simultaneous energizing of a 150 km long aerial line. At the source side the three voltage sources have a $600\ \Omega$ Thevenin impedance. The program asks for the type of source (unit step or three phase sinusoids). At the load end the line is open. Figure 14 shows the geometry of the simulated line. Figure 15 shows the sending and receiving voltages for the unit step source, while Figure 16 shows the sending and receiving voltages for the sinusoidal source.

Note at Figure 15 that waveforms for phases A and C are equal and their plots are superposed. This is because the symmetry of the line and the excitation.

Main program

```

clear all
clc
% m file to set line data
LineData
% Per unit length parameters
[Zg,Zt,Zc,Yg,ZL,YL] = LineParameters(Mu,Eo,Rsu,Geom,Ncon,Ns,w);
% Modal Parameters
for k=1:Ns
    [Yc(:,k),Vm(k,:),Hm(k,:)] = ABYZLM(ZL(:,k),YL(:,k),lenght,w(k));
end
% Characteristic Admittance Fitting
[YcPoles,YcResidues,YcConstant,YcProportional] = YcFit(Yc,f,Ns,Ncon);
% Hk fit
[HkPoles,HkResidues,HkConstant,HkProportional,md] =
HkFitTrace(Hm,Vm,ZL,YL,f,Ns,lenght,Ncon);
% m file to execute simulation loop.
SimulationLoop

```

Code to Load LineData

```

% Line Geometry
% column 1 -- conductor number
% column 2-- x position of each cond in m
% column 3-- y position of each cond in m
% column 4-- radii of each conductor
% column 5-- number of conductor in bundle
% column 6-- distance between conductors in bundle
% column 7 -- conductor resistivity
% column 8 -- conductor relative permittivity
% column 9-- line lenght in m
Geom=[1 0 20 0.0153 3 0.4 2.826e-8 1e3 150e3
      2 10 20 0.0153 3 0.4 2.826e-8 1e3 150e3
      3 20 20 0.0153 3 0.4 2.826e-8 1e3 150e3];
lenght = Geom(1,9); % Line lenght
Ncon = Geom(max(Geom(:,1)),1); % # of cond
Rsu = 100; % Earth resistivity Ohm-m
Mu = 4*pi*1E-7; % Henry's/meters
Eo = (1/(36*pi))*1E-9; % Farads/meters
Rhi = 9.09E-7; % Ohm-m resistivity of the iron.
Ral = 2.61E-8; % Ohm-m res of the aluminum.
Rhg = 2.71E-7; % Ohm-m res of the sky wires.
Ns = 500; % Number of samples
f = logspace(-2, 6, Ns); % Vector of log spaced Frequencies
w = 2*pi*f; % Vector of freqs in radian/sec.

```

Function LineParameters

```

function [Zg,Zt,Zc,Yg,ZT,YT]=LineParameters (Mu,Eo,Rsu,Geom,Ncon,Ns,w)
% Function to compute the distances between conductor
[Dij,dij,hij]=Height(Geom);

```

```

Zg = zeros(Ncon,Ncon,Ns);
Zt = zeros(Ncon,Ncon,Ns);
Zc = zeros(Ncon,Ncon,Ns);
Yg = zeros(Ncon,Ncon,Ns);
Zcd = zeros(Ncon,Ns);
Zaf = zeros(Ncon,Ns);
P = (1./sqrt(j*w*Mu/Rsu)); % Complex depth
Pmatrix = log(Dij./dij); % Potential Coeff. Matrix
Pinv = inv(Pmatrix); % Inverse of Pmatrix
% Loop to compute matrices at all frequencies
for kl = 1:Ns
    % Geometric impedance
    Zg(:,kl) = (j*w(kl)*Mu/(2*pi))*Pmatrix;
    % Earth impedance
    for km = 1:Ncon
        for kn = 1:Ncon
            if km == kn
                Zt(km,km,kl) = (j*w(kl)*Mu/(2*pi))*
                    log(1+P(kl)/(0.5*hij(km,km)));
            else
                num = hij(km,kn)^2 + 4*P(kl)*hij(km,kn) +
                    4*P(kl)^2 + dij(km,kn)^2;
                den = hij(km,kn)^2 + dij(km,kn)^2;
                Zt(km,kn,kl) = (j*w(kl)*Mu/(4*pi))*
                    log(num/den);
            end
        end
    end
    % Geometric admittance
    Yg(:,kl) = (j*w(kl)*2*pi*Eo)*Pinv;
end
% Conductor impedance
for kd = 1:Ncon;
    Rcon = Geom(kd,4); % conductor radii in m.
    Nhaz = Geom(kd,5); % # of conductor in bundle
    Rpha = Geom(kd,7); % Resistivity in Ohm-m.
    Zcd(kd,:) = (1/Nhaz)*Rpha./(pi.*Rcon.^2);
    Zaf(kd,:) = (1/Nhaz)*(1+j).*(1./(2.*pi.*Rcon)).*
        sqrt(0.5.*w.*Mu.*Rpha);
    Zc(kd,kd,:) = sqrt(Zcd(kd,:).^2 + Zaf(kd,:).^2);
end
% Outputs
ZT = Zg + Zt + Zc; % Total impedance
YT = Yg; % Total admittance

```

Function ABYZLM

```

function [Yc,Vm,Hmo] = ABYZLM(Z,Y,Lo,w)
[M, Lm] = eig(Y*Z); % Eigenvalues of YZ
Minv = inv(M); % Inverse of eigenvectors matrix
Yc = inv(Z)*(M*sqrt(Lm)*Minv); % Characteristic Admittance

```

```
Gamma = sqrt(diag(Lm)); % Propagation constants.
Vm = w./imag(Gamma); % Modal Velocities
Hmo = diag(expm(-sqrtm(Lm)*Lo)); % Modal propag. Matrix H
```

Function YcFit

```
function [YcPoles,YcResidues,YcConstant, YcProportional]=YcFit(Yc,f,Ns,Ncon)
% Trace of characteristic admittance matrix
for k = 1:Ns
    Ytrace(k,1) = trace(Yc(:,k));
end
Npol = 6; % Number of poles
[Ps] = InitialPoles(f,Npol); % Set initial poles
s = j*2*pi*f.; % Vector of values of variable "s"
Ka=2; % 1.-Strictly proper, 2.-Proper, 3.-Improper
for khg=1:20
    % Fit the trace of Yc (Poles)
    [YcPoles]=Poles(Ytrace.',s,Ps,Ns,Ka);
    Ps=YcPoles;
end
% Residues and constant term for Yc from poles of trace of Yc
for k = 1:Ncon
    for l = 1:Ncon
        Hs(:,1) = Yc(k,l,:); % k-l term of admittance
        [C,D,E]=Residue(Hs.',s,YcPoles,Ns,Ka);
        YcResidues(k,l,:) = C; % k-l residues term
        YcConstant(k,l) = D; % k-l constant term
        YcProportional(k,l)=E; %k-l proportional term
    end
end
```

Function HkFitTrace

```
function [HkPoles,HkResidues,HkConstant, HkProportional,md]=HkFit(Hm,Vm,ZL,YL,f,Ns,
lenght,Ncon);
% Minimum phase of each mode
md = ModeDelay(Hm.',f,lenght,Vm.',Ns,Ncon);
% Computing Idempotents
for k=1:Ns
    % Function to calculate Idempotents of Y*Z
    [Hk] = HmIdem(ZL(:,k),YL(:,k),lenght,f(k), md,Hm(k,:));
    HkIdem(:,k) = Hk; % Idempotents
end
for m = 1:3
    for k=1:Ns
        TraceHk(m,k) = trace(HkIdem(:,m,k));
    end
end
s = j*2*pi*f.; % Vector of the variable "s"
Ka =1;%1.-Strictly proper, 2.-Proper, 3.-Improper
Npol = 5; % Number of poles
[Ps] = InitialPoles(f,Npol); % Set the initial poles
```



```

for m = 1:3
    Hk = TraceHk(m,:);
    for khg=1:10
        [HkPol]=Poles(Hk,s,Ps,Ns,Ka);
        Ps=HkPol;
    end
    HkPoles(m,:)=Ps;
end
% Residues for Idempotent matrices of
% Hm from the poles of each trace.
for m = 1:3
    for k = 1:Ncon
        for l = 1:Ncon
            Hs(:,1) = HkIdem(k,l,m,:); % k-l term
            [C,D,E]=Residue(Hs.',s,HkPoles(m,:),Ns,Ka);
            HkResidues(k,l,m,:) = C; % k-l-m term
            HkConstant(k,l,m) = D; % k-l-m constant
            HkProportional(k,l,m) = E; % k-l-m prop
        end
    end
end
end

```

SimulationLoop program

```

Ts = 0.016; % Observation time
Nt = fix(Ts/Dt); % Number of time steps
t1 = fix(md./Dt); % Delay of H in samples
t0 = fix(max(md)./Dt); % Maximum time delay as expressed in
% number of samples
t = (0:Dt:(Nt+t1)*Dt-Dt); % Vector of time
Ks = menu('CHOOSE THE TYPE OF INPUT SOURCE', '1 -unit step', '2 -sinusoidal');
if Ks == 1 % unit step source
    Isr = ones(Ncon,Nt+t0);
elseif Ks == 2 % sinusoidal source
    Isr(1,:) = sin(337*t);
    Isr(2,:) = sin(337*t+2*pi/3);
    Isr(3,:) = sin(337*t+4*pi/3);
end
NpYc = length(YcPoles); % Number of poles of Yc
NpH = length(HkPoles); % Number of poles for the first
% Idempotent matrix
Ng = 3; %Number of groups
% Initialize the states for both nodes
ZA = zeros(Ncon,NpYc); % State variables
ZB = zeros(Ncon,NpYc); % State variables
YA = zeros(Ncon,NpH,Ng); % State variables
YB = zeros(Ncon,NpH,Ng); % State variables
IfarA = zeros(Ncon,t0+3); % Current at node A
IfarB = zeros(Ncon,t0+3); % Current at node B
VO = zeros(Ncon,1); % Voltage at node A
Vi = zeros(Ncon,Nt+t0); % Voltage at node A

```

```

VL = zeros(Ncon,1);    % Voltage at node B
Vf = zeros(Ncon,Nt+t0); % Voltage at node B
IO = zeros(Ncon,1);    % Current at node A
li = zeros(Ncon,Nt+t0); % Current at node A
IL = zeros(Ncon,1);    % Current at node B
If = zeros(Ncon,Nt+t0); % Current at node B
Iri = zeros(Ncon,Nt+t0); % Current at Y source
Irf = zeros(Ncon,Nt+t0); % Current at Y charge
IfarAint = zeros(Ncon,Ng); % Current at node A
IfarBint = zeros(Ncon,Ng); % Current at node B
% Constants for the state ZA and ZB
Ai(:,1) = (1+(Dt/2)*YcPoles)./(1-(Dt/2)*YcPoles);
Au(:,1) = ((Dt/2)./(1-(Dt/2)*YcPoles));
Bi(:,1) = (Ai+1).*Au;
Gy = zeros(Ncon,Ncon);
for nm = 1:NpYc
    Di(:,nm) = YcResidues(:,nm)*Bi(nm);
    Gy = Gy + YcResidues(:,nm)*Au(nm);
end
% Constants for the states YA and YB
for k = 1:Ng
    K1(:,k) = (1+(Dt/2)*HkPoles(:,k))./(1-(Dt/2)*HkPoles(:,k));
    Ka(:,k) = (((Dt/2)./(1-(Dt/2)*HkPoles(:,k))));
    Ku(:,k) = (K1(:,k)+1).*Ka(:,k);
end
for k = 1:Ng
    for nm = 1:NpH
        K2(:,nm,k) = HkResidues(:,nm,k).*Ka(nm,k);
        K3(:,nm,k) = HkResidues(:,nm,k).*Ku(nm,k);
    end
end
Gy = Gy + YcConstant; % Admittance of the Ish
Yi = diag(eye(3)*[1/600; 1/600; 1/600]); % Admittance of the source, connected at node A
Gys = inv(Gy + Yi); % Impedance to calculate VO
Yr = diag(eye(3)*[1/1e6; 1/1e6; 1/1e6]); % Admittance of load connected at node B
Gyr = inv(Gy + Yr); % Impedance to calculate VL
% Constants terms to perform the interpolation
tm = md - t1*Dt; % Time for the interpolation
% Linear interpolation constants
c1 = tm/Dt;
c2 = 1-c1;
c3 = ones(Ng,1);
% Pointers for the interpolation and the buffer
h1 = t1+1;
h2 = t1+2;
h3 = t1+3;
for k = t0+2:Nt+t0-3
    IfarA(:,1) = IL + Gy*VL + sum(ZB(:,2),2);
    IfarB(:,1) = IO + Gy*VO + sum(ZA(:,2),2);
    % Linear interpolation

```

```

for m = 1:Ng
    IfarAint(:,m) = c2(m)*IfarA(:,t1(m)) + c3(m)*IfarA(:,h1(m)) + c1(m)*IfarA(:,h2(m));
    IfarBint(:,m) = c2(m)*IfarB(:,t1(m)) + c3(m)*IfarB(:,h1(m)) + c1(m)*IfarB(:,h2(m));
end
IfarA(:,2:h3) = IfarA(:,1:h2);
IfarB(:,2:h3) = IfarB(:,1:h2);
for m = 1:NpYc
    ZA(:,m) = Ai(m)*ZA(:,m) + Di(:,m)*VO;
    ZB(:,m) = Ai(m)*ZB(:,m) + Di(:,m)*VL;
end
for l = 1:Ng
    for m = 1:NpH
        YA(:,m,l) = K1(m,l)*YA(:,m,l) + K2(:,m,l)*IfarAint(:,l);
        YB(:,m,l) = K1(m,l)*YB(:,m,l) + K2(:,m,l)*IfarBint(:,l);
    end
end
HistO = - sum(ZA(:,,:),2) + sum(sum(YA(:,,:),3),2);
HistL = - sum(ZB(:,,:),2) + sum(sum(YB(:,,:),3),2);
VO = Gys*(Isr(:,k)+HistO);
VL = Gyr*HistL;
IO = Gy*VO - HistO;
IL = Gy*VL - HistL;
Vi(:,k) = VO;
Vf(:,k) = VL;
Ii(:,k) = IO;
If(:,k) = IL;
end
Iri = Yi*Vi;
Irf = Yr*Vf;
vt = (0:Dt:length(Vi(1,:))*Dt-(t0+4)*Dt)';
N = length(vt);
a1 = t1+1;
a2 = Nt+t1-3;
figure(1),plot(vt,Vi(:,a1:a2),'l',vt,Vf(:,a1:a2))
ylabel('Amplitude in volts')
xlabel('Time in seconds')
legend('Sending end phase A' , 'Sending end phase B' , 'Sending end phase C' , 'Receiving end phase A' ,
'Receiving end phase B' , 'Receiving end phase C')

```

Function Height

```

function[Dij,dij,hij]=Height(Geom)
Ls = Geom(max(Geom(:,1)),1);
Req = zeros(Ls,1);
% Equivalent bundle radii
k4 = sqrt(2*(Geom(:,6)/2).^2);
for nc = 1: Ls;
    if Geom(nc,5)==1
        Req(nc) = Geom(nc,4);
    else
        Req(nc) = (Geom(nc,4).*Geom(nc,5).*k4(nc).^

```

```

(Geom(nc,5)-1)).^(1./Geom(nc,5));
end
end
% Direct and image distances among conductors
for xl = 1:Ls;
    for yl = 1:Ls;
        if xl==yl
            dij(xl,yl)=Req(xl);
            y1=Geom(yl,3);
            hij(xl,yl)=2*y1;
            Dij(xl,yl)=hij(xl,yl);
        else
            x=abs(Geom(yl,2)-Geom(xl,2));
            y=abs(Geom(yl,3)-Geom(xl,3));
            dij(xl,yl)=sqrt(x^2 + y^2);
            y1=Geom(xl,3);
            y2=Geom(yl,3);
            hij(xl,yl)=y1+y2;
            x=abs(Geom(yl,2)-Geom(xl,2));
            y=hij(xl,yl);
            Dij(xl,yl)=sqrt(x^2 + y^2);
        end
    end
end
end

```

Function InitialPoles

```

function [Ps]=InitialPoles(f,Npol)
    even = fix(Npol/2); % # of complex initial poles
    p_odd = Npol/2 - even; % Auxiliary variable to check if number
    % of initial poles is odd
    disc = p_odd ~= 0; % 0 for even Nr of initial poles & 1 - for
    % odd Nr.
    % Set a real pole in case of disc == 1
    if disc == 0 % Even Nr of initial poles
        pols = [];
    else % Odd Nr of initial poles
        pols = [(max(f)-min(f))/2];
    end
    % Set the complex initial poles
    bet = linspace(min(f),max(f),even);
    for n=1:length(bet)
        alf=-bet(n)*1e-2;
        pols=[pols (alf-j*bet(n)) (alf+j*bet(n)) ];
    end
    Ps = pols.'; % Column vector of initial poles

```

Function Poles

```

function [A]=Poles(Fs,s,Pi,Ns,Ka);
    Np = length(Pi); % Length of vector containing starting poles
    CPX = imag(Pi)~=0; % 0 for real pole and 1 for complex pole

```

```

rp = 0; % Initialize the index for real poles
cp = 0; % Initialize the index for complex poles
RePole = []; % Initialize the vector of real poles
CxPole=[];%Initialize the vector of complex poles
% Loop to separate real poles and complex poles
for k = 1:Np
    if CPX(k) == 0 % Real Pole
        rp = rp + 1;
        RePole(rp) = Pi(k);
    elseif CPX(k) == 1 % Complex pole
        cp = cp + 1;
        CxPole(cp) = Pi(k);
    end
end
Lambda = Pi.';
RePole = sort(RePole); % Sort real poles
CxPole = sort(CxPole); % Sort complex poles
Lambda = [RePole CxPole]; % Concatenate poles
I = diag(ones(1,Np)); % Unit matrix
A = []; % Poles
B = ones(Ns,1); % the weight factor
C = []; % Residues
D = zeros(1); % Constant term
E = zeros(1); % Proportional term
KQA = ones(Ns,1);

cpx = imag(Lambda)~=0; % 0 if pole is real and 1 if pole is
% complex.
dix = zeros(1,Np); % Initializes vector of pole types
if cpx(1)~=0 % If the first pole is complex
    dix(1)=1; % real part
    dix(2)=2; % imag part
    k=3; % continue dix for third position
else
    k=2; % If the first pole is real continue dix for the second position
end
% complete the classification of the poles
for m=k:Np
    if cpx(m)~=0 % If the pole is complex
        if dix(m-1)==1
            dix(m)=2; % If the previous position has the real part put 2
% to identifies the imag part
        else
            dix(m)=1; % 1 for the real part of a complex pole
        end
    end
end
end
% Creates matriz A divided in four parts A = [A1 A2 A3 A4]
% A1 = Dk
% A2 = B.*ones(Ns,1)

```

```

% A3 = B.*s
% A4 = -Dk*Fs
Dk=zeros(Ns,Np); % Initialize matrix with zeros
for m=1:Np % Iterative cycle for all poles
    if dix(m)== 0 % For a real pole
        Dk(:,m) = B./(s-Lambda(m));
    elseif dix(m)== 1 % For the real part
        Dk(:,m)=B./(s-Lambda(m)) +
            B./(s-Lambda(m)');
    elseif dix(m)== 2 % For the imag part
        Dk(:,m) = i.*B./(s-Lambda(m-1)) -
            i.*B./(s-Lambda(m-1)');
    end
end
% Creates work space for matrix A
A1 = Dk;
A2 = B.*ones(Ns,1);
A3 = B.*s;
for col = 1:Np
    A4(:,col) = -(Dk(:,col).*Fs. ');
end
% Assigns values to A
if Ka == 1
    A = [A1 A4]; % Strictly proper rational fitting
elseif Ka == 2
    A = [A1 A2 A4]; % Proper rational fitting
elseif Ka == 3
    A = [A1 A2 A3 A4]; % Improper rational fitting
else
    disp('Ka need to be 1, 2 or 3')
end
% Creates matrix b = B*Fs
b = B.*Fs. ';
% Separating real and imaginary part
Are = real(A); % Real part of matrix A
Aim = imag(A); % Imaginary part of matrix A
bre = real(b); % Real part of matrix b
bim = imag(b); % Imaginary part of matrix b
An = [Are; Aim]; % Real and imaginary part of A
bn = [bre; bim]; % Real and imaginary part of b
% Routine to applies the Euclidian norm to An
[Xmax Ymax] = size(An);
for col=1:Ymax
    Euclidian(col)=norm(An(:,col),2);
    An(:,col)=An(:,col)./Euclidian(col);
end
% Solving system
Xn = An\b;
Xn = Xn./Euclidian. ';
% Put the residues into matrix C

```

```

if Ka == 1
    C = Xn(Np+1:Ymax); % Strictly proper fitting
elseif Ka == 2
    C = Xn(Np+2:Ymax); % Proper rational fitting
elseif Ka == 3
    C = Xn(Np+3:Ymax); % Improper rational fitting
else
    disp('Ka need to be 1, 2 or 3')
end
% C complex when the residues are complex
for m=1:Np
    if dix(m)==1
        alfa = C(m); % real part of a complex pole
        betta = C(m+1); % imag part of a complex pole
        C(m) = alfa + i*betta; % the complex pole
        C(m+1) = alfa - i*betta; % the conjugate
    end
end
% Now calculate the zeros for sigma
BDA = zeros(Np);
KQA = ones(Np,1);
% Loop to calculate the zeros of sigma which are the new poles
for km = 1:Np
    if dix(km)== 0 % For a real pole
        BDA(km,km) = Lambda(km);
    elseif dix(km)== 1 % For a cp with - imag part
        BDA(km,km) = real(Lambda(km));
        BDA(km,km+1) = imag(Lambda(km));
        KQA(km) = 2;
        Aux = C(km);
        C(km) = real(Aux);
    elseif dix(km)== 2 % For a cp with + imag part
        BDA(km,km) = real(Lambda(km));
        BDA(km,km-1) = imag(Lambda(km));
        KQA(km) = 0;
        C(km) = imag(Aux);
    end
end
ZEROS = BDA - KQA*C.';
POLS = eig(ZEROS).';
%Forcing (flipping) unstable poles to make them stable
uns = real(POLS)>0;
POLS(uns) = POLS(uns)-2*real(POLS(uns));
% Sort poles in ascending order. First real poles and then complex poles
CPX = imag(POLS)~=0; % Set to 0 for a real pole and to 1 for a
%complex pole
rp = 0; % Initialize index for real poles
cp = 0; % Initialize index for complex poles
RePole = []; % Initialize the vector of real poles
CxPole = []; % Initialize the vector of cp

```

```

% Loop to separate real and complex poles
for k = 1:Np
    if CPX(k) == 0 % Real Pole
        rp = rp + 1;
        RePole(rp) = POLS(k);
    elseif CPX(k) == 1 % Complex pole
        cp = cp + 1;
        CxPole(cp) = POLS(k);
    end
end
RePole = sort(RePole); % Sort real poles
CxPole = sort(CxPole); % Sort complex poles
% For conjugate pairs store first the one with positive imag part
CxPole = (CxPole.>');
NewPol = [RePole CxPole];
A = NewPol.>'; % Output

```

Function Residue

```

function [C,D,E]=Residue(Fs,s,Pi,Ns,Ka);
Np = length(Pi);
CPX = imag(Pi)~=0; % 0 for a rp and 1 for cp
rp = 0; % Initialize the index for real poles
cp = 0; % Initialize the index for complex poles
RePole = []; % Initialize the vector of real poles
CxPole=[]; % Initialize the vector of complex poles
% Loop to separate real poles and complex poles
for k = 1:Np
    if CPX(k) == 0 % Real Pole
        rp = rp + 1;
        RePole(rp) = Pi(k);
    elseif CPX(k) == 1 % Complex pole
        cp = cp + 1;
        CxPole(cp) = Pi(k);
    end
end
RePole = sort(RePole); % Sort real poles
CxPole = sort(CxPole); % Sort complex poles
CxPole = (CxPole.>');
Lambda = [RePole CxPole];
I = diag(ones(1,Np)); % Unit diagonal matrix
A = []; % Poles
B = ones(Ns,1); % weight factor
C = []; % Residues
D = zeros(1); % Constant term
E = zeros(1); % Proportional term
cpx = imag(Lambda)~=0; % 0 for rp and 1 for cp
dix = zeros(1,Np); % Vto identifies poles
if cpx(1)~=0 % If the first pole is complex
    dix(1)=1; % put 1 in dix(1) for the real part
    dix(2)=2; % put 2 in dix(2) for the imag part
end

```



```

    k=3;      % continue dix for the third position
else
    k=2; % If the first pole is real continue dix for the second
% position
end
% complete classification of the poles
for m=k:Np
    if cpx(m)~=0 % If the pole is complex
        if dix(m-1)==1
            dix(m)=2; % If the previous position has the real part, set to % 2 to identify the imag part
        else
            dix(m)=1; % put 1 for the real part of a cp
        end
    end
end
end
% Output matrices:
Dk=zeros(Ns,Np);
for m=1:Np
    if dix(m)==0 % Real pole
        Dk(:,m) = B./(s-Lambda(m));
    elseif dix(m)==1 % Complex pole, 1st part
        Dk(:,m) = B./(s-Lambda(m)) + B./(s-Lambda(m)');
    elseif dix(m)==2 % Complex pole, 2nd part
        Dk(:,m) = i.*B./(s-Lambda(m-1)) - i.*B./(s-Lambda(m-1)');
    end
end
% Creates work space for matrices A and b
AA1=Dk;
AA2=B.*ones(Ns,1);
AA3=B.*s;
if Ka == 1
    AA = [AA1]; % Strictly proper rational fit
elseif Ka == 2
    AA = [AA1 AA2]; % Proper rational fit
elseif Ka == 3
    AA = [AA1 AA2 AA3]; % Improper fit
else
    disp('Ka must be 1, 2 or 3')
end
bb = B.*Fs.';
AAre = real(AA); % Real part of matrix A
AAim = imag(AA); % Imaginary part of matrix A
bbre = real(bb); % Real part of matrix b
bbim = imag(bb); % Imaginary part of matrix b
AAn = [AAre; AAim]; % Real and imag part of A
bbn = [bbre; bbim]; % Real and imag part of b
[Xmax Ymax] = size(AAn);
for col=1:Ymax
    Euclidian(col)=norm(AAn(:,col),2);
    AAn(:,col)=AAn(:,col)./Euclidian(col);
end

```

```

end
% Solving system X
Xxn=AA\bn;
X=Xxn./Euclidian.';
% Putting residues into matrix C
C=X(1:Np);
% C is complex when the residues are complex
for m=1:Np
    if dim(m)==1
        alfa = C(m); % real part of a complex pole
        betta = C(m+1); % imag part of a complex pole
        C(m) = alfa + i*betta; % the complex pole
        C(m+1) = alfa - i*betta; % the conjugate
    end
end
% Outputs
if Ka == 1
    A = Lambda.'; % Poles
    C = C; % Residues
    D = 0; % Constant term
    E = 0; % Proportional term
elseif Ka == 2
    A = Lambda.'; % Poles
    C = C; % Residues
    D = X(Np+1); % Constant term
    E = 0; % Proportional term
elseif Ka == 3
    A = Lambda.'; % Poles
    C = C; % Residues
    D = X(Np+1); % Constant term
    E = X(Np+2); % Proportional term
End

```

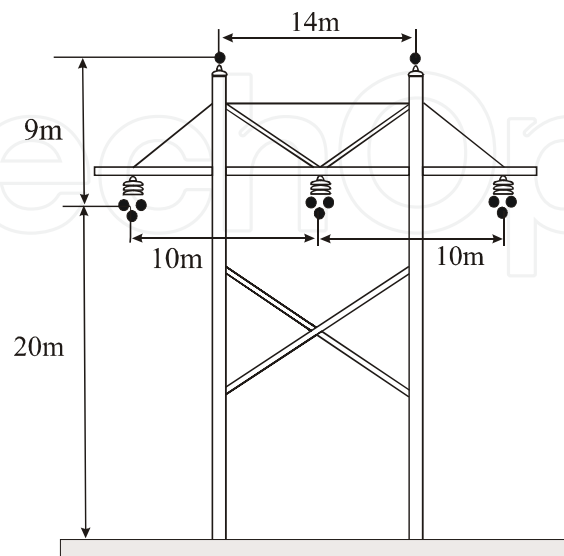


Figure 14. Transversal geometry of aerial line in example.

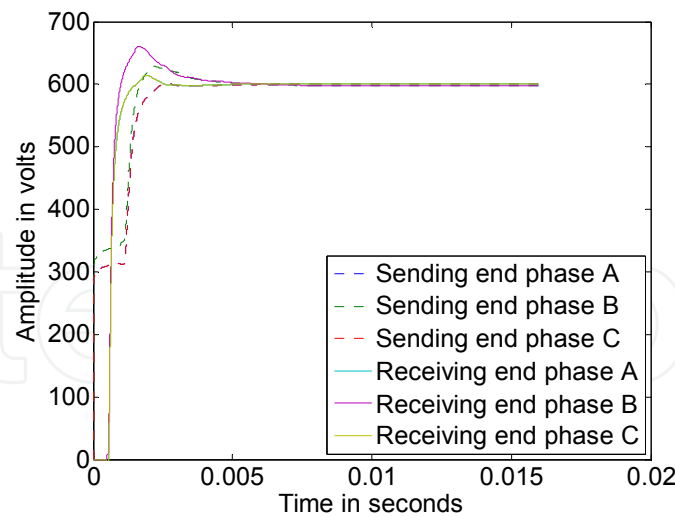


Figure 15. Voltage responses at sending and receiving ends. Unit step excitation.

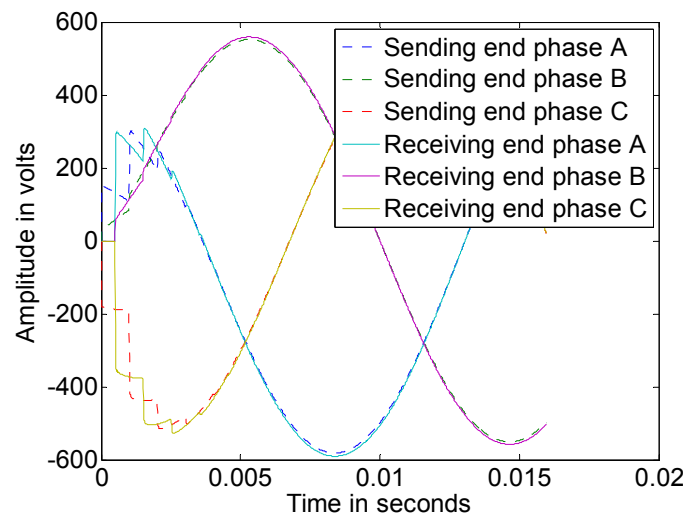


Figure 16. Voltage responses at sending and receiving ends. Sinusoidal excitation.

9. References

- Bode, H. W. (1945). *Network Analysis and Feedback Amplifier Design*, D Van Nostrand Company, London, 1945.
- Brandao Faria, J. A. & Borges da Silva, J. F. (1986). Wave Propagation in Polyphase Transmission Lines a General Solution to Include Cases Where Ordinary Modal Theory Fails, *Power Delivery, IEEE Transactions on*, vol.1, no.2, (April 1986), pp.(182-189).
- Dommel H. W. (1969). Digital computer solution of electromagnetic transients in single and multiphase networks. *IEEE Trans. On Power App. Syst.*, Vol. 88, (July 1969), pp.(388).

- Dommel, H. W. (1992). *EMTP Theory Book*, 2nd ed., Microtran Power System Analysis Corporation, Vancouver, Canada, 1992.
- Gustavsen B. & Semlyen A.(1998). Simulation of transmission line transients using vector fitting and modal decomposition. *Power Delivery, IEEE Trans. On*, Vol.13, No.2,(April 1998), pp.(605-614).
- Gustavsen, B., Semlyen, A. (1999). Rational approximation of frequency domain responses by vector fitting, *Power Delivery, IEEE Transactions on*, vol.14, no.3, (July 1999). pp.(1052-1061).
- Gustavsen, B. & Mahseredjian, J. (2007). Simulation of Internal Overvoltages on Transmission Lines by an Extended Method of Characteristics Approach, *Power Delivery, IEEE Transactions on*, vol.22, no.3, (July 2007), pp.(1736-1742).
- Gustavsen, B., Nordstrom, J. (2008). Pole Identification for The Universal Line Model Based on Trace Fitting, *Power Delivery, IEEE Transactions on*, vol. 23, no. 1, (January 2008), pp. (472–479).
- Gustavsen, B. (2008). "User's Guide for vectfit3.m", Available at <http://www.energy.sintef.no/Produkt/VECTFIT/index.asp>, Aug. 2008.
- Gutierrez-Robles, J. A., Snider, L. A., Naredo, J. L. & Ramos-Leaños (2011). An Investigation of Interpolation Methods Applied in Transmission Line Models for EMT Analysis, *Proceedings of the International Conference on Power System Transients, IPST*, Delft, The Netherlands, Jun. 2011.
- Marcano, F. J. & Marti, J. R. (1997). Idempotent Line Model: Case Studies, *Proceedings of the International Conference on Power System Transients, IPST*, Seattle, USA, Jun. 1997.
- Marti, J. R. (1982). Accurate modeling of frequency dependent transmission lines in electromagnetic transient simulations. *IEEE Trans. On Power App. Syst.*, Vol. 101, No. 1, (January 1982), pp. (147-155)
- Morched, A., Gustavsen B. & Tartibi M.(1999).A universal model for accurate calculation of electromagnetic transients on overhead lines and underground cables. *IEEE Trans. On Power Delivery*, Vol. 14, No. 3,(July 1999), pp. (1032-1038)
- Naredo, J. L., Brandao Faria, J. A., Borges da Silva, J. F. (1986). Discussion to Wave Propagation in Polyphase Transmission Lines a General Solution to Include Cases Where Ordinary Modal Theory Fails, *Power Delivery, IEEE Transactions on*, vol.1, no.2, (April 1986), pp. (188-195)
- Ramos-Leaños O. & Iracheta R. (2010). Wide-Band line model implementation in MatLab for EMT analysis. *Proceedings of the North American Power Symposium*, Arlington USA, September 2010.
- Strang, G. (1988). *Linear Algebra and its Applications*. Third Edition, Harcourt College, 1988.
- Semlyen A. & Dabuleanu A.(1975).Fast and accurate switching transient calculations on transmission lines with ground return using recursive convolutions. *IEEE Trans. On Power App. Syst.*, Vol. 94, No. 2,(April 1975), pp. (561-571)

- Semlyen, A., Abdel-Rahman, M. (1982). A state variable approach for the calculation of switching transients on a power transmission line, *Circuits and Systems, IEEE Transactions on*, vol.29, no.9, (September 1982), pp. (624-633)
- Wedepohl, L. M., (1965). Electrical characteristics of polyphase transmission systems with special reference to boundary-value calculations at power-line carrier frequencies, *Electrical Engineers, Proceedings of the Institution of*, vol.112, no.11,(November 1965), pp.(2103-2112)