APPENDIX A

Description of Computer Software

Several FORTRAN computer programs that implement the methods described in this text are included on a diskette with the textbook. This appendix gives a description of those codes and their algorithms. The codes use certain arrays, matrices, and vectors in which to store information. These array dimensions are determined primarily by the total number of conductors of the transmission line and are stored in parameters at the beginning of the code. An example is the parameter MSIZE:

```
PARAMETER (MSIZE = 99)
```

The user need only change these parameters in order for the program to handle larger numbers of conductors in the line. Each uncompiled code (source code) contains at the beginning a brief description of the code, the array dimensions, and the names and contents of the required input files and the name of the resulting output file.

The programs are written in standard FORTRAN 77 language. A conscious attempt was made to use rudimentary FORTRAN programming commands so that the codes would be compilable with a wide variety of FORTRAN compilers on a large number of platforms.

The codes fall into two distinct categories: those which compute the per-unit-length parameters of inductance and capacitance and those which solve the MTL equations and incorporate the terminal conditions. The codes that determine the per-unit-length parameters are **WIDESEP.FOR** which implements the wide-separation approximations for wires, **RIBBON.FOR** for considering ribbon cables, **PCB.FOR** and **PCBGAL.FOR** for considering lands on printed circuit boards, and **MSTRP.FOR** and **MSTRPGAL.FOR** for considering coupled microstrip structures. All of these codes require an input file which holds the two-dimensional, cross-sectional dimensions and material characteristics of the line. These files are called **WIDESEP.IN, RIBBON.IN, PCB.IN,** and **MSTRP.IN,** respectively. The output of these programs is the file **PUL.DAT.**

The upper triangle of the per-unit-length parameter inductance, L, and capacitance, C, matrices are stored by rows in this file with the inductance matrix given first followed by the capacitance matrix. The input data are printed at the end of these files but are not read by the programs that use these per-unit-length data. These and the remaining codes input the total number of conductors, $(n + 1)$.

The remaining codes solve the MTL equations and incorporate the terminal conditions to give the total solution for the line voltages and/or currents. These codes are further subdivided into those for the frequency-domain solution and those for the time-domain solution. Each of these codes require one or more input files that are described at the beginning of the code listings. For example, the code **SPICEMTL.FOR** requires the input files **SPICEMTL.IN** and **PUL.DAT**. The input file **SPICEMTL.IN** contains structural information such as line length. The file **PUL.DAT** is the output of the codes **WIDESEP.FOR, RIBBON.FOR, PCBGAL.FOR, PCB.FOR, MSTRP.FOR,** or **MSTRPGAL.FOR**. The user may alternatively provide his/her per-unit-length data from other sources. The output file of **SPICEMTL.FOR** is simply **SPICEMTL.OUT**. The output files generally contain either the magnitude and phase of the line voltages and/or currents at each frequency or the line voltages and/or currents at each time step. There is one exception to this: the programs **SPICEMTL.FOR, SPICELPI.FOR, SPICELC.FOR,** and **SPICEINC.FOR** provide as output a SPICE subcircuit that models the MTL on a port basis.

Both the input data and the output data for each code appear as one item per line in the appropriate file. These data are in a form of free field format. The READ statements in the calling program are in an unformatted type:

```
READ (5,*)
```

Each data item appears on a separate line of the input file followed on that line with several blank spaces and an equals sign followed by descriptive remarks. The annotation (in the input and output files) of =***** simply describes the data entry preceeding the equals sign to the user and is not read by the calling program. In order for this to function properly, *at least one blank space must appear between the data item and the equals sign.* This is in the same format in both the input and output files and anticipates that the subsequent program that may use this output file has unformatted reads such as READ (5,*).

A.1 PROGRAMS FOR CALCULATION OF THE PER-UNIT-LENGTH PARAMETERS

In this section we describe six codes for the calculation of the entries in the per-unit-length inductance matrix, L, and capacitance matrix, C: **WIDESEP.FOR** which implements the wide-separation approximations for wires, **RIBBON.FOR** for ribbon cables, **PCB.FOR** and **PCBGAL.FOR** for

printed circuit boards, and **MSTRP.FOR** and **MSTRPGAL.FOR** for coupled microstrip structures. The output of these codes is the file **PUL.DAT** that contains the upper triangles of these matrices with the inductance matrix given first followed by the capacitance matrix.

A.1.1 Wide-Separation Approximations for Wires: WIDESEP.FOR

Required Input Files: WIDESEP.IN
Output File: PUL.DAT

The code computes the entries in the per-unit-length inductance matrix, **L**, for three configurations of widely spaced wires in a homogeneous medium described in Section 3.2.3: (a) $(n + 1)$ wires (3.63), (b) n wires above an infinite, perfectly conducting ground plane (3.66), and (c) n wires within an overall circular cylindrical shield (3.67). The per-unit-length capacitance matrix is obtained from $\mathbf{C} = \mu_0\mu_r\varepsilon_0\varepsilon_r\mathbf{L}^{-1}$ where μ_r and ε_r are the relative permeability and permittivity, respectively, of the homogeneous medium surrounding the wires. These structures are depicted in Fig. A.1. The first data parameter in **WIDESEP.IN** is the number of wires, n, (exclusive of the reference conductor).

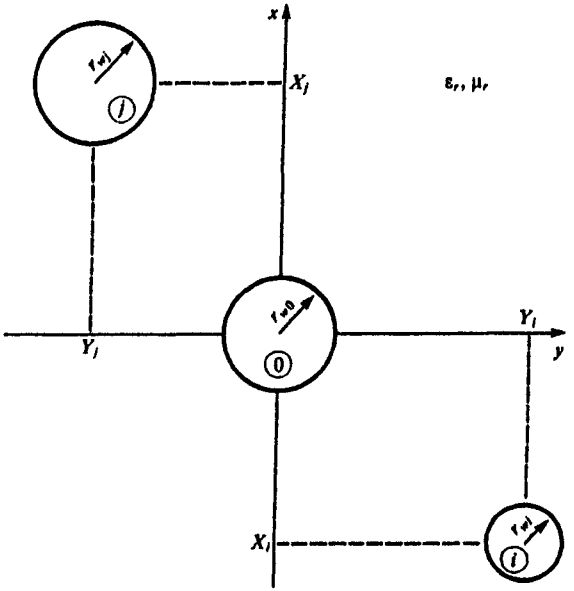
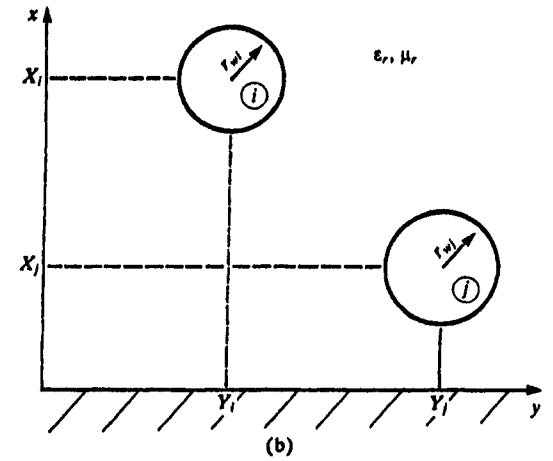
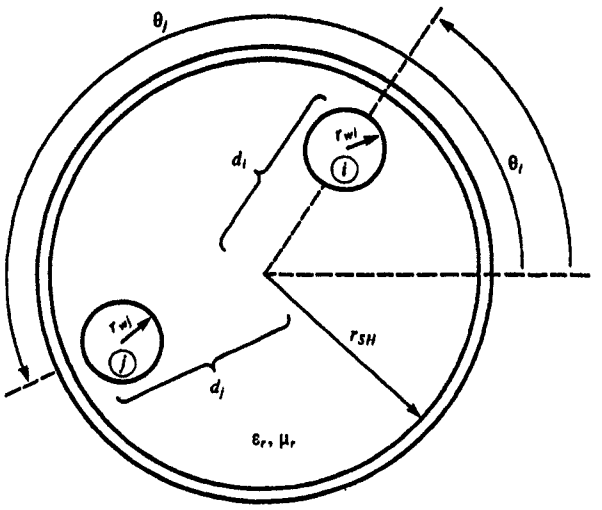


FIGURE A.1 Cross-sectional configurations of wire lines for the **WIDESEP.FOR** FORTRAN program input data: (a) $(n + 1)$ wires, (b) n wires above a ground plane, and (c) n wires within a cylindrical shield.



(b)



(c)

FIGURE A.1 *(Continued)*

The relative permittivity and relative permeability of the surrounding homogeneous medium are the next input parameters. The type of structure is input to **WIDESEP.IN** by the reference conductor parameter: 1 = wire, 2 = ground plane, and 3 = overall shield. The remaining parameters describe the wire radii and their cross-sectional positions.

The first case, *n* wires with another wire as the reference wire, is depicted in Fig. A.1(a). The reference wire is located at the origin of a rectangular coordinate system. There are five pieces of input data in **WIDESEP.IN.**

1. The number of wires, n, (exclusive of the reference wire.
2. The reference conductor type (type = 1).
3. The relative permittivity and permeability of the surrounding homogeneous medium.
4. The radius of the reference wire (in mils).
5. The radii of the other wires (in mils) along with their x coordinates (in meters) and y coordinates (in meters).

The last group are input sequentially for each wire: r_{w1}, x_1, y_1, r_{w2}, x_2, $y_2, \ldots, r_{wn}$, x_n, y_n.

The second case, n wires above an infinite ground plane as the reference conductor, is depicted in Fig. A.1(b). The ground plane forms the y coordinate axis ($x = 0$) of a rectangular coordinate system. There are four pieces of input data in **WIDESEP.IN**.

1. The number of wires, n.
2. The reference conductor type (type = 2).
3. The relative permittivity and permeability of the surrounding homogeneous medium.
4. The radii of the other wires (in mils) along with their x coordinates (height above ground) (in meters) and y coordinates (in meters).

The last group are again input sequentially for each wire: r_{w1}, h_1, y_1, r_{w2}, h_2, $y_2, \ldots, r_{wn}$, h_n, y_n. (The radius of a reference wire is again input but not read by the code.)

The last case, n wires within an overall shield of radius r_{SH} as the reference conductor, is depicted in Fig. A.1(c). There are five pieces of input data in **WIDESEP.IN**.

1. The number of wires, n.
2. The reference conductor type (type = 3).
3. The relative permittivity and permeability of the surrounding homogeneous medium.
4. The (interior) radius of the shield (in mils).
5. The radii of the other wires (in mils) along with their distances from the shield center, d_i, (in mils) and their angular locations, θ_i, (in degrees).

The last group are again input sequentially for each wire: r_{w1}, d_1, θ_1, r_{w2}, d_2, $\theta_2, \ldots, r_{wn}$, d_n, θ_n.

The entries in the upper triangle of the per-unit-length inductance and capacitance matrices are output to **PUL.DAT**. The last items in **PUL.DAT** simply restate the input parameters to insure that the correct problem was solved. These are not read by the subsequent application program.

A.1.2 Ribbon Cables: RIBBON.FOR

Required Input Files: RIBBON.IN
Output File: PUL.DAT

Consider the general N-wire ribbon cable shown in Fig. A.2. The line consists of N identical, dielectric-insulated wires. The wire radii are denoted in the code as RW, the insulation thicknesses are denoted by TD, and the identical adjacent wire spacings are denoted as S. The relative permittivity of the insulation is denoted by ER in the code. These input data are contained in **RIBBON.IN**. The results of the computation are contained in the output file **PUL.DAT**. The last items in **PUL.DAT** simply restate the input parameters to insure that the correct problem was solved. These are not read by the subsequent application program.

The structure of the main program **RIBBON.FOR** is as follows. Utilizing Tables 3.2 and 3.3 we can write the matrix that satisfies the boundary conditions.

1. The potentials of a conductor at points on that conductor due to all charge distributions in the system are set equal to the potential of that conductor.
2. The components of the electric flux density vector, $\mathscr{D} = \varepsilon\mathscr{E}$, that are normal to the dielectric–free-space boundary at points on that boundary are continuous across the boundary.

This corresponds to matrix equation (3.87) which becomes of the form

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \sigma \\ \sigma' \end{bmatrix} = \begin{bmatrix} \phi \\ \mathbf{0} \end{bmatrix} \tag{A.1}$$

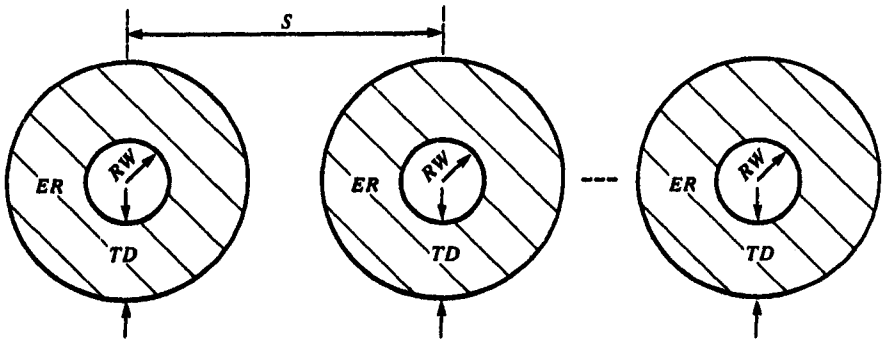


FIGURE A.2 Cross-sectional definition of the ribbon cable parameters for the **RIBBON.FOR** FORTRAN program input data.

Expanding these gives

$$\mathbf{A}\boldsymbol{\sigma} + \mathbf{B}\boldsymbol{\sigma}' = \boldsymbol{\phi} \tag{A.2a}$$

$$\mathbf{C}\boldsymbol{\sigma} + \mathbf{D}\boldsymbol{\sigma}' = \mathbf{0} \tag{A.2b}$$

The first set in (A.2a) enforce the potentials on the conductors and the second set in (A.2b) enforce continuity of the normal components of the electric flux density vector across the dielectric–free-space boundary. We will assume that *the number of Fourier expansion functions around the conductor–dielectric boundary and around the dielectric–free-space boundary are equal.* The number of expansion coefficients around each of these two boundaries is denoted as NF. The $(N \times \mathrm{NF}) \times 1$ vector of Fourier expansion coefficients for the free and bound charge around the conductor–dielectric boundary is denoted by $\boldsymbol{\sigma}$. Similarly, the $(N \times \mathrm{NF}) \times 1$ vector of Fourier expansion coefficients for the bound charge around the dielectric–free-space boundary is denoted by $\boldsymbol{\sigma}'$. The $(N \times \mathrm{NF}) \times 1$ vector $\boldsymbol{\phi}$ contains the potentials of the conductors at the matchpoints on those conductors. The $(N \times \mathrm{NF}) \times 1$ zero vector $\mathbf{0}$ results from the satisfaction of the continuity of the normal components of the electric flux density vector across the dielectric–free-space boundary. The submatrices **A**, **B**, **C**, **D** are $(N \times \mathrm{NF}) \times (N \times \mathrm{NF})$. Because of the repetitive nature of the ribbon cable structure, these submatrices have a special form:

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_N \\ \mathbf{A}_{-2} & \mathbf{A}_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{A}_2 \\ \mathbf{A}_{-N} & \cdots & \mathbf{A}_{-2} & \mathbf{A}_1 \end{bmatrix} \tag{A.3}$$

and **B**, **C**, and **D** have a similar structure. This simplifies the fill time for (A.1). In order to avoid a singular solution matrix in (A.1), the matchpoints are chosen from the following scheme [C.2, C.6]. On each conductor–dielectric interface and each dielectric–free-space interface, the NF matchpoints are separated by the angle

$$\theta = \frac{2\pi}{\mathrm{NF}} \tag{A.4a}$$

These are rotated by an angle

$$\Delta = \frac{\pi}{2\mathrm{NF}} \tag{A.4b}$$

as illustrated in Fig. A.3.

The matrix equations in (A.1) or (A.2) are solved as

$$\boldsymbol{\sigma} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\boldsymbol{\sigma} \tag{A.4a}$$

$$\boldsymbol{\sigma}' = -\mathbf{D}^{-1}\mathbf{C}\boldsymbol{\sigma} \tag{A.4b}$$

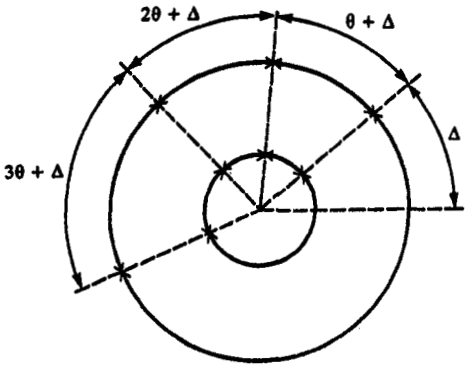


FIGURE A.3 Illustration of the scheme for rotating the matchpoints of a ribbon cable to avoid singular solution matrices.

Once these are solved, the total free charge on each conductor and the elements of the generalized capacitance matrix are obtained according to (3.90). The transmission-line-capacitance matrix, **C**, is obtained from the generalized capacitance matrix according to the algorithm given in (3.19) with the chosen reference conductor IREF. The generalized capacitance matrix *with the dielectric removed* is computed by computing the free charge with the dielectric removed:

$$\boldsymbol{\sigma} = \mathbf{A}^{-1}\boldsymbol{\phi} \tag{A.5}$$

and the transmission-line-capacitance matrix, $\mathbf{C}_o$, is computed for the chosen reference conductor IREF. The transmission-line-inductance matrix is computed according to

$$\mathbf{L} = \mu_o \varepsilon_o \mathbf{C}_o^{-1} \tag{A.6}$$

The upper triangle of **C** and **L** are printed out to the file **PUL.DAT** with **L** printed first. Observe the numbering of the conductors illustrated in Fig. A.4. The conductors to the left of the reference conductor are numbered sequentially from left to right and the remaining conductors to the right of the reference conductor are also numbered sequentially. This numbering is critical to observe when one generates the entries of the matrices and vectors of the terminal characterization as with a generalized Thévenin equivalent in the programs that use these data.

The inversion of the above real-valued matrices is accomplished with a standard Gauss–Jordan subroutine, **GAUSSJ** which is supplied as a part of the code **RIBBON.FOR**. This subroutine was taken with permission from [1].

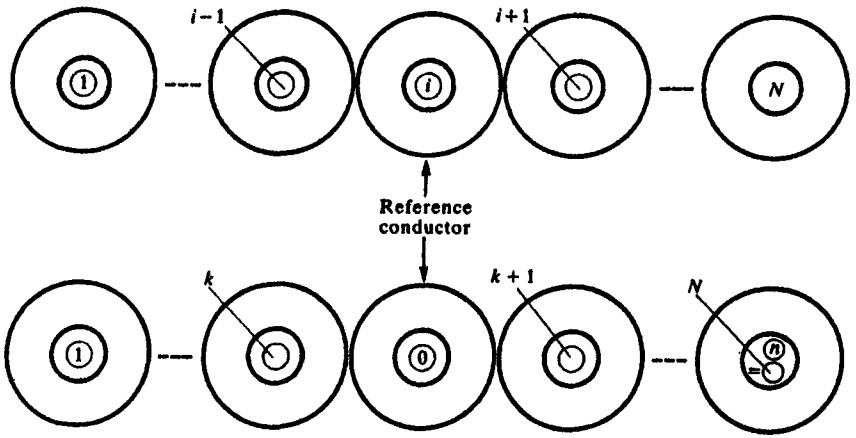


FIGURE A.4 Illustration of the numbering scheme for determining the transmission-line capacitance matrix from the generalized capacitance matrix for a ribbon cable.

A.1.3 Printed Circuit Boards: PCB.FOR, PCBGAL.FOR

Required Input Files: PCB.IN
Output File: PUL.DAT

This code determines the entries in the per-unit-length inductance and capacitance matrices, L and C, for all N-land PCB illustrated in Fig. A.5. The problem that is solved is a special case wherein all lands have the same width, W, and identical edge-to-edge separations, S. The board has thickness, T, and relative dielectric constant, ε_r. The lands are numbered from left to right as shown. The generalized capacitance matrix for the structure is first solved using the results for potential given in Section 3.3.1. One of the input data parameters provided in **PCB.IN** is the number of the desired reference land as illustrated in Fig. A.6. The code then solves for the transmission-line-capacitance matrix from the generalized capacitance matrix based on the desired reference conductor. The

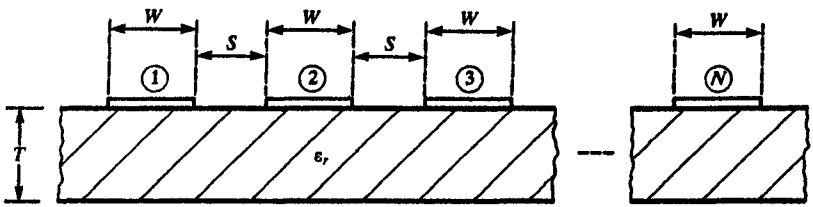


FIGURE A.5 Cross-sectional definition of the printed circuit board parameters for the **PCB.FOR** and **PCBGAL.FOR** FORTRAN program input data.

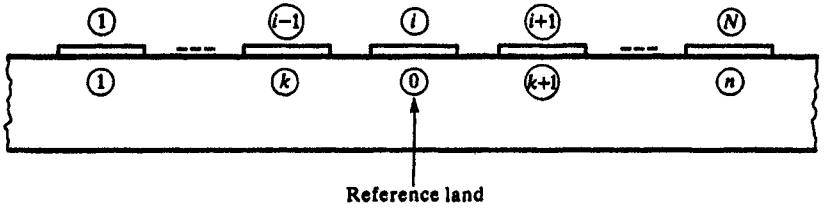


FIGURE A.6 Illustration of the numbering scheme for determining the transmission-line-capacitance matrix from the generalized capacitance matrix for a printed circuit board.

code first solves for the capacitance matrix with the board removed and replaced with free space, $\mathbf{C}_o$, from which the transmission-line-inductance matrix is obtained as $\mathbf{L} = \mu_o \varepsilon_o \mathbf{C}_o^{-1}$. Then the capacitance matrix with the board present, $\mathbf{C}$, is determined.

The matrix given in (3.101) relating the charge distributions over the land subsections to the potentials of the lands is formed and inverted as in (3.102). From this result the generalized capacitance matrix is formed as in (3.103). Then the algorithm in (3.19) is used to obtain the entries in the transmission-line-capacitance matrices, $\mathbf{C}$ and $\mathbf{C}_o$ for the chosen reference land. For both cases, the matrix equation to be solved has, because of the assumption of identical land widths and edge-to-edge spacing, the following structure:

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_N \\ \mathbf{A}_2^t & \mathbf{A}_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{A}_2 \\ \mathbf{A}_N^t & \cdots & \mathbf{A}_2^t & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_1 \\ \boldsymbol{\sigma}_2 \\ \vdots \\ \boldsymbol{\sigma}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2 \\ \vdots \\ \boldsymbol{\phi}_N \end{bmatrix} \tag{A.7}$$

where t denotes the transpose of the matrix, $\boldsymbol{\sigma}_i$ contains the unknown levels of the charge distributions (assumed constant) of each subsection of the i-th land, and $\boldsymbol{\phi}_i$ contains the potentials of the subsections of the i-th land.

The upper triangle of the per-unit-length inductance matrix is printed to **PUL.DAT** followed by the upper triangle of the per-unit-length capacitance matrix. These data are used by all subsequent analysis codes. Additionally, the per-unit-length capacitance matrix with the board removed, $\mathbf{C}_o$, is printed to **PUL.DAT** but is not used by the analysis codes. And finally the problem parameters are printed to **PUL.DAT** so that the user can insure that the problem solved is as desired. The inversion of (A.7) is again performed using the Gauss–Jordan subroutine that was used in **RIBBON.FOR**.

Additionally, the code **PCBGAL.FOR** is supplied. This code implements the pulse expansion–Galerkin method discussed in Section 3.3.1, whereas the code **PCB.FOR** implements the pulse expansion–point matching method. As was shown in Chapter 3, the Galerkin method gives better convergence for a small

number of land subdivisions, but both methods converge rapidly as the number of land subsections is increased.

A.1.4 Coupled Microstrip Structures: MSTRP.FOR, MSTRPGAL.FOR

Required Input Files: MSTRP.IN
Output File: PUL.DAT

These codes determine the $n \times n$ per-unit-length transmission line capacitance and inductance matrices, **C** and **L**, consisting of n equal-width lands with identical separations on a dielectric substrate with a ground plane on the opposite side. This code is useful in simulating a printed circuit board that has one or more inner planes that are buried at various distances within the board. Essentially it simulates the surface of the PCB where the traces are located and the innerplane closest to that surface. Since we assume a board of infinite width the remaining innerplanes are of no consequence since they are isolated in this calculation from the surface lands by the first innerplane. This code also simulates many other microstrip structures that are used in microwave circuits.

The construction of these codes requires two simple modifications of the previous codes **PCB.FOR** and **PCBGAL.FOR**. The first modification is that the generalized capacitance matrix computed in the codes **PCB.FOR** and **PCBGAL.FOR** is the transmission-line capacitance matrix for the microstrip structures. This is because the voltages of the n lands of the microstrip structure are taken *with respect to the ground plane.* It can be shown that these voltages are essentially the absolute potentials computed in the generalized capacitance matrix [B.1]. Therefore the first modification of the codes **PCB.FOR** and **PCBGAL.FOR** is to simply remove the computations that determine the transmission-line capacitance matrices for the generalized capacitance matrices.

The second modification of the codes **PCB.FOR** and **PCBGAL.FOR** is to determine the absolute potential of an infinitesimal line charge on the surface of a dielectric sheet having a ground plane on the opposite side. This basic subproblem was solved for the PCB having no ground plane in Chapter 3 by imaging across the two dielectric surfaces as shown in Figure 3.36(b) resulting in

$$\phi(d) = -\frac{\alpha q}{4\pi\varepsilon_r\varepsilon_o}\ln[d^2] - \frac{\alpha^2 q}{4\pi\varepsilon_r\varepsilon_o}\sum_{n=1}^{\infty} k^{(2n-1)}\ln[d^2 + (2nt)^2] \quad (3.121)$$

In the case of the microstrip structure we image the line charge across the top dielectric surface and across the ground plane and obtain in a similar fashion

$$\phi(d) = -\frac{\alpha q}{4\pi\varepsilon_r\varepsilon_o}\ln[d^2] - \frac{\alpha^2 q}{4\pi\varepsilon_r\varepsilon_o}\sum_{n=1}^{\infty} (-1)^n k^{(n-1)}\ln[d^2 + (2nt)^2] \quad (A.8)$$

Comparing (3.121) and (A.8) we see that the codes **PCB.FOR** and

PCBGAL.FOR can be simply modified to analyze the microstrip structures by replacing in all summations

$$\underbrace{\sum_{n=1}^{\infty} k^{(2n-1)}}_{\text{PCB.FOR, PCBGAL.FOR}} \quad \Leftrightarrow \quad \underbrace{\sum_{n=1}^{\infty} (-1)^n k^{(n-1)}}_{\text{MSTRP.FOR, MSTRPGAL.FOR}} \tag{A.9}$$

There is one additional difference. In the case of **PCB.FOR** and **PCBGAL.FOR** with the dielectric removed (replaced with free space) $k = 0$ and the summation is zero. In the case of microstrip structures, **MSTRP.FOR** and **MSTRPGAL.FOR**, the first term of the summation is nonzero for $k = 0$, i.e., with the dielectric removed. This term represents the image across the ground plane and is present even with the dielectric removed.

A.2 FREQUENCY-DOMAIN ANALYSIS

One code, **MTL.FOR**, serves to determine the frequency-domain response of a MTL. According to the input data, one can consider lossless or lossy lines as well as homogeneous media or inhomogeneous media. The lossy nature of the conductors is contained in **MTL.IN** and the inhomogeneity of the surrounding medium is determined by the per-unit-length parameters contained in **PUL.DAT**.

A.2.1 General: MTL.FOR

Required Input Files: MTL.IN, PUL.DAT, FREQ.IN
Output File: MTL.OUT

The program forms the per-unit-length impedance and admittance matrices:

$$\hat{\mathbf{Z}} = \mathbf{R}(f) + j\omega\mathbf{L} \tag{A.10a}$$

$$\hat{\mathbf{Y}} = j\omega\mathbf{C} \tag{A.10b}$$

where the entries in **R**, **L**, and **C** are given in (2.13), (2.12c), and (2.24). The entries in the upper diagonal of **L** and **C** are read from **PUL.DAT**. The entries in **R** are input data from **MTL.IN**. These are given for each conductor as the dc resistance, r_{dc}, and the break frequency, f_o, where the resistance transitions to a $\sqrt{f}$ skin-effect frequency dependence as $r_{hf} = r_{dc}\sqrt{f/f_o}$. The per-unit-length internal inductances of the conductors are also included in $\omega\mathbf{L} = \omega\mathbf{L}_i + \omega\mathbf{L}_e$ and are determined from these data according to the scheme described in Section 3.6.2.4 assuming the high-frequency resistance and internal

inductive reactance are equal, $r_{hf} = \omega l_{i,hf}$, and they both transition at the same frequency, f_o. If no conductor loss is desired, set the dc resistance to zero, and if no skin-effect dependence is desired set f_o larger than the largest analysis frequency. The analysis frequencies are read from the file **FREQ.IN**. The code determines the eigenvalues, γ, and eigenvectors, T, of **YZ** as in (4.56). The entries in the generalized Thévenin equivalent characterizations of the terminations as in (4.84) are read from the file **MTL.IN**. Incorporating these into the general solution in (4.86) gives the $2n \times 2n$ matrix to solve, (4.88), for the $2n$ undetermined constants in the vectors $\mathbf{I}_m^{\pm}$.

The eigenvectors and eigenvalues of **YZ** are determined using the International Mathematical and Statistical Libraries (IMSL) version 9.2 subroutine EIGCC. Equation (4.88) is solved using the IMSL subroutine LEQTIC. Both subroutines were taken with permission from [2].

A.3 TIME-DOMAIN ANALYSIS

Four time-domain analysis codes are included. These implement the time-domain to frequency-domain transformation (**TIMEFREQ.FOR**), Branin's method (for a homogeneous medium) extended to MTL's (**BRANIN.FOR**), the finite difference–time domain method for lossless lines (**FINDIF.FOR**), and the finite difference–time domain method for lossy lines (**FDTDLOSS.FOR**).

A.3.1 Time-Domain to Frequency-Domain Transformation: TIMEFREQ.FOR

Required Input Files: TIMEFREQ.IN, MTLFREQ.DAT
Output File: TIMEFREQ.OUT

This code determines the time-domain response of the MTL using the time-domain to frequency-domain transformation described in Section 5.2.3. The input signal is assumed to be a periodic pulse train with trapezoidal pulses. The Fourier series of this input signal is given by (5.99) and the coefficients are given by (5.100). The input file **TIMEFREQ.IN** contains the number of harmonics used, the pulse level, the repetition frequency of the pulse train, the duty cycle, the pulse rise/fall time, and the final solution time. In addition, the dc level of the transfer function, $H(0)$, is input and multiplied by the dc level of the input spectrum, c_0, to give the dc level of the output spectrum. The input file **MTLFREQ.DAT** contains the frequency-domain transfer function (magnitude and phase) at each of the desired harmonics of the input signal. These can be computed with the frequency-domain analysis code **MTL.FOR** that was described in the previous section. The program combines these data to compute the time-domain response according to (5.102).

A.3.2 Branin's Method Extended to Multiconductor Lines: BRANIN.FOR

Required Input Files: BRANIN.IN, VSVL.IN, PUL.DAT
Output File: BRANIN.OUT

This code implements the extension of Branin's method extended to MTL's as described in Section 5.2.2. Equations (5.91) are solved recursively. The source and load voltage waveforms in $\mathbf{V}_S(t)$ and $\mathbf{V}_L(t)$ in the generalized Thévenin equivalent representation of the terminations in (5.90) are described in a piecewise-linear fashion in the input file **VSVL.IN**. The *resistive* entries in the matrices $\mathbf{R}_S$ and $\mathbf{R}_L$ in (5.90) are input data in the file **BRANIN.IN**. These are assumed to be symmetric and the upper triangle of these are input. The code uses the subroutine **GAUSSJ**, described earlier, to invert **L**. It is implicit in this method that the surrounding medium is homogeneous (for which **L** and **C** were computed).

A.3.3 Finite Difference–Time Domain Method: FINDIF.FOR

Required Input Files: FINDIF.IN, VSVL.IN, PUL.DAT
Output File: FINDIF.OUT

This code implements the finite difference–time domain method described in Section 5.2.5. Equations (5.140) are solved recursively. The discretizations of position and time are contained in **FINDIF.IN**. The source and load input voltages are described in a piecewise-linear manner in the input file **VSVL.IN**. Subroutine **GAUSSJ** is used to invert **L** and **C**.

A.3.4 Finite Difference–Time Domain Method: FDTDLOSS.FOR

Required Input Files: FDTDLOSS.IN, VSVL.IN, PUL.DAT
Output File: FDTDLOSS.OUT

This code implements the finite difference–time domain method for *lossy lines* described in Section 5.3.1.3. It is essentially identical to **FINDIF.FOR** with the addition of the discrete convolution for the dc and skin-effect losses (resistance and internal inductance) described in Section 5.3.1.3. Equations (5.140) are solved recursively with the exception that (5.157) replaces (5.140d). The discretizations of position and time are contained in **FDTDLOSS.IN** along with the dc resistance and skin-effect transition frequencies of the conductors according to the scheme described in Section 3.6.2.4. The source and load input voltages are described in a piecewise-linear manner in the input file **VSVL.IN**. Subroutine **GAUSSJ** is used to invert **L** and **C**.

A.4 SPICE/PSPICE SUBCIRCUIT GENERATION PROGRAMS

This section describes three FORTRAN codes that generate SPICE/PSPICE subcircuit models that implement the SPICE method (**SPICEMTL.FOR**), the lumped-pi model (**SPICELPI.FOR**), and the low-frequency, inductive-capacitive model (**SPICELC.FOR**).

A.4.1 General Solution, Lossless Lines: SPICEMTL.FOR

Required Input Files: SPICEMTL.IN, PUL.DAT
Output File: SPICEMTL.OUT

This code implements the exact SPICE model described in Section 5.2.1.3. The line is assumed to be lossless but the surrounding medium may be inhomogeneous as determined by the per-unit-length parameters in **PUL.DAT**. The per-unit-length inductance and capacitance matrices are read from **PUL.DAT** and diagonalized as in Section 5.2.1.2 using the subroutines **DIAG** and **JACOBI**. A subcircuit model is developed with the node numbering as 101, 102, ..., 10n for the nodes of conductors 1, 2, ..., n at $z = 0$ and 201, 202, ..., 20n for the nodes of conductors 1, 2, ..., n at $z = \mathscr{L}$ as illustrated in Fig. A.7. This subcircuit model can then be incorporated into a SPICE/PSPICE program. The total number of conductors, $n + 1$, and the total line length are specified in the input file **SPICEMTL.IN**. Note that the resulting SPICE/PSPICE code can also handle frequency-domain solutions by replacing the **.TRAN** run instruction with the **.AC** instruction [A.2].

A.4.2 Lumped-pi Circuit, Lossless Lines: SPICELPI.FOR

Required Input Files: SPICELPI.IN, PUL.DAT
Output File: SPICELPI.OUT

This code implements a SPICE subcircuit for the lumped-pi model of a MTL described in Sections 4.5 and 5.2.4. The line is again assumed to be lossless but the surrounding medium may be inhomogeneous as determined by the per-unit-length parameters in **PUL.DAT**. The per-unit-length inductance and capacitance matrices are read from **PUL.DAT**. A subcircuit model is again developed with the node numbering as before as 101, 102, ..., 10n for the nodes of conductors 1, 2, ..., n at $z = 0$ and 201, 202, ..., 20n for the nodes of conductors 1, 2, ..., n at $z = \mathscr{L}$ as is illustrated in Fig. A.7. This subcircuit model can then be incorporated into a SPICE/PSPICE program. The total number of conductors, $n + 1$, and the total line length are specified in the input file **SPICELPI.IN**. Conductor losses can be incorporated by adding additional nodes and the appropriate line resistances and internal inductances to the resulting subcircuit model.

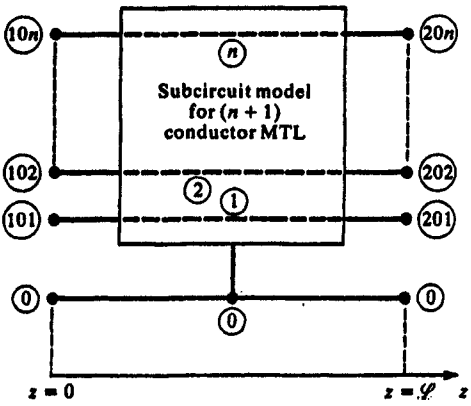


FIGURE A.7 Terminal node numbering scheme for the SPICE subcircuit models generated by the FORTRAN programs **SPICEMTL.FOR** and **SPICELPI.FOR** models.

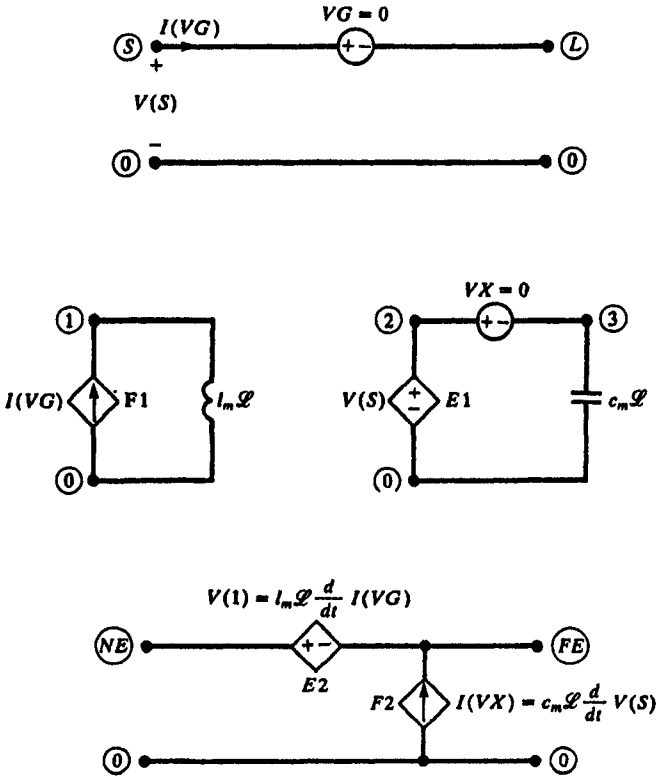


FIGURE A.8 The SPICE subcircuit model generated by the code **SPICELC.FOR**.

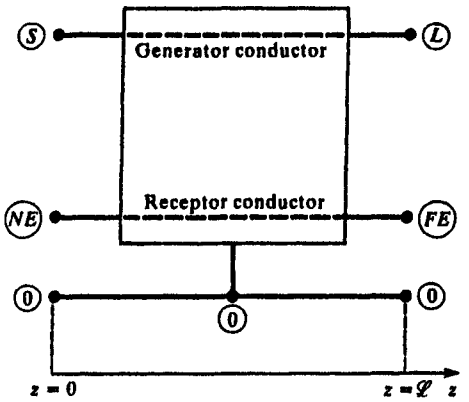


FIGURE A.9 The node numbering scheme for the SPICE subcircuit model generated by the code **SPICELC.FOR.**

A.4.3 Inductive-Capacitive Coupling Model: SPICELC.FOR

Required Input Files: SPICELC.IN
Output File: SPICELC.OUT

This code implements the low-frequency, inductive-capacitive coupling model described in Section 6.2.3 for a *three-conductor* line. The line is assumed to be lossless but the surrounding medium may be inhomogeneous as determined by the per-unit-length mutual inductance and mutual capacitance parameters in SPICELC.IN. This input file also specifies the total line length. Figure A.8 shows the SPICE model that is developed. A subcircuit model, shown in Fig. A.9, is developed with the node numbering as S, NE for the two nodes of conductors 1, 2, $z = 0$ and L, FE for the nodes of conductors 1, 2 at $z = \mathscr{L}$. This subcircuit model can then be incorporated into a SPICE/PSPICE program.

A.5 INCIDENT FIELD EXCITATION

A.5.1 Frequency-Domain Program: INCIDENT.FOR

Required Input Files: INCIDENT.IN, PUL.DAT, FREQ.IN
Output File: INCIDENT.OUT

This code determines the frequency-domain response of a general MTL to a single-frequency, uniform plane wave. Two types of line structures are provided for: an $(n + 1)$-conductor MTL and n conductors above an infinite, perfectly conducting ground plane. The terminations are characterized by generalized

Thévenin equivalents. Lumped voltage sources can also be included in these terminations. Equations (7.43) are solved for the vectors of undetermined constants in the general solution, and the terminal voltages are obtained from equations (7.44). The forcing functions due to a uniform plane-wave incident field, $\hat{\mathbf{V}}_{FT}(\mathscr{L})$ and $\hat{\mathbf{I}}_{FT}(\mathscr{L})$, are determined from equations (7.69) to (7.76). Essentially, the program **MTL.FOR** described earlier was modified to include the incident-field sources.

The code requires the input files **INCIDENT.IN**, **PUL.DAT**, and **FREQ.IN**. The file **PUL.DAT** is as before and gives the usual per-unit-length parameters of the line. The file **FREQ.IN** is also the same as for the program **MTL.FOR** and gives the solution frequencies sequentially. The file **INCIDENT.IN** describes (in this order):

1. The total number of conductors $(n + 1)$.
2. The total line length.
3. The line type (1 = no ground plane, 2 = ground plane).
4. The output conductor for the terminal solution voltages.
5. The incident uniform plane-wave description ($\hat{E}_o$, θ_E, θ_P, ϕ_P with reference to Fig. A.10).
6. The conductor cross-sectional coordinates (sequentially according to Fig. A.11).
7. The dc resistances and skin-effect break frequencies (for the reference conductor and sequentially for the other n conductors) as described for **MTL.FOR**.
8. The generalized Thévenin equivalent characterizations for the terminations as described for **MTL.FOR**.

The output file, **INCIDENT.OUT**, gives the termination solution voltages for the chosen output conductor (with respect to the reference conductor) sequentially for each solution frequency in **FREQ.IN**. These can be used in the previously described code, **TIMEFREQ.FOR**, to compute the time-domain response to a uniform plane wave that has a periodic, trapezoidal waveform.

A.5.2 SPICE/PSPICE Subcircuit Model: SPICEINC.FOR

Required Input Files: SPICEINC.IN, PUL.DAT
Output File: SPICEINC.OUT

This code generates a SPICE/PSPICE subcircuit model for an $(n + 1)$-conductor MTL excited by a uniform plane wave. The input file, **SPICEINC.IN**, contains the total number of conductors, the line length, the line type (1 = no ground plane, 2 = ground plane), the incident wave polarization and propaga-

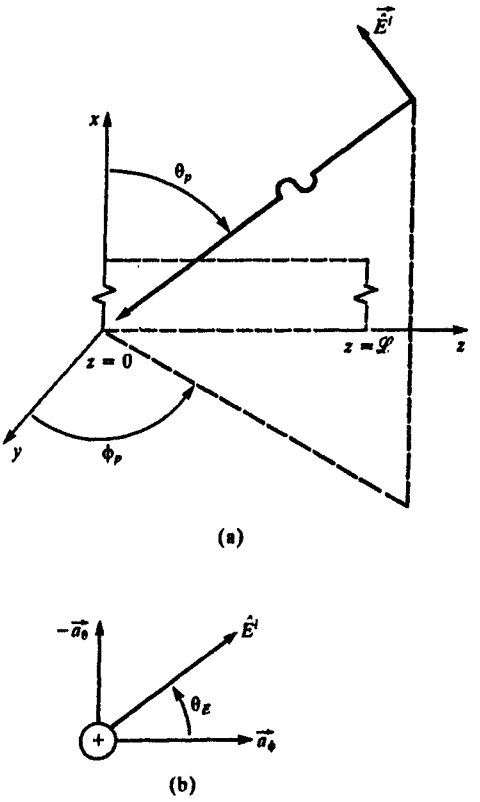


FIGURE A.10 Definition of (a) the angles of incidence and (b) polarization of the electric field for an incident uniform plane wave for the codes **INCIDENT.FOR**, **SPICEINC.FOR**, and **FDTDINC.FOR**.

tion direction with reference to Fig. A.10 (θ_E, θ_p, ϕ_p), and the cross-sectional conductor locations with reference to Fig. A.11 ($X(i)$, $Y(i)$). The cross-sectional coordinates are entered sequentially as the last items in this input file. The other input file is the usual **PUL.DAT** which contains the per-unit-length parameters. The output file, **SPICEINC.OUT**, contains the SPICE subcircuit model. The external nodes are labeled as in **SPICEMTL** and are shown in Fig. A.12. The code implements the model shown in Fig. 7.23. The external nodes at $z = 0$ are denoted as 101, 102, ..., $10n$ and the external nodes at $z = \mathscr{L}$ are denoted as 201, 202, ..., $20n$. One additional node, 100, is external to which the time waveform source of the incident wave, $\mathscr{E}_0(t)$, is attached. The code is restricted to ϕ_p positive, i.e., components of the propagation vector in the $+z$ direction. For propagation directions giving a component in the $-z$ direction, simply reverse the line.

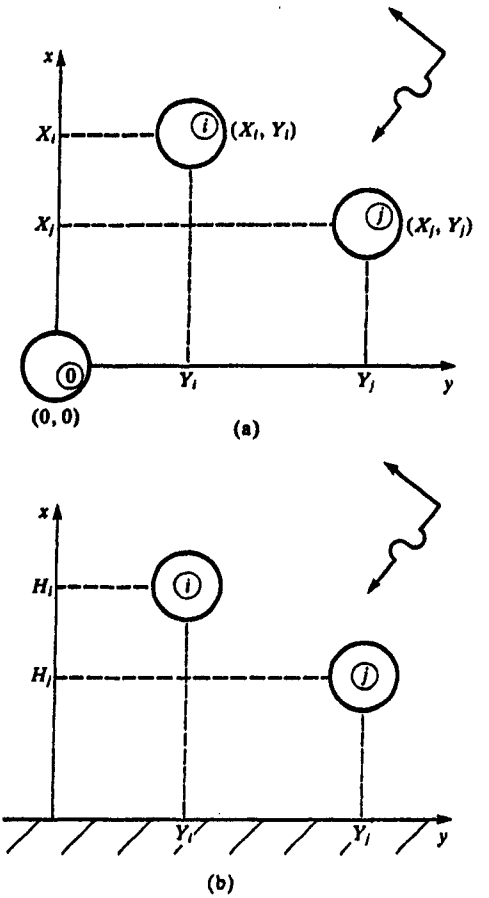


FIGURE A.11 Definition of the cross-sectional dimensions for input data to the codes **INCIDENT.FOR, SPICEINC.FOR,** and **FDTDINC.FOR** for (a) $(n + 1)$ wires and (b) n wires above a ground plane.

A.5.3 Finite Difference–Time Domain (FDTD) Model: FDTDINC.FOR

Required Input Files: FDTDINC.IN, PUL.DAT, E0.IN
Output File: FDTDINC.OUT

This code implements the FDTD method contained in equations (7.252). The input file **FDTDINC.IN** contains the total number of conductors, the line length, the line type (1 = no ground plane, 2 = ground plane), the incident wave polarization and propagation direction with reference to Fig. A.10 (θ_E, θ_p, ϕ_p), the cross-sectional conductor locations with reference to Fig. A.11 ($X(i)$, $Y(i)$), the number of line position cells, NDZ, the number of time discretizations,

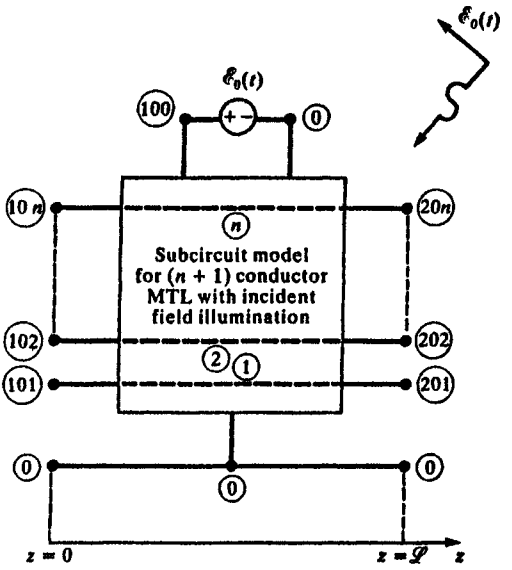


FIGURE A.12 Node numbering for the SPICE subcircuit model generated by the code SPICEINC.FOR.

NDT, the final solution time, the print solution index, the output conductor for the terminal voltages, and the line terminations in the form of a generalized Thévenin equivalent without sources ($\mathbf{R}_S$, $\mathbf{R}_L$, $\mathbf{V}_S = \mathbf{0}$, $\mathbf{V}_L = \mathbf{0}$). The input file **PUL.DAT** contains the usual per-unit-length parameters. The input file, **E0.IN**, contains a piecewise-linear specification of $\mathscr{E}_0(t)$. The output file, **FDTDINC.OUT**, contains the solution at the discrete time steps at the endpoints of the chosen output conductor.

REFERENCES

[1] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, NY, 1989.

[2] Visual Numerics, Inc., 9990 Richmond Ave., Suite 400, Houston, TX 77042-4548.