A Simplified Introduction to Circuit Simulation using SPICE OPUS

R S Ananda Murthy
Assistant Professor, Dept. of E&EE,
S J College of Engineering, Mysore 570 006

17th September 2004

Contents

1 About This Document

This document has been prepared without the help of any proprietary software products using the following:

- Slackware Linux Version 10.
- LyX Document Processor Ver. 1.3.4 with LaTeX typesetting program.
- Document Class: Article (Komascript)
- XFig Ver. 3.2.4

For the latest version of this document contact *lifelektronix@yahoo.co.in*

2 What is SPICE?

The word SPICE is an acronym which stands for

Simulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis

Using SPICE we can predetermine on a computer, the response of a circuit and thereby evaluate its working. This is faster, safer, economical and hassle-free way of testing a circuit before building a prototype. Many hardware manufacturing industries regularly use SPICE for design of electronic circuits.

3 What is EDA?

EDA stands for **E**lectronic **D**esign **A**utomation. EDA tools are software using which we can not only simulate electronic circuits, but also design the printed circuit board (PCB) of the circuit. SPICE is an important component of all EDA tools available now. In addition to SPICE, all EDA tools have programs for preparing circuit schematic, PCB drawing, and optimal layout of components on the PCB. Advanced EDA tools come with higher level hardware description languages like VHDL for designing even integrated circuits.

4 Brief History of SPICE

- In early 1970s, University of California, Bekeley, developed the first circuit simulator called CANCER.
- In mid-1970s University of California, Berkeley, developed the first version of SPICE called SPICE2. This was derived from CANCER.
- SPICE3 was developed from SPICE2. It has slight differences compared to the latter.

The SPICE developed by University of California, was able to run only on main frame computers. Many commercial companies, brought out modified versions SPICE which could also run on a PC. Most of the present day SPICE software, including commercial ones, are based on the original SPICE developed by University of California. So, they have almost similar syntax with minor variations. Most commercial versions of SPICE have user-friendly graphical user interface.

5 Some Commercial Versions of SPICE

You can search for SPICE related information on the internet by using the search string EDA + SPICE. There are now many commercial versions of SPICE supplied by various companies. Some of them are listed below (this is not an exhaustive list)

- PC Versions
  - PSPICE – supplied by OrCAD.
  - IS-SPICE – supplied by IntuSoft.
  - Z-SPICE — supplied by Z-Tech.
  - MultiSym – supplied by Electronic Work Bench (`www.interactiv.com`)
- Mainframe versions
  - HSPICE — supplied by Meta-Software. This is designed for integrated circuit design with special device models.
  - RAD-SPICE — supplied by Meta-Software. This is for simulating circuits subjected to ionizing radiation.
  - IG-SPICE – supplied by A. B. Associates.
  - Cadence-SPICE — supplied by Cadence Design
  - SPICE-Plus — supplied by Valid Logic

6 What is SPICE OPUS ?

SPICE OPUS is a circuit simulator available in two versions: (i) SPICE OPUS Lite; and (ii) SPICE OPUS Pro. It is a recompilation of the original Berkeley's source code for Windows 95/98/NT and Linux operating systems with Georgia Tech Research Institute's XSPICE mixed-mode simulator. You can simulate analog, digital, and analog+digital circuits using SPICE OPUS. This program is being developed by Faculty of Electrical Engineering, University of Ljubljana, Slovenia. SPICE OPUS Lite is free version which is available for downloading at `http://fides.fe.uni-lj.si/spice`. The lite version has no optimization tool, a limited circuit size of 100 nodes, and no user support. Except this, both lite and pro versions are functionally identical. The SPICE OPUS Pro which comes with optimization tool is a commercial version supported by SimShelf International (`www.simshelf.com`). This version has no limitations like the lite version. The simulator comes with an interpreted programming language called Nutmeg, which allows interactive SPICE sessions.

7 Types of Simulation Studies

The following types of studies can be conducted on a given analog circuit using SPICE OPUS:

- *D.C. Analysis* — Determination of steady-state response of the circuit when time-invariant D.C. sources are applied. This can be done using `op` command.
- *D.C. Sweep Analysis* — Determination of response of the circuit when excitation, or any other component is varied over a range. This can be done using `dc` command.

- *A.C. Analysis (Frequency Domain Analysis)* — Determination of steady-state response of the circuit when sinusoidal excitation is applied mine the frequency response of the circuit. The frequency of excitation can be varied to determine the frequency response of the circuit. This requires `ac` command and `ac` specification in at least one of the sources in the circuit.
- *T.F. Analysis* — Determination of small-signal transfer function with small-signal gain, input impedance, output impedance. This can be done using `tf` command.
- *Transient Analysis (Time Domain Analysis)* — Determination of variation of response of the circuit with respect to time. This be done using `tran` command.

Apart from the basic analyses mentioned above, we can also conduct the following advanced analyses:

- *Sensitivity Analysis* — This can be done using `.SENS` command.
- *Distortion Analysis* – This can be done using `.DISTO` command.
- *Noise Analysis* — This can be done using `.NOISE` command.
- *Pole-Zero Analysis* — This can be done using `.PZ` command.

However, these are not required in an introductory course on circuit simulation and hence they have not been used in the introductory simulation examples given further below.

Spice Opus comes with digital code models. These models can be used to simulate digital circuits. But on digital circuits we can conduct only `op` and `tran` analyses.

8 Starting and Quitting Spice Opus

Click on the `Spice Opus` link available on the desktop (If link is not available, then, ask your System Administrator). `Spice Opus` Command Window opens with prompt as shown in Figure 1. Now, `Spice Opus` is ready to execute commands. To quit, type quit and press `Enter` or close the `Spice Opus` window.

9 List of Code Models Available

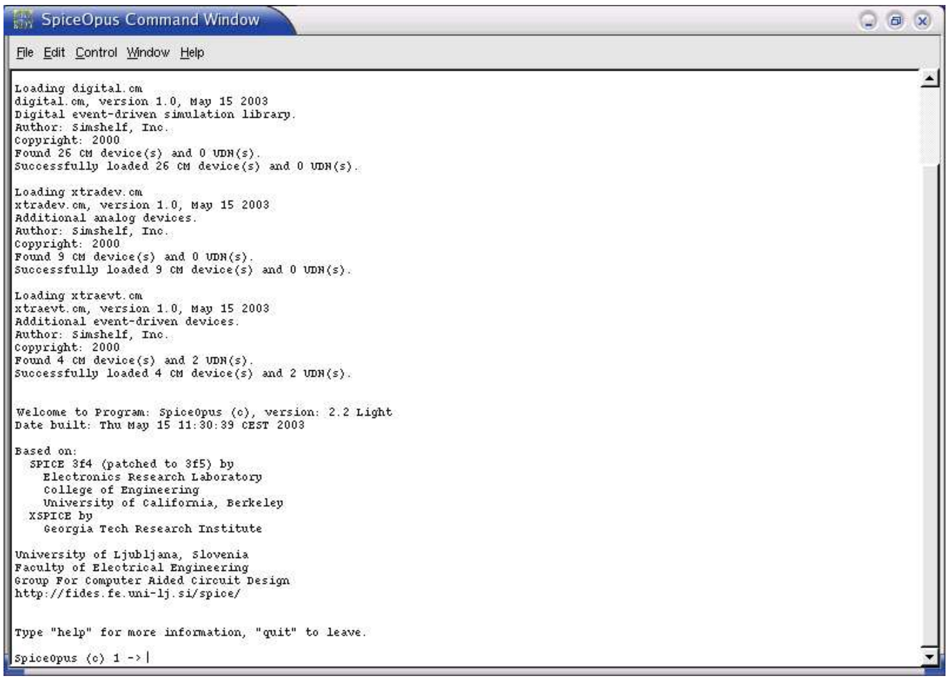
We can get a list of all code models available in Spice Opus by typing:

```
siminfo all<Enter>
```

at Spice Opus prompt.

10 Procedure for Simulation

Given a circuit, we should follow the procedure given below to conduct simulation:



```
Loading digital.cm
digital.cm, version 1.0, May 15 2003
Digital event-driven simulation library.
Author: Simshelf, Inc.
Copyright: 2000
Found 26 CM device(s) and 0 UDN(s).
Successfully loaded 26 CM device(s) and 0 UDN(s).

Loading xtradev.cm
xtradev.cm, version 1.0, May 15 2003
Additional analog devices.
Author: Simshelf, Inc.
Copyright: 2000
Found 9 CM device(s) and 0 UDN(s).
Successfully loaded 9 CM device(s) and 0 UDN(s).

Loading xtraevt.cm
xtraevt.cm, version 1.0, May 15 2003
Additional event-driven devices.
Author: Simshelf, Inc.
Copyright: 2000
Found 4 CM device(s) and 2 UDN(s).
Successfully loaded 4 CM device(s) and 2 UDN(s).


Welcome to Program: SpiceOpus (c), version: 2.2 Light
Date built: Thu May 15 11:30:39 CEST 2003

Based on:
  SPICE 3f4 (patched to 3f5) by
    Electronics Research Laboratory
    College of Engineering
    University of California, Berkeley
  XSPICE by
    Georgia Tech Research Institute

University of Ljubljana, Slovenia
Faculty of Electrical Engineering
Group For Computer Aided Circuit Design
http://fides.fe.uni-lj.si/spice/


Type "help" for more information, "quit" to leave.

SpiceOpus (c) 1 -> |
```

Figure 1: Spice Opus Command Window

- Describe the circuit connections either in a `*.cir` text file or graphically by using a schematic program. SPICE OPUS does not have a built-in schematic program. However, we can use an external schematic program called EAGLE to prepare the circuit schematic. But for a novice it is easier to use the text file. Using a text file also helps in better understanding of the basics of circuit simulation. So we will use `*.cir` to describe the circuit. Interested students can try EAGLE.

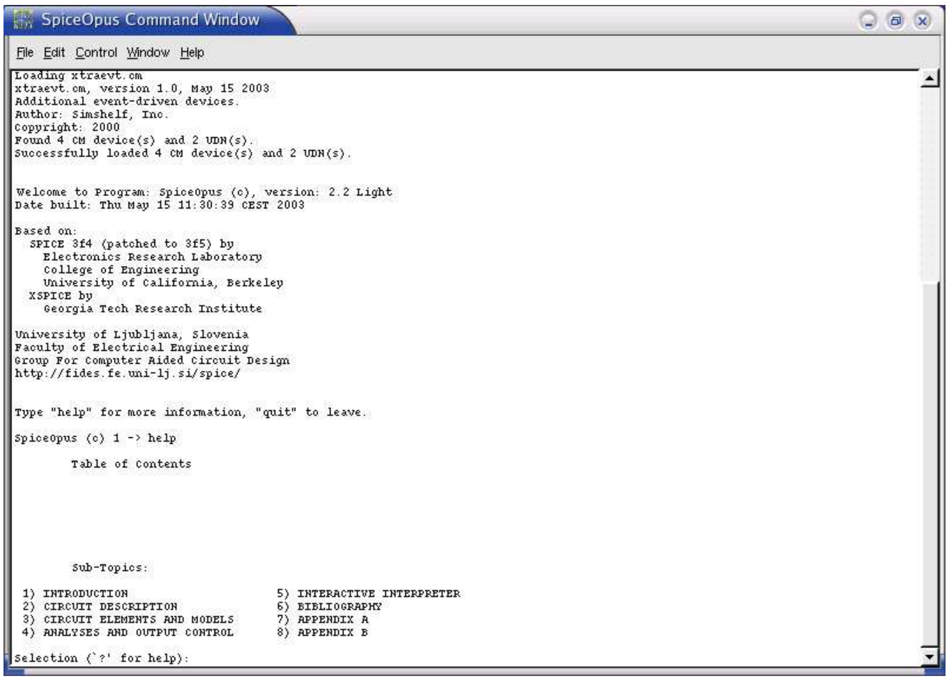
- Specify the type of simulation studies to be conducted (See Section 7 above). If text file is used, we can specify the commands for simulation within the text file it self.

- Start simulation by loading the circuit file into SPICE program.

- Print or plot the desired results. Commands for this may be included in the circuit file it self.

11 Using On-line Help

When Spice Opus window is open, at `Spice Opus` prompt, type `help` and press `Enter` to open on-line help. Then, you will see a list of sub-topics as shown in Figure 2. Type the serial number of the sub-topic you want to open and press `Enter`. To quit help type `q` and press `Enter` or repeatedly press `Enter` until you come back to `Spice Opus` prompt.



```
Loading xtraevt.cm
xtraevt.cm, version 1.0, May 15 2003
Additional event-driven devices.
Author: Simshelf, Inc.
Copyright: 2000
Found 4 CM device(s) and 2 UDN(s).
Successfully loaded 4 CM device(s) and 2 UDN(s).


Welcome to Program: SpiceOpus (c), version: 2.2 Light
Date built: Thu May 15 11:30:39 CEST 2003

Based on:
  SPICE 3f4 (patched to 3f5) by
    Electronics Research Laboratory
    College of Engineering
    University of California, Berkeley
  XSPICE by
    Georgia Tech Research Institute

University of Ljubljana, Slovenia
Faculty of Electrical Engineering
Group For Computer Aided Circuit Design
http://fides.fe.uni-lj.si/spice/


Type "help" for more information, "quit" to leave.

SpiceOpus (c) 1 -> help

        Table of Contents







        Sub-Topics:

 1) INTRODUCTION                       5) INTERACTIVE INTERPRETER
 2) CIRCUIT DESCRIPTION                6) BIBLIOGRAPHY
 3) CIRCUIT ELEMENTS AND MODELS        7) APPENDIX A
 4) ANALYSES AND OUTPUT CONTROL        8) APPENDIX B

Selection (`?' for help):
```

Figure 2: Spice Opus sub-topics in on-line help.

12 Circuit File Format in Spice Opus

The circuit file contains circuit connection details and optionally commands for conducting simulation and for outputting results. It is an ASCII text file created using any text editor. If you use a formating word processor such as Microsoft Word to create the circuit file, remember to save the file without any formating as an ASCII text file. It is better to avoid using word processors to create a circuit file. Instead, use an ASCII text editor. In Linux environment Emacs, Vi, GEdit, Kedit, Kwrite, Pico etc. are some of the popular text editors. In Windows environment, you may use Notepad, Norton Editor, TextPad etc. But remember to save the file as `.cir` file. The circuit file outline is as shown in Figure 3. The first line of the circuit file is reserved for title. You may give any title for the circuit you are simulating. Commands for simulation and outputting results must be given using only lower case letters and without a . (dot) in the beginning between `.control` and `.endc`. The last line of the circuit file must be `.end`. *To avoid confusion, it is better to use only lower case letters throughout.*

If you do not write commands for conducting simulation and outputting results in your circuit file, then, you have to give these commands interactively at `Spice Opus` prompt after the circuit file is loaded into Spice Opus.

13 Scale Factors used in SPICE

Table 1 shows scale factors used in SPICE while indicating values of parameters of circuit components.

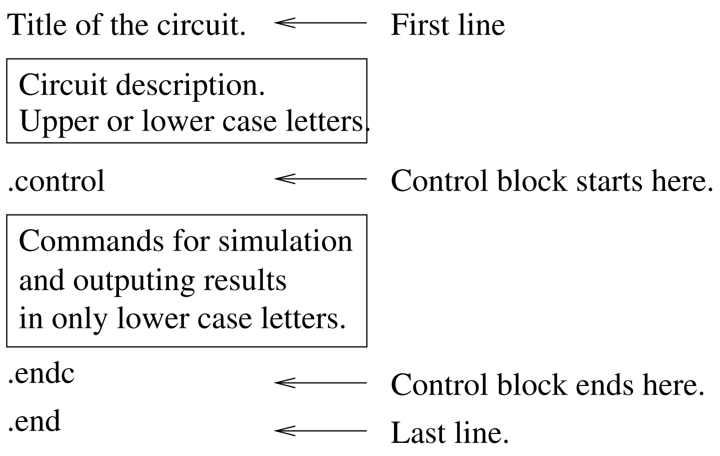


Figure 3: Circuit file format.

Scale Factor	Letter	Value
Terra	T	10^{12}
Giga	G	10^{9}
Mega	Meg	10^{6}
Kilo	K	10^{3}
Mills	mil	25.4^{-6}
Milli	m	10^{-3}
Micro	u (or M)	10^{-6}
Nano	n	10^{-9}
Pico	p	10^{-12}
Femto	f	10^{-15}

Table 1: Scale factors used SPICE.

14 Comments in Circuit File

Any line in the circuit file that begins with a `*` is treated as a comment by the SPICE program. In the programs given in the following pages, detailed comments have been given to explain the program. Go through these comments carefully. *Title of the circuit need not begin with a* `*`.

15 How to Load Circuit File into Spice Opus?

Start Spice Opus. Go to the directory in which the circuit file is stored using `cd path_to_directory` command. For example, if your circuit file is stored in `/home/anand/spicelab`, then, you type

```
cd /home/anand/spicelab <Enter>
```

Then type `source filename.cir` at `Spice Opus` prompt and press `Enter`.

16 Important Conditions to be Ensured

The following conditions should be ensured while simulating a circuit using SPICE:

- The ground or datum node of the circuit should be given number 0 and not letter o or O.
- Other nodes must be given names containing any arbitrary character strings.
- The circuit to be simulated cannot contain a loop of voltage sources and/or inductors and cannot contain a cut set of current sources and/or capacitors.
- Each node in the circuit must have a D.C. path to ground. So, there cannot be any dangling nodes.
- Every node must have at least two connections except for transmission line nodes and MOSFET substrate nodes.
- When you conduct frequency domain analysis using `ac` command, at least one of the sources in the circuit file must have `ac` values specified. This will be explained later in more detail.
- When you conduct time domain analysis using `tran` command, at least one of the sources in the circuit file must have time domain specifications. This will be explained later in more detail.
- A resistance can not have zero value because this causes divide by zero error.

If you get error messages while running simulation check whether you are violating any of the conditions given above.

17 Introductory Simulation Examples

The codes given below show examples of simulation of different types of circuits. These examples are given to introduce a novice to circuit simulation. You must go through these programs carefully and understand the statements. Extensive comments have been provided to explain the statements. You must try to simulate new circuits other than what is given in these examples to master the art of simulation.

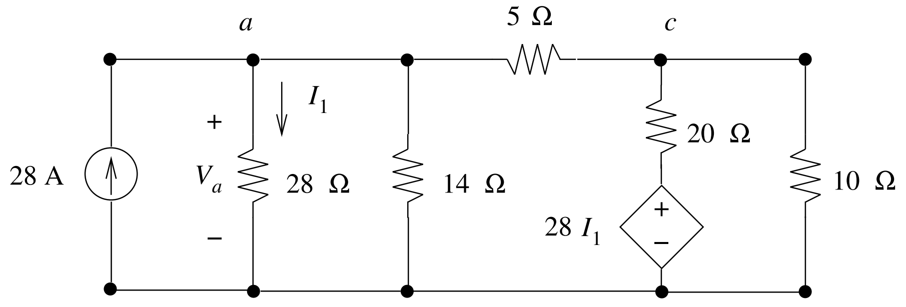


Figure 4: Circuit for Problem-1.

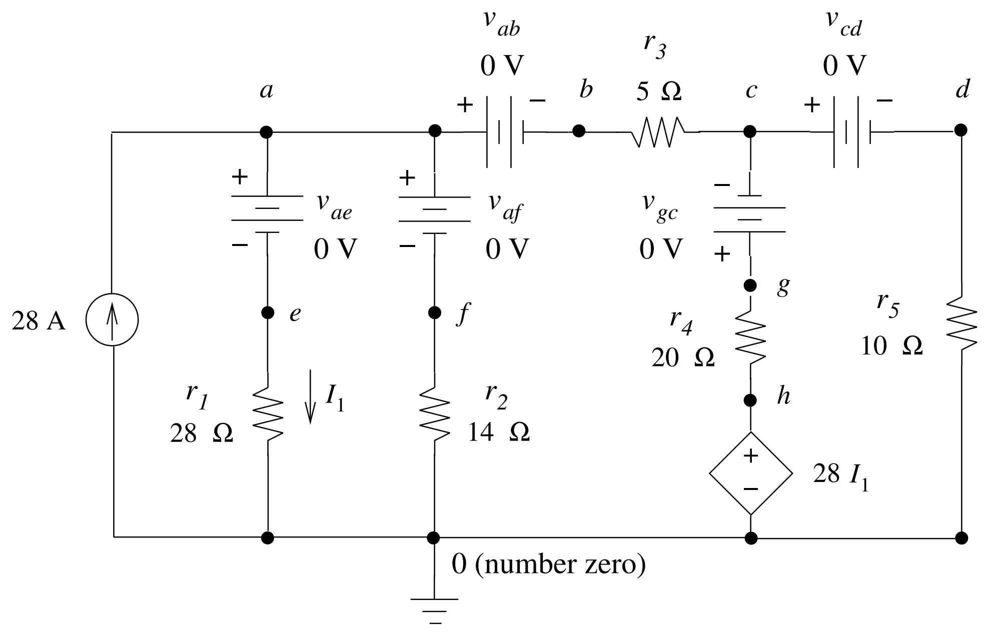


Figure 5: Circuit for Problem-1 redrawn with dummy sources in each branch.

17.1 Verification of Kirchhoff's Laws

PROBLEM-1: In the network shown in Figure 4 determine the voltage V_a and current delivered by the controlled source. Also prove Kirchhoff's Voltage Law and Kirchhoff's Current Law. (This is Problem 2-13, page no. 84, taken from, A. E. Fitzgerald, David E. Higginbotham, and Arvin Grabel, "Basic Electrical Engineering", ISBN 0- 07-021152-3, Fourth Edition, 1975, McGraw-Hill Kogakusha Ltd.)

This example is meant for novice users of Spice Opus to help them become familiar with steady-state analysis of a network excited by D.C. sources. Given a circuit, first mark the datum node with *number 0* and other nodes with either numbers or alpha-numeric labels. In SPICE, we use dummy voltage source — a voltage source of zero value — as an ammeter to measure current. Figure 5 shows the circuit of Figure 4 redrawn for simulation. Observe how dummy voltage source is introduced in each branch to measure branch current. Spice Opus always calculates current entering the positive terminal of a voltage source. Dummy voltage source polarity is decided on this basis. The circuit file to simulate the circuit shown in Figure 5 is given below.

```
KIRCHHOFF'S LAWS

* The above line is the title.

* This is how a dc current source is represented:
* i??????? fromnode tonode dc value.
* Here ??????? can be any alphabet or number.
i 0 a 28

* This is how a Current Controlled
* Voltage Source (CCVS) is represented:
* h??????? +node -node vnam value
* Here 'vnam' is the name of the voltage source
* through which the controlling current flows.
* 'value' is the transresistance in ohms.
h h 0 vae 28

* This is how a dc voltage source is represented:
* v??????? +node -node dc value.
* These dummy voltage sources work as ammeters:
vae a e dc 0
vaf a f dc 0
vab a b dc 0
vcd c d dc 0
vgc g c dc 0

* Resistors
* r??????? fromnode tonode value
r1 e 0 28
r2 f 0 14
r3 b c 5
r4 g h 20
r5 d 0 10

* Observe in the statements above that the unit for the
* quantity need not be specified. SPICE takes this
* automatically for the quantity.

.control
* op command calculates steady-state response to
* dc excitation assuming all inductances to
* be short circuits and all capacitors to be
* open circuits.
op
```

```
* Observe how to print messages using echo.
echo Answers are:

* Observe how to print node voltages and
* current through voltage sources.
print v(a) i(vgc)

* The following statement causes a blank line to be printed.
echo
echo These are KVL equations:
echo
* Observe how we can write expressions in print statement.
* Here v(a,c) means v(a)-v(c).
print v(a)-v(a,c)+v(h,g)-v(h)
print v(h)-v(h,g)-v(d)

echo
echo These are KCL equations:
echo
print 28-i(vae)-i(vaf)-i(vab)
print i(vab)+i(vgc)-i(vcd)
echo
echo Observe the values printed are almost zero.
echo
echo
.endc

* All circuit files must end with '.end' statement.
.end
```

Output of simulation will be as shown below (the actual numerical values may differ slightly):

```
Circuit: KIRCHHOFF'S LAWS

Answers are:
v(a) = 1.704348e+02
i(vgc) = 2.434783e+00

These are KVL equations:

v(a)-v(a,c)+v(h,g)-v(h) = 0.000000e+00
v(h)-v(h,g)-v(d) = 0.000000e+00

These are KCL equations:

28-i(vae)-i(vaf)-i(vab) = 3.552714e-15
```

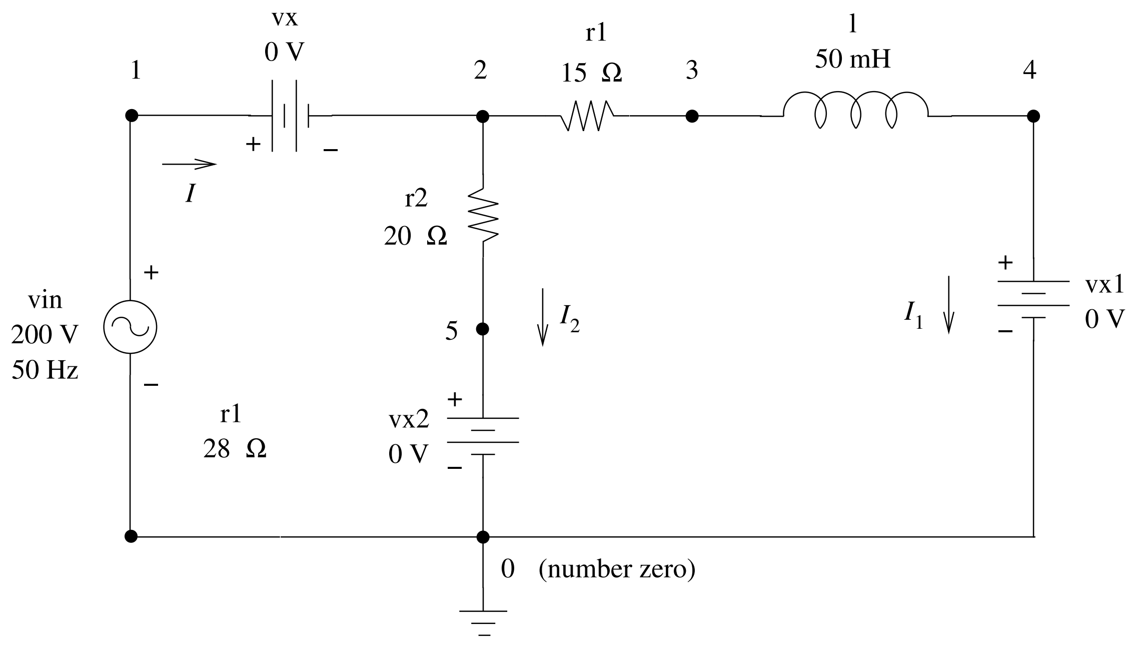


Figure 6: Circuit for Problem-2.

```
i(vab)+i(vgc)-i(vcd) = 3.552714e-15

Observe the values printed are almost zero.
```

17.2 Steady-state AC Circuit Analysis

PROBLEM-2: A coil of resistance 15 Ω and inductance 0.05 H is connected in parallel with a non-inductive resistor of 20 Ω. Find: (a) The current in each branch of the circuit; (b) The total current supplied; (c) the phase angle of the combination when a voltage of 200 V at 50 Hz is applied (Problem No. 39, taken from S. Parker Smith & N. N. Parker Smith, “Problems in Electrical Engineering”, VIII Edition, page no. 47, Asia Publishing House, 1976.). Also prove KVL and KCL. Answer: (a) 9.2A, 10A; (b) 17.6A; (c) 22°.

This example is meant for novice users of SPICE OPUS to help them become familiar with steady-state analysis of a circuit excited by constant frequency sinusoidal excitation. Figure 6 shows the circuit simulated using Spice Opus. Dummy voltage sources work as ammeters.

You know that steady-state analysis of a circuit excited by constant frequency sinusoidal excitation is done in frequency domain using the concept of impedance. For frequency domain analysis, an independent sinusoidal voltage source must be represented as:

```
v??????? +node -node dc 0 ac rmsvalue phaseangle
```

Similarly, an independent sinusoidal current source must be represented as

```
i??????? +node -node dc 0 ac rmsvalue phaseangle
```

Observe that dc value is specified as zero and frequency is not mentioned here. Only the RMS value of the source is given in this statement. If dc value is not given, Spice Opus outputs a warning message and takes its value as 0. If `rmsvalue` is not specified, it is taken as 1.

To conduct frequency domain analysis at a constant frequency the command to be included between `.control` and `.endc` is:

```
ac dec 1 source_frequency source_frequency
```

The meaning of `dec` will be explained later in detail. By default, Spice Opus outputs angle in radians. To make it output angle in degrees the command is:

```
set units=degree
```

Observe how we can print magnitude and phase angle of voltage and current. Create the circuit file given below using a text editor without any formatting and save it as a `*.cir` file.

```
STEADY-STATE AC CIRCUIT ANALYSIS

* This is a sinusoidal voltage source.
vin 1 0 dc 0 ac 200v

* Dummy sources to measure current
* vx measures total current
* vx1 measures current through inductive branch
* vx2 measures current through resistive branch
vx 1 2 dc 0
vx1 4 0 dc 0
vx2 5 0 dc 0

* Resistors
r1 2 3 15
r2 2 5 20

* Inductance
* l??????? fromnode tonode value
* Here, the first letter is L and not 1.
l 3 4 50mH

.control

* By default SPICE_OPUS outputs angles in radians.
* This command makes SPICE_OPUS output angles in degrees.
set units=degree

* This statement deletes any previously calculated
* values present in the workspace. It is better to do this
* before starting any simulation.
destroy all
```

```
* This is the command to conduct ac analysis
* at a constant frequency
ac dec 1 50 50

echo Total Current
* This command prints magnitude of a complex vector.
print mag(i(vx))
* This gives a blank line.
echo
echo Current through branch 1:
print mag(i(vx1))
echo
echo Current through branch 2:
print mag(i(vx2))
echo
echo Phase Angle of Impedance of Circuit in Deg.:
* This command prints phase angle of a complex vector
print ph(v(1)/i(vx))
echo
echo KVL equation:
print i(vx2)*@r2-(i(vx1)*(@r1+(i*@l*2*pi*50)))
* In the above equation @r2 means take the value of r2.
* Similarly, @r1 means take the value of r1.
* and @l means take the value of inductance l.
* i = square root of -1. This is a built-in constant in SPICE.
* pi = 3.14159... a built-in constant in SPICE.
echo
echo KCL equation:
print i(vx)-i(vx1)-i(vx2)
echo
echo Values printed above are almost zero.

.endc

.end
```

The output obtained by running simulation on the above mentioned circuit file is given below:

```
Circuit: STEADY-STATE AC CIRCUIT ANALYSIS

Total Current
mag(i(vx)) = 1.766299e+01

Current through branch 1:
mag(i(vx1)) = 9.208282e+00
```

```
Current through branch 2:
mag(i(vx2)) = 1.000000e+01

Phase Angle of Impedance of Circuit in Deg.:
ph(v(1)/i(vx)) = 2.215021e+01

KVL equation:
i(vx2)*@r2-(i(vx1)*(@r1+(i*@l*2*pi*50))) = 0.000000e+00,3.268219e-15

KCL equation:
i(vx)-i(vx1)-i(vx2) = -1.77636e-15,0.000000e+00

Values printed above are almost zero.
```

The example given above shows that by default Spice Opus outputs complex values in rectangular format in the form x, y where x is the real part and y is the imaginary part. It also shows how we can refer to component values by using @ operator. The next problem shows how we can simulate a three-phase circuit.

> PROBLEM-3: A three-phase 50 Hz, 220 V supply having ABC phase sequence is connected to a balanced Δ connected load having an impedance of $10\angle 30^\circ\,\Omega$ in each phase. Taking V_{BC} as reference phasor, find the line currents. Answer: $I_A = 38.1\angle 60^\circ$ A, $I_B = 38.1\angle -60^\circ$ A, $I_C = 38.1\angle -180^\circ$ A.

Figure 7 shows the circuit diagram. Since we want to measure the line currents, we have to insert dummy voltage sources in the lines. We cannot represent the supply as three Δ-connected line-to-line sources because, then, the three voltage sources form a loop which is not allowed in SPICE (refer to Section 16). Therefore, we represent the supply as three Y-connected phase voltage sources. You know from circuit theory, that, when phase sequence is ABC, the phase voltages lag behind the respective line-to-line voltages by 30°. Hence, we have the phase voltage sources

$$V_{BN} = \frac{V_{BC}}{\sqrt{3}}\angle -30^\circ = 127\angle -30^\circ\,\text{V}; \qquad V_{CN} = 127\angle 210^\circ\,\text{V}; \qquad V_{AN} = 127\angle 90^\circ\,\text{V}$$

The phase impedance of the load is inductive since its phase angle is positive. Therefore, we have

$$Z = 10\angle 30^\circ\Omega = (R + jX) = (8.666 + j5)\Omega$$

The value of inductance of each phase impedance is

$$L = \frac{X}{2\pi f} = \frac{5}{2\pi \times 50} = 15.92\,\text{mH}$$

We need to calculate the inductance because we cannot enter the reactance value in the circuit file. Taking the values calculated above, the circuit to be simulated will be as shown in Figure 8. The circuit file for simulating this will be as shown below:

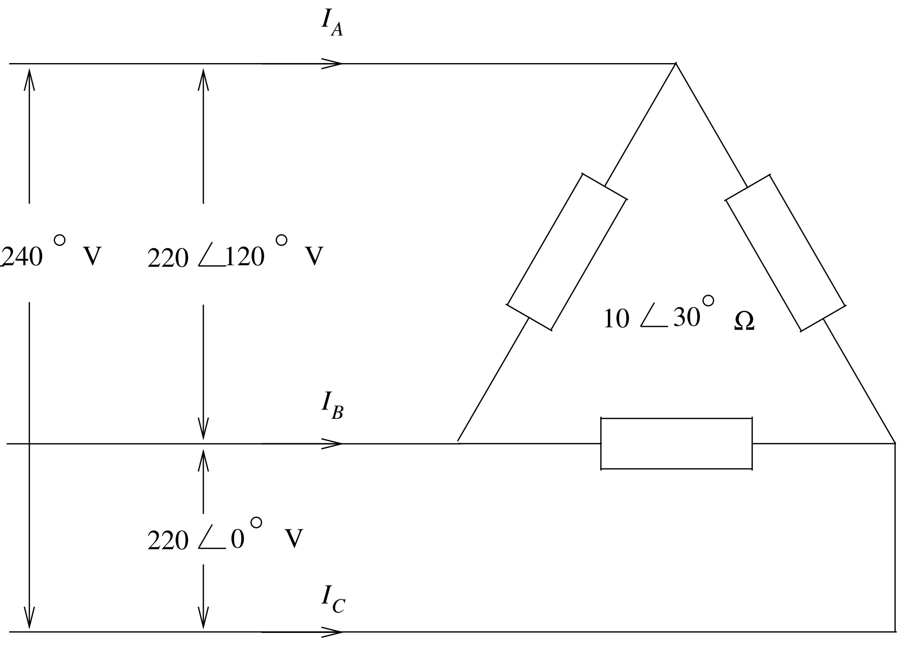


Figure 7: Circuit for Problem 3.

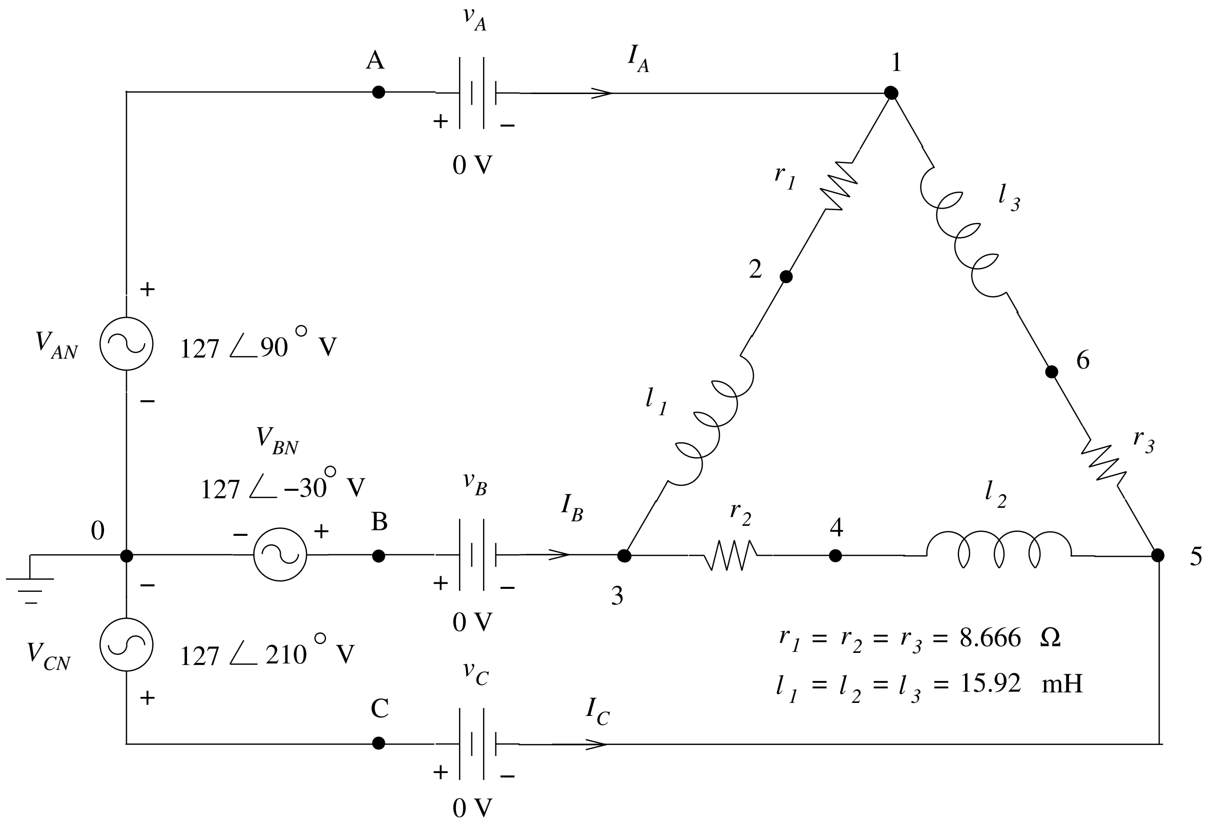


Figure 8: Circuit for simulation pertaining to Problem 3.

```
Three Phase Circuit Analysis

* Phase Voltage Sources
van A 0 dc 0 ac 127 90
vbn B 0 dc 0 ac 127 -30
vcn C 0 dc 0 ac 127 210

* Dummy sources to measure line currents
va A 1 dc 0
vb B 3 dc 0
vc C 5 dc 0

* Resistors
r1 1 2 8.666
r2 3 4 8.666
r3 5 6 8.666

* Inductances
l1 2 3 15.92mH
l2 4 5 15.92mH
l3 6 1 15.92mH

.control
* This clears the workspace.
destroy all
set units=degree
ac dec 1 50 50
echo The answers are:
print mag(i(va)) ph(i(va))
print mag(i(vb)) ph(i(vb))
print mag(i(vc)) ph(i(vc))
.endc

.end
```

The output of this simulation will be as shown below:

```
Circuit: Three Phase Circuit Analysis

The answers are:
mag(i(va)) = 3.807836e+01
ph(i(va)) = 6.000943e+01
mag(i(vb)) = 3.807836e+01
ph(i(vb)) = -5.99906e+01
mag(i(vc)) = 3.807836e+01
ph(i(vc)) = -1.79991e+02
```

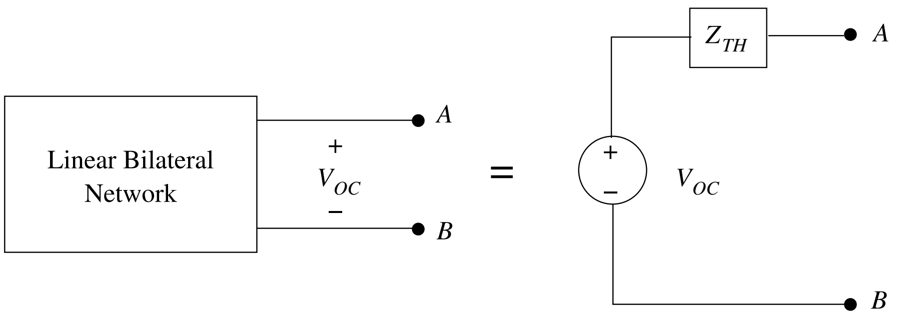


Figure 9: Thevenin equivalent network of a linear bilateral network.

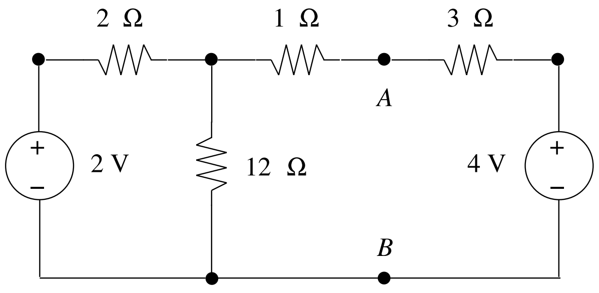


Figure 10: Circuit for Problem-4

17.3 Verification of Thevenin Theorem

In the following problems, we shall verify Thevenin Theorem. First we take a circuit excited by D.C. source/s. Then, we take a circuit excited by sinusoidal source/s.

Statement of Thevenin Theorem: Any linear bilateral network connected to terminals A and B as shown in Figure 9 can be replaced by a voltage source of value equal to the open circuit voltage V_{OC} between the terminals A and B connected in series with an internal impedance Z_{TH} where Z_{TH} is the equivalent impedance of the circuit as measured from the terminals A and B with all independent sources in the circuit reduced to zero. We also have $Z_{TH} = V_{OC}/I_{SC}$ where I_{SC} is the short circuit current flowing between A and B.

PROBLEM-4: In the circuit shown in Figure 10 find the current in a 2 Ω resistor connected between nodes A and B by applying Thevenin theorem (this is Problem-51, taken from S. Parker Smith and N. N. Parker Smith, "Problems in Electrical Engineering", VIII Edition, page no. 47). Answer: 0.8175 A

Figure 11 shows the circuit simulated using Spice Opus. The right portion of this figure shows the Thevenin equivalent of the circuit of Figure 10 as seen from the terminals A and B with a 2 Ω resistor connected. We solve this problem by two methods: (i) by using `tf` command; (ii) by using the equation $Z_{TH} = V_{OC}/I_{SC}$.

The general format of `tf` command is

```
tf output_node input_source
```

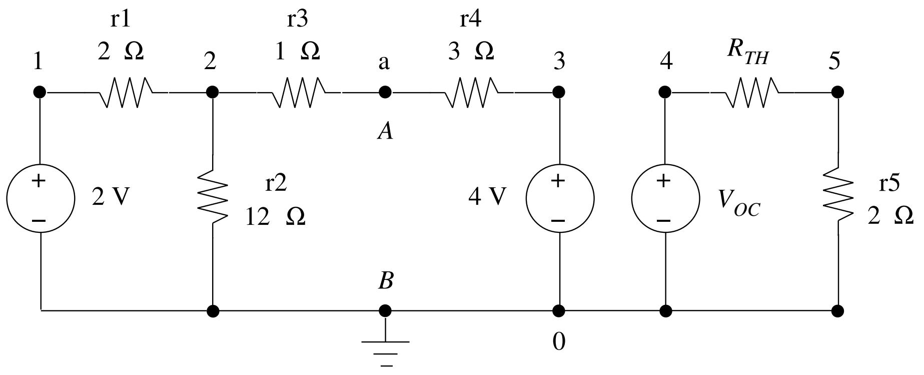


Figure 11: Circuit to be simulated for Problem-4.

The tf command performs a transfer function analysis, returning the `transfer_function` (output/input), `output resistance`, and `input resistance` between the given output node and the given input source. The circuit file given below shows how we use `tf` command:

```
THEVENIN THEOREM IN DC CIRCUIT: METHOD-1

* In this method we use tf command.
* This method can be applied only when the circuit
* has one or more independent dc sources.

* These are given voltage sources.
v1 1 0 dc 2v
v2 3 0 dc 4v

* This is Thevenin equivalent voltage source.
* This is later set equal to open circuit voltage.
voc 4 0 dc 0v

* Resistors
r1 1 2 2
r2 2 0 12
r3 2 a 1
r4 3 a 3
r5 5 0 2
* rth is initially set to some arbitrary
* value other than 0. Here it is assumed to be 10 Ohms.
* But later we set the value of this to the correct
* value using alter command.
rth 4 5 10

.control
* This clears the workspace.
destroy all
```

```
* You must conduct tf analysis with respect to each
* independent source in the given circuit.
tf v(a) v1
tf v(a) v2

* Observe how the values of sources are
* included in expressions as @v1 and @v2.
* Also how we can use let command to define new variables.
let ocvolt=(tf1.transfer_function*@v1+tf2.transfer_function*@v2)

* In the above equation we are actually applying
* principle of superposition.

let zth=output_impedance
echo Solution using tf command.
echo
echo Voc is
print ocvolt
echo
echo Rth is
print zth

* Now find current in Thevenin equivalent circuit.
alter voc=ocvolt
alter rth=zth
op

let i2ohm=-i(voc)
echo
echo Current through 2 Ohm resistor is
print i2ohm
echo
.endc

.end
```

In the above program, `tf` command is executed twice — once with respect to `v1` and then with respect to `v2`. Values calculated during the first run of `tf` will be kept in a set referred to as `tf1`. Similarly the values calculated during the second run will be in another set called `tf2`. Dot operators are used to refer to the values stored in a particular set. For example, when we say `tf1.transfer_function` we are referring to the `transfer_function` stored in the set `tf1`. Figure 12 shows the circuit to be simulated for verification of Thevenin theorem using the second method. In this method we first find V_{OC} keeping the switch open. Then, we close the switch to cause short circuit and measure I_{SC}. Then, we find $Z_{TH} = V_{OC}/I_{SC}$. The circuit file for second method is given below:

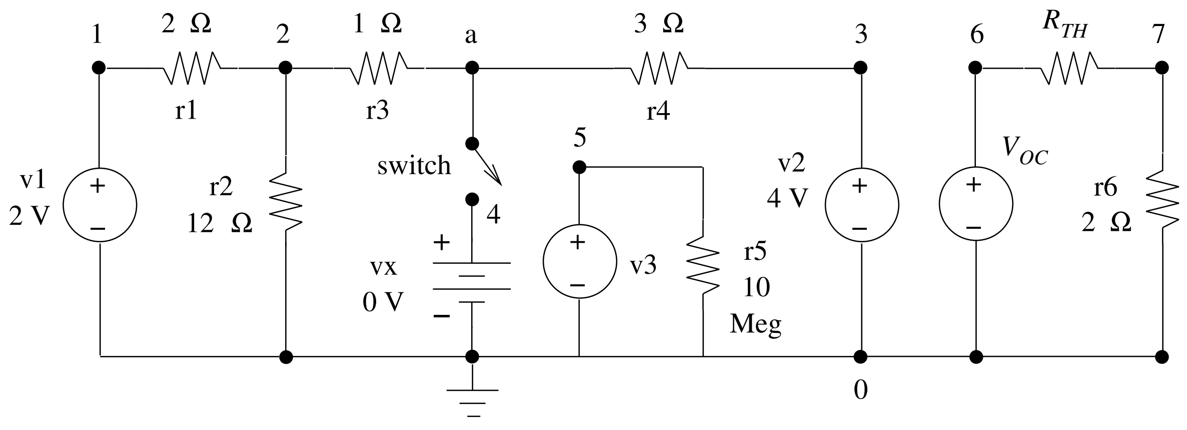


Figure 12: Circuit for verification of Thevenin theorem using a voltage controlled switch.

```
THEVENIN THEOREM IN DC CIRCUIT: METHOD-2

* In this method we first find Voc. Then, we use a
* voltage controlled switch to cause short circuit to measure Isc.
* Then, Rth is calculated as Voc/Isc. This method can be applied
* in any circuit. Observe that voltage controlled
* switch has finite on-state and off-state resistances.

* These are given voltage sources.
v1 1 0 dc 2v
v2 3 0 dc 4v

* This source controlls ON/OFF operation of the switch.
* When this is 0v the switch is OFF. The switch is closed
* by altering this to a higher value.
v3 5 0 dc 0v

* This dummy source measures Isc.
vx 4 0 dc 0v

* This is Thevenin equivalent voltage source initially
* set to 0 V arbitrarily. Later we set this to proper value.
voc 6 0 dc 0v

* A voltage controlled switch is specified as:
* S??????? N+ N- NC+ NC- MODELNAME <ON><OFF>
* Nodes N+ and N- are the nodes between which the
* switch terminals are connected.
* NC+ and NC- are the controlling voltage source terminals.
* The model name is mandatory while the initial condition
* (ON or OFF)is optional. This statement must be followed by a
* .model statement as shown below. Here sw represents
* the model of voltage controlled switch. The name given
* here for the model is switch. Observe that the ON state
* resistance ron is chosen to be very small and the OFF state
* resistance roff is chosen to be very large as compared to
* other resistors in the circuit.
s a 4 5 0 switch
.model switch sw(ron=0.00001 roff=10e+5)

* Resistors
r1 1 2 2
r2 2 0 12
r3 2 a 1
r4 3 a 3
r5 5 0 10Meg
r6 7 0 2
rth 6 7 10

.control
* This clears work space
destroy all

* Initially switch is open. So, find Voc.
op

echo Solution using voltage controlled switch.
echo
echo Voc is
print v(a)

* Apply controlling voltage to close the voltage
* controlled switch to cause short circuit and measure Isc.
alter v3=10
op
echo
echo Isc is
print i(vx)

* Calculate Rth
echo
echo Rth=Voc/Isc is
print op1.v(a)/op2.i(vx)

* Now find current in Thevenin equivalent circuit.
alter voc=op1.v(a)
alter rth=op1.v(a)/op2.i(vx)
op
```

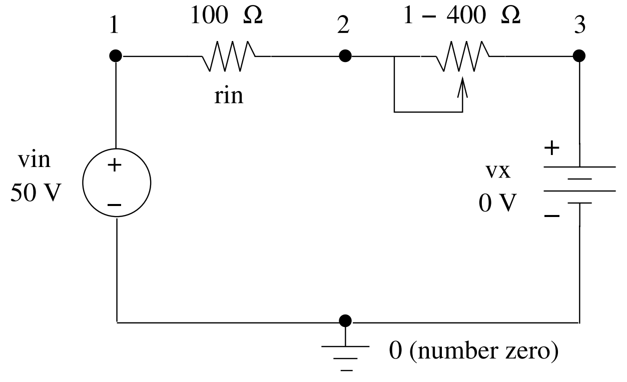


Figure 13: Circuit for verification of Maximum Power Transfer theorem.

```
echo
echo Current through 2 Ohm resistor is
print -i(voc)
echo
.endc

.end
```

17.4 Verification of Maximum Power Transfer Theorem

In the following problem we shall verify maximum power transfer theorem in a circuit excited by a D.C. source.

Statement of Maximum Power Transfer Theorem: In a linear bilateral network excited by a D.C. source, as shown in Figure 13, maximum power transfer from the source to the load takes place when the load resistance is equal to the internal resistance of the source.

In the following program, we use dc command to vary the load resistance. The general syntax of this command is

```
dc parameter startvalue stopvalue stepvalue
```

In this command, the `parameter` can also be a source. When this command is given, the `parameter` is varied from `startvalue` to `stopvalue` in steps of `stepvalue`. At each value of `parameter` the response of the circuit is found. In the following program we will also be using the `plot` command. Observe how we use this command to plot a graph of output power versus load resistance.

```
MAXIMUM POWER TRANSFER THEOREM IN DC CIRCUIT

* Voltage Sources
vin 1 0 dc 50
vx 3 0 dc 0

* Resistors
rin 1 2 100
```

```
rl 2 3 1
* Observe that we initially set rl to 1 Ohm
* and not to 0 because resistance cannot be
* 0 in SPICE.

.control
* Clear workspace.
destroy all

* Using this command rl is varied
* from 0 to 400 Ohms in steps of 1 Ohm.
* Power consumed in rl is plotted versus rl
* Value of rl corresponding to maximum power
* is read from the graph.
dc rl 1 400 1
plot i(vx)*v(2,3)
+ xlabel rL[Ohm]
+ ylabel P[Watts]
* A long statement can be continued in the next line
* by typing + in the first column as shown above.

.endc

.end
```

The output obtained from the program will be as shown in Figure 14. From this graph we see that maximum power is consumed by the load resistance when the load resistance is equal to the internal resistance of the source i.e., $100\,\Omega$in this case.

17.5 Series Resonance

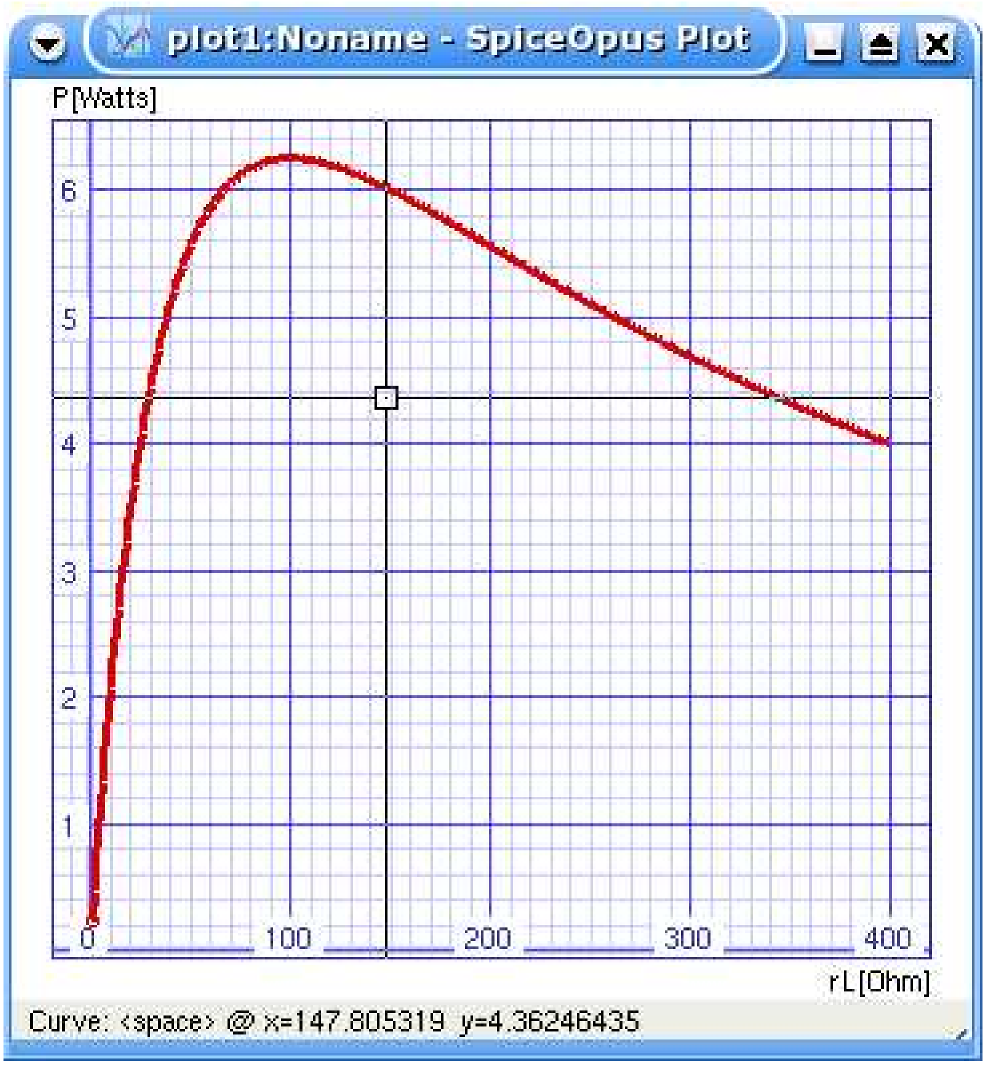
In the following problem we shall plot the frequency response of a series RLC circuit:

> PROBLEM-5: A series RLC circuit with $R = 100\,\Omega$, $L = 10\,\text{mH}$, $C = 0.1\,\mu\text{F}$ is excited by a variable frequency sinusoidal source of R.M.S. value 100 V. Plot the magnitude and phase angle of impedance of the circuit versus frequency of the source as the latter is varied from 25 Hz to 1000 kHz. Measure resonant frequency, half-power frequencies, and band width from the plots and verify their values by theoretical calculations. Repeat simulation with $R = 1000\,\Omega$ and $R = 5000\,\Omega$. What is the effect of variation of R on resonant frequency, half-power frequencies and bandwidth?

For the given circuit, the resonant frequency is given by

$$f_0 = \frac{1}{2\pi\sqrt{LC}}$$

At half-power frequency f_1 the phase angle of impedance of the circuit is -45° and at half-power frequency f_2 it is 45°. So, we can locate the half-power frequencies by plotting a graph of phase



Curve: <space> @ x=147.805319 y=4.36246435

Figure 14: Plot of output power versus load resistance.

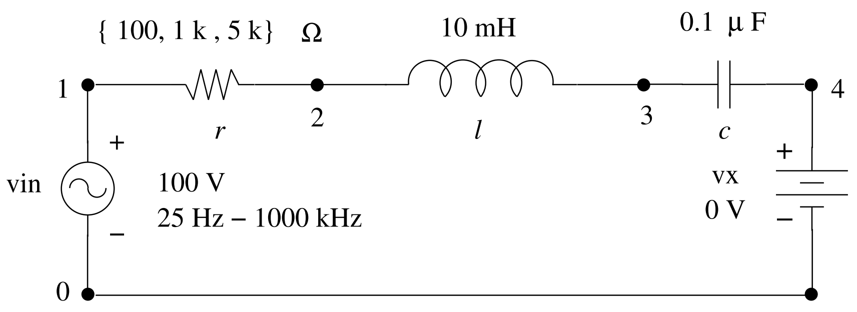


Figure 15: Circuit for studying series resonance.

angle of the impedance versus frequency. To do this, we need to vary the frequency of the sinusoidal excitation. So, we have to use `ac` command. To find frequency response, the command to be included between `.control` and `.endc` is

```
ac dec points_per_decade start_frequency end_frequency
```

For example, if we give the command

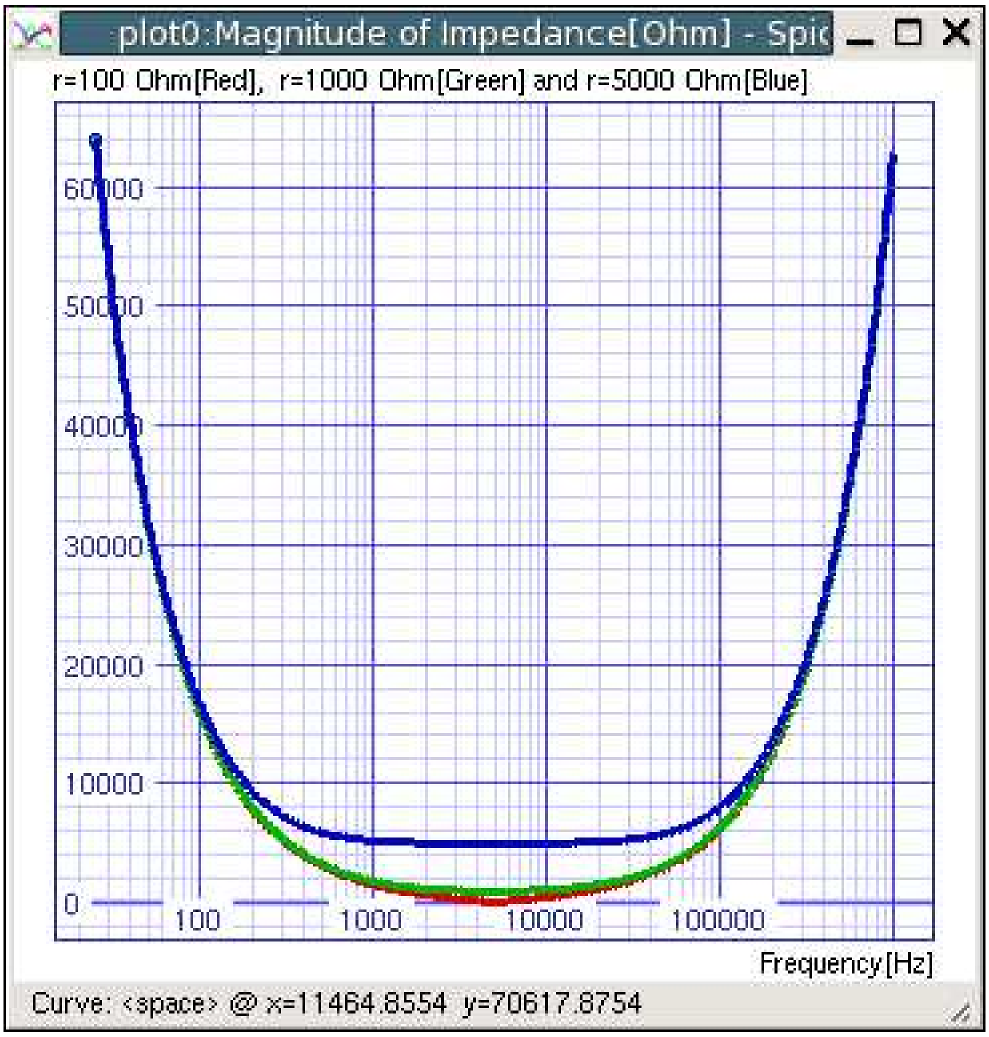
```
ac dec 100 25 1000k
```

the frequency is varied from 25 Hz to 1000 kHz taking 100 points per decade. Logarithmic scale is used for the frequency in the plot. Figure 15 shows the circuit to be simulated. Dummy source is inserted to measure the current. The program is given below:

```
SERIES RESONANCE

* Sinusoidal Voltage Source with dc value = 0
* and rms value = 100V. If dc value is not set you
* will get a warning.
vin 1 0 dc 0 ac 100V

* Dummy Source to measure Current
vx 4 0 dc 0

* Passive Components
* Observe initially we arbitrarily set r=0.1 and later
* change it to 100, 1000, and 5000 Ohm.
r 1 2 0.1
l 2 3 10mH
C 3 4 0.1uF

.control

* This makes SPICE_OPUS output angle in degrees.
set units=degree
```

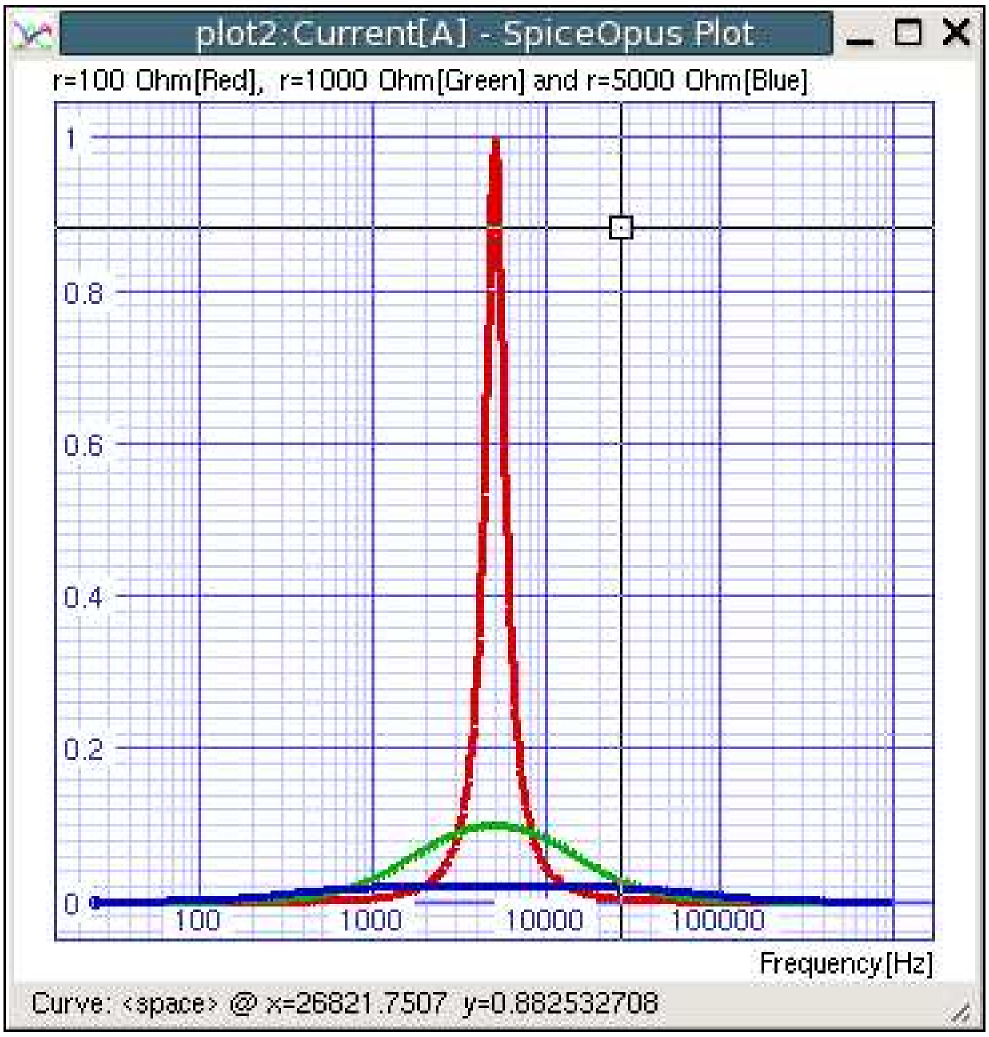
```
* This clears workspace.
destroy all

* conduct ac analysis by varying frequency from
* 25Hz to 1000kHz taking 100 points per decade.
* Check the effect of reducing/increasing number
* of points per decade. Conduct AC analysis for
* each value of r as shown below:

foreach value 100 1000 5000
alter r=$value
ac dec 100 25 1000kHz
end

* This plots |Z|
plot mag(ac1.v(1)/ac1.i(vx))
+ mag(ac2.v(1)/ac2.i(vx)) mag(ac3.v(1)/ac3.i(vx))
+ xlabel 'Frequency[Hz]' ylabel 'r=100 Ohm[Red],
+ r=1000 Ohm[Green] and r=5000 Ohm[Blue]'
+ title 'Magnitude of Impedance[Ohm]'

* This plots Angle of Z
plot ph(ac1.v(1)/ac1.i(vx))
+ ph(ac2.v(1)/ac2.i(vx)) ph(ac3.v(1)/ac3.i(vx))
+ xlabel 'Frequency[Hz]' ylabel 'r=100 Ohm[Red]
+ r=1000 Ohm[Green] and r=5000 Ohm[Blue]'
+ title 'Phase Angle of Impedance[Deg]'

* This plots Current
plot ac1.i(vx) ac2.i(vx) ac3.i(vx)
+ xlabel 'Frequency[Hz]' ylabel 'r=100 Ohm[Red],
+ r=1000 Ohm[Green] and r=5000 Ohm[Blue]'
+ title 'Current[A]'

* Suppose we want to plot magnitude of voltage
* across L and voltage across C versus frequency,
* how to do this? Add instructions here and check
* your results.
.endc

.end
```

Outputs obtained from the program given above are shown in Figures 16, 17, and 18 in the following pages.



Curve: <space> @ x=11464.8554 y=70617.8754

Figure 16: Variation of magnitude of impedance of series *R*-*L*-*C* circuit with frequency.



Curve: <space> @ x=26821.7507 y=0.882532708

Figure 17: Variation of magnitude of current in a series *R*-*L*-*C* circuit with frequency.

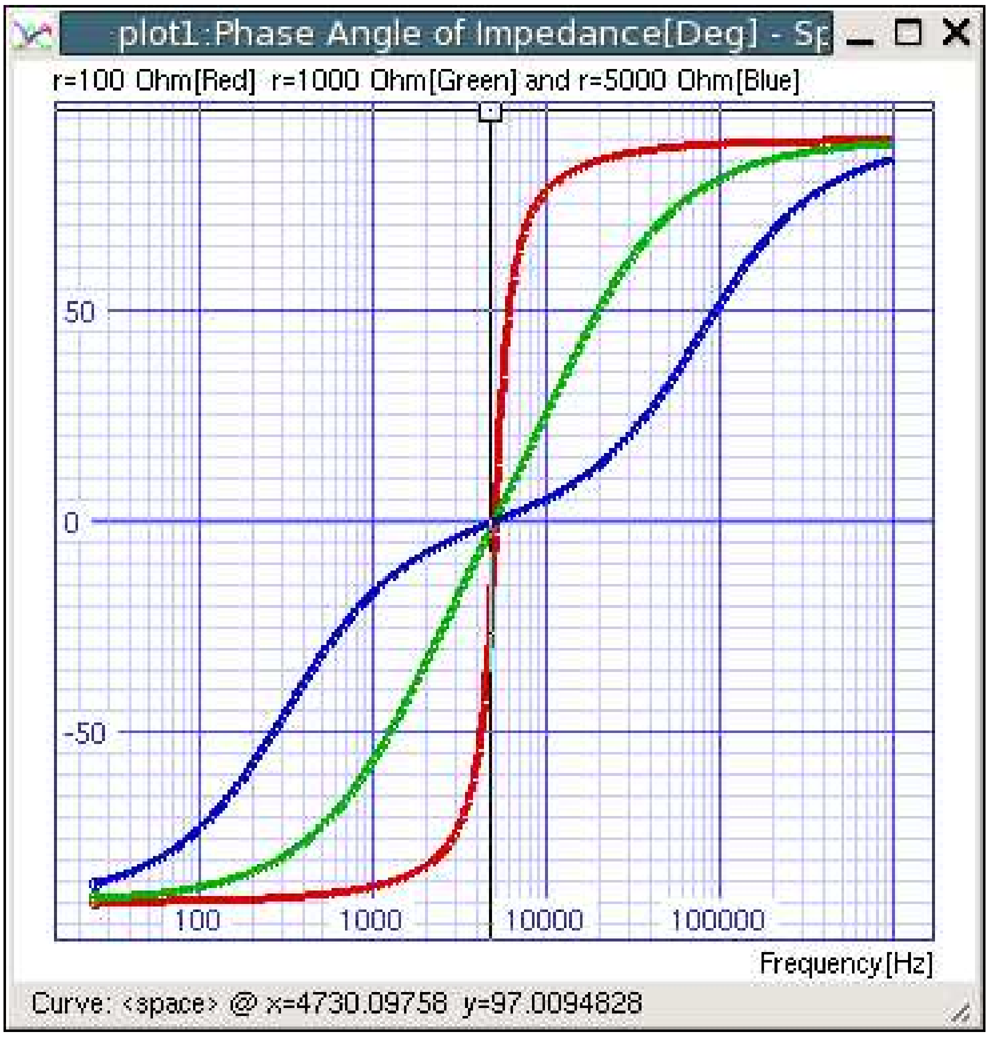


Figure 18: Variation of phase angle of impedance of series *R*-*L*-*C* circuit with frequency.

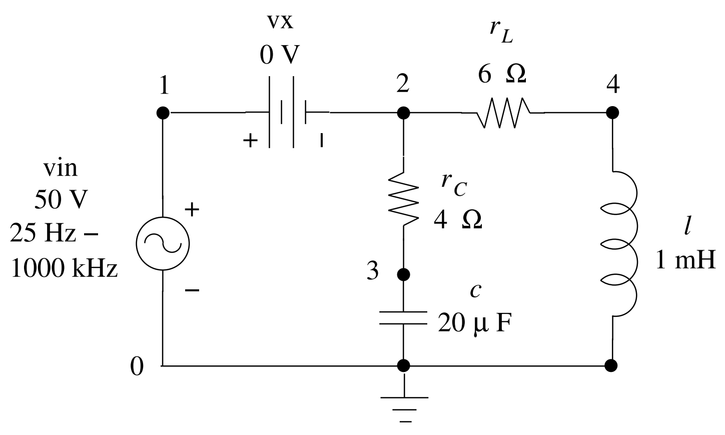


Figure 19: Circuit for studying parallel resonance.

17.6 Parallel Resonance

In the program given below we shall simulate resonance in a parallel $R-L-C$ circuit. The program is similar to that of series resonance.

Problem-6: A series *R-L* circuit with $R_L = 6\Omega$ and $L = 1$ mH, is in parallel with a series *R-C* circuit with $R_C = 4\,\Omega$ and $C = 20\mu$F. This parallel combination is excited by a variable frequency sinusoidal source of R.M.S. value 50 V. Plot the magnitude and phase angle of impedance of this parallel combination versus frequency of the source as the frequency is varied from 25 Hz to 1000 kHz. Measure resonant frequency and cross check this by theoretical calculation. Repeat simulation with $R_L = R_C = \sqrt{L/C}$. What is your conclusion ?

Figure 19 shows the circuit to be simulated. The program is given below:

```
PARALLEL RESONANCE

* Sinusoidal Voltage Source with dc value = 0 and rms
* value = 100V If dc value is not set you will get a warning.
vin 1 0 dc 0 ac 50V

* Dummy Source to measure Current
vx 1 2 dc 0

* Passive Components
rc 2 3 4
rl 2 4 6
c 3 0 20uF
l 4 0 1mH

.control
* This makes SPICE_OPUS output angle in degrees.
```

```
set units=degree

* This cleans the workspace.
destroy all

* conduct ac analysis by varying frequency from
* 25Hz to 1000kHz taking 100 points per decade.

* Check the effect of reducing/increasing number
* of points per decade.
* Conduct AC analysis for each value of r as shown below:

ac dec 100 25 1000kHz
alter rl=7.0711
alter rc=7.0711
ac dec 100 25 1000kHz

* This plots |Z|
plot mag(ac1.v(1)/ac1.i(vx)) mag(ac2.v(1)/ac2.i(vx))
+ xlabel 'Frequency[Hz]'
+ ylabel 'Red: rL=6 Ohm and rC=4 Ohm, Green: rL=rC=7.0711 Ohm'
+ title 'Magnitude of Impedance[Ohm]'

* This plots Angle of Z
plot ph(ac1.v(1)/ac1.i(vx)) ph(ac2.v(1)/ac2.i(vx))
+ xlabel 'Frequency[Hz]'
+ ylabel 'Red: rL=6 Ohm and rC=4 Ohm, Green: rL=rC=7.0711 Ohm'
+ title 'Phase Angle[Deg]'

* This plots Current
plot ac1.i(vx) ac2.i(vx)
+ xlabel 'Frequency[Hz]'
+ ylabel 'Red: rL=6 Ohm and rC=4 Ohm, Green: rL=rC=7.0711 Ohm'
+ title 'Current'
.endc

.end
```

The outputs obtained from the program given above are shown in Figures 20 , 21, and 22. From this we can see that when $R_L = R_C = \sqrt{L/C}$ the circuit is resonant at all frequencies.

17.7 R-C Coupled Amplifier

In the following circuit file we simulate an *R-C* coupled amplifier. We obtain the frequency response and time response of the amplifier when a sinusoidal signal is applied at the input. Figure 23 shows the amplifier circuit to be simulated. To obtain the time response, the command is

```
tran Tstep Tstop [ Tstart [ Tmax ] ] [ UIC ]
```

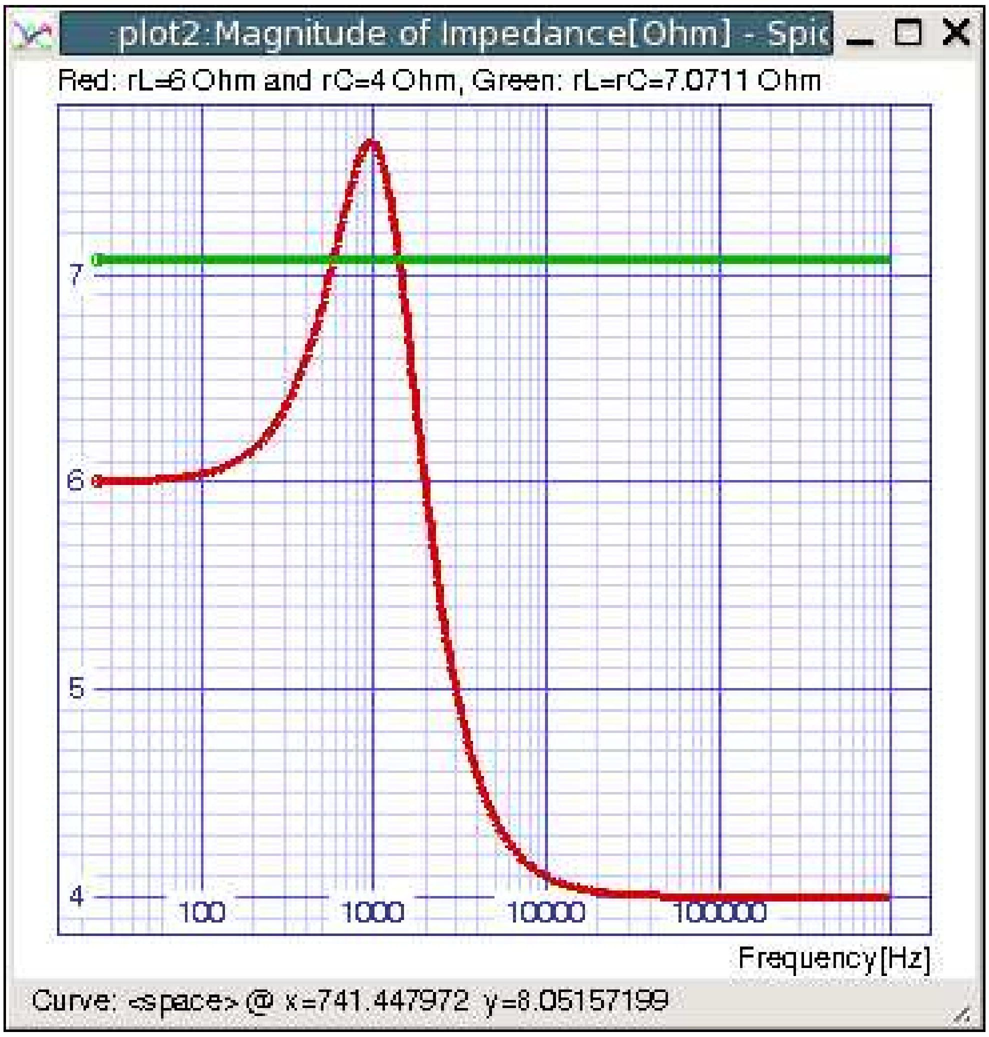


Figure 20: Variation of magnitude of impedance of parallel RLC circuit with frequency.

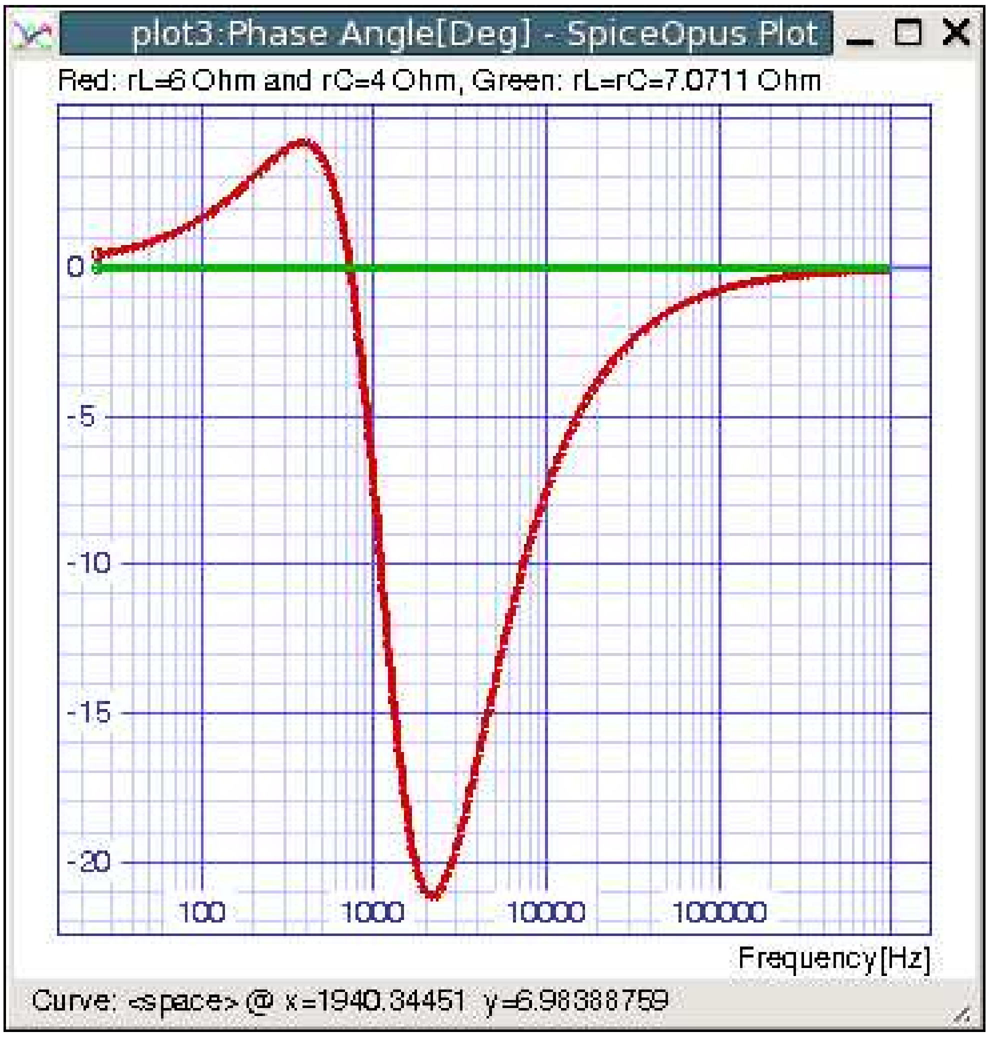


Figure 21: Variation of phase angle of impedance of parallel RLC circuit with frequency.

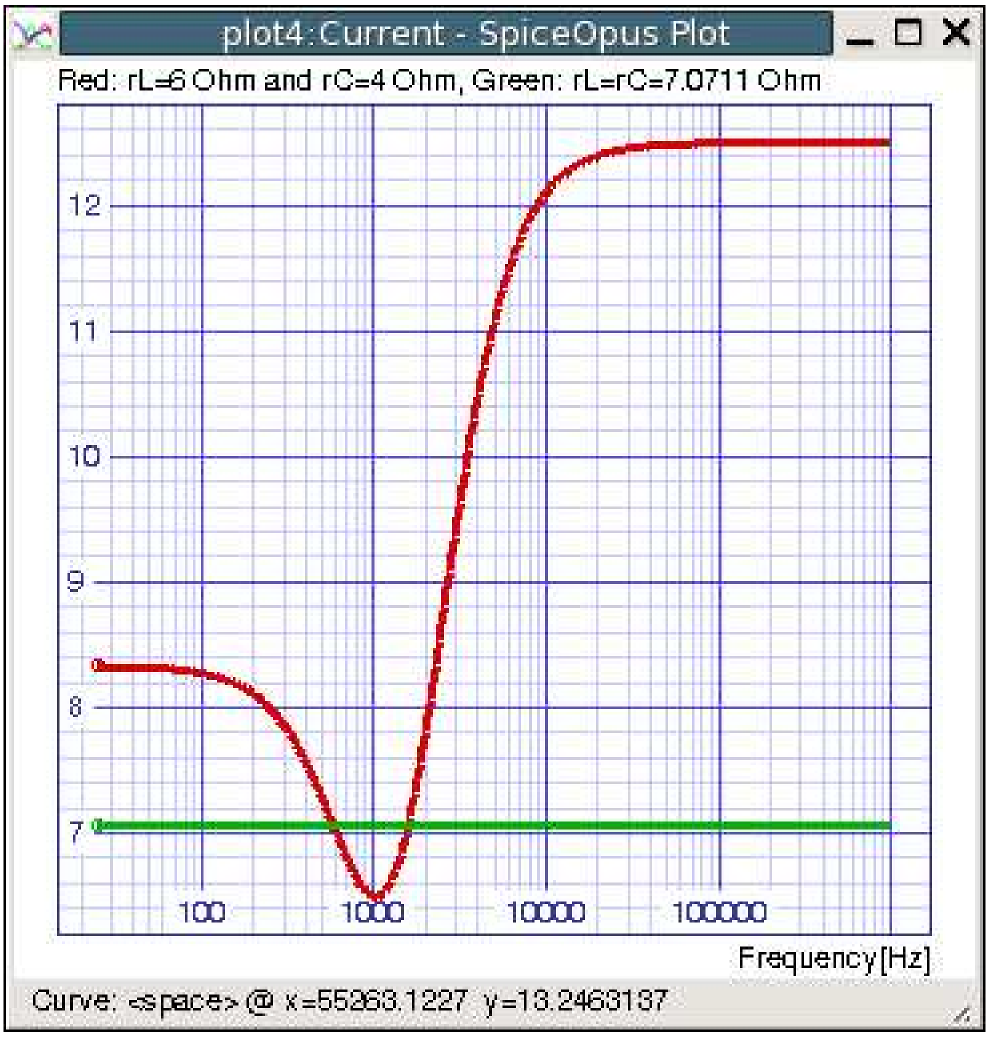


Figure 22: Variation of current drawn by parallel RLC circuit with frequency.

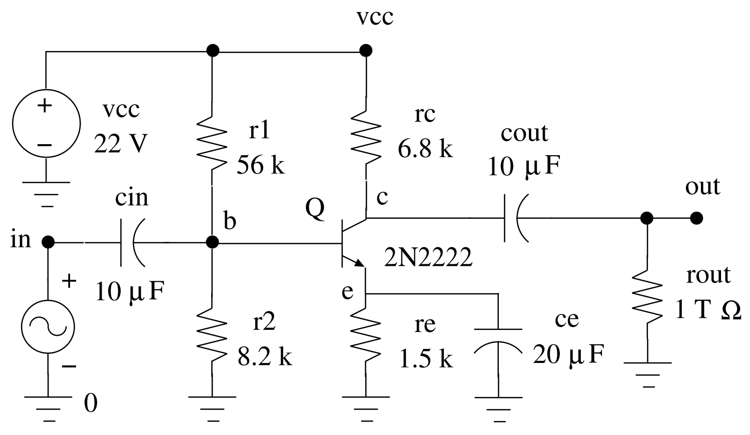


Figure 23: *R-C* coupled amplifier.

`TSTEP` is the time increment. `TSTOP` is the final time, and `TSTART` is the initial time. If `TSTART` is omitted, it is assumed to be zero. The transient analysis always begins at time zero. In the interval `<zero, TSTART>`, the circuit is analyzed (to reach a steady state), but no outputs are stored. In the interval `<TSTART, TSTOP>`, the circuit is analyzed and outputs are stored. `TMAX` is the maximum step-size that SPICE uses; for default, the program chooses either `TSTEP` or `(TSTOP-TSTART)/50.0`, whichever is smaller. `TMAX` is useful when one wishes to guarantee a computing interval which is smaller than the printer increment, `TSTEP`.

`UIC` (use initial conditions) is an optional keyword which indicates that the user does not want SPICE to solve for the quiescent operating point before beginning the transient analysis. If this keyword is specified, SPICE uses the values specified using `IC=...` on the various elements as the initial transient condition and proceeds with the analysis. If the `.IC` control line has been specified, then the node voltages on the `.IC` line are used to compute the initial conditions for the devices. Look at the description on the `.IC` control line for its interpretation when `UIC` is not specified. In most of the cases, we use `tran` command in the following format:

```
tran Tstep Tstop
```

If `tran` command has to be executed, at least one of the sources in the circuit should have time domain specification. In the following simulation, we are obtaining both frequency response and time response of the circuit. So, we specify both time-domain and frequency domain specification for the input signal source. For example, we write

```
vin in 0 dc 0 ac 1mv 0 sin(0 1mv 10kHz)
```

Here `ac 1mv 0` is the frequency domain specification which is required by `ac` command and `sin(0 1mv 10kHz)` is the time-domain specification which is required by `tran` command. An NPN BJT is specified in the circuit file by two statements having the format given below:

```
Q???????  collector_node base_node emitter_node MODELNAME
.model MODELNAME NPN(Parameter1=x Parameter2=x ...)
```

Here `???????` is the label given to a particular transistor in the circuit, and `MODELNAME` should be same in both the statements. For each type number of transistor, its `.model` statement should appear only once in the circuit file. Usually, `.model` statement is provided for transistors by manufacturers at their web sites. This statement contains parameters of the transistor which depend upon the construction. For example, the `.model` statement of the NPN BJT q2N2222 taken from a manufacturer's web site is given below:

```
.model q2N2222 NPN(Is=14.34f Xti=3 Eg=1.11 Vaf=74.03 Bf=255.9 Ne=1.307
+ Ise=14.34f Ikf=.2847 Xtb=1.5 Br=6.092 Nc=2 Isc=0 Ikr=0 Rc=1
+ Cjc=7.306p Mjc=.3416 Vjc=.75 Fc=.5 Cje=22.01p Mje=.377 Vje=.75
+ Tr=46.91n Tf=411.1p Itf=.6 Vtf=1.7 Xtf=3 Rb=10)
```

Similarly we can also specify PNP BJT. At this stage, it is unnecessary to go in to the meaning of each of the parameters given in the statement above. Obviously typing such a lengthy `.model` statement is cumbersome. So, all such `.model` statements for different devices can be collected together and stored in a library file. You can give an appropriate name to that file, say `parts.lib`. This file can be stored in a directory which is accessed by Spice Opus (Usually `/usr/lib/spice/lib/scripts` in Linux machines. In Windows machines `c:\SpiceOpus\lib\scripts`). Then, the contents of this library file can be included in the circuit file by writing the statement

```
.include parts.lib
```

in the circuit file. In the program give below, it is assumed that such a library file is available. If it is not available, then, you have to create it by collecting the SPICE models from the internet. The circuit file for simulating a BJT amplifier is given below:

```
BJT C-E AMPLIFIER USING VOLTAGE DIVIDER BIAS

* Biasing Supply
vcc vcc 0 dc 22

* Input Signal Source
vin in 0 dc 0 ac 1mv 0 sin(0 1mv 10kHz)

* Biasing Resistors
r1 vcc b 56k
r2 b   0 8.2k
re e   0 1.5k
rc vcc c 6.8k

* Coupling Capacitors
cin in b 10uf
cout c out 10uf

* Emitter Capacitor
ce e 0 20uf

* Load Resistance
rout out 0 1e12

* Transistor
q c b e q2n2222

* This library file contains model of q2n2222
.include parts.lib

.control
set units=degree
destroy all

* This finds time response.
tran 1us 500us
plot v(out) (v(in)*100) vs (time*1000)
+ xlabel 't[ms]'
+ ylabel 'v(out)[Red], v(in)*100[Green]'
+ title 'Time Response of BJT C-E Amplifier'
```

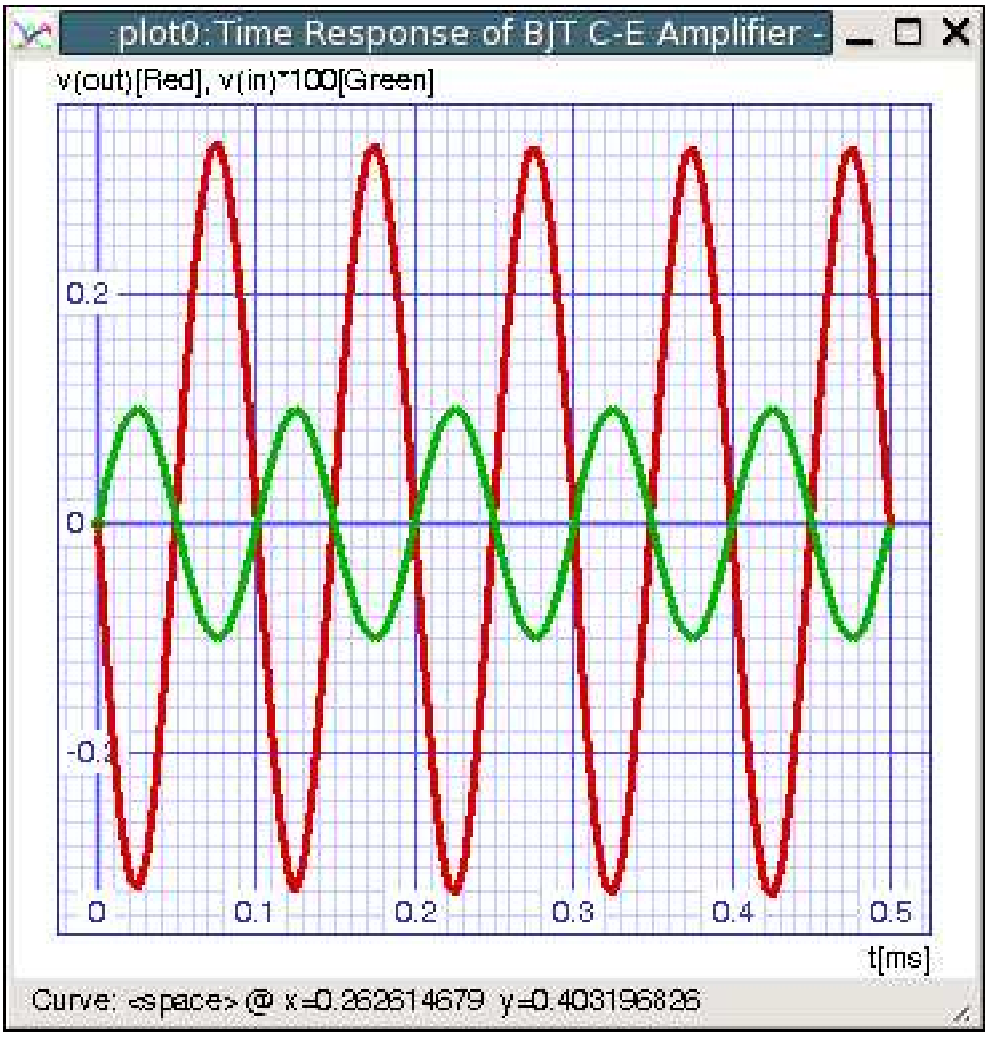
```
* This finds frequency response.
ac dec 10 10Hz 100MegHz

let gain=v(out)/v(in)

* Observe how decibels can be calculated.
plot db(gain)
+ xlabel 'f[Hz]'
+ ylabel 'Gain in dB'
+ title 'Frequency Response of BJT C-E Amplifier'

plot ph(gain)
+ xlabel 'f[Hz]'
+ ylabel 'Angle in Degrees'
+ title 'Phase Angle of Gain'
.endc

.end

* How is v(out) affected by variations in the following:
* (a) vcc; (b) rout; (c) r1 or r2 ?
* How is v(out) affected if cout is shorted ?
```

Results of simulation of BJT amplifier are given in Figures 24, 25, and 26.

17.8 Inverting and Non-inverting Amplifiers using Op-amp

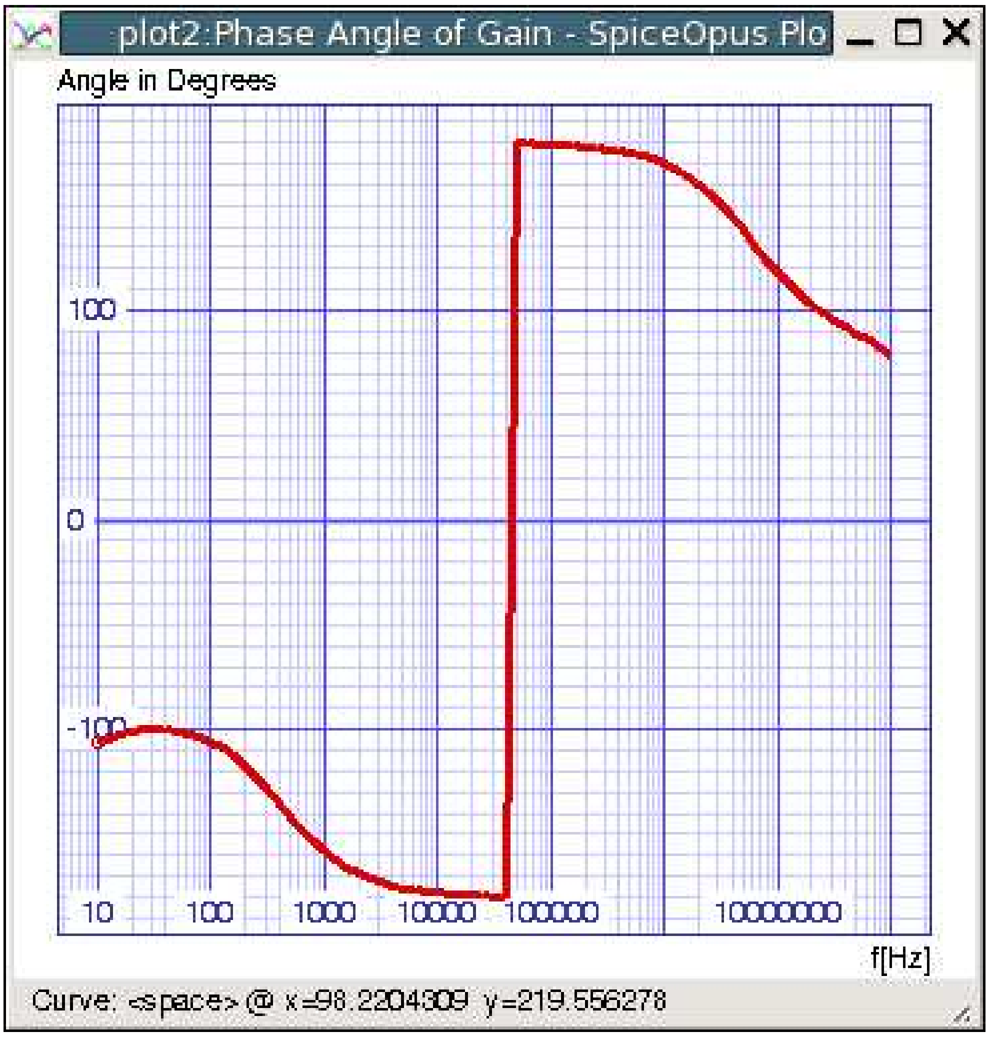
In the program given below, we simulate an inverting and non-inverting amplifier which uses op-amp μA741. This op-amp has a complex circuit in side. In SPICE we can define any circuit as a sub-circuit. For example, the op-amp μA741 can be modeled as a sub-circuit as shown below:

```
*//////////////////////////////////////////////////////////
* uA741/A/C/E OPERATIONAL AMPLIFIER MACRO-MODEL
*//////////////////////////////////////////////////////////
*
* connections:      non-inverting input
*                   |   inverting input
*                   |   |   positive power supply
*                   |   |   |   negative power supply
*                   |   |   |   |   output
*                   |   |   |   |   |
*                   |   |   |   |   |
.SUBCKT uA741       1   2  99  50  28
*
* Features:
* Improved performance over industry standards
* Plug-in replacement for LM709,LM201,MC1439,748
```


plot0: Time Response of BJT C-E Amplifier -
v(out)[Red], v(in)*100[Green]
0.2
0
-0.2
0
0.1
0.2
0.3
0.4
0.5
t[ms]

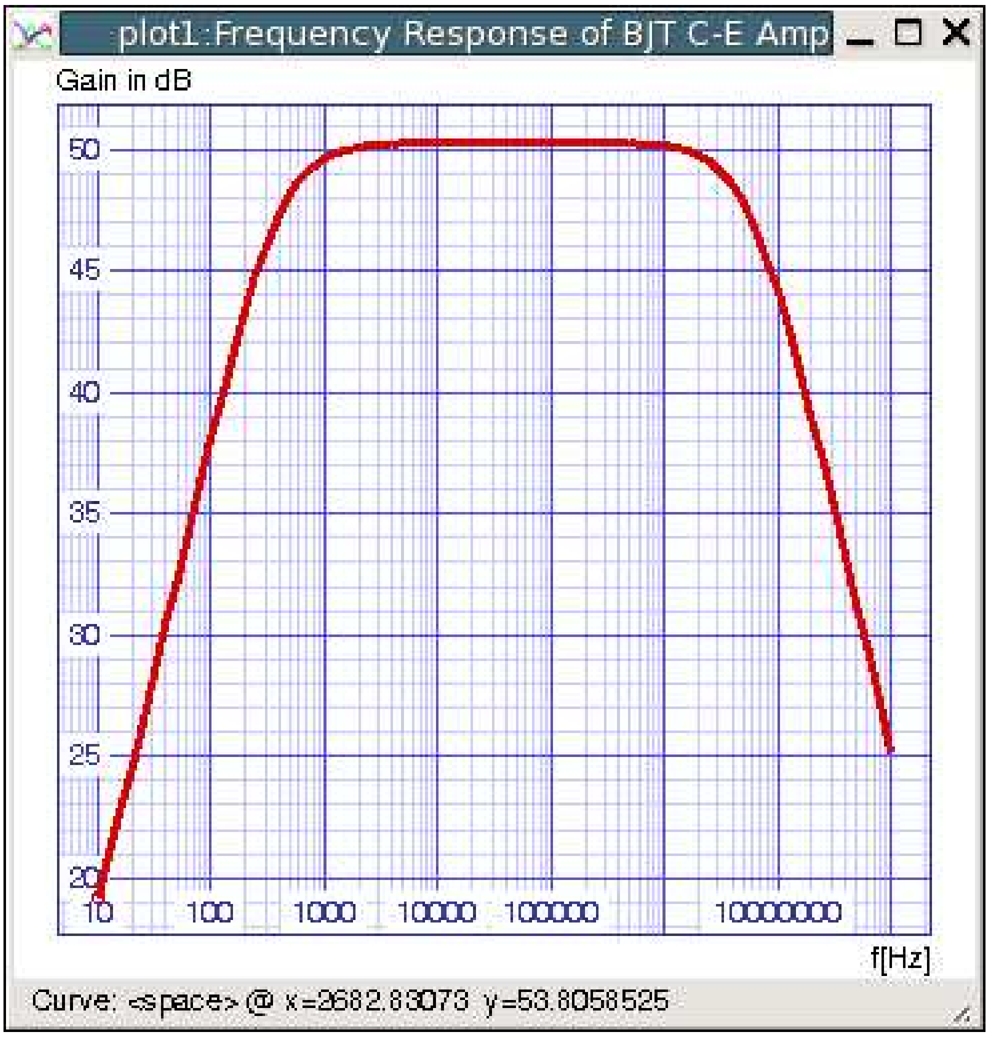
Curve: <space> @ x=0.262614679 y=0.403196826

Figure 24: Time response of BJT amplifier.



Curve: <space> @ x=98.2204309 y=219.556278

Figure 25: Phase angle of gain of BJT amplifier plotted versus frequency.



```
Curve: <space> @ x=2682.83073 y=53.8058525
```

Figure 26: Decibel gain of BJT amplifier plotted versus frequency.

```
* Input and output overload protection
*
****************INPUT STAGE**************
*
IOS 2 1 20N
*^Input offset current
R1 1 3 250K
R2 3 2 250K
I1 4 50 100U
R3 5 99 517
R4 6 99 517
Q1 5 2 4 QX
Q2 6 7 4 QX
*Fp2=2.55 MHz
C4 5 6 60.3614P
*
***********COMMON MODE EFFECT***********
*
I2 99 50 1.6MA
*^Quiescent supply current
EOS 7 1 POLY(1) 16 49 1E-3 1
*Input offset voltage.^
R8 99 49 40K
R9 49 50 40K
*
*********OUTPUT VOLTAGE LIMITING********
V2 99 8 1.63
D1 9 8 DX
D2 10 9 DX
V3 10 50 1.63
*
**************SECOND STAGE**************
*
EH 99 98 99 49 1
G1 98 9 5 6 2.1E-3
*Fp1=5 Hz
R5 98 9 95.493MEG
C3 98 9 333.33P
*
***************POLE STAGE***************
*
*Fp=30 MHz
G3 98 15 9 49 1E-6
R12 98 15 1MEG
C5 98 15 5.3052E-15
*
*********COMMON-MODE ZERO STAGE*********
```

```
*
*Fpcm=300 Hz
G4 98 16 3 49 3.1623E-8
L2 98 17 530.5M
R13 17 16 1K
*
**************OUTPUT STAGE**************
*
F6 50 99 POLY(1) V6 450U 1
E1 99 23 99 15 1
R16 24 23 25
D5 26 24 DX
V6 26 22 0.65V
R17 23 25 25
D6 25 27 DX
V7 22 27 0.65V
V5 22 21 0.18V
D4 21 15 DX
V4 20 22 0.18V
D3 15 20 DX
L3 22 28 100P
RL3 22 28 100K
*
***************MODELS USED**************
*
.MODEL DX D(IS=1E-15)
.MODEL QX NPN(BF=625)
*
.ENDS
```

The above SPICE model has been copied from one of the manufacturer's web site. At this stage, it is not required to understand how the model shown above is derived. Such models are available freely on the internet for most of the commonly used op-amps. We can copy this sub-circuit model in to our `parts.lib` file. After this we can simply call this sub-circuit from our circuit file by the two statements given below:

```
x???????  Input+ Input- Supply+ Supply- Output uA741
.include parts.lib
```

Here `x` refers to the sub-circuit, `???????` can be any label, and nodes must be specified in the same order in which they are given in the sub-circuit definition in `parts.lib` file. Figure 27 shows the inverting amplifier circuit to be simulated and the circuit file is given below:

```
INVERTING AND NONINVERTING AMPLIFIER USING OPAMP 741

vin Input 0 dc 0 ac 0.5v 0 sin(0 0.5v 1kHz)
vp 7 0 dc 15
vn 0 4 dc 15
```

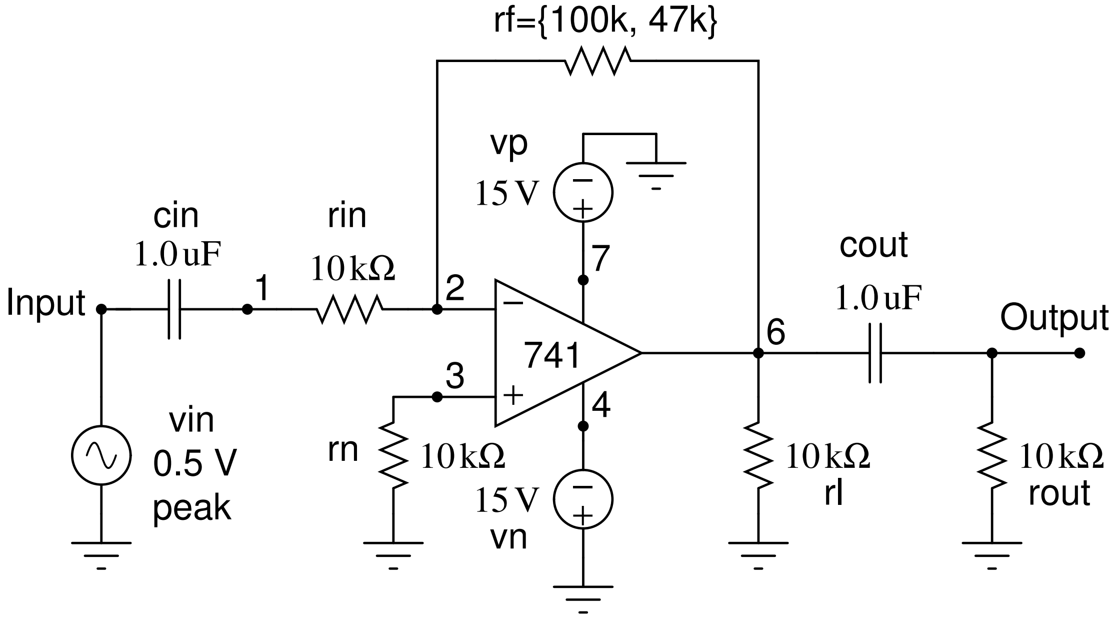


Figure 27: Inverting amplifier using op-amp μA741.

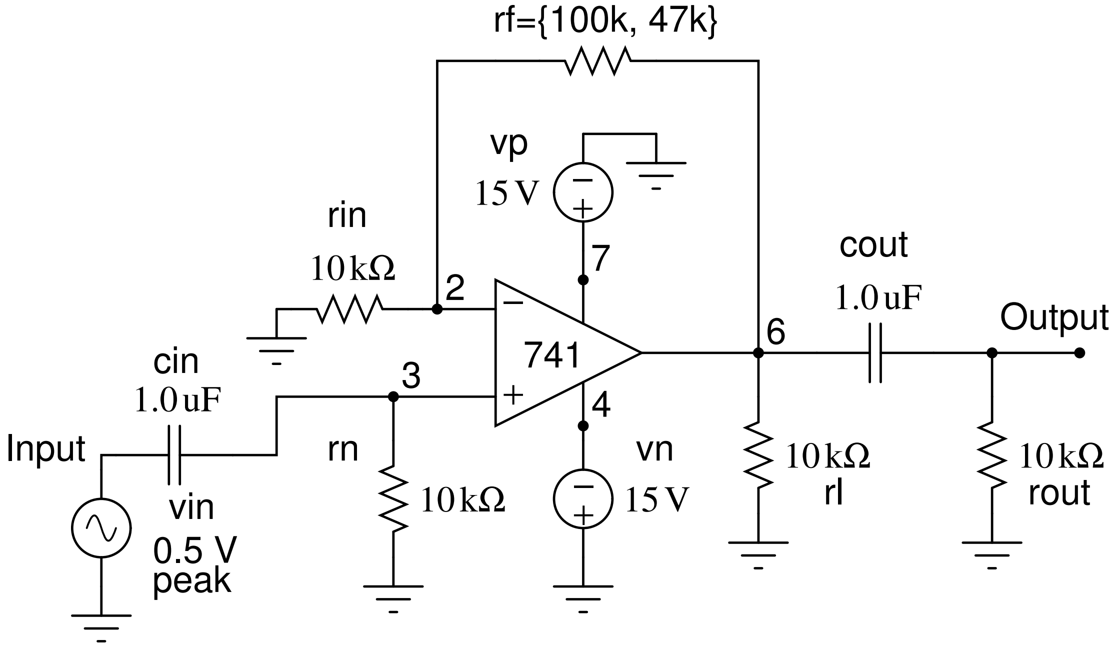


Figure 28: Non-inverting amplifier using op-amp μA741.

```
rin 1 2 10k
* Comment out the above line and uncomment the
* following line for non-inverting amplifier.
* rin 0 2 10k
rf  2 6 100k
rn  3 0 10k
rl  6 0 10k
rout Output 0 10k

cin Input 1 1uF
* Comment out the above line and uncomment the
* following line for non-inverting amplifier.
* cin Input 3 1uF
cout 6 Output 1uF

x 3 2 7 4 6 ua741

.include parts.lib

.control
destroy all
tran 0.01ms 4ms
alter rf=47k
tran 0.01ms 4ms
plot v(Input) vs (time*1000)
+ (tran1.v(Output)) vs tran1.time*1000
+ (tran2.v(Output)) vs tran2.time*1000
+ xlabel 'Time[ms]'
+ ylabel 'v(Input), v(Output) with rf=100k[Green], 47k[Blue]'
+ title 'Response of Inverting Amplifier having uA741 OpAmp'

destroy all

ac dec 10 10Hz 1MegHz
alter rf=100k
ac dec 10 10Hz 1MegHz

plot mag(ac1.v(Output))/mag(ac1.v(Input)) vs ac1.frequency
+ mag(ac2.v(Output))/mag(ac2.v(Input)) vs ac2.frequency
+ xlabel 'Frequency[Hz]'
+ ylabel 'Gain with rf=47k[Red], 100k[Green]'
+ title 'Frequency Response of Inverting Amplifier having uA741 OpAmp'

.endc

.end
```

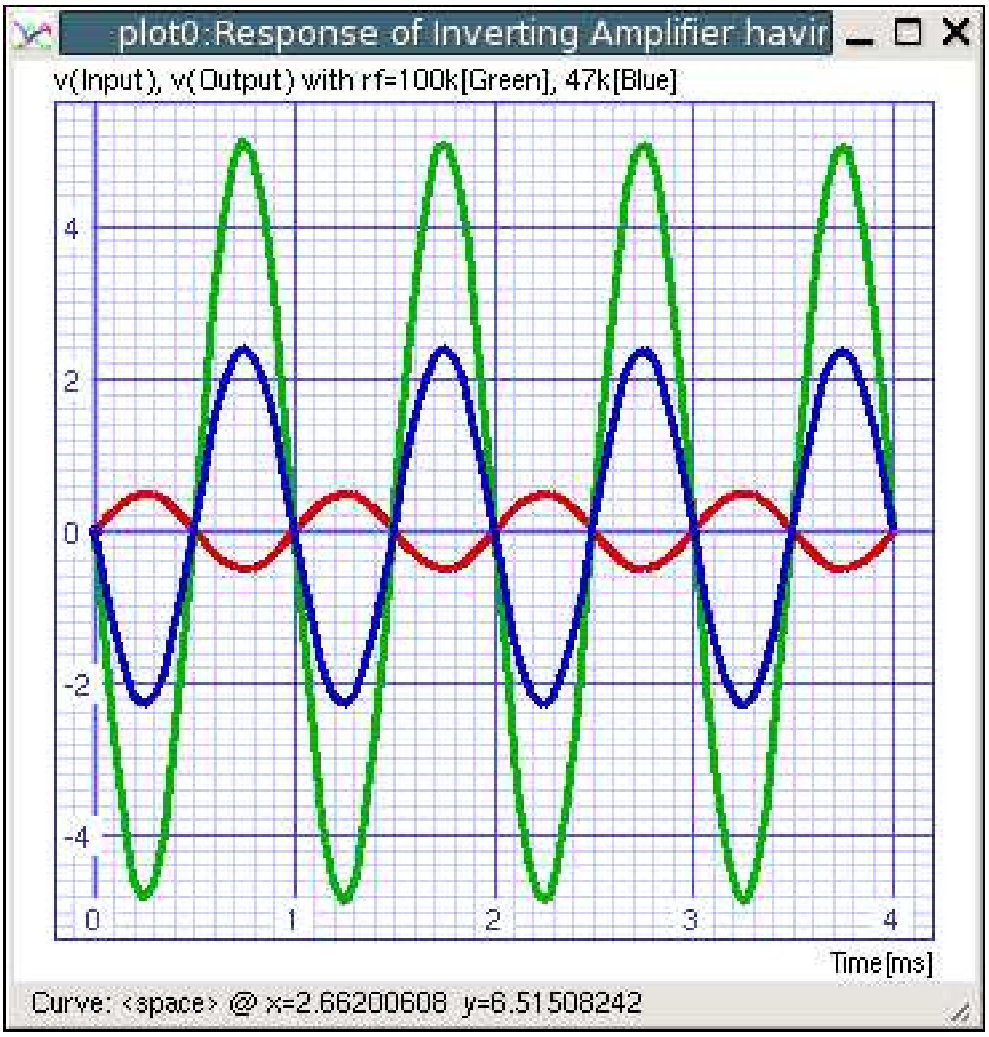


Figure 29: Time response of inverting amplifier using op-amp μA741.

The results of simulation are given in Figures 29 and 30. Figure 28 shows the non-inverting amplifier. Change the circuit file given above to simulate this circuit. The results of simulation are shown in Figures 31 and 32.

17.9 Bridge Rectifier

In Figure 33 a bridge rectifier with a step-down transformer and a Π filter is shown. Let us plot the output voltage of this circuit versus time. It is given that the input voltage is $v_s = 169.7 \sin 377t$. So, the duration of one period is $T = (2\pi/377) = 16.67$ ms and the frequency is 60 Hz. Let us plot the output voltage for about 5 cycles. So, the duration to be specified in `tran` command is $5 \times 16.67 \approx 85$ ms. Step down transformer has a ratio of $12:1$. The secondary voltage is therefore given by $v_1 = v_s/12 = 0.083 v_s$. So, the secondary voltage source may be modelled as a voltage controlled voltage source as shown in Figure 34. The circuit file for simulation is given below:

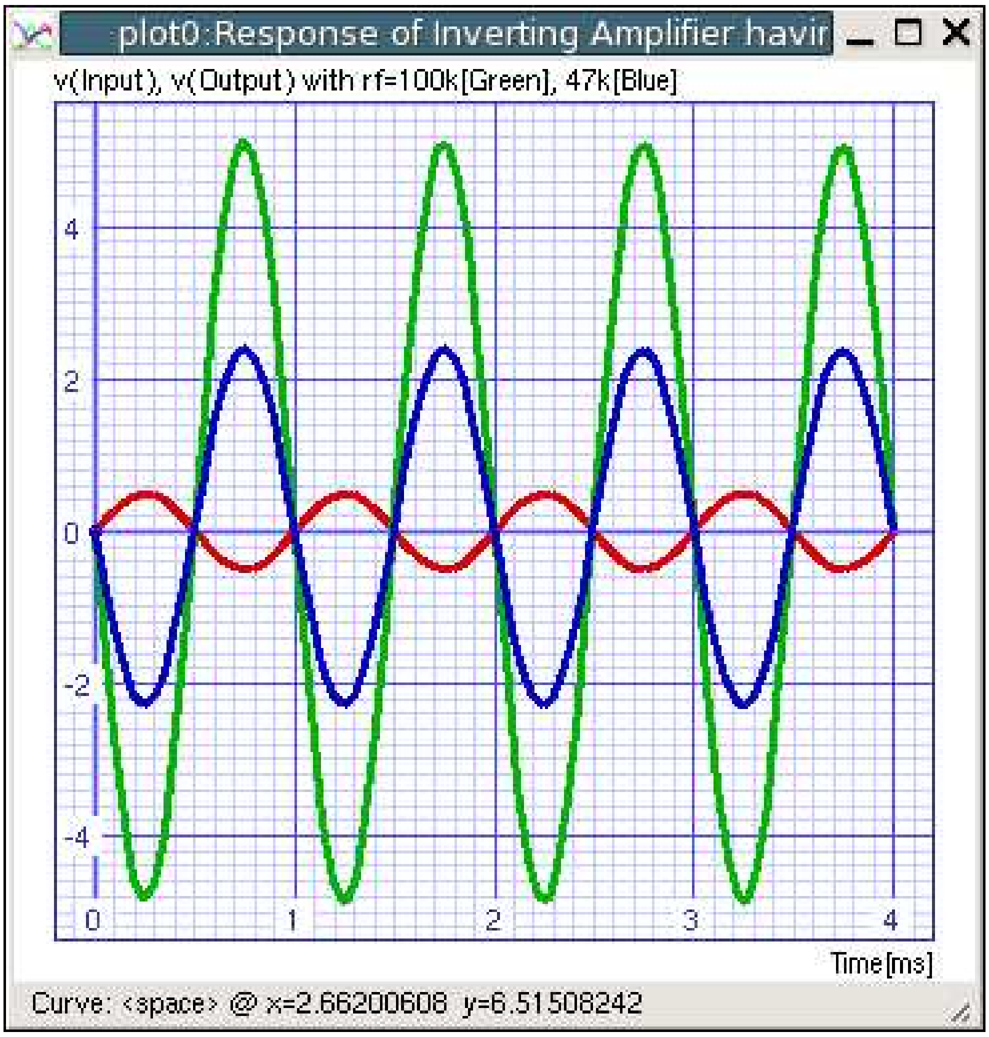
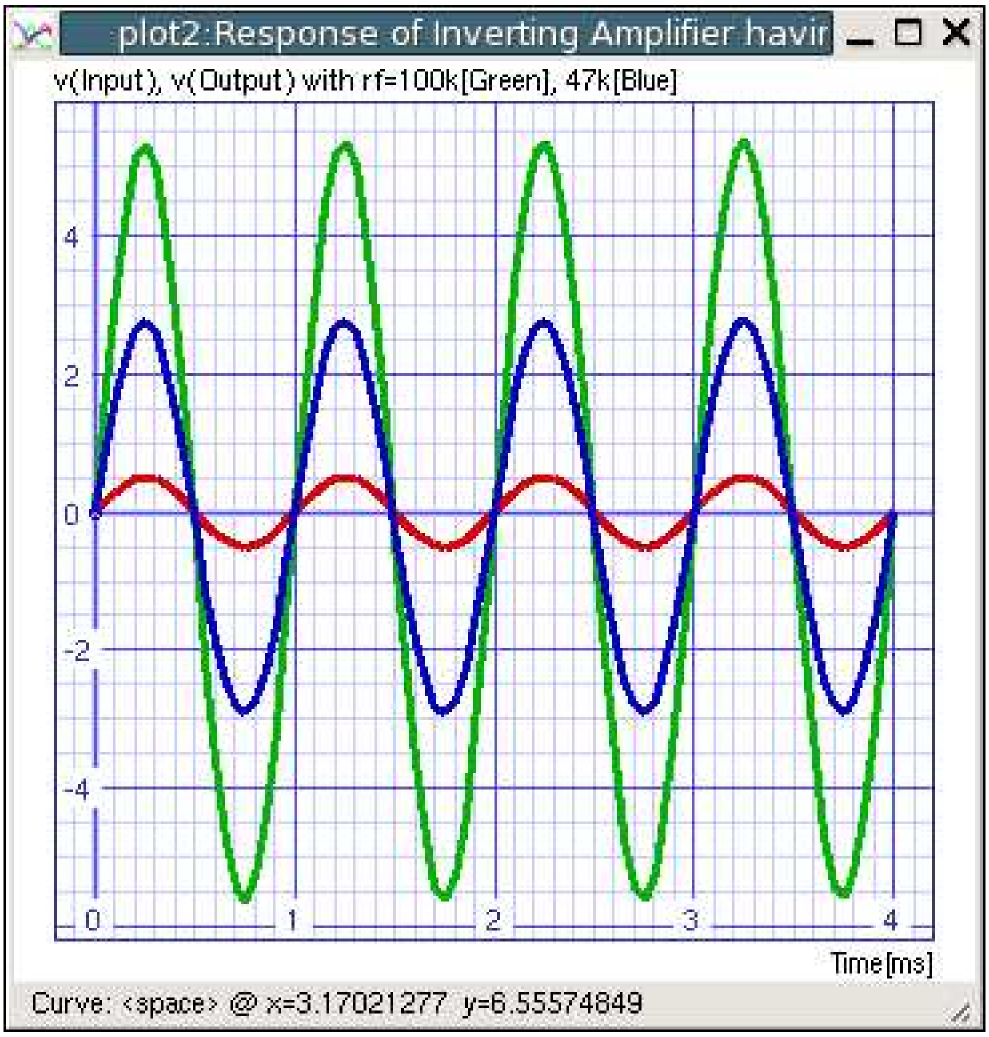
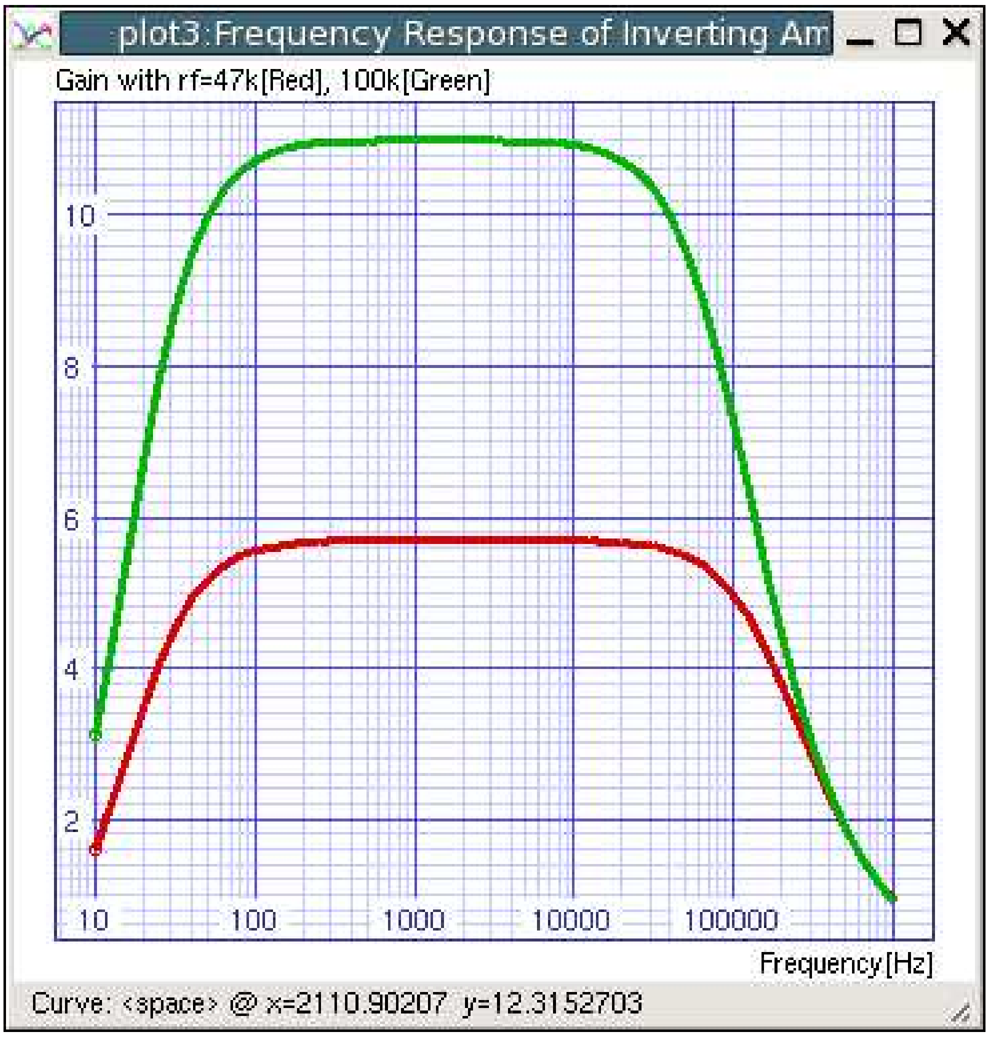


Figure 30: Frequency response of inverting amplifier using op-amp μA741.



Curve: <space> @ x=3.17021277 y=6.55574849

Figure 31: Time response of non-inverting amplifier using op-amp μA741.



Curve: <space> @ x=2110.90207 y=12.3152703

Figure 32: Frequency response of non-inverting amplifier using op-amp μA741.

```
BRIDGE RECTIFIER WITH PI-FILTER

* Sources
* This is the sinusoidal input voltage
* source of peak value 169.7 V @ 60 Hz frequency.
Vs 1 0 dc 0 sin(0 169.7 60)

* This is the secondary source.
* This a voltage controlled voltage source.
* The secondary voltage = (1/12)Vs = 0.083Vs
E1 4 5 1 0 0.0833

* Resistors
R1 3 0 100
R2 1 0 1T

* Diodes
D1 4 2 1N4007
D2 0 5 1N4007
D3 5 2 1N4007
D4 0 4 1N4007

* If the model statement given below is
* copied into parts.lib file, then, this
* can be replaced by .include parts.lib
.MODEL  1N4007  D(IS = 3.872E-09 RS = 1.66E-02
+ N = 1.776 XTI = 3.0 EG = 1.110 CJO = 1.519E-11
+ M = 0.3554 VJ = 0.5928 FC = 0.5 BV = 1000.0 IBV = 1.0E-03)

* Filter
C1 2 0 10u
C2 3 0 100u
L1 2 3 0.1m

.control
destroy all
tran 0.01ms 85ms
plot v(3) vs (time*1000)
+ xlabel 'Time[ms]'
+ ylabel 'V(out)'
+ title 'Time Response of Bridge Rectifier'
.endc

.end
```

The output obtained from simulation of bridge rectifier circuit is shown in Figure 35.

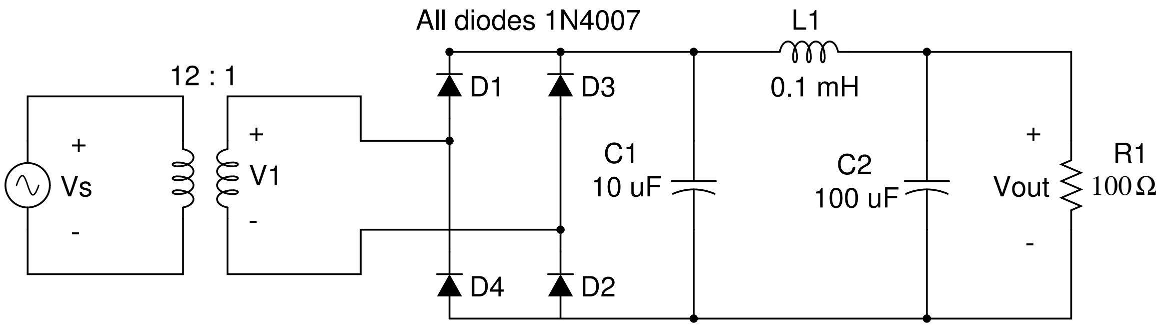


Figure 33: Bridge rectifier circuit with Π filter.

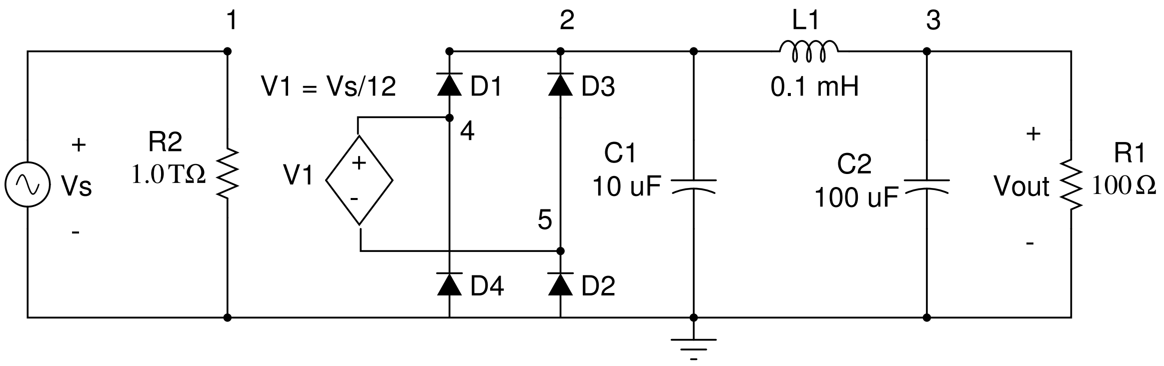
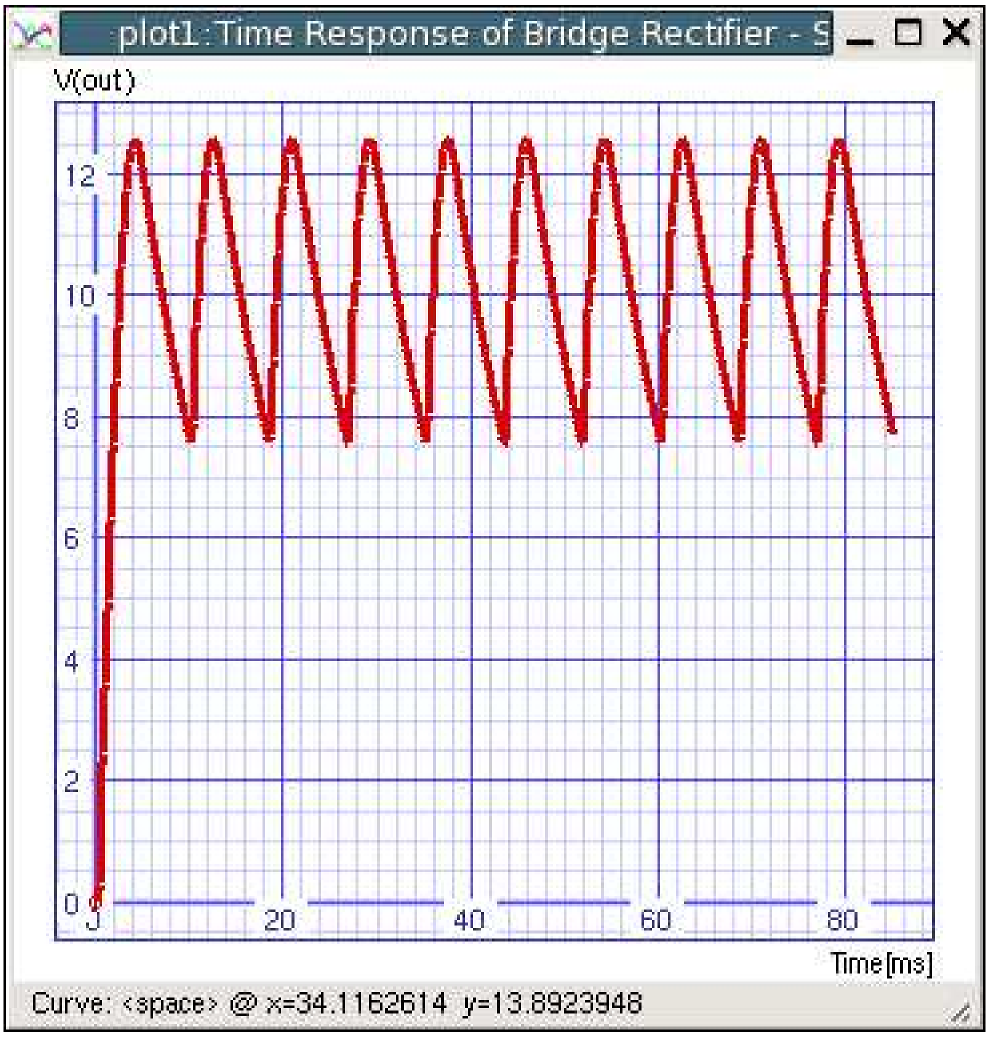


Figure 34: Bridge rectifier circuit modified for simulation.



Curve: <space> @ x=34.1162614 y=13.8923948

Figure 35: Output of bridge rectifier circuit with Π filter.

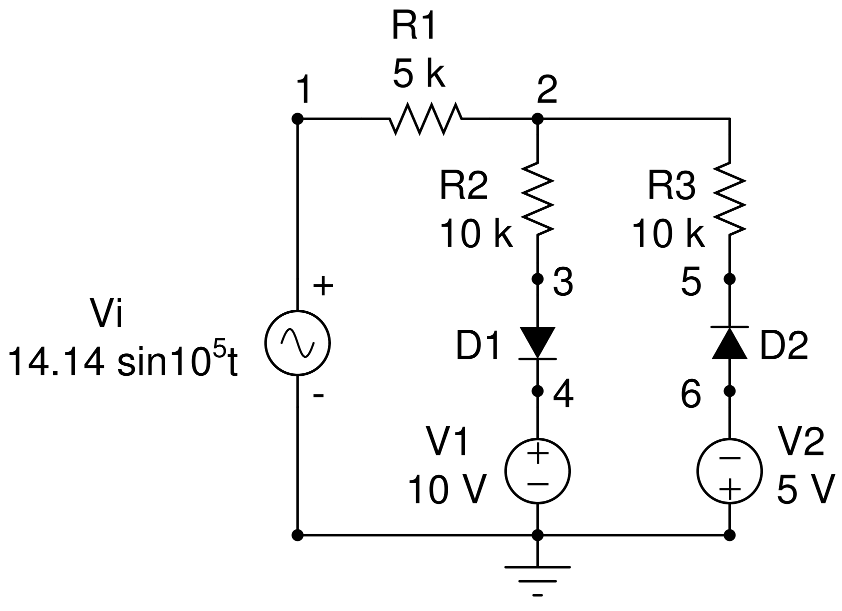


Figure 36: Diode clipper circuit.

17.10 Diode Clipping Circuit

Figure 36 shows a diode clipper circuit. It is required to plot a graph of v_i and v_o versus time and also to plot its transfer characteristics which is a plot of v_o versus v_i. The circuit file is given below:

```
DIODE CLIPPER

* Input Source
Vi 1 0 dc 0 sin(0 14.14 1e5)

* DC Sources
V1 4 0 dc 10
V2 0 6 dc 5

* Resistors
R1 1 2 5k
R2 2 3 10k
R3 2 5 10k

* Diodes
D1 3 4 1N4007
D2 6 5 1N4007
.include parts.lib

.control
destroy all
tran 0.01u 12u
```

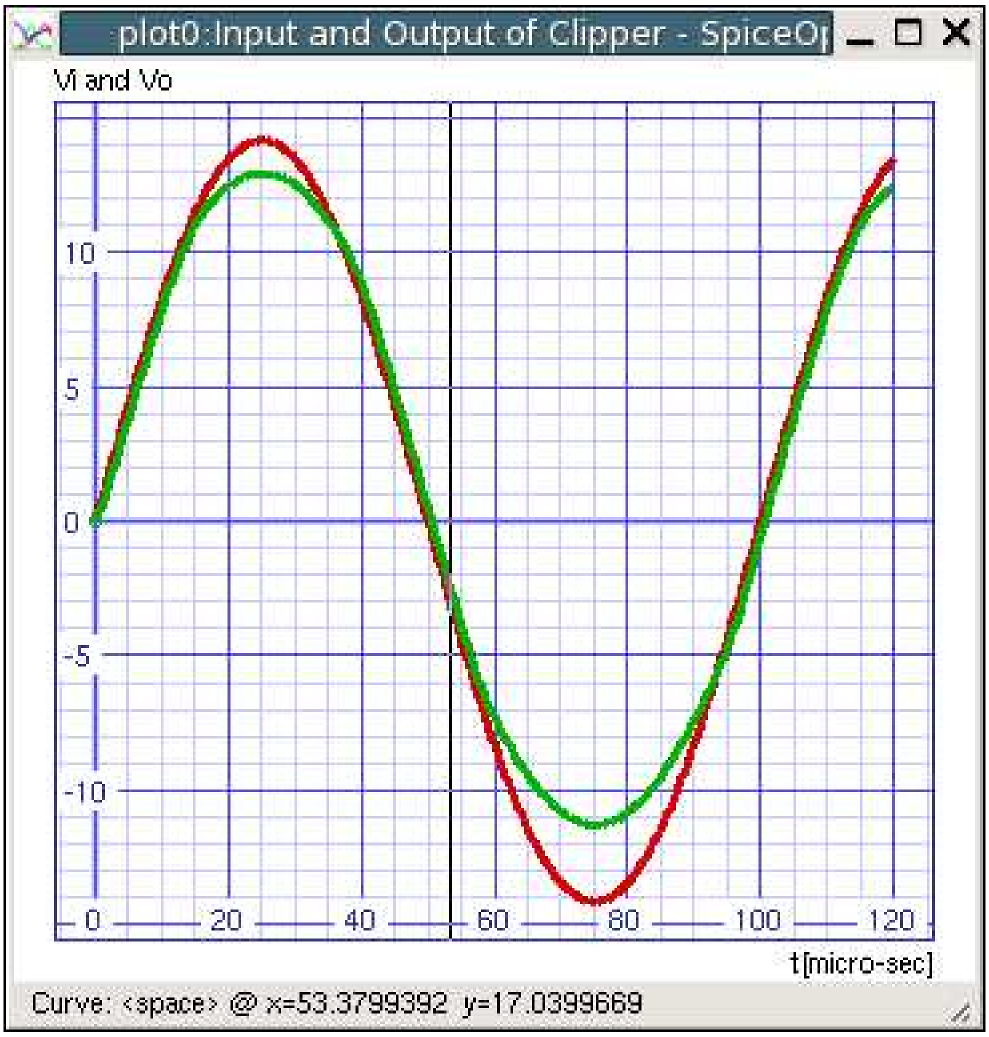
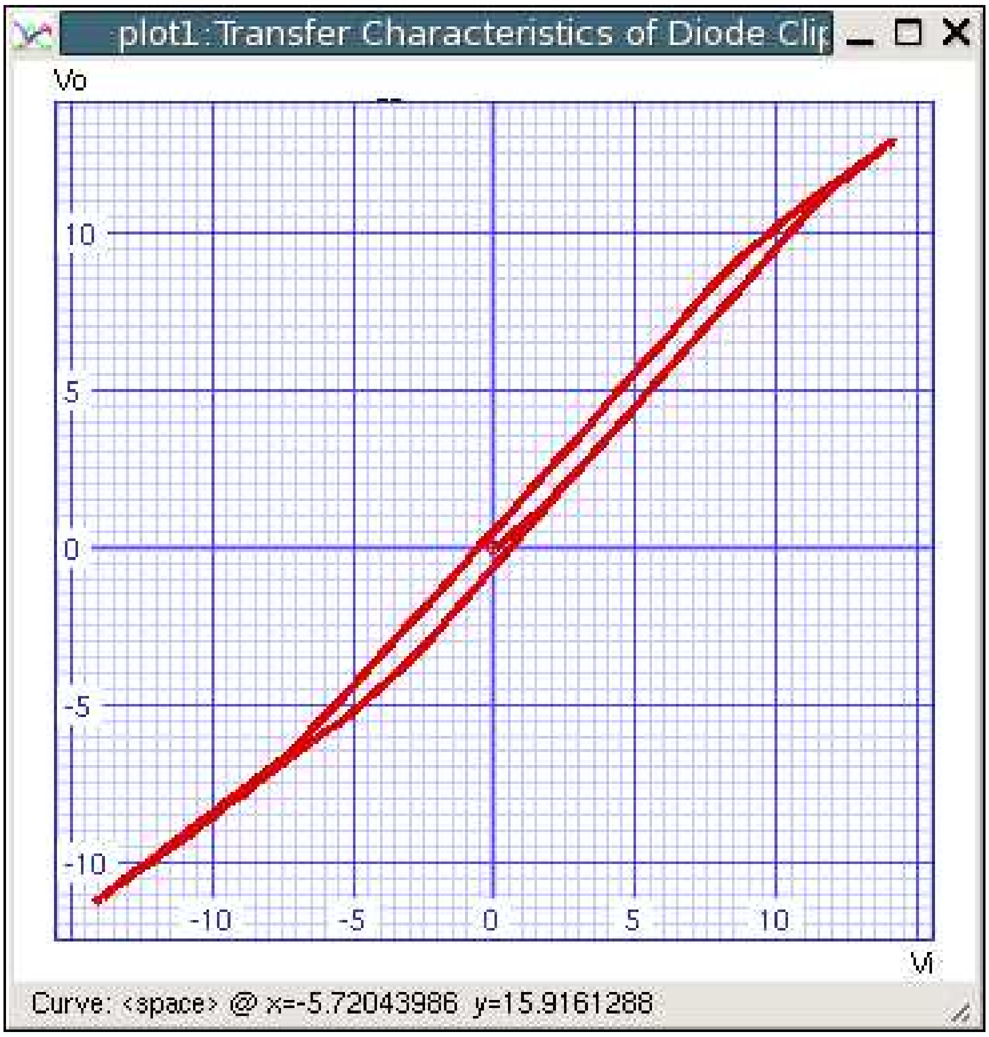


Figure 37: Input and output voltages of diode clipper.

```
plot V(1) V(2) vs (time*10e6)
+ xlabel t[micro-sec]
+ ylabel 'Vi and Vo'
+ title 'Input and Output of Clipper'

plot V(2) vs V(1)
+ xlabel Vi
+ ylabel Vo
+ title 'Transfer Characteristics of Diode Clipper'
.endc

.end
```

Results of simulation are shown in Figures 37 and 38.



Curve: <space> @ x=-5.72043986 y=15.9161288

Figure 38: Transfer characteristics of diode clipper.

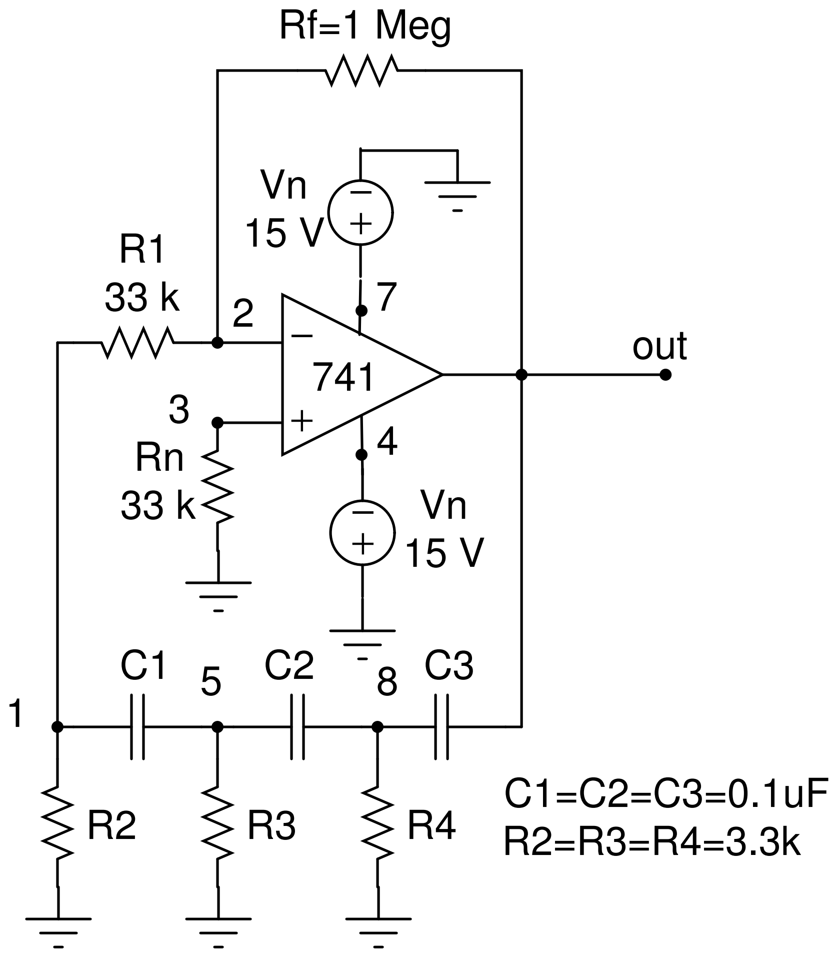


Figure 39: Phase shift oscillator using op-amp.

17.11 R-C Phase Shift Oscillator using Opamp

Figure 39 shows an *R-C* phase shift oscillator built using op-amp 741. The frequency of oscillation of this circuit is given by

$$f = \frac{1}{2\pi\sqrt{6}RC}$$

where $R = R_2 = R_3 = R_4$ and $C = C_1 = C_2 = C_3$. At frequency f given by the above equation, the gain $A_v = R_f/R_1$ must be at least 29 for the oscillations to sustain. For the values of parameters given in the figure, the frequency of oscillation is 200 Hz. This oscillator gives sinusoidal output. Circuit file for simulating this circuit is given below. Here we use `tran` command since we want to plot the output voltage waveform versus time.

```
R-C Phase Shift Oscillator Using Opamp uA741

* This is dual power supply.
VP 7 0 DC 15
VN 0 4 DC 15

* Resistors
R1 1 2 33K
R2 1 0 3.3K
R3 5 0 3.3K
R4 8 0 3.3K
RF 2 OUT 1MEG
RN 3 0 33K

* Capacitors
C1 1 5 0.1UF
C2 5 8 0.1UF
C3 8 OUT 0.1UF

* This refers to subcircuit of op-amp uA741
* which is stored in the library file parts.lib
X 3 2 7 4 out uA741
.include parts.lib

.control
destroy all
* Here, TSTART is assumed to be 15ms. So, the waveform
* from t=0 to t=15ms is not shown in the plot.
* Try simulation without TSTART and observe the output
* voltage plot.
tran 0.01ms 50ms 15ms

plot v(out) vs (time*1000)
+ xlabel 't[ms]'
+ ylabel 'v(out)[V]'
```

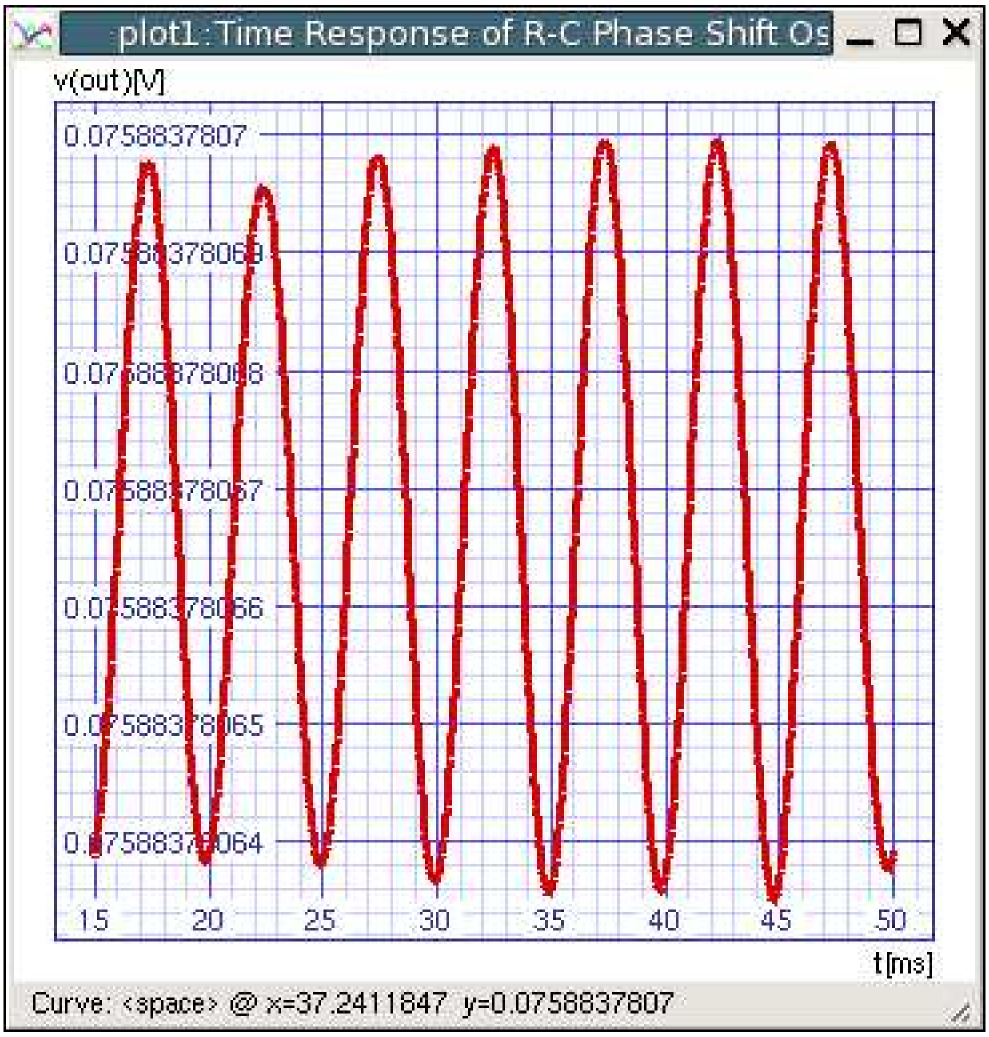


Figure 40: Output of phase shift oscillator.

```
+ title 'Time Response of R-C Phase Shift Oscillator'
.endc

.end
```

The results of simulation are shown in Figure 40.

17.12 Diode Clamper

Figure 41 shows a diode clamper and its input voltage waveform. The diode is assumed to be 1N4007 whose model is assumed to be present in `parts.lib` file. If it is not present in `parts.lib` file, copy the same into that file and then proceed with simulation. The circuit file is given below:

```
DIODE CLAMPER CIRCUIT

* Input has a rectangular waveform which can
* be described for time-domain analysis by
* a pulse specification. General format of pulse
* specification is:
* pulse(V1 V2 TD TR TF PW PER) where
* V1 -- Initial value.
* V2 -- Pulsed value
* TD -- Delay time
* TR -- Rise time
* TF -- Fall time
* PW -- Pulse width
* PER -- Period
V1 1 0 dc 0 PULSE(-20 10 0 2NS 2NS 0.5ms 1ms)
V2 3 0 dc 5
R1 2 0 100K
C1 1 2 1U
D1 3 2 1N4007
.include parts.lib

.control
destroy all
tran 0.01ms 5ms
plot v(1) v(2) vs (time*1000)
+ xlabel 'Time[ms]'
+ ylabel 'Input: V(1)[Red]; Output: V(2)[Green]'
+ title 'Diodle Clamper Response'
.endc

.end
```

The output and input voltage waveforms will be as shown in Figure 42.

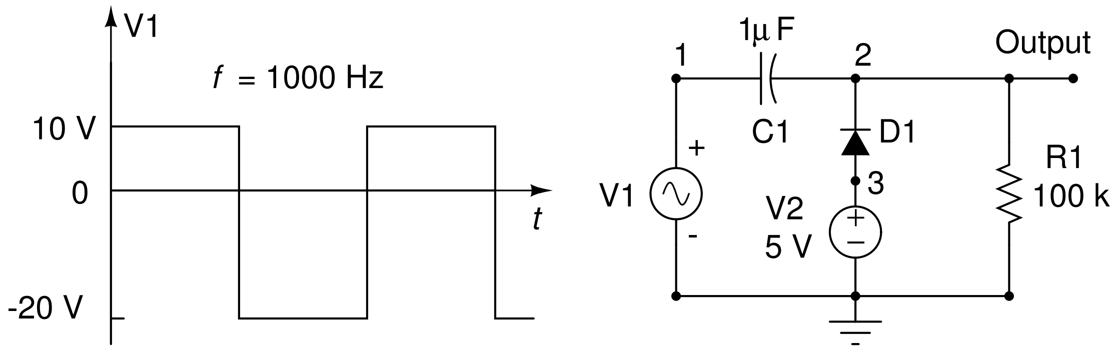
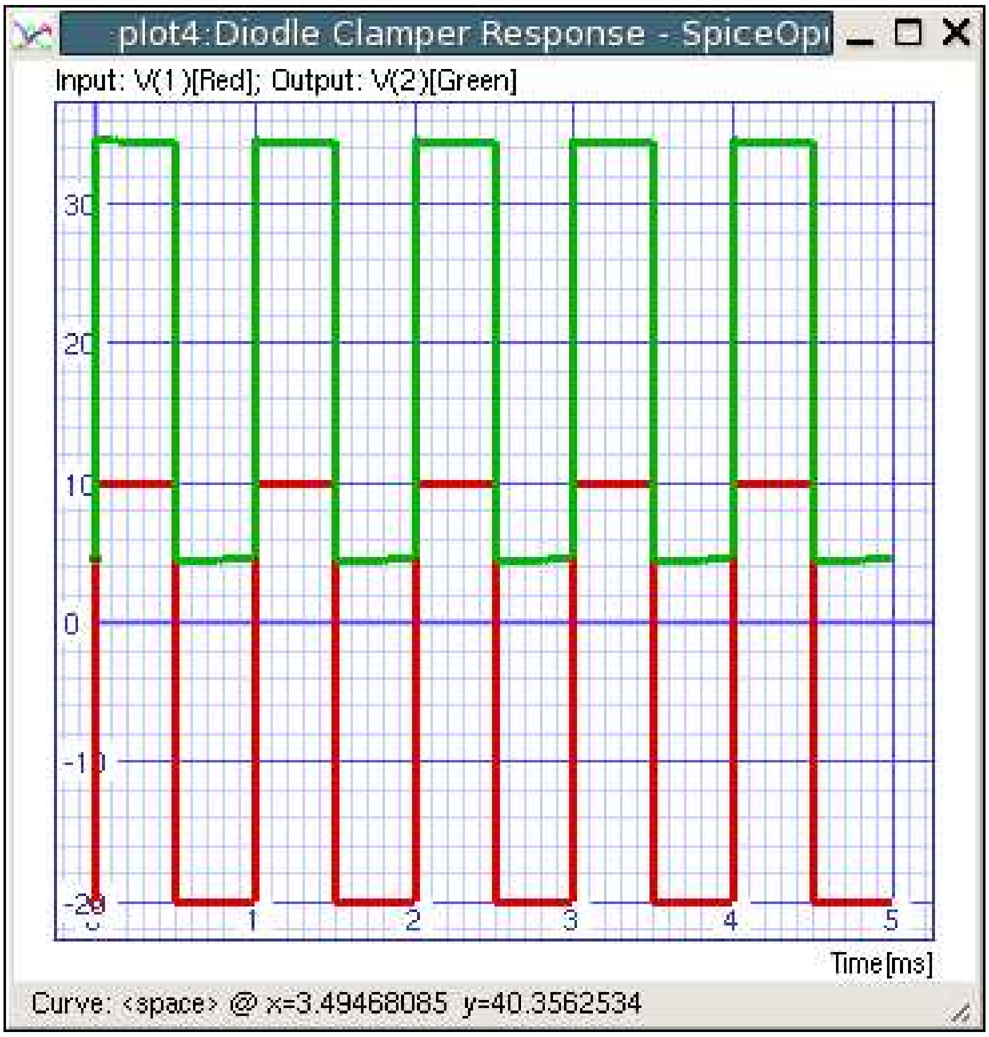


Figure 41: Diode clamper circuit.



Input: V(1)[Red]; Output: V(2)[Green]

Curve: <space> @ x=3.49468085 y=40.3562534

Figure 42: Input and output voltage waveforms in diode clamper circuit.