NGSPICE: Circuit Simulator

User guide for ECE 391

Last Updated: August 2015

Preface

This user guide contains several page references to the ngspice re-work manual version 26. The official ngspice manual can be found at `http://ngspice.sourceforge.net/` as well as older versions available for download.

Acknowledgments

Contents

1 Introduction

This ngspice user guide has been developed for the ECE 391 course at Oregon State University to assist students to further their understanding on the behavior of transmission lines. This guide covers the basic concepts to using ngspice to simulate ideal (lossless) and non-ideal (lossy) transmission lines in DC/AC circuits and other related topics discussed in the course. This user guide summarizes the useful, pertinent information from the near 600 page ngspice manual needed to run the ngspice simulator for this course, while adding several extra examples. For more in-depth details and other topics, see the full ngspice manual.

2 How to Install Ngspice

Ngspice is a free downloadable program that runs on Windows, Linux, and Mac computer systems. This section will cover how to install and run ngspice (re-work 26) for Windows, Mac, Linux systems, and remote access to ngspice.

2.1 Mac OS X

- First make sure you have XQuartz installed.
- Get it at: `http://xquartz.macosforge.org/landing/`
- Current release is: XQuartz 2.7.5
- Download XQuartz-2.7.5.dmg and install via drag and drop.

Three methods to install:

1. (Easiest way, binary package)
   - Go to: `http://sourceforge.net/projects/ngspice/files/ng-spice-rework/26/`
   - Download and install: ngspice-26.pkg
   - Install instructions are in: INSTALL_MAC_OS_X. It installs like any Mac package file.
2. (Less easier way, compile from source)
   - Install macports, see: `http://www.macports.org/install.php`
   - Using the Mac OS X Package (.pkg) Installer is recommended.
   - Once installed, in a terminal window, type: sudo port install ng-spice
   - This will install ngspice into the /opt directory.
   - After this, ngspice will be available at the command prompt.
   - Type ngspice, not ng-spice.

3. (Allows very latest version install)

   - Install XCode Tools if you haven't already.
   - Go to: `http://sourceforge.net/projects/ngspice/files/ng-spice-rework/26/`
   - Download ng-spice-rework-26.tar.gz to a working area.
   - In your working area type: tar zxvf ng-spice-rework-25.tar
   - Go into the newly created ngspice directory and type:
   - ./configure
   - ./make
   - sudo make install
   - when finished, ngspice will be installed in: /usr/local/bin/ngspice

2.2 Linux Distros

   - sudo vi /etc/apt/sources.list
     (remove all comments on path lines)
   - sudo apt-get update
   - sudo apt-get install build-essential linux-headers-‘uname -r‘
   - sudo apt-get install libtool automake autoconf
   - sudo apt-get install flex bison texinfo
   - sudo apt-get install libx11-dev libxaw7-dev
   - Go to: `http://sourceforge.net/projects/ngspice/files/ng-spice-rework/25/`
   - download ng-spice-rework-26.tar.gz to a working area.
   - gunzip ng-spice-rework-26.tar.gz
   - tar xvf ng-spice-rework-26.tar
   - Go into the newly created ngspice directory and type:
   - ./autogen.sh
   - ./configure
   - ./make
   - sudo make install

   when finished, ngspice will be installed in /usr/local/bin/ngspice

2.3 Windows

- Go to: `http://sourceforge.net/projects/ngspice/files/ng-spice-rework/26/`
- Download: ngspice-26_130104.zip
- Extract the zip file

2.4 Alternative Method - remotely (PuTTY)

Ngspice can be accessed through the College of Engineering's shell servers (i.e. flip & access) remotely without having to install ngspice locally onto your computer. To access the COE servers, a SSH program such as PuTTY is needed. For Windows users, another program called Xming is also needed to output the GUI display windows of ngspice.

To install PuTTY and Xming go to `http://engineering.oregonstate.edu/computing/personal/134` for instructions and links to download them.

3 How to Run Ngspice

Ngspice can be run on multiple platforms. PuTTY is used to connect to a Unix/Linux OSU server remotely. The command prompt will be similar to the Mac and Linux Distros. To run ngpsice on a Mac or Linux, simply open a command prompt window and enter the text "ngspice" (without quotes). Note, you may need to change your current working directory.

3.1 Using PuTTY

Once you have successfully downloaded and installed PuTTY as well as Xming if needed, open the server window to access the UNIX shell seen below in figure 1.

Figure 1

Login using your ONID username and COE password. Note: If the prompt displays the text "Terminal type? [Xterm]", leave it blank and press the enter key to continue to get to the command line. To run ngspice, simply type the text "ngspice" (without quotes) into the command prompt. The message seen in figure 2 will be displayed if ngspice was successfully opened.

ngspice.png

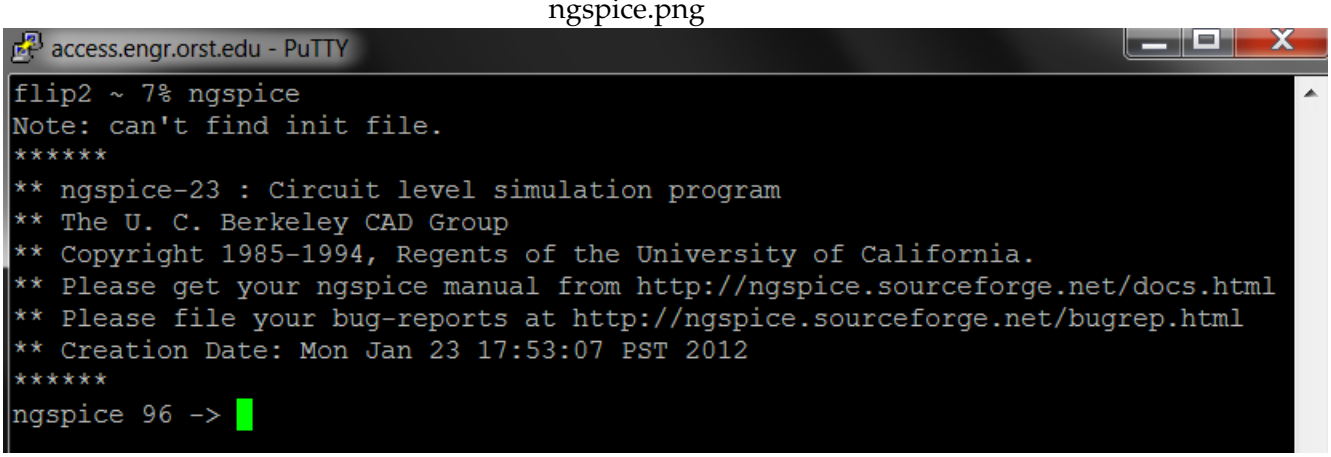


Figure 2

3.2 Using Windows

Ngspice is a MS Windows executable program, which also includes XSPICE code models, examples, and the quick user manual. Once the file has been extracted, the ngspice.exe file will be located in the spice/bin folder. This folder is also the default directory that ngspice will look for netlist files in. Figure 3 displays the ngspice command window when ngpisce.exe is opened and waiting for command inputs.

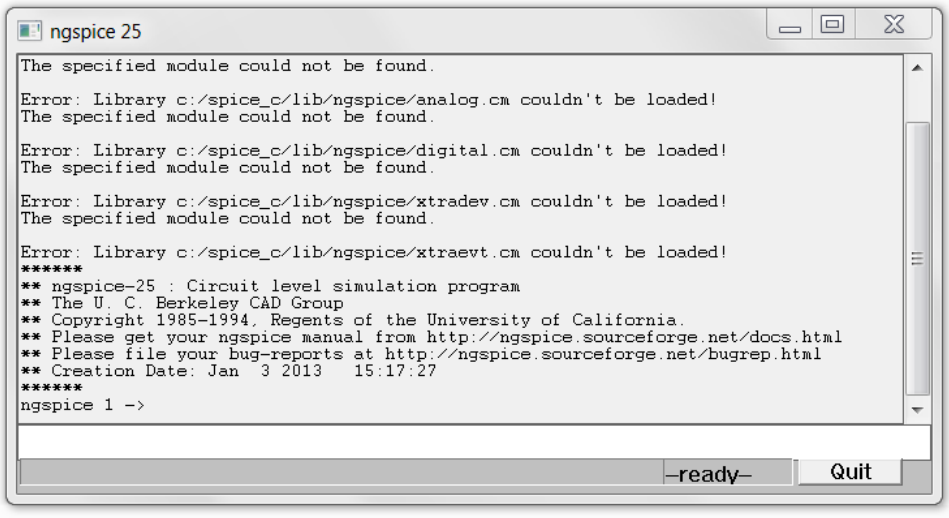


Figure 3

4 Ngspice Overview

Ngspice is a code-based mixed-level/mixed-signal circuit simulator. Unlike popular circuit simulators, such as LTSpice that are capable of drawing symbolic objects (schematic captures) to describe circuit topology, ngspice simulated circuits are described as text only inside a file called a netlist. A netlist is primarily composed of a list of defined circuit components and the nodes that connect them together. A netlist is written using a text-editor program such as notepad and notepad++ for Windows or VI and emacs for Linux to name a few. Avoid using word processing programs, such as Microsoft Word, as they embed hidden formatting objects within the text file that may disrupt the simulator from working correctly and may result in a run-time error during simulation.

4.1 Getting Started

Ngspice runs only from a command line from a pre-defined set of command instructions. Therefore, there is no GUI menus to interact with like there are with other spice simulators such as LTSpice. Before you can begin simulating a netlist, ngspice must know where the netlist files are located in order to perform circuit analyses and simulation on them. If your netlist files are located outside of the ngspice default folder, you must direct ngspice to point at that path directory.

This is accomplished by first opening up ngspice and simply typing in the ngspice command at the prompt:

cd <path to your files>

Examples:

- cd ngspice/demo_ngspice
- cd c:\spice\ngspice\demo_ngspice (windows .exe version)

Now that ngspice is pointing to the directory containing the netlist files, the next command is to indicate which netlist file to open. This is done by typing in the command:

source <filename.file extension>

Examples:

- source voltagedivider.sp
- source lowpassfilter.cir

One of the advantages of ngspice is the ability to edit a netlist file quickly within ngspice without having to exit the program, open a text editor, make changes and then reopen ngspice.

Editing a netlist file within ngspice is accomplished by simply typing the command: edit

After the changes are saved within the editor, ngspice will prompt if you want to re-run the circuit.

To exit ngspice, type the command: quit

These ngspice commands are the same for the Windows, Mac and the Linux versions. However, to use the edit command using the Windows version, you may need to modify the spinit file to indicate where and which editor to use.

To do this:

- Go to the spice\share\ngspice\scripts folder
- Open the spinit file using a text editor such as notepad
- Add in the line: set editor="start notepad"
- For the notepad++ text editor add the line: set editor="start notepad++"

Take note when using the *Windows version only*, plotting several functions within the same window plot are set at the default line color values and cannot be currently changed. The following steps summarize how to begin using ngspice once the program is opened and waiting for commands:

1. Type the current directory path to the folder that contains the netlist file(s) if they exist outside the default folder using the "cd <file path>" command.
2. Type "source <netlist file name>" command to run the simulator.
3. Type "edit" in the command line to make changes to the netlist file if desired.
4. Repeat step 2 to rerun the new source file.
5. Type "quit" to exit the program.

4.2 Creating a Netlist

To get started writing a netlist, begin by opening a text editor. There are two required lines to the netlist code structure. The first line in a netlist must be the title line and does not require any special syntax or format. The last line of the netlist must include the syntax ".end" only (without the quotes). The other lines (with no particular order) in between the first and last can be categorized as either being an element instance line, a control line, or a comment line.

The topology of the circuit is described by the element instance lines and their corresponding values. Control lines tell the simulator which model & run parameters to perform, such as transient analysis or plot.

The syntax to make an entire line a *comment* is by placing an asterisk star '*' at the beginning of a line.

General form:

* <this line is a comment>

Examples:

* A simply RC low pass filter circuit with R = 100 ohms, C = 10uF.

To add an end-of-a-line comment after a command, Ngspice allows the following characters: ';' , '//' or the '$' character. Take note, that when using the single $ character as an end-of-line comment it must be outside of a .control section.

General form:

<command> ; this is a comment

<command> // this is a comment

Example:

set color0 = white ; background color

set color1 = blue ; text and grid color

4.3 Creating Circuit Elements

To create circuit components, every component follows the general outline:

- Name of the type of component
- Name of its connecting nodes
- Values/parameters.

The following short list contains the most common circuit components used for this course and the general syntax format to use:

Resistor
R[name] [node 1] [node 2] [resistance value]

Capacitor
C[name] [node 1] [node 2] [capacitance value]

Inductor
L[name] [node 1] [node 2] [inductance value]

Voltage Source (*See pg. 79 - ngspice manual*)
V[name] [pos. node] [neg. node] DC [value/Tran.] AC [<ACMAG> <ACPHASE>]

Nodes and name values are not case sensitive and can be any arbitrary length of characters. Values can be represented by modifiers such as $10e-6$, 10u, or 10uF. Any letter following a scale factor (e.g. 10uF) is ignored. Note, when specifying a 10^6 value with an abbreviated suffix, use MEG notation since m or M equals a 10^-3 value. The table below lists the ngspice scale factors *(see page 47 in the manual for further details)*.

Suffix	Name	Factor
T	Tera	10^{12}
G	Giga	10^{9}
Meg	Mega	10^{6}
K	Kilo	10^{3}
mil	Mil	25.4×10^{-6}
m	milli	10^{-3}
u	micro	10^{-6}
n	nano	10^{-9}
p	pico	10^{-12}
f	femto	10^{-15}

Table 2.2: Ngspice scale factors

EXAMPLE: Voltage divider Circuit

Here is a schematic capture of a simple voltage divider circuit. The circuit contains a 12VDC input source connected to node "vin" and node "0". R1 is connected between node "vin" and node "vout" with R2 completing the circuit from the nodes "vout to 0".

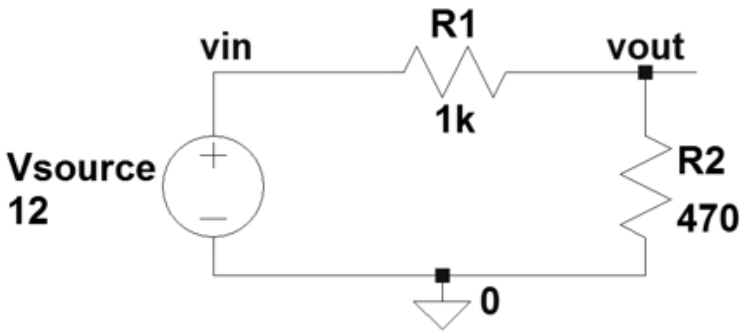


Figure 4

Figure 5 shows the ngspice equivalent circuit of figure 4 as a netlist file:

```
Voltage Divider Circuit - ECE 391 -

*12V DC Voltage source (Note: node #0 = ground)

Vsource vin 0 DC 12

*R1 = 1k, R2 = 470 ohms.

R1 vin vout 1k

R2 vout 0 470


.control

tran .5s 1s ;transient analysis (pg. 238 ngspice manual)

.endc

.end
```

Figure 5

Notice in figure 5's netlist, the code between ".control" & ".endc" are where the control lines go and in this case the only control parameter is "tran", which performs a simple DC transient analysis and outputs the results to the command prompt window seen below in figure 6.

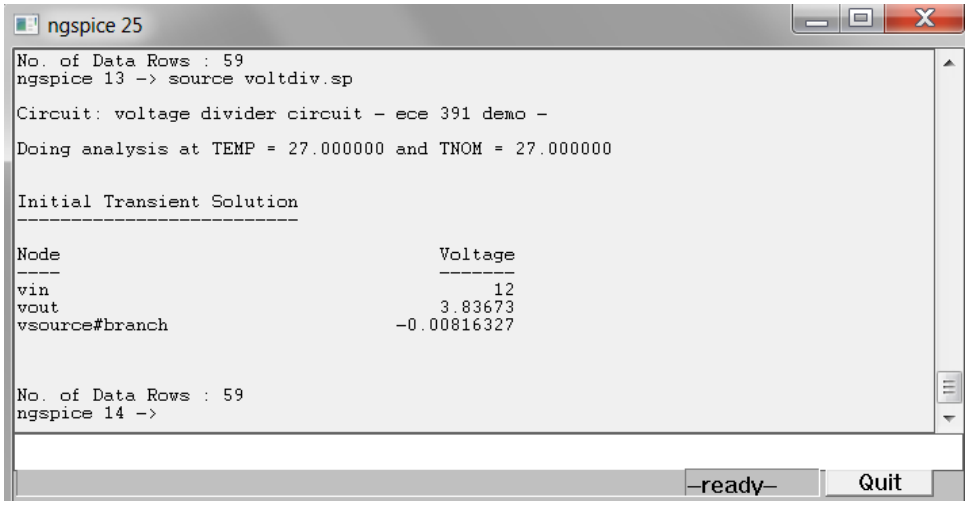


Figure 6

5 Transmissions Lines

The primary focus in this course is to understand the behavior of transmission line effects on DC and AC circuits. The characteristics of transmissions line behavior can be modeled and analyzed in two ways. The first model is to ignore any transient components on the t-line and consider it to be *lossless (ideal)*.

The second model includes the transient components on the t-line and is called a *lossy (non-ideal)* transmission line. Ngspice has the capability to model both types of transmissions lines. Take note, that *lossy* transmission line modeling is a bit more difficult than *lossless* models and requires more parameter values.

5.1 Transmission Line (lossless)

This section will cover how to write a netlist that will define, analyze, and plot several different transmission line configurations. These T-lines examples will have various load conditions such as:

- Shorted
- Open
- Inductive/Capacitive
- Un-matched load resistance

5.1.1 Voltage Sources

For DC based transmission lines, a pulsed square wave model is best used as the input voltage source. To create a square wave voltage source in a netlist, use the following format found on page 80 of the ngspice manual (Re-work 26):

4.1.1 Pulse

General form:

PULSE(V1 V2 TD TR TF PW PER)

Examples:

VIN 3 0 PULSE(−1 1 2NS 2NS 2NS 50NS 100NS)

Name	Parameter	Default Value	Units
V1	Initial value	-	V, A
V2	Pulsed value	-	V, A
TD	Delay time	0.0	sec
TR	Rise time	TSTEP	sec
TF	Fall time	TSTEP	sec
PW	Pulse width	TSTOP	sec
PER	Period	TSTOP	sec

Figure 7

The example in figure 7 creates a voltage source called, "VIN" connected between the nodes 3 to 0. The source is pulsed from -1V to 1V, with 2ns edges, 50ns pulse width and a 100ns period. It is recommended to place line comments above all the circuit components created in the netlist with a brief detail description. Here is an example netlist with a pulsed voltage source with a brief description commented above it:

```
*0 to 3.3V input source with 5ns delay, 5ns edges, 20ns pulse width, 40ns cycle time
Vin vin 0 3.3 PULSE(0 3.3 5e-9 5ns 5ns 20e-9 40e-9)
```

Figure 8

Note: *Maintain caution* when defining the rise and fail times as you will want to be sure that they at a lower value than the T-line's defined flight time (Td).

To create a unit step function voltage source, the pulse parameter only needs the voltage limits.

Example: Vin vin 0 3.3 pulse(0 3.3)

5.1.2 Creating a Transmission Line

To create a transmission line, ngspice follows this general form (*see manual pg.*101 *for more details*):

T[name] [node 1] [node 2] [node 3] [node 4] Z0=[value] Td=[value] F=[FREQ] NL =[NRMLEN]

Where,

Z0 = characteristic impedance

Td = flight time

F = frequency

NL = normalized electrical length

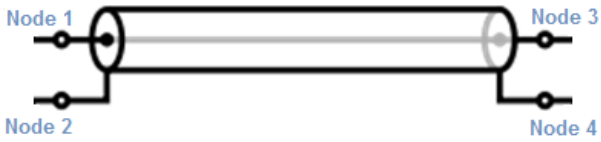
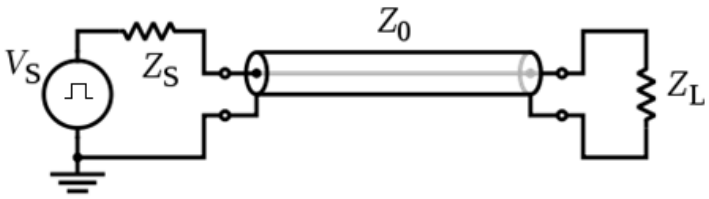


Figure 9

Figure 9 provides a visual aid that describes the connection nodes of a transmission line in ngspice before a circuit analysis can be performed.

EXAMPLE: Open-Circuit transmission line

In this example, a simple open-ended transmission line circuit is constructed in a netlist file. To simulate an open-circuit, a resistive load ZL will be used and set to 10 MΩ. Figure 10 shows the general T-line circuit topology and its corresponding netlist elements with defined values for Vs, Zs, Zo and ZL.




```
*0-1V Input source with 5ns delay, 1ns edges, 20ns pulse width, 40ns period
Vs vsrc 0 1 pulse(0 1 5e-9 1ns 1ns 20e-9 40e-9)

*50 ohm Source impedance (Zs)
Rsrc vsrc node1 50

*Transmission line with Zo = 50, td=3ns
T1 node1 0 node2 0 z0=50 td=3ns

*T-line Termination. Load impedance (ZL) set as:10 mega ohms (~infinite)
Rload node2 0 10MEG
```

Figure 10

To quickly change from an open-ended circuit to a short or matched/unmatched circuit, simply change the ZL value. For the short circuit case, set ZL to a small value that is close to zero (i.e 0.0001). To change from a resistive load to either an inductive or capacitive load, replace the instance line with an L or C element.

The waveform shown below in figure 11 is a combined plot of V(t) at the nodes Vsrc, node1, and node2 from the open-end T-line circuit example from figure 10's generated by the netlist.

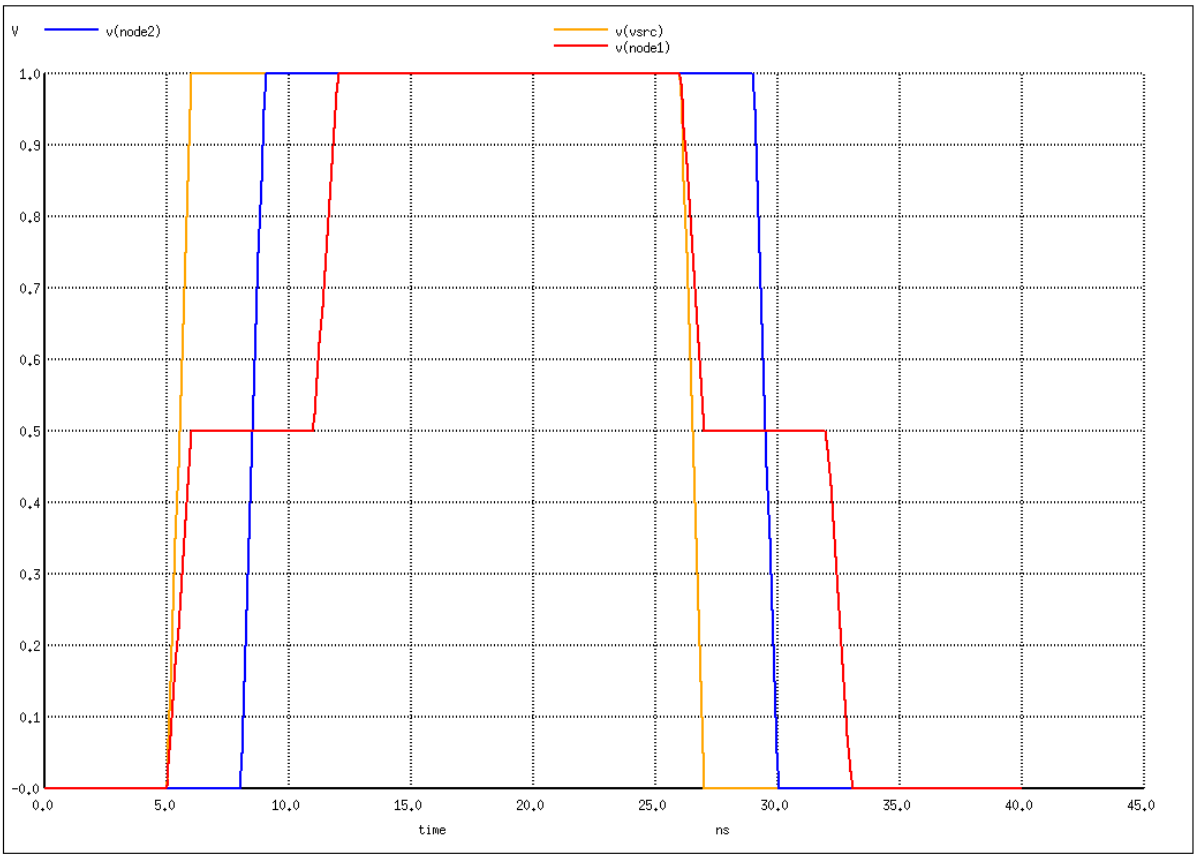


Figure 11: Open-circuit T-line waveform

To generate plots within a netlist, the plot parameters are set within the ".control to .endc " code.

General Form:

```
plot exprs [ylimit ylo yhi] [xlimit xlo xhi] [xindices xilo xihi]
[xcompress comp] [xdelta xdel] [ydelta ydel] [xlog] [ylog] [loglog]
[vs xname] [xlabel word] [ylabel word] [title word] [samep]
[linear]
```

This general form for the plot feature *does not require* all those parameters to be filled in and they may be omitted all together *(see pg.* 298 *in the manual for further details)*.

Examples of using the plot command:

- plot V(node1) V(node2)
- plot V(node1) V(node2) V(Vsrc) xl 1ns 45ns ; x-axis limit (xl) set to 1ns and 45ns

Here is the complete netlist for the open-ended circuit example:

```
open circuit T-Line

*0-1V Input source with 5ns delay, 1ns edges, 20ns pulse width, 40ns period
Vs vsrc 0 1 pulse(0 1 5e-9 1ns 1ns 20e-9 40e-9)

*50 ohm Source impedance (Zs)
Rsrc vsrc node1 50

*Transmission line with Zo = 50, td=3ns
T1 node1 0 node2 0 z0=50 td=3ns

*T-line Termination. Load impedance (ZL) set as:10 mega ohms (~infinite)
Rload node2 0 10MEG

.control
  set color0 = white ; plot window - background color
  set color1 = black ; plot window - grid and text color

 *plot V(t) with a x-axis limit of 1ns to 45ns
  plot  v(node1) v(node2) v(vsrc) xl 1ns 45ns
.endc
.end
```

Figure 12

The lines "set color0" & "set color1" inside the .control body *are optional* and only change the look of the output window from the default background colors of black & white text. This option is useful for wanting printed copies of the plots to save on printer ink. Just note, that *some features may not be available for the Windows version*.

5.2 Transmission Lines (lossy)

To create a lossy transmission line, a model line statement needs to be included. A model statement is a list specifying the device parameters of the component(s) to be simulated. Devices are given device codes to indicate which type of circuit element it is. For a lossy transmission line, the model code is, "LTRA". See table 2.3 on page 50 for a complete list of model types and their corresponding model code.

General Form for a lossy T-line:

O[name] [node 1] [node 2] [node 3] [node 4] [Model Name]

.model [Model Name] [Model Code] [p1 = value] ... [p15 = value]

Note: p1 ... p15 refers to the list of parameters to assign certain values too. Some parameters are optional, but the length parameter "LEN" *must be specified*. For the most part, only R,L,C and G parameter values are needed in this course.

EXAMPLE: Shorted lossy transmission line

This example simulates a lossy transmission line shorted at the end from a frequency range of 1Mhz to 1Ghz with a $4V_{pp}$ voltage source. The source resistance is 75 ohms and the lossy parameters per meter are: R=0.2 Ω/m, L=190nH/m, and C=140pF. The length of the transmission line is 50cm. The plot of the resonant frequencies can be seen in figure 13 with its corresponding netlist in figure 14.

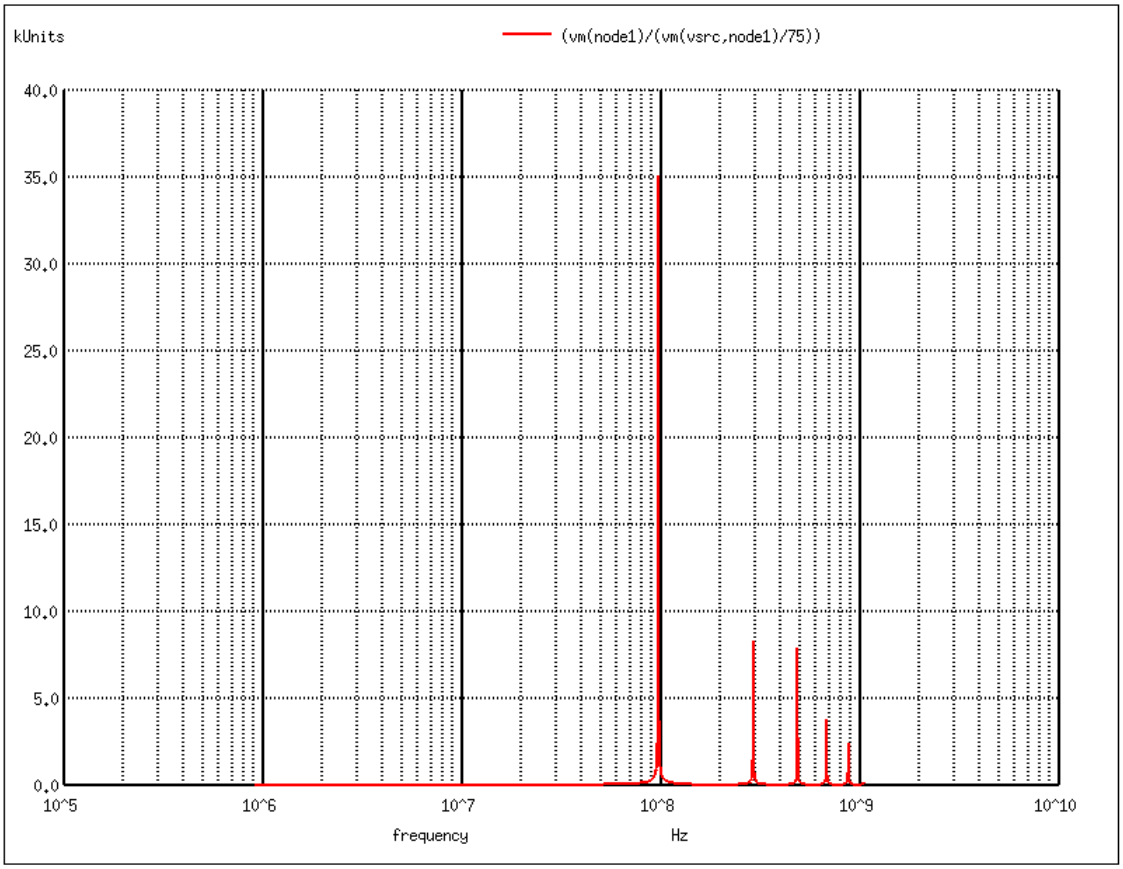


Figure 13: Shorted lossy t-line waveform

```
short circuit T-Line - lossy

*AC voltage source with peak voltage = 4V, amplitude = 2V
Vs vsrc 0 ac sin(0 2)

*Source impedance R = 75 ohms
Rsrc vsrc node1 75

*Lossy Transmission line
O1 node1 0 node2 0 lossyTline
.model lossyTline LTRA r=.02 l=190nH g=0 c=140pF len=0.5

*T-line termination = shorted end -> set Rload = 0 ohms
Rload node2 0 0

.control
  set color0 = white ; plot window - background color
  set color1 = black ; plot window - grid and text color
  ******************************************************************************
  *  performs ac small signal analysis - 1000/decade start:0.9e6, end:1.2e9
  *                            See pg. 233 for more details

  ac dec 1000 0.9e6 1.05e9 ; note dec stands for decade variation

  ******************************************************************************

  *this will plot magnitude of Zin - see pg. 252 for vm and vp output controls
   plot (vm(node1)/(vm(vsrc,node1)/75)) ; <-- 75 is the source impedance

.endc
.end
```

Figure 14

6 Subcircuits

Subcircuits are lumped circuit elements that are treated as a single block. A subcircuit is analogous to a function call in a c/c++ programming language. In other words, it is called from another body of code. A subcircuit is supplied arguments like a function call, but they are actually the ports on the subcircuit block (figure 15). To model a lossless transmission line with only L's & C's, would require several individual elements. However, creating a sub circuit allows the use of creating large sums of elements within only a few lines of code. This is just one advantage of using a netlist generated circuit vs. a schematic capture circuit that may otherwise be nearly impossible to create. Below is a subcircuit internal representation and equivalent block diagram of a T-line subcircuit.

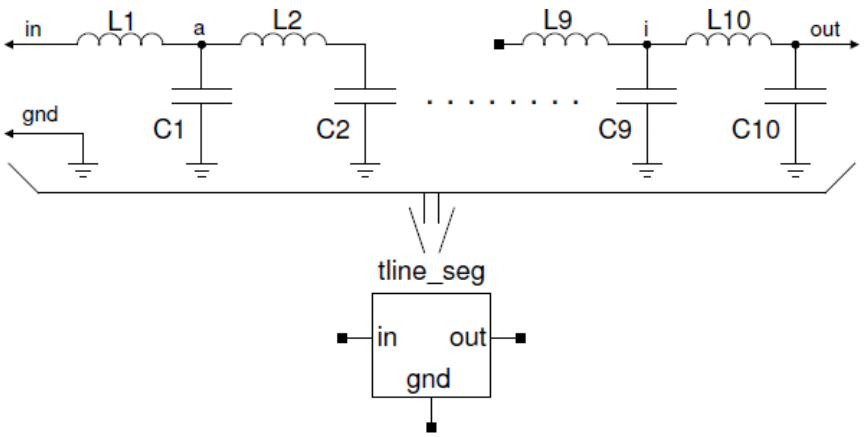


Figure 15

6.1 Creating a Subcircuit

Subcircuits begin with a control line ".subckt", a subcircuit name, and then the names of its ports or external nodes. The port names or external nodes are understood internal to the subcircuit. The elements in the subcircuit are connected as with any spice netlist. The subcircuit ends with the ".ends " command. In cases where there are many elements of the same value, or where you may want to override values, parameters may be added to the subcircuit to set a single value to all circuit elements with that parameter value.

To use the subcircuit in the top level circuit, it is called with a line that begins with an X, followed by a unique number if there are multiple instances of the subcircuit. Lower case x will also work, but the capital version stands out more in the netlist. Wires are connected to the subcircuit in the same order in which they were declared in the subcircuit declaration. At the end of the line, the name of the subcircuit being used is named.

General form:

. SUBCKT [Subname] [Node1] ... [Node 2, 3, 4, ...]

Example:

. SUBCKT TLINE 1 2 GND

"Subname" is the sub circuit name to be referenced by and the nodes are the external connecting nodes. The nodes can be named with any string of characters with the exception that they cannot be "0" (*see pg.* 50 *in the manual for further details*).

EXAMPLE: Subcircuit

The following example creates a subcircuit that consists of 10 LC elements to model a lossless transmission line. Once the 10 LC elements are defined within the subcircuit block, 10 instances of the blocks will be tied together in series, thus in-effect creating 100 LC elements lumped together. Below is the circuit that is being modeled with its waveform plotted and netlist shown.

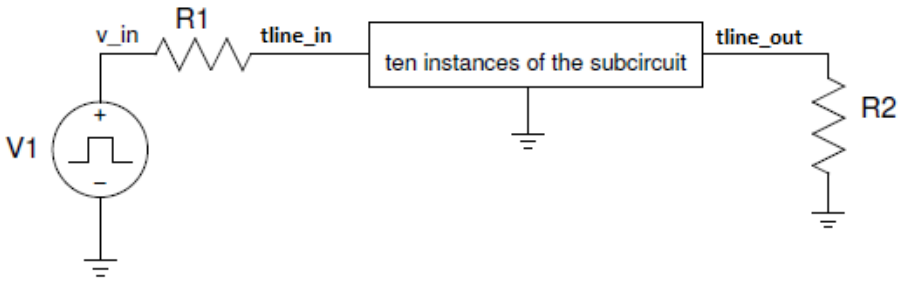


Figure 16

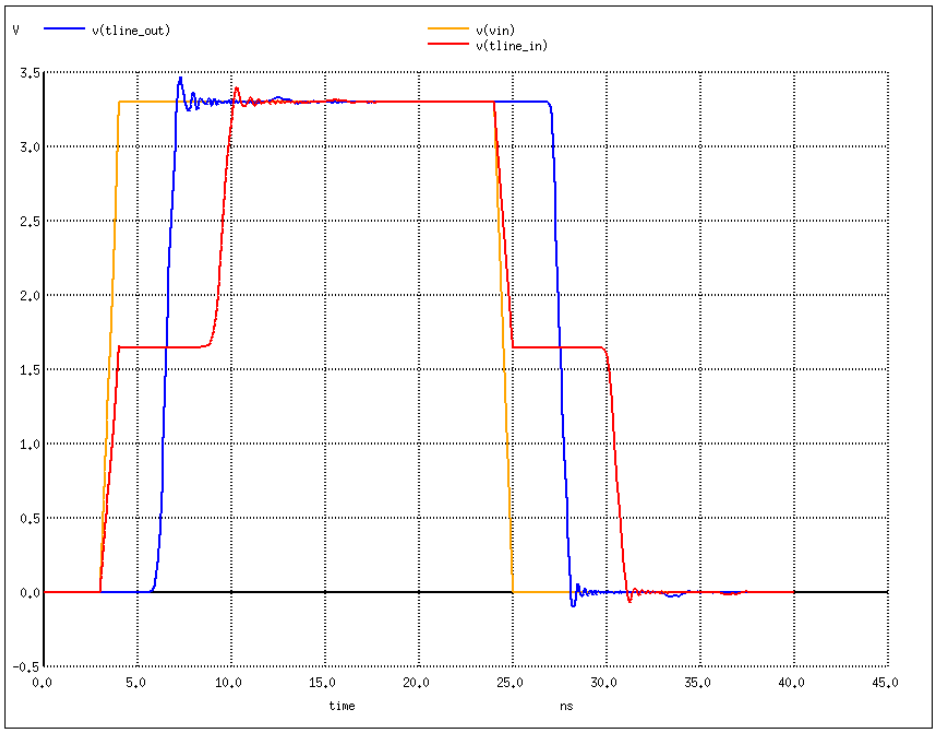


Figure 17

```
Transmission Line - LC Network 2ns, 150 ohm line

*input source with 3ns Delay, 1nS edges, 20ns pulse width, 40ns cycle time
Vin vin 0 PULSE(0 3.3 3e-9 1n 1n 20e-9 40e-9)

.param C_val=0.2pf
.param L_val=4.5nh

********************************************************************************
*Sub circuit composed of 10 LC elements
.subckt tline_seg in out gnd
l1   in      a     L_val
l2    a      b     L_val
l3    b      c     L_val
l4    c      d     L_val
l5    d      e     L_val
l6    e      f     L_val
l7    f      g     L_val
l8    g      h     L_val
l9    h      i     L_val
l10   i      out   L_val
c1    a             gnd    C_val
c2    b      gnd    C_val
c3    c      gnd    C_val
c4    d      gnd    C_val
c5    e      gnd    C_val
c6    f      gnd    C_val
c7    g      gnd    C_val
c8    h      gnd    C_val
c9    i      gnd    C_val
c10   out    gnd    C_val
.ends tline_seg
********************************************************************************
*source impedance
rsrc   vin           tline_in     150
*call subcircuit
X1     tline_in             a             gnd   tline_seg
X2     a                    b             gnd   tline_seg
X3     b                    c             gnd   tline_seg
X4     c                    d             gnd   tline_seg
X5     d                    e             gnd   tline_seg
X6     e                    f             gnd   tline_seg
X7     f                    g             gnd   tline_seg
X8     g                    h             gnd   tline_seg
X9     h                    i             gnd   tline_seg
X10    i             tline_out            gnd   tline_seg
.control
tran 100ps 40ns

set color0 = white      ;background
set color1 = black      ;text and grid
plot  V(tline_in) V(tline_out) V(vin) xl 100ps 45ns
.endc
********************************************************************************
*Measure the time delay
.meas tran tdelay trig v(tline_in) val=1.65 rise=1 td=100ps
+              targ v(tline_out) val=1.65 rise=1 td=100ps

*measure the input impedance during the rising edge
.meas tran z_in find par('(v(tline_in)/((v(vin)-v(tline_in))/150))') when v(tline_in)=1.65
.end
```

Figure 18

Notice at the bottom of the netlist after the plot command, ngspice is instructed to perform a transient analysis and reports the measurement values onto the output window. This is done with the .meas command. The ".meas" command is complex with many options and possibilities for making measurements *(see pg.* 243 *in the manual for further details)*. From the netlist, the first .meas command measures the delay and the result generated is called, "tdelay". The measurement is triggered when the first rising edge of node v(tline_in) reaches Vth (threshold voltage). The check for the trigger begins after a 100ps delay. The target or point at which the measurement is ended, is reached when the first rising edge of node v(tline_out) reaches Vth.

The second .meas command measures the input impedance. Vth was chosen as the point on the rising edge to begin taking the measurement. The result generated is called, "z_in". The input impedance is determined by dividing the voltage at the input of the network by the current flowing into the network. With the measure command, it can be confirmed that the t-line delay and input impedance match the theoretical calculated values seen here:

```
Measurements for Transient Analysis

tdelay               =  2.509494e-009 targ=  6.501931e-009 trig=  3.992436e-009
z_in                 =  1.522991e+002
```

Figure 19

The characteristic impedance z_in was calculated to be about 152 ohms, which is close to the theoretical value of 150 ohms. Thus, this model of 100 LC elements is a very close approximation.

7 AC - Standing Waves

Standing waves are generated when a transmission line is not matched or properly terminated to match the line. To create transmission lines that will generate standing waves when not properly terminated and others that won't due to having a matched line in ngspice, look at the following examples and their corresponding waveform plots respectively.

7.1 Standing Wave Examples

Example A:

```
*terminated 75 ohm line will show no standing waves.
*Vin vin 0 ac 1.0 sin
*rsrc  vin  tl_in  75
*t1 tl_in 0 tl_out 0 z0=75 td=1u
*rload tl_out 0 75

.control
ac lin 1000 1m 0.5Meg
 plot vm(vin) vm(tl_in) vm(vin,tl_in)  vm(tl_out)
 .endc
.end
```

Example B:

```
*unterminated (1e7) 75 ohm line will show standing waves.
Vin vin 0 ac 1.0 sin
rsrc  vin  tl_in  75
t1 tl_in 0 tl_out 0 z0=75 td=1u
rload tl_out 0 1e7
.control
ac lin 1000 1m 0.5Meg
 plot vm(vin) vm(tl_in) vm(vin,tl_in)  vm(tl_out)
 .endc
.end
```

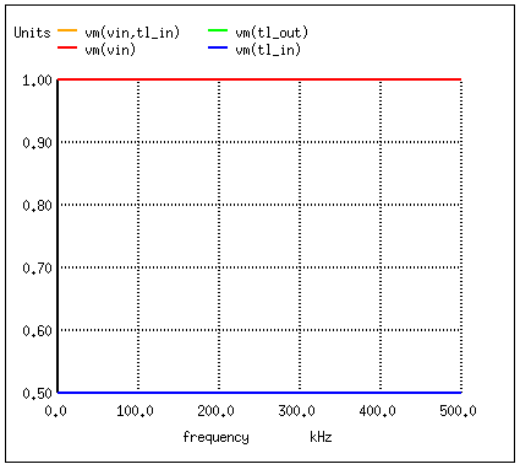
Example C:

```
*unterminated (0.01 ohm) 75 ohm line will show standing waves.
*Vin vin 0 ac 1.0 sin
*rsrc  vin  tl_in  75
*t1 tl_in 0 tl_out 0 z0=75 td=1u
*rload tl_out 0 0.01
.control
ac lin 1000 1m 0.5Meg
 plot vm(vin) vm(tl_in) vm(vin,tl_in)  vm(tl_out)
 .endc
.end
```

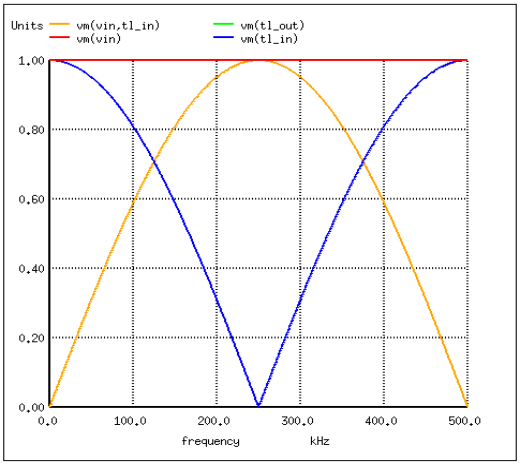
Example D:

```
*terminated (75 ohm at end point) line will not show standing waves.
*Vin vin 0 ac 1.0 sin
*rsrc  vin  tl_in  0.01
*t1 tl_in 0 tl_out 0 z0=75 td=1u
*rload tl_out 0 75
.control
ac lin 1000 1m 0.5Meg
 plot vm(vin) vm(tl_in) vm(vin,tl_in)  vm(tl_out)
 .endc
.end
```

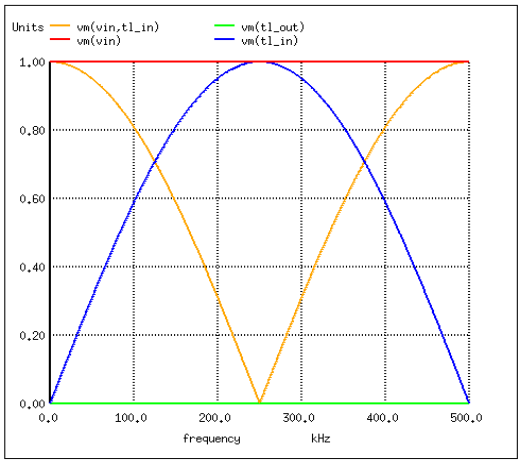
7.2 Standing Wave plots



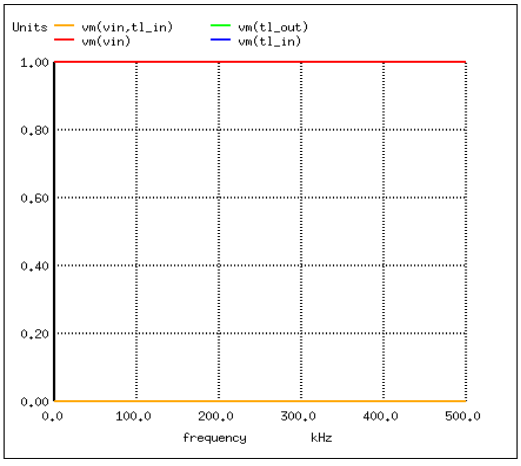
Example A



Example B



Example C



Example D