PAIRGAIN WIRELINE LOOP MODEL GENERATOR
USER'S MANUAL

PAIRGAIN WIRELINE LOOP MODEL GENERATOR USER'S MANUAL

TABLE OF CONTENTS

PAIRGAIN WIRELINE MODEL GENERATOR USER'S MANUAL

1.0 Introduction

The PairGain wireline loop model generator is a set of programs for the IBM PC (and compatibles) that produces drop-in PSPICE models of TELCO copper plant wireline loops. Generated wireline loop models are accurate over the frequency range of DC to 3.2 MHz. and may include typical loop impairments such as bridge taps and wire gauge changes. Each PSPICE model consists of a subcircuit netfile which describes a two-port Y-parameter model built from controlled sources. The controlled sources within each model are defined by frequency response tables. As a result, usage of the PSPICE models is restricted to SPICE programs that support a frequency response table extension for controlled sources (i.e. PSPICE or HSPICE). Computer memory requirements for each table-driven controlled source have been reduced to a minimum by employing an algorithm that eliminates redundant look-up table entries. This feature is important because the reduced memory requirement allows the models to work with the evaluation version (NOT the student version) of the PSPICE program. An additional utility program is provided for converting frequency-domain data (derived from PSPICE simulations) into time-domain impulse response data.

The next section discusses the procedures and algorithms by which the wireline models are constructed. The third section provides a detailed description of the usage and command syntax for each of the model construction programs. The last section contains some example models from the standard Bellcore CSA loops.

2.0 Technical Description

This section discusses the procedures and algorithms by which the wireline models are constructed. The first part describes the basic process of creating wireline models with the available program set. The second part contains a detailed discussion of the model generating algorithms used within each of the programs. The first part should be read by all users, but the second parts is optional and may be skipped over.

2.1 Basic Operation

The basic operation of the wireline model generator is as follows. Binary data files are first built to represent the electrical characteristics of individual loop sections (and bridge taps). This process is performed with the programs TMATXL.EXE and MAKBTAP.EXE. Each binary data file contains the T matrix (i.e. chain matrix or ABCD matrix) two-port circuit parameters for a given loop section numerically evaluated at a predetermined set of discrete frequencies. Wireline cable electrical characteristics (transmission line primary constants *L, R, C, G*) necessary to produce these T matrix parameters are found in the file MODEL.DAT. This file contains polynomial expressions of transmission line primary constants for various cable types. These polynomial expressions are derived by curve-fitting available ANSI primary constant data of typical wireline cables (Appendix G of ANSI specification T1.601-1988). The loop section binary data files are combined using the program TCHAIN.EXE to produce a complete loop model binary data file (T matrix parameters) which represents the series cascade of the individual loop sections. Finally, this loop model binary data file is processed by another program, DAT2SPC.EXE, to produce a Y-parameter two-port PSPICE model file. Note that since all transmission line segments are reciprocal circuit elements, the final model will also be reciprocal and thus can be used bidirectionally. Detailed descriptions of the individual construction programs and the necessary adjunct data files are given in section 3.

The model construction programs are designed to run in batch mode in the PC-DOS environment. A particular loop model is constructed in a single batch file using these programs as a "pseudolanguage" to define the individual segments of a given loop model. Although this method seems crude, it allows the construction of very sophisticated loop models (including models with compound bridge taps) with minimal effort (assuming one is familiar with batch files). Sample batch files describing typical Bellcore test loops are described in section 4.

Once a PSPICE model file has been constructed, it may be included in any PSPICE netfile by means of .INC or .LIB command. The complete loop model is built in the form of a PSPICE subcircuit. The PSPICE format was chosen because PSPICE (or HSPICE) is an efficient (and widely available) tool for general purpose simulation of analog circuits. It is often necessary to include the effects of analog loop interface circuitry (i.e transformers, filters, hybrids, etc) when evaluating complete wireline loop channels. A PSPICE loop model provides the convenience of using PSPICE for evaluating the interactions of the analog line interface components with the wireline loop. An additional utility program, FREQ2IMP.EXE, is provided with the model generator programs for converting frequency domain output data (from PSPICE) into time-domain impulse response sample data (see section 3 for details). This program is useful for creating sampled-data channel and echo models for use in subsequent high-level system simulation programs.

2.2 Technical Background

The following paragraphs provide a technical description of the algorithms used by the model generation programs. This section does not describe the impulse response conversion utility FREQ2IMP.EXE. A description of this program can be found in section 3. Since this detailed knowledge is not necessary to use the software package, this section may be skipped over. Nevertheless understanding the contents of this section will allow one to use the model generator more effectively.

The model generator programs use T matrix two-port circuit parameters to represent the electrical characteristics of loop sections. Conversion to the final PSPICE format occurs only at the last processing stage. The basic descriptions and properties of T matrices are shown in Fig. 1. This particular modeling technique is used for two main reasons: T matrices allow simple mathematical formulation for cascaded networks (i.e cable sections) and the T matrix coefficients for generalized transmission line segments are relatively simple mathematical expressions. It will be shown that our application requirements make this modeling technique more efficient than traditional lumped circuit methods.

The twisted pair cables in wireline loop channels (or any general transmission line for that matter) may be described by a series cascade of lumped circuit component cells (see Fig. 2). Each cell of lumped components (L, R, C, and G) approximates the electrical characteristics of a finite length of transmission line. The values of these lumped components are derived from the known primary constant data (lumped circuit characteristic parameters) of the simulated transmission line. The primary constants of a transmission line are defined as quantities (Ohms, Henries, etc.) per unit length and in general are ***not*** constant with respect to frequency. The primary constant values for various wireline cable types may be found in Appendix G of ANSI specification T1.601-1988. The physical cable length represented by a lumped circuit cell is equal to the actual cell component values divided by the corresponding primary constant values. When lumped circuit cells are used to model transmission line segments, the simulated length of each circuit cell must be limited so that the total phase shift introduced by each circuit cell ***at the highest frequency of interest*** is less than twenty degrees. If this guideline is not followed, the group delay characteristics of the simulated line section will not match the true group delay response of the real line section ***for the signal under consideration.*** In practical terms, this means that the line segment length simulated by each lumped circuit cell should be less than one-tenth the electrical wavelength (on the cable) of the maximum frequency of interest. For typical wireline cables at frequencies above 100 KHz., the electrical wavelength λ at frequency f can be estimated by the following formula:

$$\lambda \approx \frac{1}{f\sqrt{LC}}$$ where L and C are the respective primary constants evaluated at f.

Long cable segments and wide bandwidth requirements necessitate large numbers of lumped circuit cells for an accurate model representation. Furthermore, inspection of the ANSI primary constant tables reveals that the primary constants for all commonly used cable types show considerable variation in value over large frequency bandwidths. The combination of a large total component count and frequency-dependent component values precludes efficient simulation of twisted-pair cable sections using traditional lumped circuit transmission line modeling methods. In contrast, T matrix representation of such networks is simpler to implement.

The T matrix coefficient formulas for a uniform general transmission line segment are shown in Fig. 2. These expressions are based upon the general transmission line parameters *Zo* (characteristic impedance), γ (propagation constant), and the length of the segment. The expressions for *Zo* and γ, in turn, are derived from the primary constant values contained in the ANSI data tables. Since the primary constant values are frequency dependent, the values of *Zo* and γ are also frequency dependent. Note that the two-port described by any transmission line segment is reciprocal (AD-BC=1) and consequently all constructed loop models will be reciprocal. This property allows the final models to be used bidirectionally in a circuit netfile. The expression for γ is defined on a per unit length basis. The line section generation program (TMATXL.EXE) requires section length on a per foot basis, but primary constant values derived from the ANSI tables are defined on a per mile basis. The primary constant values derived from the ANSI tables must be scaled (divided by 5280) before they can be incorporated into the formula for γ.

In order to calculate the T matrix coefficients at any arbitrary frequency, it is first necessary to calculate the primary constant values at that frequency. The frequency dependence of each primary constant is modelled by curve-fitting a fourth order interpolating polynomial (in frequency) for each primary constant parameter to the available ANSI table data. The actual mechanics of curve-fitting the ANSI table data were performed by the Lotus 1-2-3 spreadsheet program (see reference 1). Although the ANSI data tables provide cable parameter data to the frequency of 5 MHz., the accuracy of the resulting polynomials is degraded at frequencies beyond 3.2 MHz. The ANSI data tables contain relatively fewer data entries at the higher frequencies than at the lower frequencies. When the curve fitting algorithm is applied, the total influence of the low frequency error terms is greater than that of the high frequency error terms simply because there are more low frequency error terms. The resulting polynomials therefore become very accurate at the lower frequencies and less accurate at the higher frequencies. This problem can be eliminated by replacing the interpolating polynomials with the more accurate cubic spline interpolation algorithm. A future software version will contain this enhancement. The interpolating polynomial algorithm, however, is slightly faster. Polynomial sets have been precalculated for each of the following cable models: 26 Gauge PIC at 70 F., 24 Gauge PIC at 70 F., and 22 Gauge PIC at 70 F. This information is required by TMATXL.EXE and is found in the ASCII file MODEL.DAT. The format for MODEL.DAT is described in section 3. The above formulas will calculate the T matrix coefficients for a transmission line segment at a single specified frequency. Complete characterization, however, requires information over a wide range of frequencies.

The model generation programs perform all loop model characterization in the frequency domain. T matrix coefficients for each line section are numerically calculated at a predetermined set of five hundred discrete frequency points. This same set of points is used for all line sections and models. The set of frequency points is defined by the following formula:

$$\text{freq}(n) = 10^{1.049904\ln(n)}; \qquad n=\{1, 2, 3,..., 500\}$$

This formula generates an adaptive logarithmic frequency sweep from 1 Hz. to 3.3 MHz, where the number of points per log cycle continuously increases as the frequency increases. This method combines the wide coverage of a log scale sweep with acceptable detailed resolution at higher frequencies. The same high frequency resolution using a straight log scale sweep would require an excessive number of low frequency points (<10 KHz.). These low frequency points would be redundant because typical wireline loop sections (and complete loop models) do not exhibit radical frequency response changes (high Q peaking) at lower frequencies. Wireline loop response peaking from bridge taps, however, can occur at higher frequencies (>100 KHz.) hence the necessity of extra resolution in that range. The resulting set of T matrices is stored as a binary data file with all T matrix coefficients stored as double precision floating point complex numbers. All line sections, partial loop models, and complete loop models are stored as binary data files until the final processing stage.

From the T matrix data files of separate cable sections, we can construct a T matrix data file for any given loop model including typical impairments such as gauge changes and bridge taps. Loop models are constructed by a series cascade of separate cable sections. In general, we start at one end of the loop termination and add cable sections until the loop model is complete. The cascading of one cable section onto an existing cable section is performed by multiplying the corresponding T matrices of each section (in the proper order) with TCHAIN.EXE. The resulting product matrix becomes the T matrix representation of the two cascaded sections. The construction of a bridge tap section is slightly more involved. First we use TMATXL.EXE to build the actual bridge tap line section. We observe that a bridge tap is simply a section of unterminated cable connected at some point in the loop model. At the point of connection, a bridge tap appears to be a separate impedance load connected in parallel with the remainder of the loop model. Using the T matrix formulas shown in Fig. 1, we calculate the apparent impedance (using the open-loop input impedance formula) of the bridge tap section at that point. This parallel impedance value is converted to an admittance and inserted into the general T matrix representation for a parallel admittance (also shown in Fig. 1). This conversion process is performed by the program MAKBTAP.EXE. The resulting T matrix may be treated as a series cable segment section and is subsequently cascaded onto the existing loop model construct using TCHAIN.EXE. An example of the model construction procedure for a simple loop with a single bridge tap is shown in Fig. 3. Note that the input data required and output data produced by each construction phase is a complete T matrix binary data file. Eventually, the entire loop model is built into a T matrix binary data file.

After the complete loop model has been built into a T matrix binary data file, the program DAT2SPC.EXE converts the binary data file into a PSPICE subcircuit netfile. This netfile describes a Y-parameter two-port using four voltage controlled current sources (VCCS) defined by frequency response tables. One useful property of the final two-port model is its reciprocity. The final PSPICE model may be used bilaterally facilitating the modeling of reverse channels. The conversion algorithm is as follows. First standard formulas are used to convert each T matrix data item into its corresponding two-port Y matrix representation. Afterwards, the corresponding Y matrix coefficients from each data item are processed as four separate groups. Each of these groups will become a frequency response table that defines one of the four VCCS needed to describe the loop model. First each group data item is converted into magnitude/phase format where the magnitude is defined in decibels and the phase is defined in degrees. The phase value entries are further processed to eliminate cycle discontinuities. A cycle discontinuity occurs when the phase jumps from -180 degrees to +179 degrees. At each discontinuity an additional 360 degrees is subtracted from all subsequent original phase values. The above phase discontinuity would now appear as -180-(360n) degrees to 180-(360(n+1)) degrees where n=0,1,2.... This process is necessary for PSPICE to perform transient analysis properly with VCCS frequency response tables. The group table is now a VCCS frequency response table with five hundred entries. PSPICE uses linear interpolation on the frequency response table to calculate a transconductance value at any arbitrary frequency. Note that not all of the table entries are necessary to achieve a reasonable degree of accuracy. The final processing step consists of compressing each frequency response table by eliminating unnecessary table entries.

The table compression algorithm, which is also performed by DAT2SPC.EXE, is important for two main reasons. First it reduces the disk space size occupied by loop model files and the resulting PSPICE output files. More importantly, however, the internal memory space occupied by each lookup table is reduced allowing reasonably sized networks to be simulated on the evaluation software version of PSPICE. This particular version of PSPICE (for the PC) is available for less than one hundred dollars. The compression algorithm is as follows. A linear interpolation estimate of the magnitude and phase is calculated for every data point between two segment anchor points (initially two points apart). If the difference between the actual and estimated values is less than a specified error threshold for both magnitude (0.1 dB.) and phase (0.5 degree) for every data point inside the segment, the segment size is increased by one data point (by moving the upper anchor point forward one item). The above process is repeated until the error threshold is exceeded by some point within the current segment. When an error occurs, the current segment size is reduced by one data point (the upper anchor point is moved backwards one item) and both resulting anchor points are flagged for printing. The data points contained between the anchor points are not printed since they can be estimated accurately from the existing anchor points. The process is now repeated with the current upper anchor point becoming the new lower anchor point and the new upper anchor point set two points ahead of the lower anchor point. The algorithm is finished when the last data point is processed. The net result of this algorithm is a dramatic reduction (by a factor of 5 to 15) in the size of the PSPICE model without sacrificing accuracy (only unnecessary table items are eliminated). The resulting compressed tables are formatted in a PSPICE subcircuit netfile.

The T matrix parameter approach for generating wireline loop models is much simpler to implement than lumped circuit models when long section lengths and frequency-dependent behavior are present. It should also be mentioned that the final table-driven PSPICE model runs considerably faster in PSPICE than lumped circuit models. The price of this technique, however, is increased memory requirements to store the T matrix binary data files. Another weakness is that transmission line segments may not contain any nonlinear voltage/current relationships. In spite of this, some nonlinearities can be simulated outside the loop model by including nonlinear elements within the complete channel PSPICE netlist.

3.0 Program Reference

This section is a description of each of the programs in the PairGain wireline loop model generator software package. It provides instructions and examples of the correct usage of each of the various programs. In addition, complete example loop construction batch files are provided for further reference.

3.1 Summary of Files

The PairGain wireline loop model generator package consists of the following files:

TMATXL.EXE -- executable; generates a T matrix binary data file for a single uniform line section

MAKBTAP.EXE -- executable; converts a line section T matrix binary data file into a bridge tap element

TCHAIN.EXE -- executable; calculates the T matrix binary data file for a series cascade of two input line section T matrix binary data files

DAT2SPC.EXE -- executable; converts a T matrix binary data file into a PSPICE subcircuit netfile

FREQ2IMP.EXE -- executable; converts frequency response data into sampled impulse response data

MODEL.DAT -- ASCII text; contains interpolating polynomial coefficients for the primary constants of the following cable types: 26 Ga. PIC @70 F., 24 Ga. PIC @70 F., and 22 Ga. PIC @70 F.

DOLOOP?.BAT -- a series of sample batch files for generating the eight Bellcore CSA test loops

SAMPLE?.CIR -- a series of sample PSPICE netfiles incorporating wireline models

CSALOOP.LIB -- pre-built PSPICE model library of the eight Bellcore CSA test loops

Installation of the simulator package onto a PC hard disk consists of copying all of the distribution disk to a selected subdirectory. The model generator package, however, may also be operated directly from the distribution diskette or a suitable backup copy.

3.2 Description of Executable Files

This subsection provides a detailed description of the usage and proper syntax of the five executable files (i.e programs) that are provided with the simulator package.

3.2.1 Cable Section Generator -- TMATXL

Purpose: This program generates T matrix data files for cable sections. Each cable section is specified by model type (i.e. 26 Gauge PIC, 24 Gauge PIC, etc) and length (in feet). The program is designed to run from a model construction batch file.

Command Syntax:

`tmatxl /m=`*<cable model>* `/l=`*<length>* `/o=`*<output file>*

General Information:

Cable model information is contained in the file MODEL.DAT. This file must exist and be located in the current directory path. If MODEL.DAT does not exist or cannot be located, TMATXL.EXE will terminate without producing any output. Information on the following cable models is contained in a "default" version of MODEL.DAT that comes with the program set:

m26pic70 -- 26 Gauge PIC at 70 degrees F.
m24pic70 -- 24 Gauge PIC at 70 degrees F.
m22pic70 -- 22 Gauge PIC at 70 degrees F.

The name string at the left of each item is the actual cable model argument used in the command line. The default model used when no model is specified (assuming the "default" version of MODEL.DAT exists) is 26 Gauge PIC at 70 degrees F.

The cable section length is specified in feet. If no cable length is specified, the program will break without generating any output.

The output file defines the name of the generated T matrix data file. The default name if none is specified is TMATXL.DAT.

All the command line parameters and their associated arguments are case insensitive. Furthermore, the command line parameters may occur in any sequence; they do not have to appear in the sequence shown above.

Examples:

```
tmatxl /m=m26pic70 /l=9000 /o=testloop.dat
```

The above command will generate a T matrix data file for a 9000 foot section of 26 Gauge PIC cable at 70 degrees F. The T matrix output file is TESTLOOP.DAT.

```
tmatxl /m=m24pic70 /l=12000 /o=loop8.dat
```

This example will generate a T matrix data file for a 12000 foot cable section of 24 Gauge PIC cable at 70 degrees F. The T matrix output file is named LOOP8.DAT.

Model Data Format:

Cable model data in file MODEL.DAT is written in the following format:

<start of file>
.MODEL *<model #1 name>* *<start of model definition>*
.PROMPT "*<model #1 prompt string>*"
.RCOF *<Nr=number of polynomial coefficients for primary constant R>*
<R0>
<R1>
:
<RNr-1>
.LCOF *<Nl=number of polynomial coefficients for primary constant L>*
<L0>
<L1>
:
<LNl-1>
.GCOF *<Ng=number of polynomial coefficients for primary constant G>*
<G0>
<G1>
:
<GNg-1>
.CCOF *<Nc=number of polynomial coefficients for primary constant C>*
<C0>
<C1>
:
<CNc-1>
.ENDM *<end of model definition>*

MODEL *<model #2 name>*
.PROMPT "*<model #2 prompt string>*"
<repeat above sequence for coefficients>
:
.ENDM
:
:
:
.MODEL *<model #K name>*
.PROMPT "*<model #K prompt string>*"
:
.ENDM
<end of file>

The model name refers to the string identifier used by the /m switch. The model prompt string refers to a short description (i.e. "26 Gauge PIC @ 70F.") that is used to annotate the model name when TMATXL is ran interactively. The other data define the polynomial coefficients used to describe primary constant parameters. See file MODEL.DAT for an example.

3.2.2 Convert Cable Section to Bridge Tap -- MAKBTAP

Purpose: This program converts the T matrix data file of a cable section to a T matrix data file of a bridge tap section. The cable type and length of the bridge tap are defined by the cable section that is converted. This program is designed to run from a model construction batch file.

Command Syntax:

`makbtap` *<infile> <outfile>*

infile -- This is the full name for the T matrix data file of the input cable section. If no input file name is specified (i.e. no command line arguments), the program queries the user interactively for input and output file names. The first command line argument is assigned to infile.

outfile -- This is the full name of the converted bridge tap section output file. If no output file name is specified, the output file name automatically becomes MAKBTAP.DAT. The second command line argument is assigned to outfile.

Example:

```
tmatxl /m=m24pic70 /l=400 /o=temp.dat
makbtap temp.dat tap1.dat
```

The above example generates the T matrix data file of a bridge tap which consists of 400 feet of 24 Gauge PIC cable at 70 degrees F. The bridge tap T matrix data file is TAP1.DAT.

3.2.3 Cascading Cable Sections -- TCHAIN

Purpose: This program calculates the T matrix data file for the series cascade of two line sections defined by input T matrix data files. This program is designed to run from a model construction batch file.

Command Syntax:

`tchain` *<file1> <file2> <outfile>*

file1 -- This is the full name for the T matrix data file of the first input cable section. The first command line argument is assigned to *file1*.

file2 -- This is the full name for the T matrix data file of the second input cable section. The second command line argument is assigned to *file2*.

outfile -- This is the full name for the T matrix data file generated by the series cascade of the two input networks (i.e cable sections). The third command line argument is assigned to *outfile*.

General Information:

The order of T matrix multiplication is identical to the order of appearance in the command line parameters. The port 2 side of the T matrix defined in *file1* is connected to the port 1 side of the T matrix defined in *file2*. The port 1 side of *file1* becomes the port 1 side for the T matrix in *outfile*, and the port 2 side of file2 becomes the port 2 side for the T matrix in *outfile*.

If less than two command line arguments are specified (i.e. no arguments or *file1* only), the program will interactively query the user for input and output file names. If exactly two command line arguments are specified (*file1* and *file2*) then the output file name defaults to TCHAIN.DAT.

3.2.4 Producing Final PSPICE Model Format -- DAT2SPC

Purpose: This program converts a loop model T matrix binary data file into a two-port Y-parameter model defined by a PSPICE subcircuit netfile. Execution of this program is the final stage of the model construction process. This program is designed to run from a loop construction batch file.

Command Syntax:

`dat2spc` *<infile> <outfile> <modname>*

infile -- This is the full name for the input T matrix data file of the complete line model. If no file name is specified (i.e. no command line arguments), the program queries the user interactively for input file, output file, and subcircuit names. The first command line argument is assigned to *infile*.

outfile -- This is the full name of the output PSPICE subcircuit netfile. If no file name is specified, the output file name automatically becomes TMATCOF.MOD. The second command line argument is assigned to *outfile*.

modname -- This is the subcircuit name that will appear in the output PSPICE subcircuit netfile. If not specified, the subcircuit name automatically becomes WIRELINE_LOOP. The third command line argument is assigned to *modname*.

General Information:

If no command line arguments are specified the program will query the user for input file, output file, and subcircuit names. If one or more command line arguments are present (i.e. *infile*) then the remaining unspecified arguments are assigned their default values.

The final PSPICE models are described as subcircuits with the following format:

```
<start of file>
.SUBCKT   <modname>   1   2   3   4
                      |   |   |   |
                      |   |   |   initial side (-) node
                      |   |   initial side (+) node
                      |   final side (-) node
                      final side (+) node
GY11   1   2   FREQ { V(1,2) } =
+ (<freq1>    <dbmag1>    <phase1>)
     :           :           :
+ (<freqN1>   <dbmagN1>   <phaseN1>)
GY21   3   4   FREQ { V(1,2) } =
+ (<freq1>    <dbmag1>    <phase1>)
     :           :           :
```

```
+ (<freqN2>  <dbmagN2>  <phaseN2>)
GY12  1  2  FREQ { V(3,4) } =
+ (<freq1>   <dbmag1>   <phase1>)
    :          :          :
+ (<freqN3>  <dbmagN3>  <phaseN3>)
GY22  3  4  FREQ { V(3,4) } =
+ (<freq1>   <dbmag1>   <phase1>)
    :          :          :
+ (<freqN4>  <dbmagN4>  <phaseN4>)
.ENDS
<end of file>
```

"Initial side" refers to the termination where the loop model construction process began and "final side" refers to the termination where the last cable section was added. The models can be used bidirectionally and with differential or single-sided analog interface configurations.

Examples:

```
dat2spc testloop.dat testloop.mod testloop
```

This example creates a PSPICE subcircuit named "testloop" in output file TESTLOOP.MOD from the T matrix binary data file TESTLOOP.DAT.

```
dat2spc testloop.dat
```

This example creates a PSPICE subcircuit named "WIRELINE_LOOP" (default loop model name) in output file TMATCOF.MOD (default output file name) from the T matrix binary data file TESTLOOP.DAT.

3.2.5 Impulse Response Conversion Utility -- FREQ2IMP

Purpose: This is a utility program for converting frequency-domain sampled data into equivalent time-domain sampled data. The sample period for the time-domain output data is user-selectable. This program may be used interactively or in batch mode.

Command Syntax:

`freq2imp` [*switches*] <*infile*> <*outfile*>

Switches:

/p Create default settings in file FREQ2IMP.PRM.
/b Run in batch mode with settings found in FREQ2IMP.PRM.

infile -- This is the full name for the input frequency domain data file. The contents of this file must follow the formatting rules defined below. If no file name is specified, the program queries the user for information. The first non-switch command line argument is assigned to *infile*.

outfile -- This is the full name of the output time-domain impulse response data file. If outfile is not specified, the output file name automatically becomes IMPRSP.DAT. The second non-switch command line argument is assigned to *outfile*.

General Information:

When the /p switch is detected, the program queries the user for new default settings and updates the file FREQ2IMP.PRM with the new settings. No other action is taken, and no other output is produced. All other switches and file name arguments are ignored.

When the /b switch is detected, the program automatically uses the default settings found in FREQ2IMP.PRM. The default input file name is FREQSWP.DAT, and the default output file name is IMPRSP.DAT. These file names are used when their corresponding command line arguments are not specified.

When no command line switches are present, all impulse response settings are entered interactively by the user. The default settings are contained in FREQ2IMP.PRM.

The number of output points selected must be a power of 2 and no greater than 4096. If the number is not a power of 2 then the number will be rounded to the nearest power of 2 greater than the original value. The time sample interval selected must be greater than $1/2f_{max}$ where f_{max} is the maximum frequency defined in the available input data.

Data File Formats:

The frequency-domain input data is not required to have any specific sample interval (i.e. linear or log scale in frequency) but is required to be in the following ASCII text format:

```
<start of file>
DATA  <N>  <2*K>
<freqval(1)>  <Re(1,1)> <Im(1,1)> ... <Re(1,K)> <Im(1,K)>
<freqval(2)>  <Re(2,1)> <Im(2,1)> ... <Re(2,K)> <Im(2,K)>
    .           .          .           .          .
    .           .          .           .          .
    .           .          .           .          .
<freqval(N)>  <Re(N,1)> <Im(N,1)> ... <Re(N,K)> <Im(N,K)>
<end of file>
```

where: N = number of data points (1 <= N <= 4096)
K = number of data sets (1 <= K <= 4)
Re() --> real part of complex value
Im() --> imaginary part of complex value
freqval(n+1) > freqval(n) for all n=1,2,...,N.

Frequency domain data must be in complex rectangular form (not dB. magnitude and phase degrees).

An example PSPICE command to produce output data for node X would be as follows:

```
.PRINT <analysis type> VR(X)   VI(X)
```

See section 4 for a complete PSPICE netfile example.

The the time-domain output data is contained in an ASCII text file with the following format:

```
<start of file>
#IMPULSE RESPONSE DATA
DATA  <N>  <K>
<timeval(1)>  <impval(1,1)> ... <impval(1,K)>
<timeval(1)>  <impval(2,1)> ... <impval(2,K)>
```

```
    .              .          .          .
    .              .          .          .
    .              .          .          .
<timeval(N)>  <impval(N,1)> ... <impval(N,K)>
<end of file>
```

Description of Algorithm:

The algorithm implemented by FREQ2IMP is fairly straightforward. The user selects the desired sample period T for the output data and the required number of sample points N (must be a power of 2). The available frequency-domain data (formatted PSPICE output data) is read into an internal data lookup table. From this lookup table a new internal binary data table is constructed. This new table contains estimates of the input frequency-domain data evaluated at the following set of frequencies {0F, 1F, 2F,...,(N/2)F, ((N/2)+1)F,...,(N-1)F} where F=1/NT. At frequency points less than or equal to (N/2)f, the new frequency-domain data is derived by performing cubic spline interpolation using the input frequency-domain lookup table. At frequency points greater than (N/2)F the following formula is applied:

Data[((N/2)+k)F] = (Data[((N/2)-k)F])* where * denotes complex conjugate

This "mirrored" complex conjugate padding formula fills in the missing "negative" frequencies and guarantees that the resulting time-domain sequence will be completely real. Afterwards, the new interpolated frequency-domain data table is windowed (using an Exact Blackman window) to eliminate any edge discontinuities, and an inverse FFT is performed on each separate complex-valued data set. The imaginary portion of each time-domain data set is discarded (the imaginary part is guaranteed to be zero as a result of the "conjugate padding" scheme employed in a previous step), and the final data table is formatted and written to the selected output file.

Althought the cubic spline interpolation alogorithm provides very good estimates of frequency-domain data at the necessary points, its accuracy ultimately depends on the "density" of the available lookup table. If the set of available input frequency-domain data does not contain enough points, there will be slight errors in the interpolation estimates. These errors will produce ghost signals in the final time-domain data. The effect is most noticable if insufficient data points are available in the frequency band of 50 KHz to 500 KHz. It is recommended that input frequency-domain data to FREQ2IMP contain at least 40 points per decade resolution (linear or logarithmic sweep) at frequencies above 10 KHz. to minimize this effect.

Finally, it should be mentioned that this implementation of the cubic spline interpolation does not extrapolate for data points outside the range of the available lookup table. The returned data values at points outside the domain of the lookup table will equal to the data value at the nearest domain end point. Hence if available frequency-domain data has a lower limit of 100 Hz., the returned data value for all frequencies below 100 Hz. will be the table entry for 100 Hz. It is especially important to be aware of this fact when generating (with PSPICE) frequency-domain data for conversion into high sample rate time-domain data. The highest data entry (in frequency) of the lookup table must be greater than or equal to 1/2T or the program will respond with a warning message. No warning message, howver, is returned for low frequency limit violations.

4.0 Examples

This section presents some examples of model construction batch files and the usage of final models in PSPICE netfiles. The sample files shown here are also found in the files DOLOOP?.BAT (? = any digit from 1 to 8) and SAMPLE?.CIR.

4.1 Sample Model Generation Batch Files

Loop model construction is performed using batch files containing the programs described in the previous section. The distribution diskette contains a set of sample batch files for the construction of the eight Bellcore CSA test loops (DOLOOP?.BAT) and the actual pre-built loop models (CSALOOP.LIB). The actual physical configurations of these loops are shown in Fig. 4. The batch files for two simple examples are repeated below for convenience. Note that these models are built from the RD termination towards the CO termination.

```
  rem  Batch file to construct Bellcore CSA test loop #6
  rem  Create 9000 foot section of 26 Ga. PIC
tmatxl /m=m26pic70 /l=9000 /o=loop6.dat
dat2spc loop6.dat loop6.mod LOOP6
```

```
  rem  Batch file to construct Bellcore CSA test loop #1
  rem  Start from RD term -- Create first cable section
tmatxl /m=m26pic70 /l=1800 /o=temp1.dat
  rem  Create cable section for bridge tap
tmatxl /m=m26pic70 /l=600 /o=temp2.dat
  rem  Convert cable section to bridge tap section
makbtap temp2.dat tap.dat
  rem  Cascade bridge tap section to previous cable section
tchain tap.dat temp1.dat temp1.dat
  rem  Create final cable section
tmatxl /m=m26pic70 /l=5900 /o=temp2.dat
  rem  Cascade final cable section to existing construct
tchain temp2.dat temp1.dat loop1.dat
dat2spc loop1.dat loop1.mod LOOP1
  rem  Complete model construction -- Erase temporary files
erase tap.dat
erase temp1.dat
erase temp2.dat
```

4.2 Sample PSPICE Netfiles

We now present some usage examples of the wireline PSPICE models within actual netfiles. These examples also include necessary PSPICE commands for obtaining the proper data to use as input for FREQ2IMP. For the following examples assume that the model for Bellcore CSA loop #1 has been built using the batch file defined in the previous section. Note that all included CSA loop model batch files are built from the RD termination towards the CO termination. These models will have the following terminal orientation:

```
X!  A  B  C  D  LOOP?
    |  |  |  |
    |  |  |  RD (-) node
    |  |  RD (+) node
    |  CO (-) node
    CO (+) node
```

The first example calculates the voltage gain (CO to RD) of Bellcore CSA loop #1 doubly terminated by 135 Ohm resistors:

```
*** EXAMPLE TEST FILE -- VOLTAGE GAIN LOOP 1 (CO TO RD) ***
.INC "loop1.mod"
V1  1  0  AC  1.0
R1  1  2  135                ; this is the CO termination
X1  2  0  3  0  LOOP1
R2  3  0  135                ; this is the RD termination
.AC  DEC 10  10 1MEG
.PRINT AC VDB(3)             ; V(R2)/V1
.END
```

Note that since the loop model is reciprocal we can generate the reverse direction voltage gain by inserting the model backwards as follows:

```
*** EXAMPLE TEST FILE -- VOLTAGE GAIN LOOP 1 (RD TO CO) ***
.INC "loop1.mod"
V1  1  0  AC  1.0
R1  1  2  135                ; this is now the RD termination
X1  3  0  2  0  LOOP1        ; note order of nodes
R2  3  0  135                ; this is now the CO termination
.AC  DEC 10  10 1MEG
.PRINT AC VDB(3)
.END
```

Using FREQ2IMP to generate time-domain data requires frequency-domain data expressed as rectangular complex values ($x + jy$). The following netfile will caclulate the raw data required to generate the CO to RD voltage gain impulse response:

```
*** EXAMPLE TEST FILE -- VOLTAGE GAIN LOOP 1 (CO TO RD) ***
*** FREQUENCY-DOMAIN DATA REQUIRED FOR IMPULSE RESPONSE ***
.INC "loop1.mod"
V1  1  0  AC  1.0
R1  1  2  135                ; this is the CO termination
X1  2  0  3  0  LOOP1
R2  3  0  135                ; this is the RD termination
.AC  DEC 40  10 1MEG         ; note the higher decade resolution
.PRINT AC VR(3) VI(3)   ; output in rect. complex format
.END
```

The PSPICE output for the above netfile is found in SAMPLE3.OUT. Since f_{max} is 1MHz., the minimum sample interval available with FREQ2IMP is 0.5 uSec. Note that it does not matter whether the frequency sweep format is logarithmic or linear. In addition, the data point density (decade resolution) has been increased to minimize interpolation error ghosts (see section 3.2.5). The output file SAMPLE3.OUT must now be edited and formatted properly before being processed by FREQ2IMP. The resulting formatted data file (derived from SAMPLE3.OUT) is found in file SAMPLE3.SWP. This formatting procedure can be performed manually with a text editor or automatically (using a simple C program or AWK script).

The final example illustrates transient analysis to generate the channel (CO to RD) pulse response and near-end echo (CO is local termination) pulse response for a 2.5 uSec. square pulse:

```
*** EXAMPLE TEST FILE -- PULSE REPONSE ***
*** CHANNEL AND NEAR-END (CO) ECHO ***
.INC "loop1.mod"
V1  1  0  PULSE(0 5.0 1U 0.01U 0.01U 2.5U 100U)
```

```
R1   1   2   135                    ; this is the CO termination
X1   2   0   3   0   LOOP1
R2   3   0   135                    ; this is the RD termination
.OPTIONS LIMPTS=401                 ; increase number of points limit
.TRAN 100U 0.25U
.PRINT TRAN V(2) V(3)               ; node 2 is echo, node 3 is channel
.END
```

5.0 References

[1] Orvis, W. J.. *1-2-3 For Scientists and Engineers.* San Francisco: Sybex, Inc., 1987, pp 170-172.

[2] Hayt Jr, W. H. *Engineering Electromagnetics.* New York: McGraw-Hill, Inc., 1974, pp 408-413.

[3] Temes, G. C., and J. W. LaPatra. *Introduction to Circuit Synthesis and Design.* New York: McGraw-Hill, Inc., 1977.

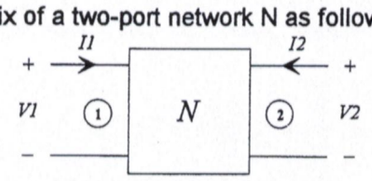
[4] Oppenheim, A. V., and R. W. Schafer. *Digital Signal Processing.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975.

[5] American National Standards Institute (ANSI). *Integrated Services Digital Network (ISDN) - Basic Access Interface for Use on Metallic Loops for Application on the Network Side of the NT (Layer 1 Specification) ANSI T1.601-1988.* New York: American National Standards Institute, Inc., 1988, pp 69-77.

[6] MicroSim Corporation. *The Design Center: Circuit Analysis User's Guide.* Irvine, California: MicroSim Corporation, 1992.

BASIC T MATRIX RELATIONS

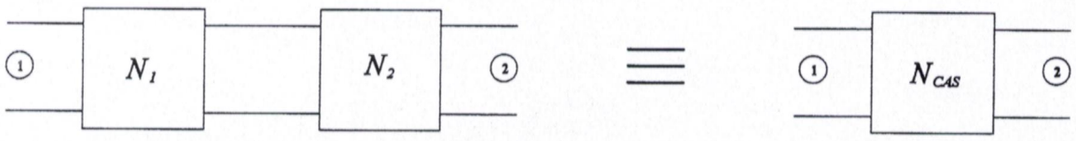
We define the T matrix of a two-port network N as follows:



$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix} \qquad \text{Where } T = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

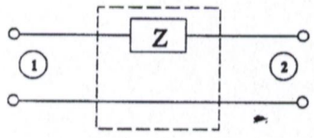
Basic Relations:

1) Given two series cascaded networks N_1 and N_2:



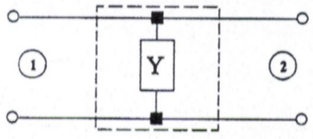
$$T_{NCAS} = T_{N1} \bullet T_{N2}$$ (Note the order of terms.)

2) T Matrix for series impedance Z:



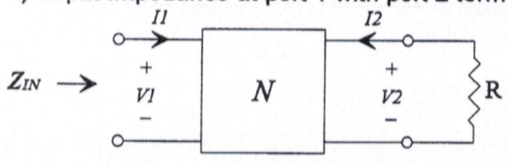
$$T = \begin{bmatrix} 1 & Z \\ 0 & 1 \end{bmatrix}$$

3) T Matrix for parallel admittance Y:



$$T = \begin{bmatrix} 1 & 0 \\ Y & 1 \end{bmatrix}$$

4) Input Impedance at port 1 with port 2 terminated:



$$Z_{IN} = \frac{V_1}{I_1} = \frac{AR + B}{CR + D}$$

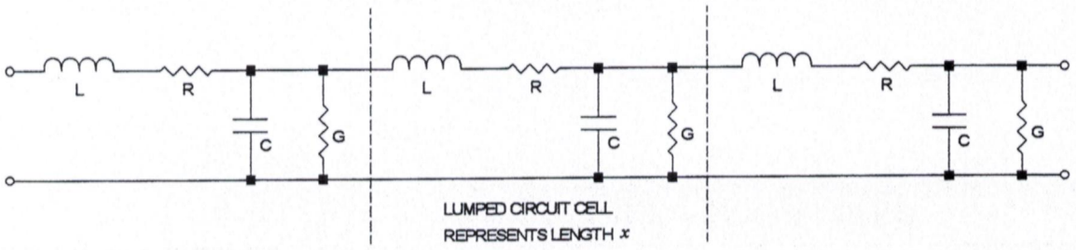
Note if port 2 is open (i.e. R=∞) then $Z_{IN} = \frac{A}{C}$

Figure 1

CIRCUIT MODELS FOR TRANSMISSION LINES

Lumped Circuit Models:

Any uniform transmission line can be represented by a series cascade of lumped circuit cells:



with component values **L**=Lx, **R**=Rx, **G**=Gx, and **C**=Cx. The values of L, R, C, G are known as the primary constants (defined per unit length) of the transmission line.

Limitations:

1) The primary constant values must be frequency independent.

2) The physical line length represented by each lumped circuit cell must be less than one-tenth of the electrical wavelenth of the maximum frequency of interest.

Distributed Parameter Modeling:

All uniform transmission line segments (of any length) can be completely characterized electrically by three parameters: Z_o (characteristic impedance), γ (propagation constant), and the physical length of the line segment. The value of Z_o is independent of length, but γ is defined per unit length.

Relations between Z_o, γ and the primary constants:

$$Z_o = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

Note that "per length" dependencies cancel out.

$$\gamma = \sqrt{(R + j\omega L)(G + j\omega C)}$$

Unit length is the same as that upon which the primary constants are defined.

Formulas for T Matrix Coefficients of Transmission Line Segments:

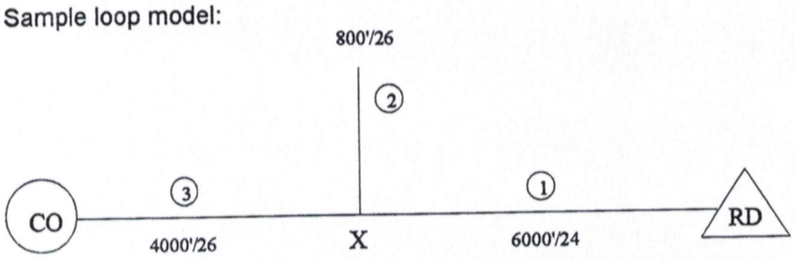
$$A = \cosh(\gamma l) \qquad B = Z_o \sinh(\gamma l) \qquad C = \frac{1}{Z_o}\sinh(\gamma l) \qquad D = \cosh(\gamma l)$$

where l=length of the transmission line segment.

Figure 2

EXAMPLE OF LOOP MODEL CONSTRUCTION

Sample loop model:



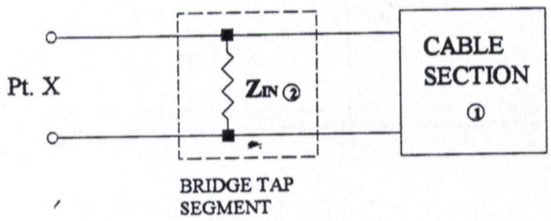
1) Start at RD termination. Generate a T matrix binary data file for cable segment 1.

2) Generate a T matrix binary data file for cable segment 2.

3) Convert cable segment 2 into a bridge tap segment as follows:

For segment 2 let $T = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$

Now, $Z_{IN2} = \frac{A}{C}$ at the tap junction termination (point X) when the far end is open. We see the following equivalent circuit at point X:



Hence, the conversion of cable section 2 into a bridge tap section is as follows:

$T = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Rightarrow T = \begin{bmatrix} 1 & 0 \\ \frac{C}{A} & 1 \end{bmatrix}$ T matrix representation of a parallel Y where $Y = \frac{1}{Z} = \frac{C}{A}$

4) Cascade the bridge tap section to cable section 1.

5) Generate a T matrix binary data file for cable section 3.

6) Cascade cable section 3 to the existing series cascade of bridge tap section and cable section 1.

7) Convert resulting T matrix binary data file into a PSPICE model file.

Figure 3

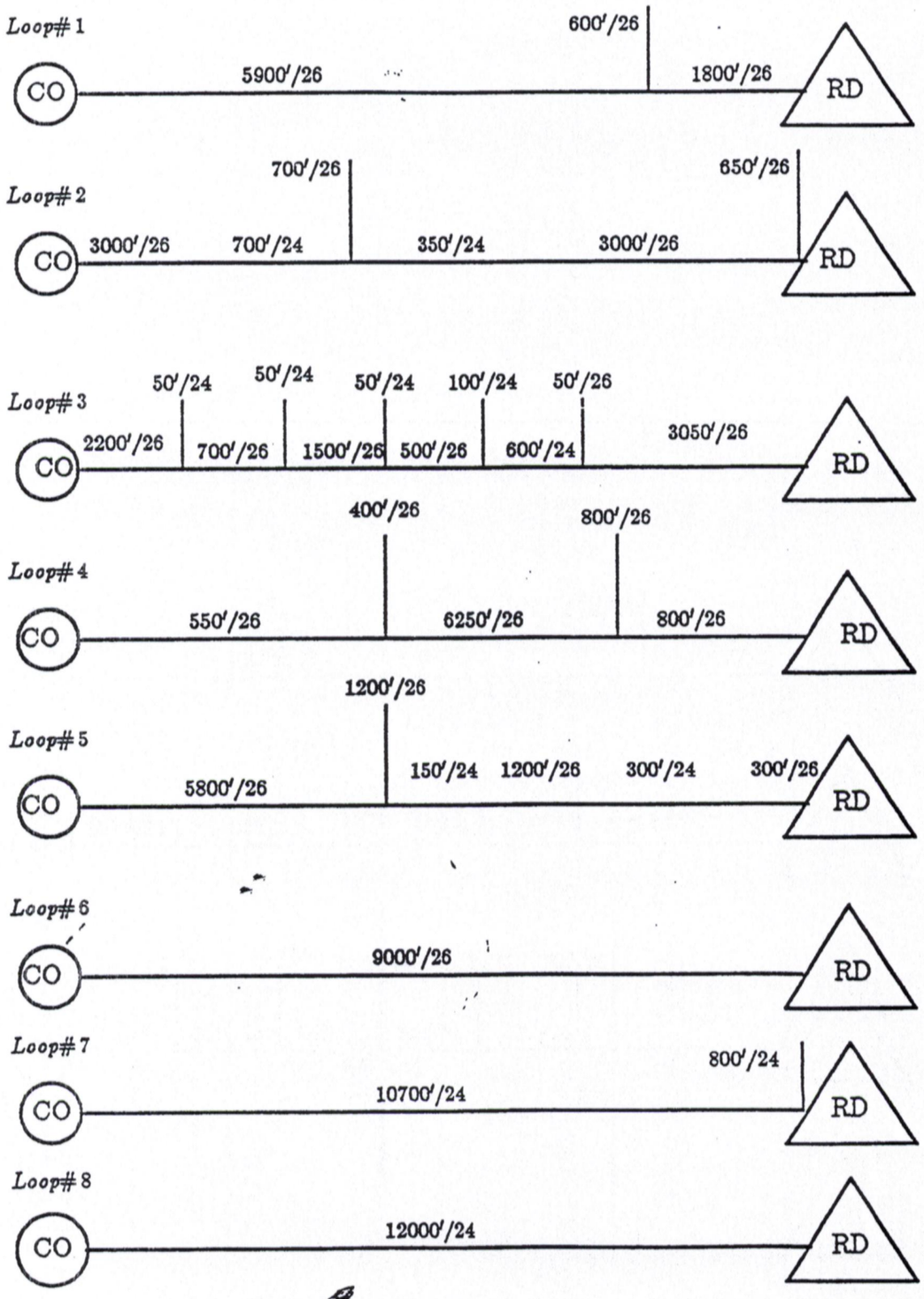


Figure 4. Make-up of Tentative CSA Test Loops