Abstract

A data structure and algorithm for representing, and simultaneously solving for all nodes within complex transmission line systems is presented. The method is based on representing the network as a recursive tree structure and solving for the voltage, current, and impedance at each node using recursive programming techniques. First, all frequency dependent parameters within the tree structure are updated, then in a post-order traversing of the tree, the impedances at each node are computed followed by a pre-order traversing of the tree to compute node voltages and currents. The technique is useful in networks with many branches and mixed transmission line characteristics. Applications include the modeling and simulation of pulse propagation in distribution line carrier networks, local area networks, and the digital subscriber loop with bridged taps.
Table of Contents

I. Introduction .......................................................... 2
II. Transmission Line Networks ....................................... 2
III. Recursive Programming and Data Structures .................. 7
IV. Traversing Trees ..................................................... 9
V. Recursive Calculation of Network Impedances ................. 11
VI. Recursive Calculation of Node Currents and Voltage .......... 12
VII. Computer Simulation Results .................................... 14
VIII. Conclusions ...................................................... 22
IX. References .......................................................... 22
I. Introduction

In many digital communication systems, complex transmission line networks are encountered which contain mixed transmission lines with different characteristics and many branches. Examples include the distribution line carrier network, the digital subscriber loop with bridge taps, and certain local area network configurations. The transmission line network usually introduces severe magnitude and phase distortion resulting in the degradation of bit error rate performance in digital transmission systems.

In this paper we present a program for computer modeling and simulation of complex transmission line networks. The network is represented in the computer by a recursive binary tree data structure. Using recursive programming techniques, the node voltage current, and impedance at each node within the tree structure is computed. In this manner the frequency response of the network, from the source node to the receiving node is computed. The impulse response or the pulse response of the network is then calculated from the frequency response using Fast Fourier Transforms.

Computer programs for modeling transmission line networks have been written using ABCD parameters [1]. In this paper a new technique in which the frequency response at all nodes within the network are obtained simultaneously is presented. The technique is also suitable for the computer aided design and modeling of digital communication systems, with complex transmission line networks. A CAD tool is currently being developed for this purpose and a technical report describing the graphical capture, editing, and analysis of networks is available by in [5]. The interactive graphics is based on the graphical kernel system (GKS) [4]

II. Transmission Line Networks

Consider the basic problem of simulating pulse transmission through a loaded transmission line. Assuming that the pulse is bandlimited with a cutoff frequency of $f_c$, we can obtain the pulse response by computing the inverse FFT of the complex multiplication of the frequency response of the pulse and the transmission line network. Therefore, as a first step in calculating the frequency response of the network, we analyze the network response to a
single sinusoid of frequency $f_0$. Consider the loaded transmission line connected to the generator $E_g$ through a source impedance $Z_s$ as shown in Fig. 1 [2].

![Diagram of generator connected to loaded transmission line](image)

**Figure 1. Generator connected to loaded transmission line**

The voltage and current at any point on the transmission line can be obtained from the following expressions:

\[
v(x) = \frac{v_s z_0}{z_0 + z_s} e^{-\gamma x} \frac{1 + \Gamma_L e^{-2\gamma L}}{1 - \Gamma_s \Gamma_L e^{-2\gamma L}}
\]

\[
i(x) = \frac{v_s}{z_0 + z_s} e^{-\gamma x} \frac{1 - \Gamma_L e^{-2\gamma L}}{1 - \Gamma_s \Gamma_L e^{-2\gamma L}}
\]

In the above expressions

\[
\gamma = \sqrt{(r + j\omega)(g + j\omega)}
\]

is the propagation constant and

\[
z_0 = \sqrt{\frac{r + j\omega}{g + j\omega}}
\]

is the characteristic impedance of the transmission line. The expressions for the source and load reflection coefficients are,
\[ \Gamma_L = \frac{Z_L - Z_0}{Z_L + Z_0} \]  
(5) 

\[ \Gamma_s = \frac{Z_s - Z_0}{Z_s + Z_0} \]  
(6) 

The expression for \( v(x) \) includes the superposition of all waves reflecting from the source and load mismatches. This can be seen by a Taylor series expansion of (1)

\[
v(x) = \frac{v_s Z_0}{Z_0 + Z_s} \left[ e^{-\gamma x} + \Gamma_L e^{-\gamma(Lx)} + \Gamma_L \Gamma_s e^{-\gamma(2Lx)} + \frac{1}{2} \frac{\gamma}{\Gamma_L} \Gamma_s e^{-\gamma(3Lx)} + \Gamma_L \Gamma_s e^{-\gamma(3Lx)} + \ldots \right]
\]  
(7) 

To obtain the shape of the pulse at the load we evaluate \( v(L) \) at frequencies from \( f=0 \) to \( f=f_c \) in discrete steps where \( f_c \) is the cutoff frequency of the bandlimited pulse. The number of points must be a power of 2 such that the inverse FFT may be used to obtain the sampled pulse response at the load.

Consider now the case where the boundary voltage and current are known on a section of transmission line. See Fig. 2. Evaluate \( v(0) \) in (4) and then compute.

\[
\frac{v(x)}{v(0)} = \frac{e^{-\gamma x} + \Gamma_L e^{-2\gamma(Lx)}}{1 + \Gamma_L e^{-2\gamma L}}
\]  
(8) 

Also

\[
\frac{i(x)}{i(0)} = \frac{e^{-\gamma x} - \Gamma_L e^{-2\gamma(Lx)}}{1 - \Gamma_L e^{-2\gamma L}}
\]  
(9)

Thus using (8) and (9) the voltage and current can be evaluated at any point on the transmission line given the boundary voltage and current.
Figure 2. Section of transmission line with boundary voltages and currents

With the above preliminaries, we will examine the simple network in Fig. 3 and present a methodology for its solution. In Fig. 3, the nodes have been labeled n1 through n5. To solve this network, that is to obtain the voltage and current at each node and at any location within the network, consider equation (1). This equation suggests that if the impedance at node n1 was known then the voltage and current at node n1 can be calculated from the generator and source impedance. Thus the first step is to obtain the impedance at n1. This impedance is seen to consist of the parallel combination of the impedance looking into n5 and n2 from n1.

Figure 3. Example transmission line network
These impedances can be obtained by noting that (Fig. 4),

\[
Z_{in}(x) = \frac{1 + \Gamma e^{-2\kappa (Lx)}}{1 - \Gamma e^{-2\kappa (Lx)}} Z_0
\]

(10)

Figure 4. Input impedance of a loaded transmission line.

Thus, the first step is to calculate the impedances looking into \( n_3 \) and \( n_4 \) from \( n_2 \). The parallel combination forms the impedance at \( n_2 \). The impedance at \( n_1 \) is thus calculated by the parallel combination of the impedances looking into \( n_2 \) and \( n_5 \).

Therefore, the following methodology is suggested for solving the network. In the first pass, starting from the three loaded end nodes, the impedances are calculated and the parallel combination of these impedances at the parent node forms the parent node impedance. Working backward in this manner, the impedance at the root node (\( n_1 \) in the example) is calculated. Using (1) the voltage and current at the root node \( n_1 \) is calculated. Using (8) and (9) and the boundary voltages and currents, calculated at the parent node, the voltage and current at each node in the network can be calculated. Note that the current at each node is split into two currents flowing into each node.

In the next section, a computer program will be introduced which uses recursion and recursive data structures available in C to solve complex transmission line networks.
III. Recursive Programming and Data Structures

To introduce the algorithm for solving a complex transmission line network, we first consider the case where the network is limited to the binary tree structure shown in Fig. 5. In the figure, the generator is connected to the root of the tree through a source impedance $Z_s$. The tree consists of nodes which are either parents or leaves. A leaf is a node which is terminated on a load. For example, $n_3$, $n_4$, $n_6$, $n_7$, $n_9$, $n_{11}$, and $n_{12}$. Parent nodes have two branches. A left branch and a right branch. Nodes $n_1$, $n_2$, $n_5$, $n_8$, and $n_{10}$ are parent nodes.

In general each branch represents a transmission line with different characteristics and lengths. Each section of transmission line is associated with the node on which it terminates. Thus the section of transmission line from the generator to the root node $n_1$ is described in the data structure pointed to by $n_1$. This concept is described below.

Each node has an associated data structure which occupies memory locations. A pointer can be defined which points to the data structure in memory. As nodes are added to the tree, memory is
dynamically allocated for the data structure and a pointer is defined. Thus, for the nodes of the network in Fig. 5 the following data structure can be defined in C.

```c
struct node {  
    struct node *left;  
    struct node *right;  
    struct node *parent;  
    char name [16];  
    float r,l,c,g;  
    float length  
    complex Z_Left;  
    complex Z_Right;  
    complex Z_L;  
    complex node_voltage;  
    complex left_current;  
    complex right_current;  
    complex input_current;  
    complex Z_0;  
    complex gamma;  
}
```

Within the data structure definition are three pointers to data structures of the same type. Thus the data structure is recursive. Two pointers point to the left and right nodes while the third pointer points to the parent node. Three cases are immediately evident. If the node is a leaf then the left and right node pointers are NULL. Otherwise, they will point to the left and right child nodes attached to the node. If the node is the root node, then the pointer to the parent will be NULL.

The other data types within the structure represent data necessary to describe the node. These can be classified into two groups. One group defines the name of the node and the characteristics of the transmission line (e.g., r,l,c,g and Z_0 and γ). The other group represents data which are calculated and depend on the network. These include the voltage at the node, the current flowing into the right and left nodes, and the impedances looking into the nodes.

It is very convenient to access data in a data structure using pointers to data structures. For example, to assign the variable Z the value of the characteristic impedance at the node pointed to by np we write,
\[ Z = np \rightarrow Z_0 \]

To access the characteristic impedance of the left child node of the node pointed to by \( np \) we write,

\[ Z = np \rightarrow left \rightarrow Z_0 \]

At this stage, we will describe recursive functions which are used to compute the impedances, voltage and currents at each node. First we introduce two methods for traversing tree data structures.

**IV. Traversing Trees**

There are general methods for traversing trees [3]. We will apply two of these methods to solve the tree network.

**Postorder Listing**

Postorder traversing of trees is illustrated in Fig. 6. This method is useful in the first pass needed prior to solving for the voltages and currents of the network. As pointed out earlier the impedances at each node must be computed. Thus in Fig. 6, the impedance at 3 is the parallel combination of the impedances of the loaded transmission lines 1 and 2. Similarly, the impedance at 6 is computed from 4 and 5. Once the impedance of 3 and 6 are computed, the impedance at 7 can be calculated and so on. Careful study of the figure will show that the numbering schemes corresponds to the order in which the impedance calculations must be carried out. This order of traversing the tree is termed postorder listing. The method is summarized below [3]:

1. If a tree is composed of only a single node, the post order listing consists of just that single node.
2. If a tree consists of more than one node, the postorder listing consists of the postorder listing of each subtree, in left-to-right order, followed by the root.
Preorder Listing

This method is used in the calculation of the voltages and currents at each node once the impedances have been determined. Preorder listing is illustrated in Fig. 7. Thus, once the boundary voltage at node 1 is known, the voltage at node 2 can be computed (since the impedance at 2 is also known from the first postorder traversing in computing the impedance). From 2, the voltage at 3 and 6 can be computed and so on. The preorder listing method is summarized below [3]:
(1) If a tree is composed of a single node, the preorder listing consists of just that single node.
(2) If a tree consists of more than one node, the preorder listing consists of the root, followed by the preorder listing of each sub tree in left-to-right order.
V. Recursive Calculation of Network Impedances

The following C code presents a recursive function that calculates the impedance at each node using a post order traversing of the tree structure:

(1) `Complex calculate_impedance (root)`
    `struct node * root;`
    `{`
    
    (2) `if ((root-> left == NULL) && (root -> right == NULL)) {`
        `/*
           * we are at a leaf
           * calculate impedance a distance root -> length away from
           * the load, root -> Z_L
           * calculate reflection coefficient at load
           */`
        `rcl = calc_rcl (root-> Z_L,root -> Z_0);
        return (line_impedance (rcl, root -> Z_0,`
                `root -> gamma,root -> length));`
    }
    
    (3) `root -> left_impedance = calculate_impedance (root -> left);`
    `root -> right_impedance = calculate_impedance (root -> right);`
/*
 * calculate parallel combination of left and right
 * impedance
 */

(4) root -> Z_L = Z_impedance_parallel (root -> left_impedance,
     root-> right_impedance);

rcl = calc_rcl (root ->Z_L, root -> Z_0);

(5) return ((line_impedance(rcl, root ->Z_0 , root -> gamma,
     root->length));

Function explanation:

(1) The function argument, root, is a pointer to a node within the tree. The function returns the impedance "looking into a node," and it is complex.

(2) The function checks to see if the node is a leaf. If it is, then it calculates and returns the impedance looking into the leaf node. This is one of the terminating conditions of the recursive function. In other words, once a leaf node is reached, the function returns. The function line_impedance () calculates the impedance based on equation (10)

(3) If the node is not a leaf, then the function recursively calls itself by passing the left branch node pointer. After returning the impedance looking into the left node, the function is recursively called to calculate the right branch node impedance.

(4) Once the left and right impedances at the node are known, the parallel combination is computed. This produces the total node impedance.

(5) At this point the impedance looking into the node is computed and returned by the function. This is also another terminating condition.

After this function is called, the impedances at all nodes are stored in the data structures pointed to by each node within the network.

VI. Recursive Calculation of Node Currents and Voltage.

After the function calculate_impedance (root) is called, each node contains the terminating or left and right branch impedance. The next recursive function, calculates the total current flowing into each
node including the left and right branch currents. The function makes use of equation (9) to compute the node current $I(L)$ based on knowledge of the boundary current $I(0)$. The function performs a pre-order traversing of the tree network.

(1) calc_current (root, i_input)
    struct node * root;
    complex i_input;
    {
        complex line_current();
        /*
         * evaluates boundary current based on Eq. (9)
         */
        complex calc_left_current ();
        complex calc_right_current ();
        /*
         * calculate the total node current based on equation (9)
         * i_input is the boundary current corresponding to I(0) in
         * the equation
         */
        (3) root -> input_current = line_current (root, i_input, root ->ZL ,
            root -> length);
        (4) if (root -> left != (struct node *) NULL) {
            root -> left_current = calc_left_current (root -> left_impedance, 
                root -> right_impedance, root -> input_current);
                calc_current (root -> left, root -> left_current);
        }
        (5) if (root -> right != (struct node *) NULL) {
                root -> right_current = calc_right_current (root -> left_impulse, 
                    root -> right_impulse, root -> input_current);
                    calc_current (root ->right , root-> right_current);
        }

Explanation:

(1) The root argument is a pointer to the current node, i_input is the complex boundary current corresponding to $I(0)$ in equation (9). That is, it is the total current flowing into the transmission line associated with the node.

(2) The function line_current () computes the current $I(L)$ in equation (9). The functions calc_left_current () and calc_right_current (), calculate the currents flowing into the left and right branches.
This operation calculates $I(L)$ and stores it in the data structure of the node. Note that since this is a recursive procedure for traversing all the nodes of the tree, this operation will occur for each node.

(4) If the left branch is not a termination, then recursively call the function by moving to the node attached to the current node, root. However, first compute the boundary current $I(0)$ flowing into the node, in this case left_current.

(5) The same as in step 4 but for the right branch.

After this procedure is called, all node data structures will contain the total current, and the left and right currents. Note that when the procedure is first called, $i_{input}$ is the total current flowing from the generator into the network. It is computed based on equation (2).

VII. Computer Simulation Results

In the previous sections, we showed how the voltage and currents at each node can be computed. In general, for bandlimited digital transmission over a complex transmission line network, the frequency response of the network and the input impedance looking into the network over a frequency range is required to completely model the network. Thus, the response of the network to a generator with arbitrary source impedance may be computed.

Fig. 8 shows a test network consisting of coaxial cable transmission lines with different characteristics. We are interested in obtaining the impulse response from the sending station to the receiving station.
The computer program computed the voltage at the receiving station at discrete frequency intervals up to 100 MHz. The program then performed a 1024 point inverse FFT to produce the impulse response at a sampling rate of $f_s = 200$ MHz. Due to skin effect at higher frequencies the frequency response of the network is effectively bandlimited. The computed impulse response at node n6 is displayed in Fig. 9.

**Impulse Response**

![Figure 8. Test Network for Computer Simulation](image)

Figure 8. Test Network for Computer Simulation

![Figure 9. Test network impulse response](image)

Figure 9. Test network impulse response
Notice the pure delay corresponding to traveling a distance of 180 m from the sending station to the receiving station. Also, immediately following the main impulse, a large second impulse appears with a delay corresponding to traveling a distance of 40 m. This second impulse is due to reflections from the two open circuit transmission lines of 20 m length each (RG 58/U) which constructively add and propagate to the receiving station.

The response of the test network to a series of three pulses, shown in Figure 10, is given in Figure 11. The pulse widths are 1 µs. The rise and fall times are 100 ns and the period is 4 µs.

![Unwindowed Pulse](image)

Figure 11. Sequence of pulses.
The eye diagram corresponding to exciting the network with a pseudo random sequence of pulses with 1 μs pulse widths, 100 ns rise and fall times and pulse separation of 1.5 μs is shown in Figure 12. It is interesting to compare the eye diagram with the pulse response of Figure 10. The eye diagram gives very useful information about the performance of the network for pulse propagation.
Figure 12. Eye diagram of the test network

An interesting phenomenon associated with complex transmission line systems is that while reliable communication to one node is possible, another node may exhibit significantly lower reliability within the same network. This can be seen by examining Figure 13 through Figure 16. In these figures the eye diagram at various nodes is displayed. Obviously some nodes will behave better than others in terms of error rate performance.
Figure 13. Eye diagram at node n2

Figure 14. Eye diagram at node n4
Eye Diagram

Figure 15. Eye diagram at node n6

Eye Diagram

Figure 16. Eye diagram at node n7
The results presented above were obtained from the program CAPNET written by Joseph Hall [5] which solves complex transmission line networks based on the technique presented in this paper. Using CAPNET a complicated network is graphically captured using flexible graphical editing capabilities. The program also models junctions such as bends, "tees", and transitions between different types of transmission lines. The plots presented in this paper are from sessions in CAPNET in which the user can change parameters, plot the results, and obtain hard copies of the plots without leaving the program. CAPNET uses GKS for its interactive graphics and workstation independent capabilities.

Finally in Fig. 17, the voltage distribution along the 50 m cable connecting the receiving station to the main RG 8/U coaxial cable transmission line is shown at a frequency of 10 MHz. Note the standing wave pattern and loss associated with skin effect in the cable. The source was 5 volts.

![Voltage Distribution](image)

Figure 17. Voltage distribution along a transmission line segment n6.
VIII. Conclusions

In this technical report, a computer program is described which simultaneously solves for all nodes within complex networks of transmission lines. A tree data structure was introduced for representing the network in the computer. Recursive procedures were presented for traversing the tree data structure to compute the impedance, voltage and current at each node within the network. Simulation results were then presented in which the impulse response of a test network composed of transmission lines of various characteristics and lengths was computed. The impulse response was then related to the network in terms of the predicted reflections and delays.

The program efficiently solves complex transmission line networks and has applications in the area of Computer Aided Design (CAD) of digital communication networks. Specific applications include the Distribution Line Carrier Network, the digital subscriber loop, and Local Area Networks.

IX. References.


