

## C H A P T E R 1

# Introduction to SPICE

The history of the SPICE program starts in 1968 at the University of California, Berkeley. A young faculty member, Ron Rohrer, introduced a new course on circuit simulation [Rohrer, 1992]. For this course, a circuit simulator, **CANCER** (Computer Analysis of Nonlinear Circuits Excluding Radiation) was developed by a student, Larry Nagel. The improved version of **CANCER**, named **SPICE** (Simulation Program with Integrated Circuit Emphasis), was developed in 1971 and released into the public domain. Other similar programs were developed at the same time such as **ECAP** and **ECAP-II** by IBM, **TIME** and **MTIME** by Motorola, and **TRAC** by Rockwell. Because **SPICE** was in the public domain, the program dominated both academia and industry. During the rapid development of integrated circuits, the demand for simulation tools was very large, and the **SPICE** program became the industry standard. In 1975 the new and significantly improved version **SPICE2** was released. From 1975 to 1983 several additional versions of **SPICE2** were developed. The most popular version was **SPICE2G.6** released in 1983. **SPICE2** was written in **FORTRAN** and was used on main frame and mini computers.

Because the source code of **SPICE2** was available to everyone, versions of the **SPICE** program were soon developed in C language. These commercial versions could also run on personal computers. Popular programs are **PSPICE** by MicroSim, **ICAP** and **IS\_SPICE** by IntuSoft, **VBA** by VeriBest, **HSPICE** by Meta-Software, **MICRO-CAP** by Spectrum Software, **ProSPICE** by EI Software, **B2SPICE** by Beige Bag Software, and **RSPICE** by RCG Research. **PSPICE** was the first and became the most popular circuit simulator for several reasons. (1) Limited function versions of **PSPICE** were distributed freely, (2) The MicroSim company introduced a very convenient graphical postprocessor called **PROBE**, and (3) Powerful input file error checking routines were introduced.

In the early 1990's the new **SPICE3** program from the University of California at Berkeley was released. This version was also written in C language. The first versions had numerous bugs, which were corrected in successive versions. The first mature version was **SPICE3f**, released in 1993. The currently distributed version is **SPICE3f5**. Similar to **SPICE2**, the source code of **SPICE3** is available to everyone, and this fact generated many **SPICE**-type simulators that use the **SPICE3** engine. The most popular programs with the **SPICE3** engine are **AIM-SPICE** by Ytterdal, Lee, Shur & Fjeldly; **B2SPICE-version2** by Beige Bag Software; **TurboSim** by Island Logic, **Dr.SPICE** by Deutsch Research, **Simetrix** by Newbury Tech. **S-SPICE** by SVC Production and **ViewSpice** by ViewLogic.

## 1.1. Fast Start

In order to use the computer for circuit analysis, information about the circuit topology and its elements must be communicated to the computer. This can be done in two ways: by drawing the circuit on a computer screen, or by describing the circuit using a special language. The first approach, known as schematic capture, is only a partial solution, since only circuit topology and element values can be entered. Circuit elements still require description. With a special language for circuit description, not only circuit topology and element descriptions but also information about required analyses can be entered in one file.

Close to 20 versions of SPICE have already been developed. Some have schematic capture capabilities, and with a little experience with windows, these are easy to learn. Although each program is different, all contain easy-to-follow routines for drawing a circuit on a screen. SPICE versions with schematic capture entry are easy to use, but in many cases some SPICE features are lost (for example, the subcircuit creation ability for hierarchical design). Schematic capture is often implemented differently in various versions even from the same developer, and the schematic capture interface is specific to each computer platform being used.

Since the schematic capture portions are not standard and have not matured at this time, we will concentrate on the standard SPICE language for circuit description. This simple text file is universal and can be used on any computer platform. There are some small differences between various versions of SPICE programs, but usually each SPICE version is close to one of three SPICE standards: SPICE2, PSPICE, or SPICE3.

It is usually easy to learn about programs by examples, and SPICE is not an exception. In order to analyze a circuit, a version of the SPICE program is needed. For the case of a simple circuit, all these versions should behave similarly. The SPICE input file has to be first created with an editor, and then the file is used as an input to the SPICE program.

### Example 1. Uncompensated voltage divider

Let us consider the simple circuit diagram shown in Fig. 1.1.1. The creation of an input SPICE file always begins with assigning numbers to circuit nodes and names to circuit elements. Node numbers need not be sequential, but the number 0 must be always assigned to the reference (ground) node. Names of elements must always start with a specific identifying letter. This must be the letter R for resistors, the letter C for capacitors, and the letter V for independent voltage sources. For a complete list of letters dedicated to specific elements, see Chapter 5.

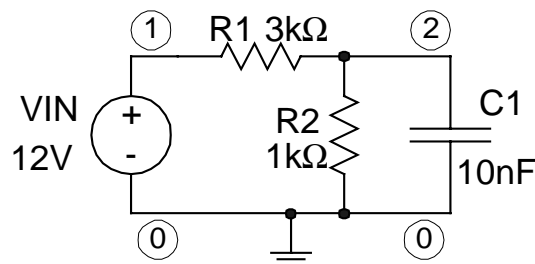


Fig. 1.1.1. Diagram of a voltage divider (or lossy integrator).

An input SPICE file for the circuit of Fig. 1.1.1 is presented in Fig. 1.1.2. The first line of a SPICE input file is always the title line. The contents of this line are usually printed in the output file or on

the plot. The next four lines describe the circuit topology. Any text after the “;” character represent comments and is skipped by PSPICE. Unfortunately this type of comment is not allowed in SPICE2 and SPICE3. However, lines that start with “\*” will always be treated as comments in all SPICE versions.

VOLTAGE DIVIDER	; title line
VIN 1 0 DC 12	; 12V voltage source connected between nodes 1 and 0
R1 1 2 3000	; 3k $\Omega$ resistor connected between nodes 1 and 2
R2 2 0 1K	; 1k $\Omega$ resistor connected between nodes 2 and 0
C1 2 0 10N	; 10nF capacitor connected between nodes 2 and 0
.OP	; command <b>.OP</b> statement to calculate the biasing point
.END	; <b>.END</b> statement indicating the end of circuit description

Fig. 1.1.2. Listing of SPICE input file FS1.CIR describing the circuit from Fig. 1.1.1.

There are four circuit elements: voltage source **VIN** connected between nodes 1 and 0; two resistors, between nodes 1 and 2, and between nodes 2 and 0; and capacitor **C1**, connected between nodes 2 and 0. Each element has its characteristic value, which is specified at the end of the line. An element value can be written directly or using abbreviating letters. A list of abbreviations is given in Table 1.1.1. For example, for the case of resistor **R1**, it could be written as 3000, 3E3, or 3K. In the case of capacitor **C1**, it could be 0.00000001, 1E-8, 10E-9, 10N, 10000P, 0.01U, or 0.00001M. The original version of SPICE accepted only capital letters, but current versions are not case-sensitive (with the exception of some PSPICE versions, where m stands for milli and M stands for mega). A symbol or text written after an abbreviating letter is ignored by SPICE. Therefore, to make an input file more readable, one may write 12V, 3Kohms, 3kOHMs, or 10nF.

**Table 1.1.1 Scaling letters**

Letter	Pronunciation	Scale factor
F	femto	$10^{-15}$
P	pico	$10^{-12}$
N	nano	$10^{-9}$
U	micro	$10^{-6}$
M	milli	$10^{-3}$
K	kilo	$10^3$
MEG	mega	$10^6$
G	giga	$10^9$
T	tera	$10^{12}$
MIL	mil	$25.4 \cdot 10^{-6}$

All statements which start with a “.” (period) are commands. The **.OP** command tells SPICE to calculate a bias point. Each circuit description must end with the **.END** statement. Many circuits can be described in one file, but each circuit must be terminated with the **.END** line. SPICE will perform analysis for all circuits in the same sequence as they were written in the input file. However, PSPICE and SPICE3 produce slightly different output files.

```

**** 05/24/96 14:03:22 ***** NT Evaluation PSpice (April 1995) *****
VOLTAGE DIVIDER
****      CIRCUIT DESCRIPTION
*****
VIN 1 0 12
R1 1 2 3000
R2 2 0 1K
C1 2 0 10N
.OP
.END
**** 05/24/96 14:03:22 ***** NT Evaluation PSpice (April 1995) *****
VOLTAGE DIVIDER
****      SMALL SIGNAL BIAS SOLUTION          TEMPERATURE = 27.000 DEG C
*****
NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE
(      1)  12.0000  (      2)   3.0000
VOLTAGE SOURCE CURRENTS
NAME          CURRENT
VIN           -3.000E-03
TOTAL POWER DISSIPATION  3.60E-02  WATTS
**** 05/24/96 14:03:22 ***** NT Evaluation PSpice (April 1995) *****
VOLTAGE DIVIDER
****      OPERATING POINT INFORMATION          TEMPERATURE = 27.000 DEG C
*****
JOB CONCLUDED
TOTAL JOB TIME                .88

```

Fig. 1.1.3. Example of an output file generated by PSPICE using the FS1.CIR input file from Fig. 1.1.2.

```

***** 05/30/96 ***** SPICE2 V2G.6 03/15/83 ***** 2:39 pm *****
VOLTAGE DIVIDER
****      INPUT LISTING          TEMPERATURE = 27.000 DEG C
*****
VIN 1 0 12
R1 1 2 3000
R2 2 0 1K
C1 2 0 10N
.OP
.END
***** 05/30/96 ***** SPICE2 V2G.6 03/15/83 ***** 2:39 pm *****
VOLTAGE DIVIDER
****      SMALL SIGNAL BIAS SOLUTION          TEMPERATURE = 27.000 DEG C
*****
NODE   VOLTAGE     NODE   VOLTAGE
(      1)  1.2000E+01  (      2)  3.0000E+00
VOLTAGE SOURCE CURRENTS
NAME          CURRENT
VIN           -3.000E-03
TOTAL POWER DISSIPATION  3.60E-02  WATTS
JOB CONCLUDED
TOTAL JOB TIME                0.06

```

Fig. 1.1.4. Example of an output file generated by SPICE2 version 2G.6 using the FS1.CIR input file from Fig. 1.1.2.

```

Circuit: VOLTAGE DIVIDER
Date: Fri May 24 13:59:38 1996
Operating point information:
  Node   Voltage
  ----   -
  V(2)   3.000000e+000
  V(1)   1.200000e+001
Source Current
-----
vin#branch  -3.00000e-003
Capacitor models (Fixed capacitor)
model      C
  cj       0
  cjsw     0
  defw     1e-005
  narrow   0
Resistor models (Simple linear resistor)
model      R
  rsh      0
  narrow   0
  tc1      0
  tc2      0
  defw     1e-005
Capacitor: Fixed capacitor
device     c1
model      C
capacitanc 1e-008
  i 1.87e-306
  p 1.87e-306
Resistor: Simple linear resistor
device     r2      r1
model      R      R
resistance 1e+003  3e+003
  i 0.003  0.003
  p 0.009  0.027
Vsource: Independent voltage source
device     vin
  dc      12
  acmag   0
  i -0.003
  p 0.036
elapsed time since last call: 0.000 seconds.
Total elapsed time: 0.000 seconds.
Current dynamic memory usage = 0,
Dynamic memory limit = 0.

```

Fig. 1.1.5. Example of an output file generated by SPICE3 using the FS1.CIR input file from Fig. 1.1.2.

Such output can be obtained by redirecting standard output to a file FS1.OUT using the following parameters in the command line: “-b FS1.CIR >FC1.OUT”.

All three programs, PSPICE, SPICE2, and SPICE3, generate slightly different output formats (Figs. 1.1.3. through 1.1.6), but the results are the same. The programs calculate the voltage on node 2 as 3 V, the VIN source current as 3 mA, and the power delivered by VIN as 36 mW. In addition to computing results, PSPICE prints the data from the input file (Fig. 1.1.3). In SPICE3, results can be directed to the screen (default), to an output file if the standard output is redirected (Fig. 1.1.5), or to a “raw” file using the -r switch and the file name (Fig. 1.1.6). For the .OP command, SPICE3 prints all model parameters to the output. When the -r switch is used, only essential output results are printed to the “raw” file (Fig. 1.1.6). The results can then be used for plotting characteristics with a graphical package or the NUTMEG program, which works as a postprocessor for SPICE3.

```

Title: VOLTAGE DIVIDER
Date: Fri May 24 21:43:00 1996
Plotname: Operating Point
Flags: real
No. Variables: 3
No. Points: 1
Command: version 3f5
Variables:
    0      V(1)  voltage
    1      V(2)  voltage
    2      vin#branch  current
Values:
0          1.2000000000000000e+001
          3.0000000000000000e+000
          -3.0000000000000000e-003

```

Fig. 1.1.6. Example of a “raw” file generated by SPICE3 using the FS1.CIR input file from Fig. 1.1.2. A raw file can be obtained using the following parameters in the command line: “-b -r FS1.RAW FS1.CIR”.

There are minor differences in input files between SPICE2, PSPICE, and SPICE3 programs. For the purpose of clarity in this book, the \*.CIR extension will be used for PSPICE input files and the \*.CKT extension will be used for SPICE3 input files. Examples in this book will be illustrated alternatively by input formats for PSPICE and SPICE3. For SPICE3 an input file extension is not important, since the full file name must always be used. PSPICE uses \*.CIR as default, but it can also run with any input file name extension.

## 1.2. dc Analysis

In the previous section, SPICE solved a simple problem of finding the bias point for a passive voltage divider, but SPICE also contains built-in models for most semiconductor devices.

### Example 1. Diode characteristics

Let us find the current-voltage relation for a diode using the circuit diagram of Fig. 1.2.1. This example will be analyzed using PSPICE; the next example of dc analysis will use SPICE3.

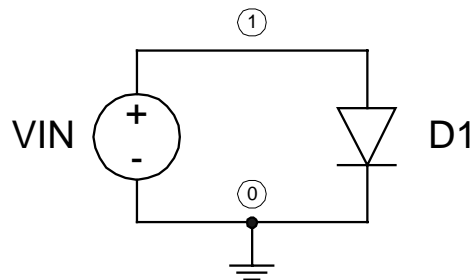


Fig. 1.2.1. Circuit to simulate static diode characteristic.

DIODE CHARACTERISTIC	; title line
VIN 1 0	; voltage source connected between nodes 1 and 0
*	note that default 0 value for voltage is used
D1 1 0 DMOD	; diode D1 using model DMOD
.MODEL DMOD D IS=1.0E-16A	; diode model with definition of <b>IS</b> parameter
.DC VIN -0.8V 0.8V 20MV	; <b>.DC</b> statement to calculate dc characteristics
.PRINT DC I(VIN)	; <b>.PRINT</b> statement with one variable
.END	; end of circuit

Fig. 1.2.2. Listing of SPICE input file DC1.CIR describing the circuit from Fig. 1.2.1. The **.PRINT** statement is used to print results to an output file.

The listing of SPICE input is shown in Fig. 1.2.2. The third line of the code starts with “\*” and is ignored by the SPICE program as a comment line. A diode is described in a manner similar to a resistor, by writing its name, which must start with the letter **D**, and the nodes where the diode is connected. In the case of semiconductor devices, such as diodes and transistors, it is also mandatory to specify the name of a model. This can be any name, but it is good practice to start the model name with a characteristic letter for the element. Model parameters must be declared in a separate **.MODEL** statement. In our example, only one model parameter **IS** is declared.

```

**** 05/25/96 13:38:55 ***** NT Evaluation PSpice (April 1995) *****
DIODE CHARACTERISTIC
****
CIRCUIT DESCRIPTION
*****
VIN 1 0
*
D1 1 0 DMOD
.MODEL DMOD D IS=-1.0E-16A
.DC VIN -0.8V 0.8V 20MV
.PRINT DC I(VIN)
.END
**** 05/25/96 13:38:55 ***** NT Evaluation PSpice (April 1995) *****
DIODE CHARACTERISTICS
****
Diode MODEL PARAMETERS
*****
DMOD
IS -100.000000E-18
**** 05/25/96 13:38:55 ***** NT Evaluation PSpice (April 1995) *****
DIODE CHARACTERISTICS
****
DC TRANSFER CURVES TEMPERATURE = 27.000 DEG C
*****
VIN I(VIN)
-8.000E-01 7.999E-13
-7.500E-01 7.499E-13
-7.000E-01 6.999E-13
.
.
6.000E-01 1.188E-06
6.500E-01 8.211E-06
7.000E-01 5.675E-05
7.500E-01 3.922E-04
8.000E-01 2.711E-03
JOB CONCLUDED
TOTAL JOB TIME 1.54

```

Fig. 1.2.3. Example of an output file generated by PSPICE using DC1.CIR input file from Fig. 1.2.2. Only the beginning and end of the DC1.OUT file are shown.

The `.DC` statement indicates that independent voltage `VIN` is to be changed from  $-0.8\text{ V}$  to  $+0.8\text{ V}$  with  $20\text{ mV}$  steps. Note that there is no ammeter in the diagram of Fig. 1.2.1 to measure the diode current. Since SPICE always calculates currents through all voltage sources, the current through voltage source `VIN` can be used as an output. Current through a voltage source is considered to be positive when it flows from the positive terminal through the source to the negative terminal, and in our circuit the source current corresponds to negative diode current. Using the `.PRINT` statement, we can display the current of the `VIN` voltage source, which has an opposite sign to the diode current.

Fragments from the PSPICE output file for input file `DC1.CIR` of Fig. 1.2.2. are shown in Fig. 1.2.3. Another way to print results is to use the `.PLOT` statement. Output characteristics can be plotted to the output file using ASCII characters and creating a pseudo graph. In this case the input file `DC2.CIR` should have the format shown in Fig. 1.2.4, and Fig. 1.2.5 presents the resulting pseudo graph.

<code>PSEUDO PLOT OF DIODE CHARACTERISTIC</code>	; title line
<code>VIN 1 0</code>	; voltage source connected between nodes 1 and 0
<code>D1 1 0 DMOD</code>	; diode D1 using model DMOD
<code>.MODEL DMOD D IS=-1.0E-16A</code>	; diode model with defined <code>IS</code> parameter
<code>.DC VIN 0 0.8 50MV</code>	; <code>.DC</code> statement to calculate dc characteristics
<code>.PLOT DC I(VIN)</code>	; <code>.PLOT</code> statement with one variable
<code>.END</code>	; end of circuit

Fig. 1.2.4. Listing of a SPICE input file `DC2.CIR` describing the circuit from Fig. 1.2.1. The `.PLOT` statement is used to plot I-V characteristic as a pseudo graph.

```

**** 05/25/96 15:55:27 ***** NT Evaluation PSpice (April 1995) *****
PSEUDO PLOT OF DIODE CHARACTERISTIC
****      DC TRANSFER CURVES          TEMPERATURE = 27.000 DEG C
*****
VIN          I (VIN)
(*)-----  -1.0000E-03  0.0000E+00  1.0000E-03  2.0000E-03  3.0000E-03
-----
0.000E+00  0.000E+00  .          *          .          .
5.000E-02 -4.941E-14  .          *          .          .
1.000E-01 -9.532E-14  .          *          .          .
1.500E-01 -1.171E-13  .          *          .          .
2.000E-01  2.808E-14  .          *          .          .
2.500E-01  1.327E-12  .          *          .          .
3.000E-01  1.060E-11  .          *          .          .
3.500E-01  7.498E-11  .          *          .          .
4.000E-01  5.202E-10  .          *          .          .
4.500E-01  3.598E-09  .          *          .          .
5.000E-01  2.487E-08  .          *          .          .
5.500E-01  1.719E-07  .          *          .          .
6.000E-01  1.188E-06  .          *          .          .
6.500E-01  8.211E-06  .          *          .          .
7.000E-01  5.675E-05  .          .*         .          .
7.500E-01  3.922E-04  .          .          *         .          .
8.000E-01  2.711E-03  .          .          .          .          *
-----

```

Fig. 1.2.5. Example of an output file generated by PSPICE using the `DC2.CIR` input file from Fig. 1.2.4. The `.PLOT` statement was used to generate a pseudo graph.



In the case of PSPICE, the graphical postprocessor PROBE can be used to present output characteristics as true graphics. To do this the **.PROBE** statement has to be added to the input file. In this case PSPICE will create a temporary file PROBE.DAT which can then be read by the PROBE program. When the **.PROBE** statement is used without parameters, all results are stored in the PROBE.DAT file. For large circuits this may lead to excessively large PROBE.DAT files. It is possible to store only data of interest by specifying a list of variables after the **.PROBE** keyword in a manner similar to the **.PRINT** or **.PLOT** statements. Fig. 1.2.6 shows a sample input file using the **.PROBE** statement and Fig. 1.2.7 presents the graph from the PROBE program.

DIODE CHARACTERISTIC USING PROBE	; title line
VIN 1 0	; voltage source connected between nodes 1 and 0
D1 1 0 DMOD	; diode D1 using model DMOD
.MODEL DMOD D IS=-1.0E-16A	; diode model with defined IS parameter
.DC VIN 0 0.8 50MV	; .DC statement to calculate DC characteristics
.PROBE I(VIN)	; .PROBE statement with one variable
.END	; end of circuit

Fig. 1.2.6. Listing of PSPICE input file DC3.CIR describing the circuit from Fig. 1.2.1. The **.PROBE** statement is used to plot I-V characteristic as true graphics.

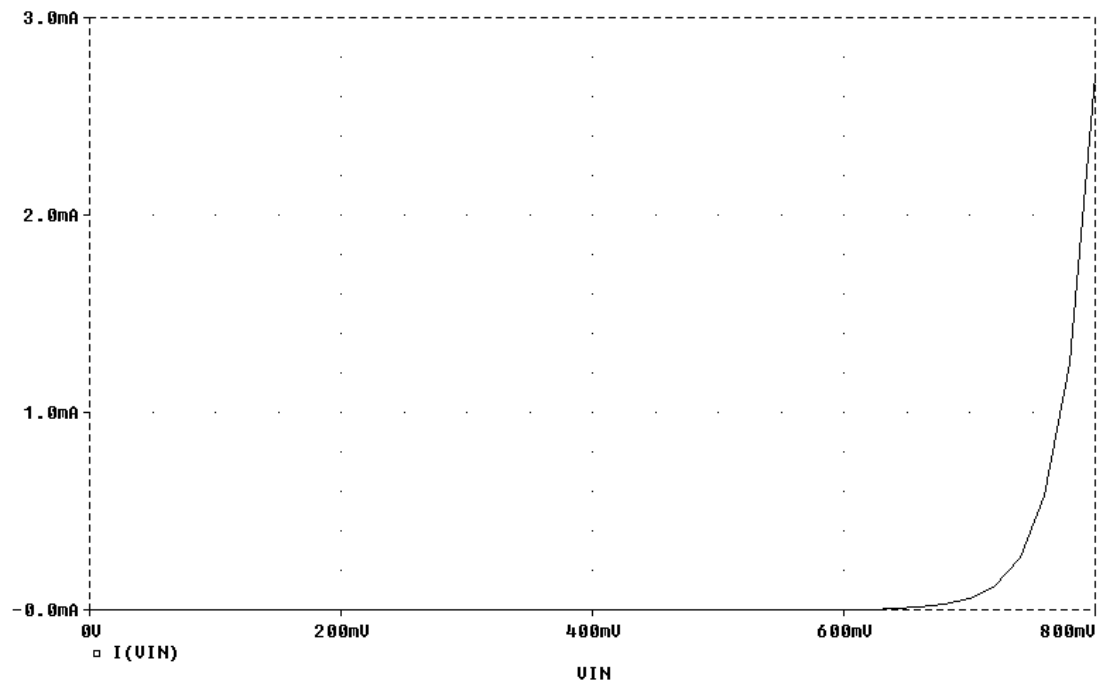


Fig. 1.2.7. Diode forward characteristic obtained with PSPICE and PROBE programs using the DC3.CIR input file from Fig. 1.2.6.

### Example 2. *npn-pnp* transistor amplifier

Let us analyze the amplifier circuit shown in Fig. 1.2.8. Node 2 is the input and node 6 is the output. The PSPICE input description is shown in Fig. 1.2.9. Both statements **.PLOT** and **.PROBE** are used to generate output as pseudo graphics and as true graphics.

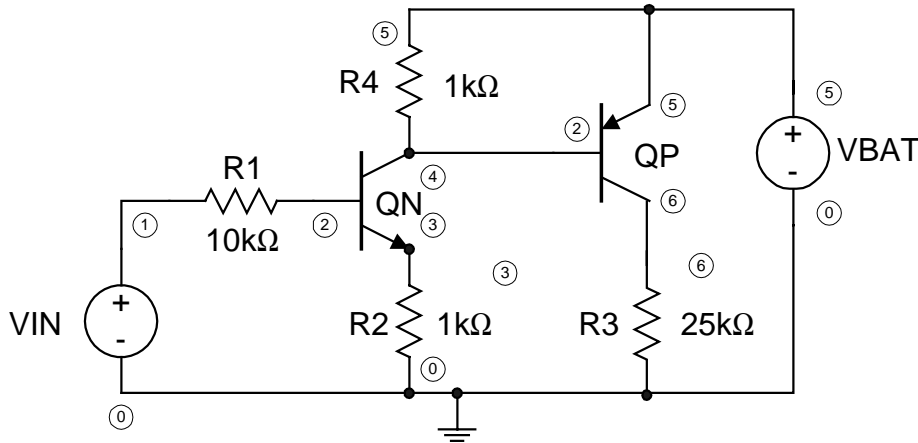


Fig. 1.2.8. *npn-pnp* transistor amplifier circuit.

NPN-PNP amplifier circuit	; title line
VBAT 5 0 3V	; supply battery connected between nodes 5 and 0
VIN 1 0 1V	; input voltage source
R1 1 2 10k	; resistor R1
R2 3 0 1k	; resistor R2
R3 6 0 25k	; resistor R3
R4 5 4 1k	; resistor R4
QN 4 2 3 QNPN	; NPN transistor QN nodes: C = 4, B = 2, E = 3
QP 6 4 5 QPNP	; PNP transistor QP nodes: C = 6, B = 4, E = 5
.MODEL QNPN NPN IS=2.0E-15A	; model statement for NPN transistor
+ BF=150 VAF=100 BR=2 VAR=30	; continuation of previous line using "+" as the first character in the line
.MODEL QPNP PNP IS=1.0E-15A	; model statement for PNP transistor
+ BF=100 VAF=80 BR=1 VAR=25	; continuation of previous line
.DC VIN 0.5 2 50MV	; .DC statement to sweep VIN from 0 to 2 V with 50mV steps
*.PROBE DC V(1) V(2) V(3)	; .PROBE statement for PSPICE only - remove "*" from both lines
*+ V(4) V(5) V(6)	; the .PROBE statement must not be used for SPICE3
.SAVE V(1) V(2) V(3)	; .SAVE statement for SPICE3 only
+ V(4) V(5) V(6)	; the .SAVE statement must not be used for PSPICE
.PLOT DC V(1) V(2) V(3)	; .PLOT statement for pseudo graphics
+ V(4) V(5) V(6)	; extension of previous line using "+" as the first character in the line
.END	; end of circuit description

Fig. 1.2.9. Input file DC4.CKT for *npn-pnp* amplifier from Fig. 1.2.8.

Fig. 1.2.10 presents the pseudo graphics output obtained with SPICE3, and Fig. 1.2.11 presents a similar output obtained with PSPICE. Note that in both cases, the characteristics were automatically scaled. In the case of SPICE3, Fig. 1.2.10, the same scale is used for all plots. In the case of PSPICE,

Fig. 1.2.11, each plot has its own scale. For both programs a scale can be set manually by changing the **.PLOT** statements in DC4.CIR to

```
.PLOT DC V(1) V(2) V(3) V(4) V(5) V(6) (0,3)
```

Then all characteristics will be plotted using the scale specified in brackets (in this case in the range of 0 to 3 V).

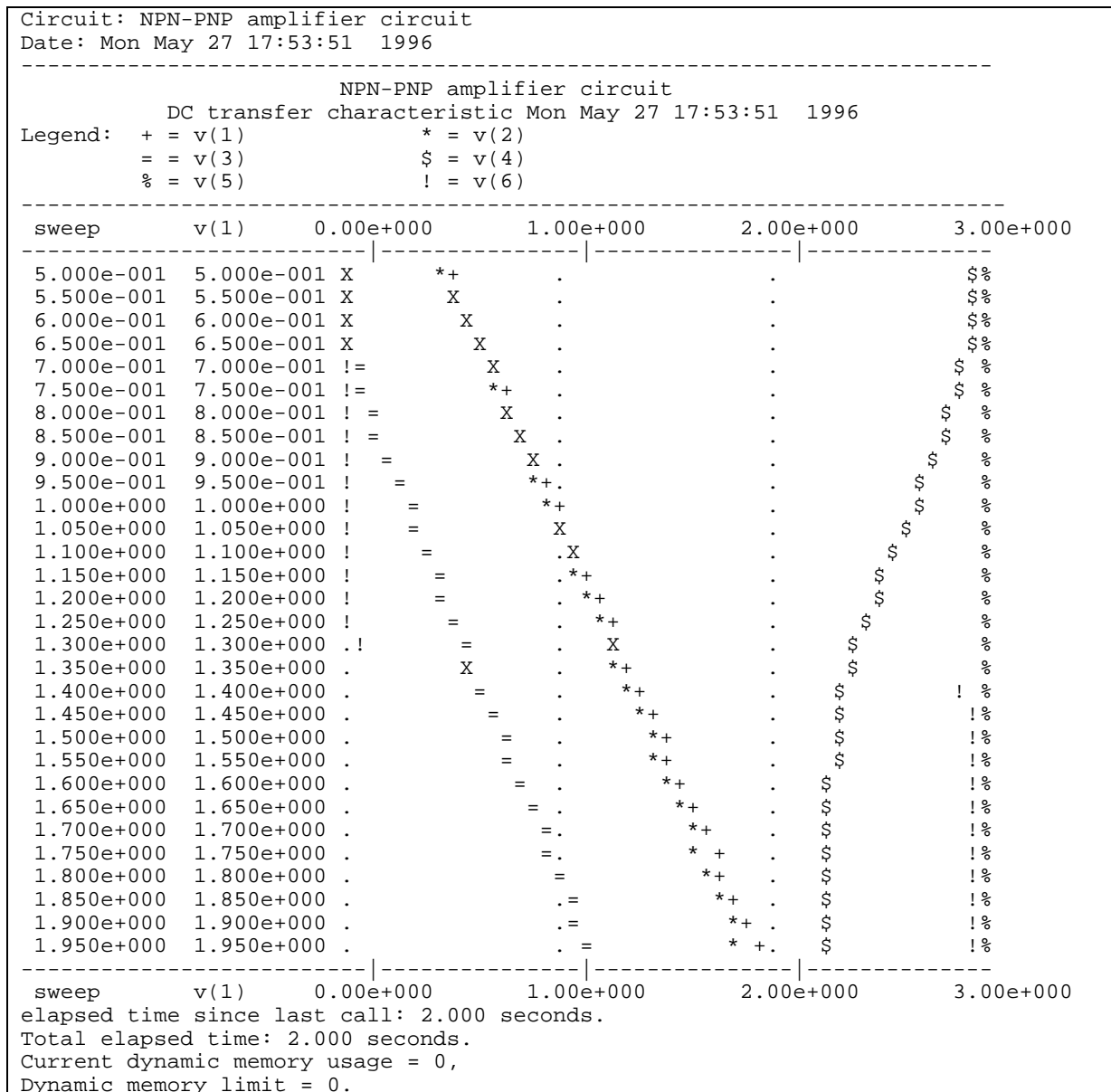


Fig. 1.2.10. A pseudo graph generated by SPICE3 using the input file DC4.CKT from Fig. 1.2.9. The **.PROBE** statement was commented out.

True graphic output can be generated with a **.PROBE** statement in the PSPICE and PROBE programs, or using **.SAVE** statement in SPICE3 and NUTMEG programs, or using MATLAB as a

postprocessor for both PSPICE or SPICE3. Resulting characteristics are shown in Figs. 1.2.12 through 1.2.14. Both programs PROBE and NUTMEG create graphics with black backgrounds, which can be inverted using the INVERTER program. SPICE3, NUTMEG, and INVERTER programs and MATLAB postprocessors can be found on <http://nn.uwyo.edu/sp/>.

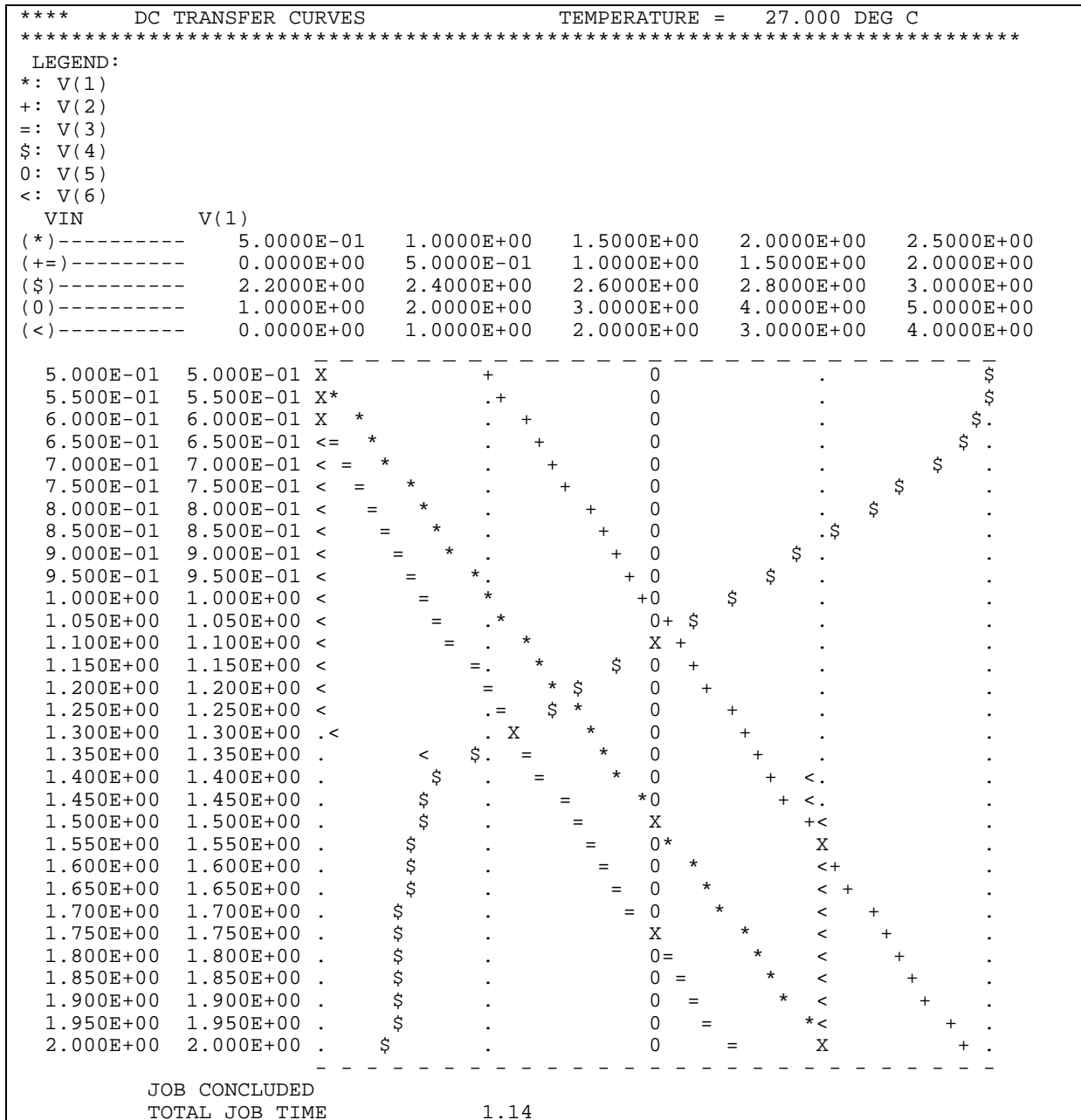


Fig. 1.2.11. A pseudo graph generated by PSPICE using the input file DC4.CIR, which is a modification of DC4.CKT from Fig. 1.2.9. The .SAVE statement was commented out.

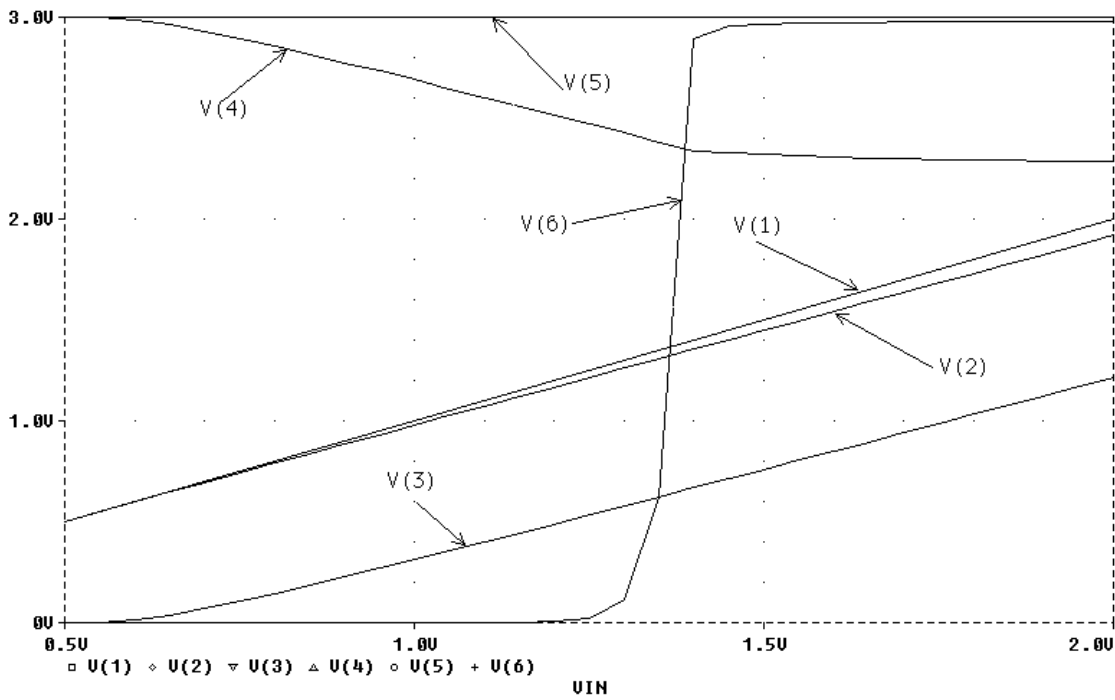


Fig. 1.2.12. dc transfer characteristics for the circuit of Fig. 1.2.8 obtained using PSPICE and input file DC4.CIR, which is a modification of DC4.CKT from Fig. 1.2.9. The **.PROBE** statement was used, and the **.SAVE** statement was commented out.

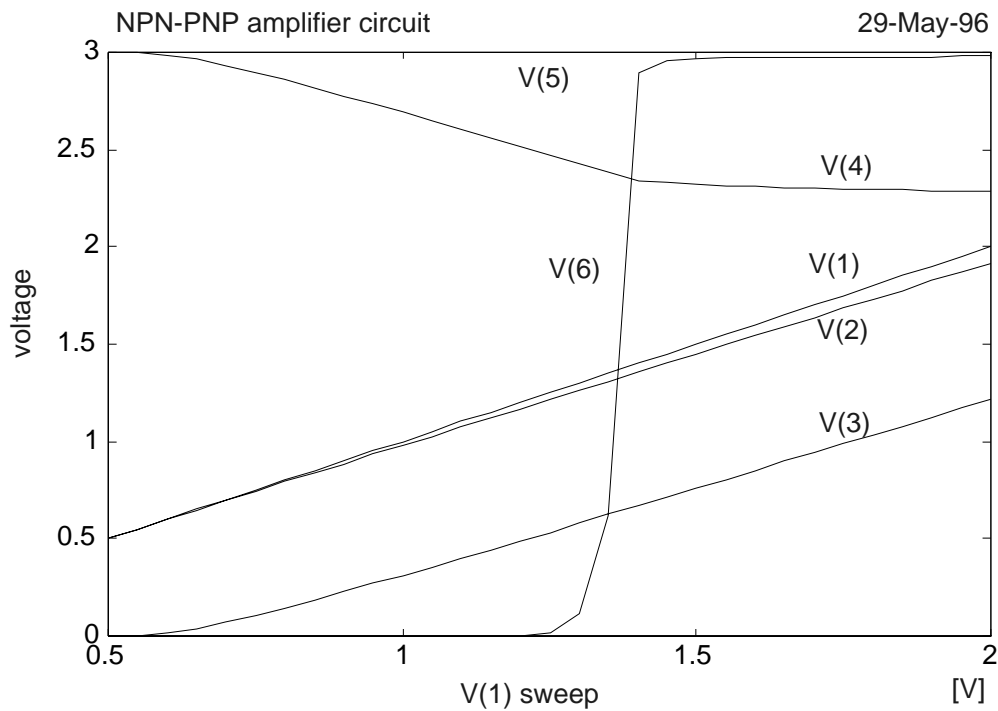


Fig. 1.2.13. The same dc transfer characteristic as on Fig. 1.2.12 obtained using MATLAB as a postprocessor. Data for MATLAB can be generated using both PSPICE and SPICE3.

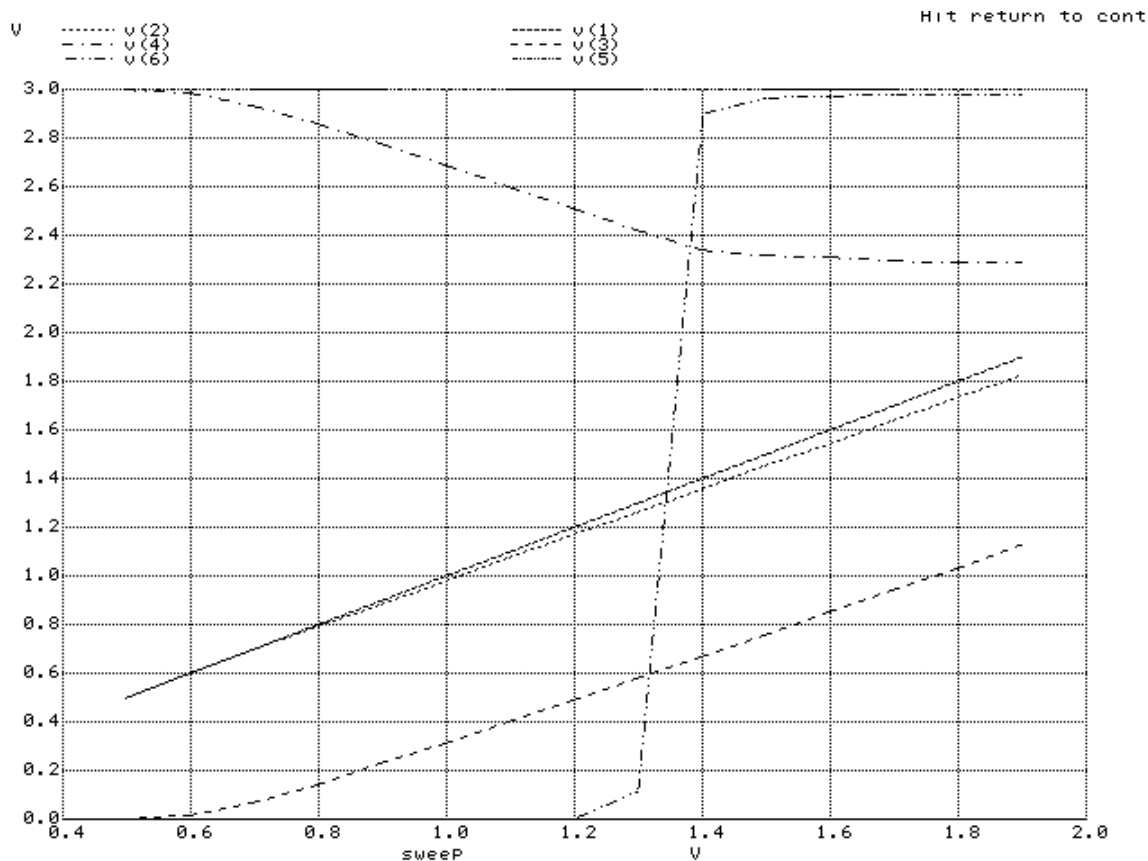


Fig. 1.2.14. dc transfer characteristics for the circuit of Fig. 1.2.8 obtained using SPICE3 and NUTMEG programs

SPICE3 has a graphical postprocessor NUTMEG which was primarily designed for Xwindows in the UNIX environment. PSPICE and PROBE also work in the PC environment, but the screen can be captured only as bitmap and cannot be scaled. Also, special care must be taken to remove the usual PROBE black background. This makes it difficult to paste and scale the results of PROBE to reports or other documents. It is possible to capture PROBE printer output in \*.PS or \*.HGL format and then convert it into high-quality vector graphics, but this is a very tedious procedure. It is much easier to save PSPICE or SPICE results as numerical data and use any graphical package or spreadsheet to create high-quality graphics. In this book, the MATLAB package will be used for this purpose. MATLAB was chosen because it has not only high-quality graphics, but also very powerful numerical routines to analyze results including axis scaling, Fourier transformations, statistical analysis, and others.

A simple way to generate numerical output data is to use the **.PRINT** statement. With any text editor, the required columns of data can be extracted and used as input to spreadsheet or graphic programs.<sup>1</sup> Instead of using the **.PRINT** statement, it is even easier to use the dedicated statements to generate raw numerical data. In the case of PSPICE, one can use the **.PROBE/CSDF** statement. The **/CSDF** switch was introduced so that the data file for the PROBE program could be generated in ASCII format and could therefore be transferred from one computer platform to another. This ASCII file can be easily read and converted to the format required by MATLAB or any other graphical package.

SPICE3 generates all results in ASCII raw file form for future use by the NUTMEG program. This raw file can also be easily read and converted to any other format. A problem may occur when a large

<sup>1</sup> The process can be automated using developed software which can be obtained from <http://nn.uwyo.edu/sp/>

circuit is analyzed and very large data is generated. This will happen when the **.PROBE** statement is used without parameters in the PSPICE case or no **.SAVE** statement with parameters is used in the SPICE3 case. It is therefore recommended to always use **.PROBE** and **.SAVE** statements, for PSPICE and SPICE3 respectively, with a list of variables to be plotted. For a series of different plots, such as ac, dc, and transient analyses, it is worthwhile to run the SPICE program a few times to generate different data for different plots.

### Example 3. dc characteristics of bipolar transistor

For the next example, let us analyze the characteristics of a bipolar transistor using the circuit diagram shown in Fig. 1.2.15. A SPICE input file for the circuit of Fig. 1.2.15 is shown in Fig. 1.2.16. Results are presented in Fig. 1.2.17. MATLAB was used as a graphical postprocessor in this case. Note the inverted output characteristic. This occurs because the current of a voltage source was used as output instead of the actual collector current. To remedy the problem, an additional voltage source with zero value, which works as an ammeter, has to be added to the circuit. Figure 1.2.18 illustrates the change.

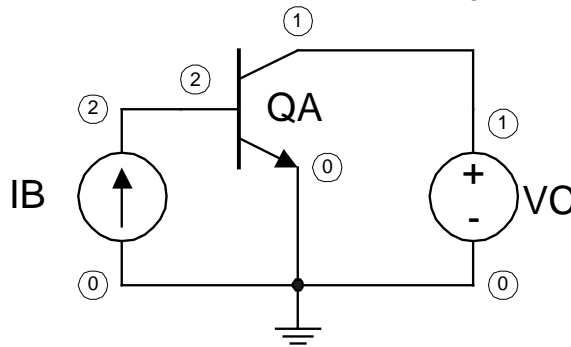


Fig. 1.2.15. Circuit to plot dc characteristics of an *npn* bipolar transistor.

TRANSISTOR	; title line
CHARACTERISTICS	; bipolar transistor model with parameters
.MODEL QMOD NPN IS=1.0E-15	; continuation of previous line using "+" as the first character
+ BF=100 BR=3 VAF=80 VAR=30	; current source with default 0 current
IB 0 2	; voltage source with default 0 voltage
VC 1 0	; bipolar transistor with specified nodes and model name
QA 1 2 0 QMOD	; for bipolar transistor nodes are listed in C, B, E order
* comment line	; DC sweep for collector voltage from 0 to 10V with 0.25V step
.DC VC 0 5V 0.05V	; and stepping base current with 10 $\mu$ A step
+ IB 10U 100U 10U	; commented statement for PSPICE only
*.PROBE I(VC)	; set proper output for SPICE3
.SAVE I(VC)	; end of circuit description
.END	

Fig. 1.2.16. SPICE3 input file DC5.CKT for plotting output characteristics of the bipolar transistor shown in Fig. 1.2.15. When PSPICE is used, the **.SAVE** statement must be replaced with the **.PROBE** statement.

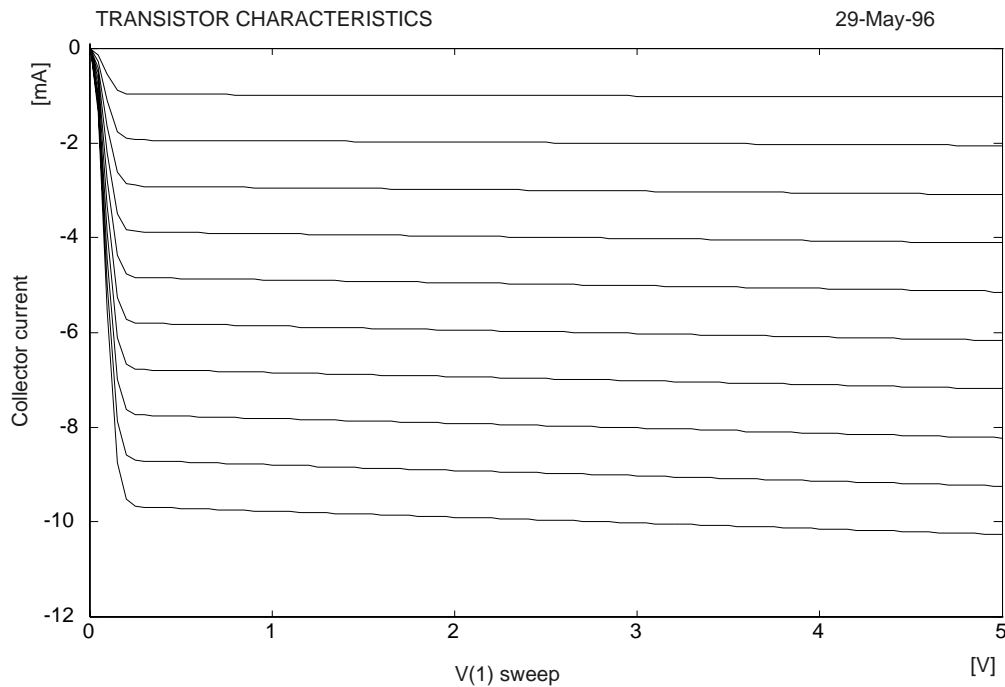


Fig. 1.2.17. Results generated by the input file from Fig. 1.2.16.

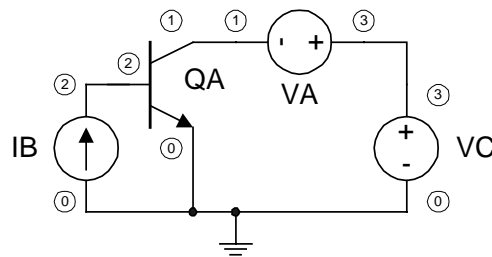


Fig. 1.2.18. Circuit to plot dc characteristics of bipolar transistor.

```

TRANSISTOR CHARACTERISTICS 2 ; title line
.MODEL QMOD NPN IS=1.0E-15 ; bipolar transistor model with parameters
+ BF=100 BR=3 VA=200 ; continuation of previous line using "+" as the first character
IB 0 2 10U ; 10 μA current source between nodes 0 and 2
VC 3 0 5V ; 5V voltage source between nodes 3 and 0
VA 3 1 0 ; zero voltage source which acts as ammeter
QA 1 2 0 QMOD ; bipolar transistor with specified nodes and model name
.DC VC 0 5V 0.05V ; DC sweep for collector voltage from 0 to 5V with 0.05V
+ IB 10U 100U 10U ; step base current with 10 μA step
* .PROBE I(VA) ; commented statement for PSPICE
.SAVE I(VA) ; generates output to raw file in SPICE3 only
.END ; end of circuit description

```

Fig. 1.2.19. SPICE input file DC6.CKT for plotting output characteristics of the bipolar transistor shown in Fig. 1.2.18. Additional zero voltage source VA is used as an ammeter.



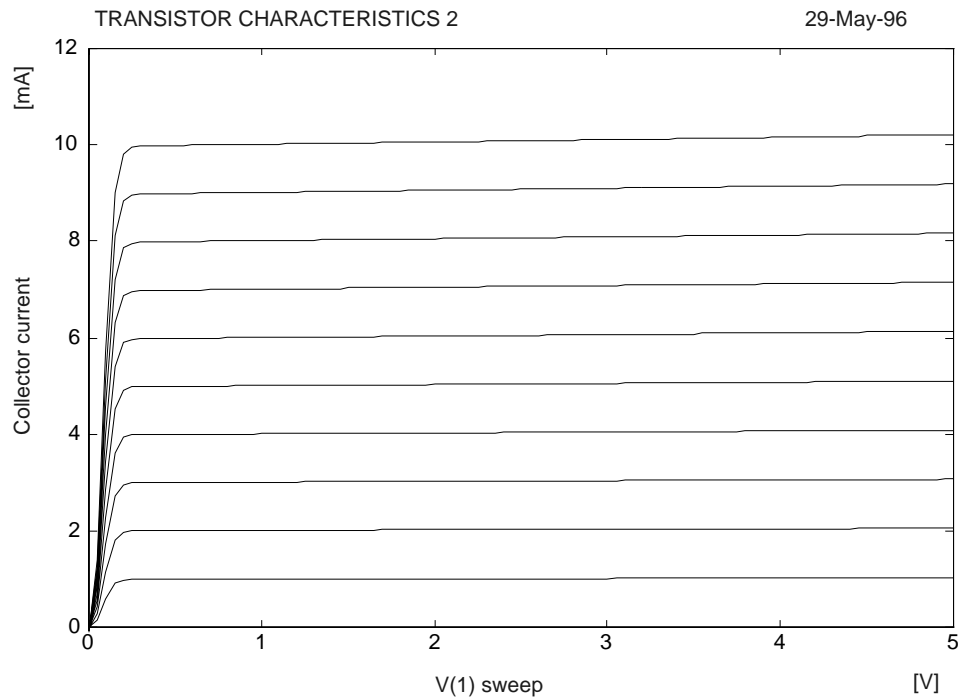


Fig. 1.2.20. Bipolar transistor output characteristic obtained using circuit from Fig. 1.2.18 and DC6.CKT file as input. Additional zero voltage source VA is used as an ammeter.

### 1.3. ac analysis

SPICE programs use the **.AC** statement to perform circuit analysis in the frequency domain. The **.AC** statement has the form:

```
.AC LIN|DEC|OCT Npoints F_start F_stop
```

One of the **LIN**, **DEC**, or **OCT** keywords must be used to specify linear (**LIN**) or logarithmic (**DEC**, **OCT**) scales. *Npoints* specifies the number of frequency points in the linear case, or the number of points per decade or per octave for the **DEC** and **OCT** cases. *F\_start* and *F\_stop* specify the beginning and end of the frequency sweep.

#### Example 1. Series resonant circuit

Let us analyze the simple *LC* resonant circuit shown in Fig. 1.3.1 using SPICE input file AC1.CIR shown in Fig. 1.3.2. The voltage source VIN has an ac magnitude of 1 mV. The dc bias of VIN is ignored during the ac analysis. The **.AC** statement defines the logarithmic sweep from 1 kHz to 100 kHz with 10 points per decade. In this case, 20 data points will be printed to an output file. Voltages on node 4 are printed by PSPICE as a magnitude of a complex number. This is an equivalent of the VM(4) term, where the magnitude is specifically requested. VP(4), VR(4), and VI(4) terms are for the phase and real and

imaginary parts of voltage V(4). A fragment of the output file, with printed voltages, is shown in Fig. 1.3.3. Note that V(4) and VM(4) are equivalent and the same data is printed.

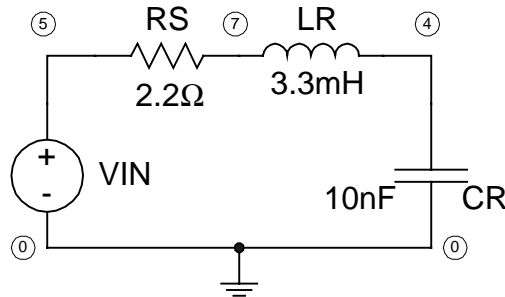


Fig. 1.3.1. Series resonant circuit.

```

RESONANT CIRCUIT      ; title line
VIN 5 0 1 AC 1m       ; voltage source between nodes 0 and 5 with 1V DC and 1mV AC values
RS 5 7 2.2            ; 2.2Ω resistor between nodes 5 and 7
LR 7 4 3.3m          ; 3.3mH inductor between nodes 7 and 4
CR 4 0 10n            ; 10nF capacitor between nodes 4 and 0
.AC DEC 10 1k 100k   ; .AC statement for log. sweep with 10 points per decade from 1kHz to 100kHz
.PRINT AC V(4) VM(4) ; print AC voltage of node 4 and magnitude of V(4) to an output file
+ VP(4) VR(4) VI(4) ; print line continuation - print phase, real, and imaginary part of V(4) voltage
.OPTION NOPAGE        ; option statement - remove page breaks in the output file
.END                  ; end of circuit description

```

Fig. 1.3.2. SPICE input file AC1.CIR for the frequency analysis.

```

****      AC ANALYSIS      TEMPERATURE = 27.000 DEG C
FREQ      V(4)      VM(4)      VP(4)      VR(4)      VI(4)
1.000E+03  1.001E-03  1.001E-03  1.800E+02  -1.001E-03  1.386E-07
1.259E+03  1.002E-03  1.002E-03  1.800E+02  -1.002E-03  1.747E-07
1.585E+03  1.003E-03  1.003E-03  1.800E+02  -1.003E-03  2.205E-07
1.995E+03  1.005E-03  1.005E-03  1.800E+02  -1.005E-03  2.787E-07
2.512E+03  1.008E-03  1.008E-03  1.800E+02  -1.008E-03  3.530E-07
3.162E+03  1.013E-03  1.013E-03  1.800E+02  -1.013E-03  4.487E-07
3.981E+03  1.021E-03  1.021E-03  1.800E+02  -1.021E-03  5.738E-07
5.012E+03  1.034E-03  1.034E-03  1.800E+02  -1.034E-03  7.405E-07
6.310E+03  1.055E-03  1.055E-03  1.799E+02  -1.055E-03  9.702E-07
7.943E+03  1.090E-03  1.090E-03  1.799E+02  -1.090E-03  1.303E-06
1.000E+04  1.150E-03  1.150E-03  1.799E+02  -1.150E-03  1.827E-06
1.259E+04  1.260E-03  1.260E-03  1.799E+02  -1.260E-03  2.764E-06
1.585E+04  1.486E-03  1.486E-03  1.798E+02  -1.486E-03  4.840E-06
1.995E+04  2.077E-03  2.077E-03  1.797E+02  -2.077E-03  1.190E-05
2.512E+04  5.617E-03  5.617E-03  1.789E+02  -5.616E-03  1.096E-04
3.162E+04  3.302E-03  3.302E-03  8.271E-01  3.302E-03  4.767E-05
3.981E+04  9.391E-04  9.391E-04  2.961E-01  9.391E-04  4.854E-06
5.012E+04  4.401E-04  4.401E-04  1.747E-01  4.400E-04  1.342E-06
6.310E+04  2.389E-04  2.389E-04  1.194E-01  2.389E-04  4.976E-07
7.943E+04  1.385E-04  1.385E-04  8.713E-02  1.385E-04  2.106E-07
1.000E+05  8.314E-05  8.314E-05  6.585E-02  8.314E-05  9.555E-08
JOB CONCLUDED
TOTAL JOB TIME      1.09

```

Fig. 1.3.3. Fragment of the output file generated by PSPICE using the AC1.CIR input file.

```

RESONANT CIRCUIT 2      ; title line
VIN 5 0 DC 1 AC 1      ; voltage source between nodes 0 and 5 with 1V DC and 1V AC values
RS 5 7 2.2              ; 2.2Ω resistor between nodes 5 and 7
LR 7 4 3.3M             ; 3.3mH inductor between nodes 7 and 4
CR 4 0 10N              ; 10nF capacitor between nodes 4 and 0
.AC LIN 500 25K 30K     ; .AC statement for linear sweep with 500 points from 25kHz to 30kHz
.PROBE V(4)             ; PSPICE only! save voltage of node 4 for PROBE post processor
.END                    ; end of circuit description

```

Fig. 1.3.4. PSPICE input file AC2.CIR for the frequency analysis.

In order to obtain a graphical output, the **.PROBE** statement in the PSPICE program can be used. A modification of the AC1.CIR file is shown in Fig. 1.3.4. VIN is the ac voltage source with a magnitude of 1 V. The dc bias of VIN is ignored during the ac analysis. The **.AC** statement defines the linear sweep from 25 kHz to 30 kHz with 500 points. The input file AC2.CIR is for PSPICE only and voltages on node 4 are saved for the PROBE postprocessor. In the case of SPICE3, the statement **.SAVE V(4)** should be used instead. The graphical output obtained with the PROBE postprocessor is shown in Fig. 1.3.5 using a lin-lin scale. Magnitude VM(4), real part VR(4), and imaginary part VI(4) are plotted using volts, and phase VP(4) is plotted using degrees.

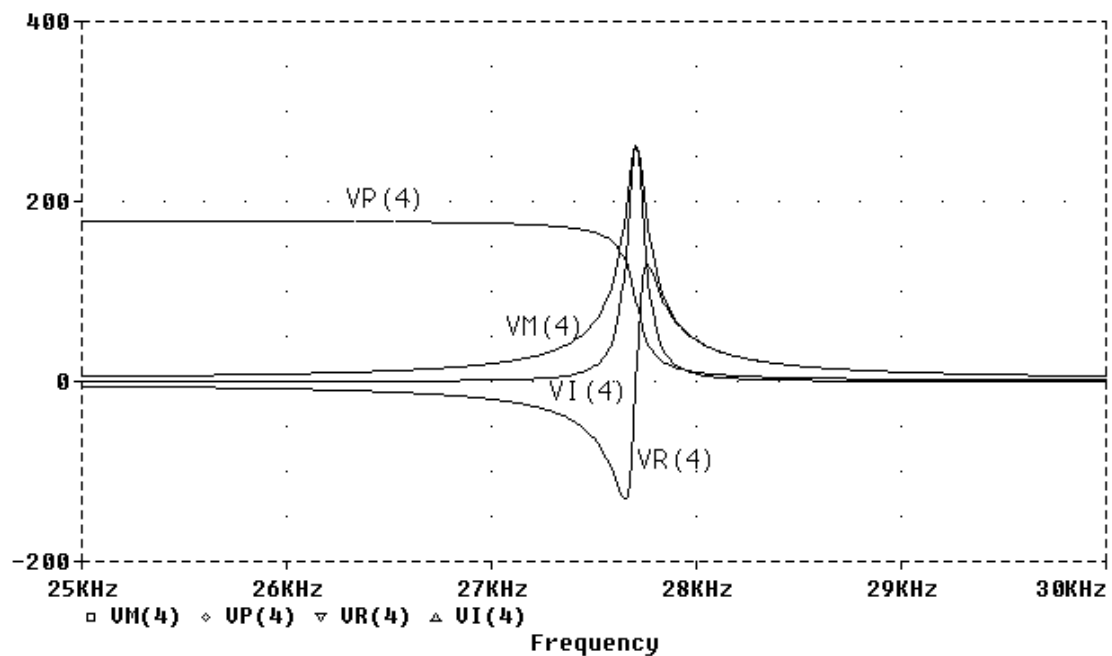


Fig. 1.3.5. Magnitude, phase, real part, and imaginary part of V(4) voltage versus frequency obtained using PSPICE and PROBE with AC2.CIR as the input file.

## 1.4. Transient Analysis

The transient analysis is initiated by the **.TRAN** statement, and a shape of excitation is defined in independent sources using the keywords **PULSE**, **SIN**, **EXP**, **PWL**, and **SFFM**. The **.TRAN** statement has the format:

**.TRAN**[**/OP**] *Tstep Tstop* [*Tstart*[*Maxstep*]] [**UIC**]

The optional **/OP** switch causes the initial values to be printed to the output file. *Tstep* defines the time step for the output. The internal time step is usually much smaller. *Tstop* defines the stop time. The default value for *Tstart* is zero and can be changed by specifying an optional *Tstart* parameter. As default SPICE must calculate at least 50 time steps for each analysis. This value can be changed using an optional *Maxstep* parameter. Usually, before transient analysis, the dc biasing point is calculated and used as initial condition for transient analysis. This biasing point calculation can be skipped using the keyword **UIC**. In this case all initial values will be set to zero unless other values are specified by the **.IC** statement.

The definition of waveforms is included in independent voltage or current sources after node specification. Five types of waveforms can be used in the SPICE programs:

**PULSE**(V1 V2 TD TR TF PW PER) - pulse waveform

V1 - initial value of voltage or current  
 V2 - pulsed value of voltage or current  
 TD - delay time  
 TR - rise time  
 TF - fall time  
 PW - pulse width  
 PER - period

**SIN**(VO VA FREQ TD DF ) - sinusoidal waveform

VO - offset  
 VA - amplitude  
 FREQ - frequency  
 TD - delay  
 DF - damping factor

**EXP**(V1 V2 TD1 TAU1 TD2 TAU2) - exponential waveform

V1 - initial value  
 V2 - pulsed value  
 TD1 - rise delay time  
 TAU1 - rise time constant  
 TD2 - fall delay time  
 TAU2 - fall time constant

**PWL**(T1 V1 [Tn Vn] ... ) - piecewise linear waveform

Tn - time at corner  
 Vn - voltage or current at the corner

**SFFM**(VO VA FC MDI FS) - single-frequency FM waveforms  
 VO - offset  
 VA - amplitude  
 FC - carrier frequency  
 MDI - modulation index  
 FS - signal frequency

For a more detailed description of these waveforms and their parameters, see Chapter 5.

### Example 1. Simple RC circuit

Let us consider the transient response of a simple RC circuit shown in Fig. 1.4.1. To perform transient analysis two elements were added to the input file. The **PULSE** waveform was added to the independent voltage source VIN definition, and the **.TRAN** statement was included in the input file. Analysis was performed using PSPICE, and results are shown in Fig. 1.4.3. In the case of SPICE3, the **.PROBE** statement should be replaced by the **.PRINT** or **.SAVE** statement.

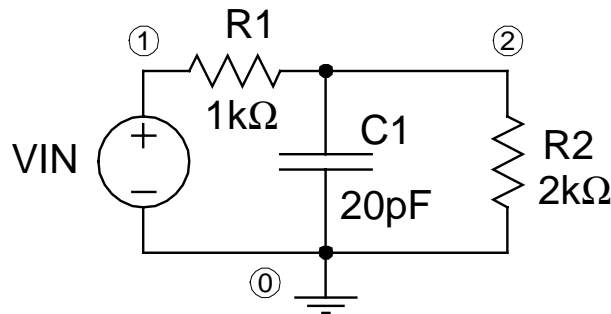


Fig. 1.4.1. Simple RC circuit.

```

Simple RC Circuit           ; title line
R1 1 2 1k                  ; 1kΩ resistor between nodes 1 and 2
R2 2 0 2k                  ; 2kΩ resistor between nodes 2 and 0
C1 2 0 20p                 ; 20pF capacitor between nodes 2 and 0
VIN 1 0 PULSE (-3 6       ; voltage source between nodes 1 and 0 with PULSE waveform VL=-3V VH=6V
+ 10n 10n 10n 100n 200n) ; Tdelay=10ns Trise=10ns Tfall=10ns Twidth=100ns Period=200ns
.TRAN 1n 600n              ; .TRAN statement for transient analysis from 0 to 600ns with 1ns step
.PROBE V(1) V(2)           ; PSPICE only! save voltages of node 1 and 2 for PROBE post processor
.END                        ; end of circuit description

```

Fig. 1.4.2. PSPICE input file TR1.CIR for the transient analysis of the simple RC circuit from Fig. 1.4.1.

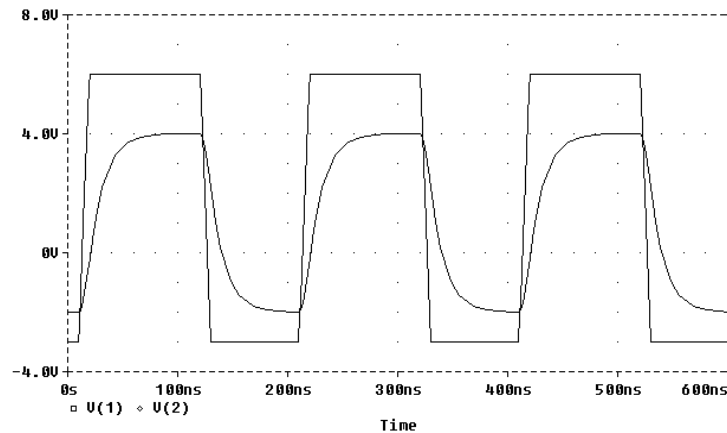


Fig. 1.4.3. Input and output waveforms of the simple RC circuit from Fig. 1.4.1 obtained using PSPICE and PROBE with the TR1.CIR input file.

### Example 2. Voltage quadrupler circuit

Let us consider the transient response of the voltage quadrupler circuit shown in Fig. 1.4.4.

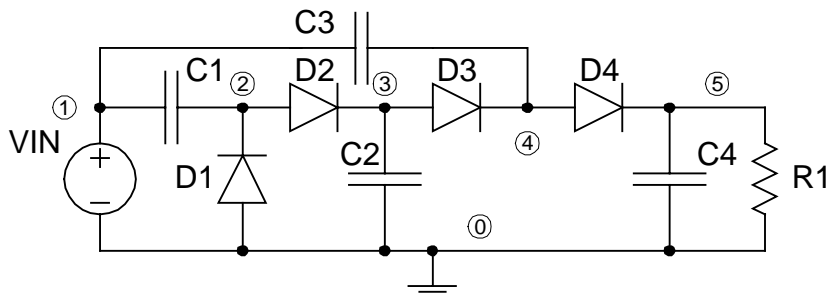


Fig. 1.4.4. Voltage quadrupler circuit.

```

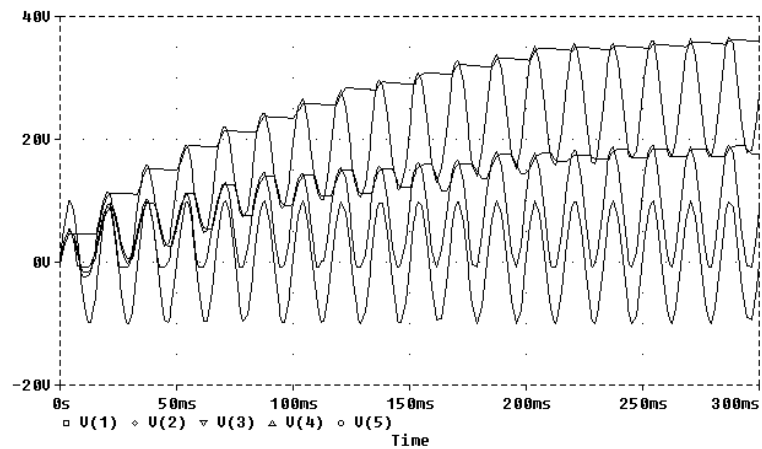
Voltage Quadrupler Circuit           ; title line
.MODEL DMOD D IS=1.0E-16             ; model DMOD line for diode only saturation current IS specified
C1 1 2 1u                             ; 1uF capacitor C1 between nodes 1 and 2
C2 3 0 1u                             ; 1uF capacitor C2 between nodes 3 and 0
C3 1 4 1u                             ; 1uF capacitor C3 between nodes 1 and 4
C4 5 0 1u                             ; 1uF capacitor C4 between nodes 5 and 0
D1 0 2 DMOD                          ; diode D1 with DMOD connected between nodes 0 and 2
D2 2 3 DMOD                          ; diode D2 with DMOD connected between nodes 2 and 3
D3 3 4 DMOD                          ; diode D3 with DMOD connected between nodes 3 and 4
D4 4 5 DMOD                          ; diode D4 with DMOD connected between nodes 4 and 5
R1 5 0 1MEG                          ; resistor R1 with 1MW resistance connected between nodes 5 and 0
VIN 1 0 SIN (0 10 60)                ; sinusoidal voltage source between nodes 1 and 0
.TRAN 1m 0.3                          ; transient analysis from 0 to 300ms with 1ms step
.PROBE V(1) V(2) V(3) V(4) V(5)      ; PSPICE only! save voltages of five nodes for PROBE post processor
.PRINT TRAN V(1) V(2) V(3) V(4)      ; save data of five nodes to the output file TR2.OUT
V(5)                                  ; this data can be then read by other programs(for example MATLAB)
* comment line                        ; end of circuit description
.END

```

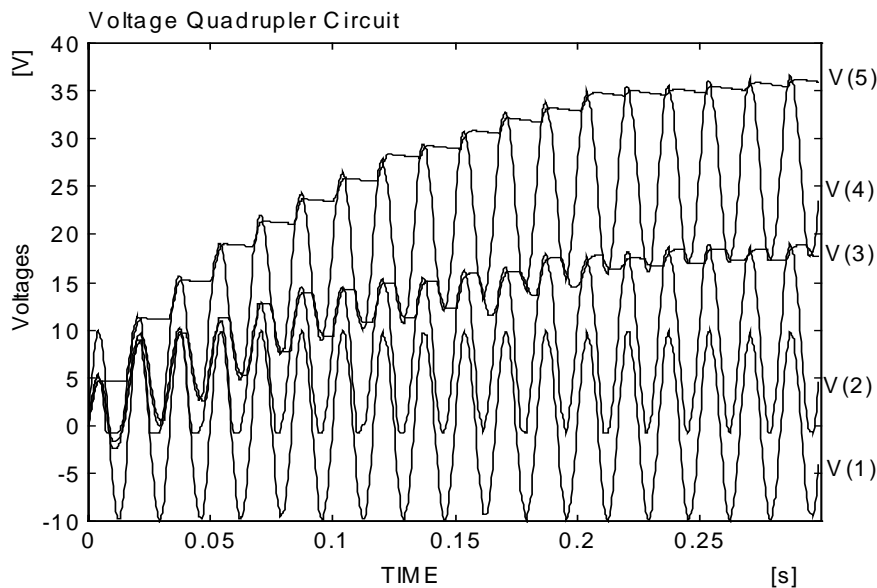
Fig. 1.4.5. PSPICE input file TR2.CIR for transient analysis of the voltage quadrupler circuit.

To perform transient analysis, two elements were added to the input file. The **SIN** waveform was added to the independent voltage source **VIN** definition, and the **.TRAN** statement was included in the input file. The sinusoidal voltage source is defined using the keyword **SIN** with three parameters: dc offset is set to 0, magnitude to 10 V, and frequency to 60 Hz. Other parameters such as delay, damping factor, and phase use default zero values. Semiconductor elements such as diodes require definition of model parameters. This is done using the **.MODEL** statement. Usage of the **.MODEL** statement is described in Section 1.6.

Analysis was performed using PSPICE, and results are shown in Fig. 1.4.6. Both the PROBE and the MATLAB outputs are presented there. In the case of SPICE3, the **.PROBE** statement can be replaced by the **.SAVE** statement.



(a)



(b)

Fig. 1.4.6. Waveforms of the voltage quadrupler circuit from Fig. 1.4.4 obtained using PSPICE and the TR2.CIR input file; (a) graph obtained using the PROBE program, and (b) graph obtained using the **.PRINT** statement and the MATLAB program.

## 1.5. Subcircuits

When the same fragment of a circuit is used repetitively, then this fragments can be described in the input file only once using the **.SUBCKT** statement. Each appearance of the subcircuit is then reduced to a one-line statement which starts with the letter **X**.

### Example 1. Ring oscillator

Use of the **.SUBCKT** statement is illustrated with the ring oscillator circuit shown in Fig. 1.5.1. The input file for the circuit is shown in Fig. 1.5.2. First, the model parameters for both the PMOS and the NMOS transistor are specified.

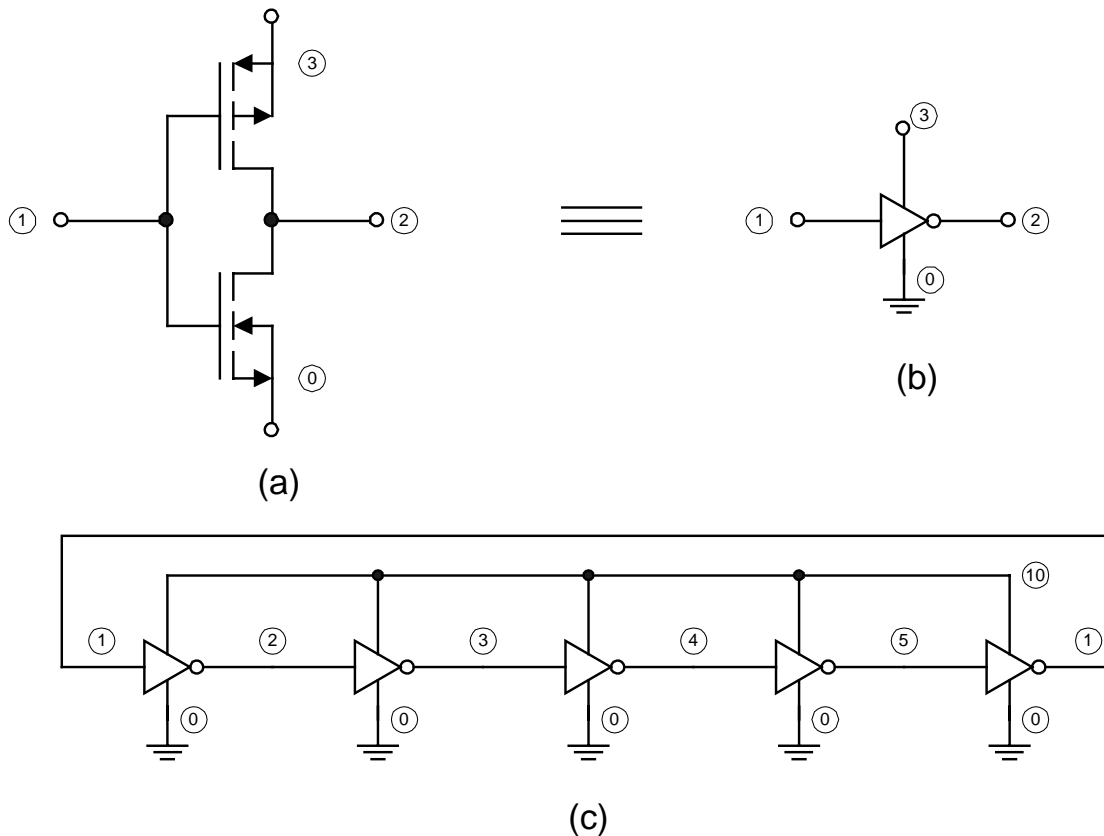


Fig. 1.5.1. Ring oscillator: (a) internal circuitry of the subcircuit INVERTER, (b) macro diagram of the subcircuit INVERTER, and (c) circuit of the ring oscillator using subcircuit INVERTER.



```

RING OSCILLATOR ; title line
.MODEL MODEL_P PMOS ; PMOS transistor model
+ ( VTO=-1 KP=200U GAMMA=0.35 ; parameters of PMOS transistor model
LAMBDA=0.02 ; parameters of PMOS transistor model
+ CBS=10P CBD=10P CJSW=25P ; parameters of PMOS transistor model
+ CGSO=30P CGDO=30P CGBO=40P ) ; NMOS transistor model
.MODEL MODEL_N NMOS ; parameters of NMOS transistor model
+ ( VTO=1 KP=400U GAMMA=0.25 LAMBDA=0.01 ; parameters of NMOS transistor model
+ CBS=10P CBD=10P CJSW=25P ; parameters of NMOS transistor model
+ CGSO=30P CGDO=30P CGBO=40P ) ; subcircuit definition
.SUBCKT INVERTER 1 2 3 ; subcircuit description
M1 2 1 3 3 MODEL_P W=10U L=3U ; subcircuit description
M2 2 1 0 0 MODEL_N W=5U L=3U ; end subcircuit definition
.ENDS INVERTER ; subcircuit INVERTER connected to nodes 1, 2 and 10
XC 3 4 10 INVERTER ; subcircuit INVERTER connected to nodes 2, 3 and 10
XD 4 5 10 INVERTER ; subcircuit INVERTER connected to nodes 3, 4 and 10
XE 5 1 10 INVERTER ; subcircuit INVERTER connected to nodes 4, 5 and 10
XA 1 2 10 INVERTER ; subcircuit INVERTER connected to nodes 5, 1 and 10
XB 2 3 10 INVERTER ; power supply of 5V between nodes 0 and 10
VSUP 10 0 5 ; transient analysis 0 to 150ns with 1ns step
.TRAN 1NS 150NS UIC ; initial voltages on five nodes before the analysis
.IC V(1)=2 V(2)=1 V(3)=-1 V(4)=-1 V(5)=1 ; printing to an output file voltages on five nodes
.PRINT TRAN V(1) V(2) V(3) V(4) V(5) ; end of circuit description
.END

```

Fig. 1.5.2. PSPICE input file SU1.CIR for transient analysis of the ring oscillator circuit from Fig. 1.5.1.

The subcircuit definition starts with the line **.SUBCKT INVERTER 1 2 3**, and it is terminated with the line **.ENDS INVERTER**. The name of subcircuit **INVERTER** is required after the keyword **.SUBCKT** and it is optional at the end of subcircuit definition. Numbers specified at the end of the **.SUBCKT** line are node numbers with which the subcircuit is connected with the external circuitry. See Fig. 1.5.1. In our example, all four nodes are accessible from outside, but the “0” node for ground must not be declared. The sequence of node numbers is important. When the circuit is called, the same sequence is used. In our example the subcircuit is called five times using statements which start with the letter **X**. For example, the first call has the form **XC 3 4 10 INVERTER**. This instance of subcircuit has name **C**, and nodes 1, 2, 3 of subcircuit correspond to nodes 3, 4, 10 of the external circuit. After node declaration, the subcircuit name must be listed, since in general more than one subcircuit definition can be used.

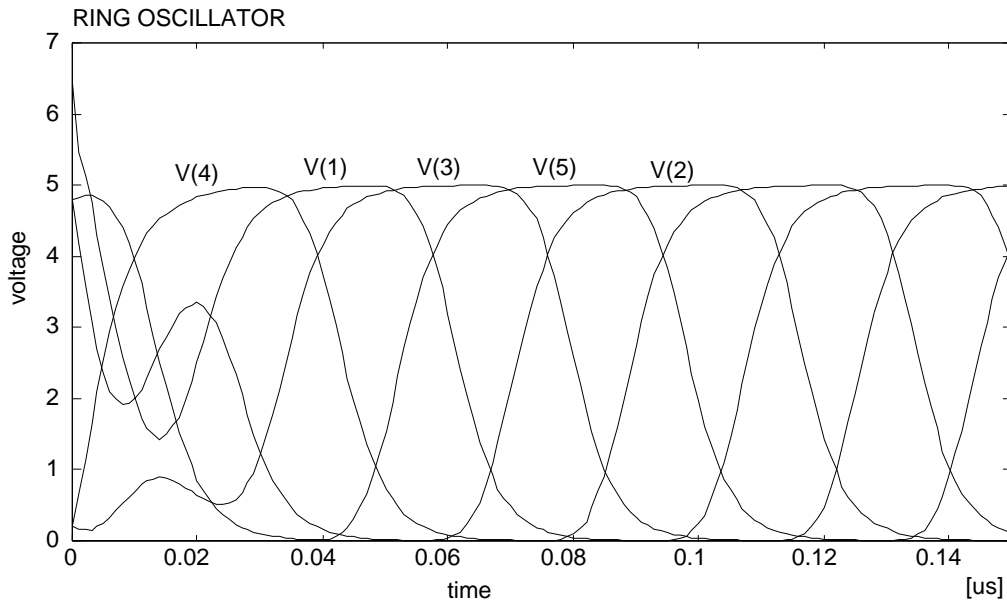


Fig. 1.5.3. Waveforms of the ring oscillator circuit from Fig. 1.5.1 with the TR2.CIR input file obtained using PSPICE with the **.PRINT** statement and MATLAB as graphical postprocessor.

In this example the transient analysis is done using the **.TRAN** statement with keyword **UIC** for using initial node voltages specified in the **.IC** statement. Different random voltages are set so the circuit can start to oscillate. This would not be possible with identical voltages on all nodes. The output waveform is presented in Fig. 1.5.3.

## 1.6. Device models and Monte Carlo analysis

All semiconductor devices require the **.MODEL** statement to specify parameters of the device model in the form:

**.MODEL** *Name* *Type* [*List\_of\_parameters*]

*Name* is the name of the model. It is a good habit to start the *Name* with the same letter as the *Type* of the model. After *Type*, *List\_of\_parameters* specifies the parameters in the format:

*Name\_of\_parameter* = *Value*. For example:

**.MODEL** NPNBIG NPN BF=120 BR=20 IS=1E-14 CJE=10P CJC=3P

is the **.MODEL** line for an *npn* transistor with  $\beta_{\text{FORWARD}} = 120$ ,  $\beta_{\text{REVERSE}} = 20$ , saturation current = 0.01 pA, capacitance the base-emitter junction = 10 pF, and capacitance of the base-collector junction = 3 pF. Only five parameters are specified; for others about 40 default values will be used by SPICE programs. See reference material in the Chapter 6 for detailed information about models of semiconductor devices. Table 1.6.1 shows keywords allowed for *Type*.

**Table 1.6.1. Keywords indication model type**

<i>Type</i>	<i>Description</i>	<i>Restrictions</i>
<b>CAP</b>	capacitor	
<b>CORE</b>	nonlinear magnetic core (transformer)	PSPICE only
<b>CSW</b>	current-controlled switch	SPICE3 only
<b>D</b>	diode	
<b>GASFET</b>	GaAs field-effect transistor with n-type channel	PSPICE only
<b>IND</b>	inductor	
<b>ISWITCH</b>	current-controlled switch	PSPICE only
<b>LPNP</b>	lateral <i>npn</i> transistor	PSPICE only
<b>LTRA</b>	lossy transmission line	SPICE3 only
<b>NJF</b>	JFET with <i>n</i> -type channel	
<b>NMF</b>	<i>n</i> -channel MESFET	SPICE3 only
<b>NMOS</b>	MOS transistor with n-type channel	
<b>NPN</b>	<i>npn</i> bipolar transistor	
<b>PJF</b>	JFET with <i>p</i> -type channel	
<b>PMF</b>	<i>p</i> -channel MESFET	SPICE3 only
<b>PMOS</b>	MOS transistor with <i>n</i> -type channel	
<b>PNP</b>	<i>npn</i> bipolar transistor	
<b>RES</b>	resistor	
<b>SW</b>	voltage controlled switch	SPICE3 only
<b>URC</b>	uniform distributed <i>RC</i> line	SPICE3 only
<b>VSWITCH</b>	voltage controlled switch	PSPICE only

In addition newer versions of PSPICE have several models for digital circuits which start with the letter **D** or **U**.

In the case of passive elements such as resistors, capacitors, and inductors **.MODEL** statements are not required, but they can be used when advanced models are used. **.MODEL** statements for passive elements are also useful when Monte Carlo analysis is performed.

### Example 1. Low-pass 8th order Chebyshev filter

Let us consider the low-pass 8th order Chebyshev filter shown in Fig. 1.6.1 and the input file MC1.CIR as shown in Fig. 1.6.2. The Monte Carlo analysis is implemented in the PSPICE program only. To obtain similar results with SPICE3, advanced programming with SPICE3 Interactive Language would be required. A block diagram is shown in Fig. 1.6.1(e) with four second-order Sallen-Key filters. Figure 1.6.1(a) presents a subcircuit for an ideal operational amplifier with input resistance of 1 M $\Omega$  and output resistance of 1 k $\Omega$ . Voltage-controlled voltage source EOA is used to obtain the gain of 10<sup>6</sup>. There are three external nodes (1, 2, and 12) and one internal node (11). The macro diagram of this subcircuit is shown in Fig. 1.6.1(b). The subcircuit has the name IOPAMPS, and it is defined in the input file MC1.CIR using lines 4 through 8. Numbers with circles in Fig. 1.6.1(a), (c), and (e) indicate node numbers. Numbers without circles in Fig. 1.6.1(b) and (d) indicate the calling order of subcircuits.

Four subcircuits of second-order Sallen-Key filters are defined in the input file using lines 9 through 40. All these subcircuits have the same circuit topology as presented in Fig. 1.6.1(c) and different capacitor values. The circuit was designed using the FILTER program [22] for the cut off frequency of

3 kHz or  $\omega_c = 18,850$  rad/s. Assuming that all resistors have the value of  $R1 = R2 = R = 10$  k $\Omega$  capacitor values can be calculated using formulas:

$$C1 = \frac{2Q}{R\omega_o\omega_c} = 10.61 \frac{Q}{\omega_o} \text{ [nF]} \quad C2 = \frac{1}{2QR\omega_o\omega_c} = 2.6525 \frac{1}{Q\omega_o} \text{ [nF]}$$

Note that **.MODEL** lines for resistors and capacitors have scaling factors 1k and 1n respectively. The keyword **DEV** is used in the model lines to set a 5% fluctuation of resistances and capacitances for each consecutive run of Monte Carlo analysis. Default uniform distribution is assumed. The Gauss or uniform type distribution can be set in **.MODEL** lines using keywords **GAUSS** and **UNIFORM**, respectively. The keyword **DEV** means that for each resistor a different random value is chosen. In order to set the same random multiplication factor for all elements of a given model the keyword **LOT** has to be used. Both keywords can be used jointly. For example **DEV = 8% LOT = 3%** means that on top of the 3% random deviation for all elements, an 8% random deviation is set for each element. In the **.MC** statement, 10 **AC** indicates 10 Monte Carlo runs with randomly chosen parameters for ac analysis. Keyword **YMAX** finds the greatest difference in each waveform from nominal output. Keyword **LIST** causes printing of model parameters for each run. Keyword **OUTPUT ALL** generates output for all runs including nominal.

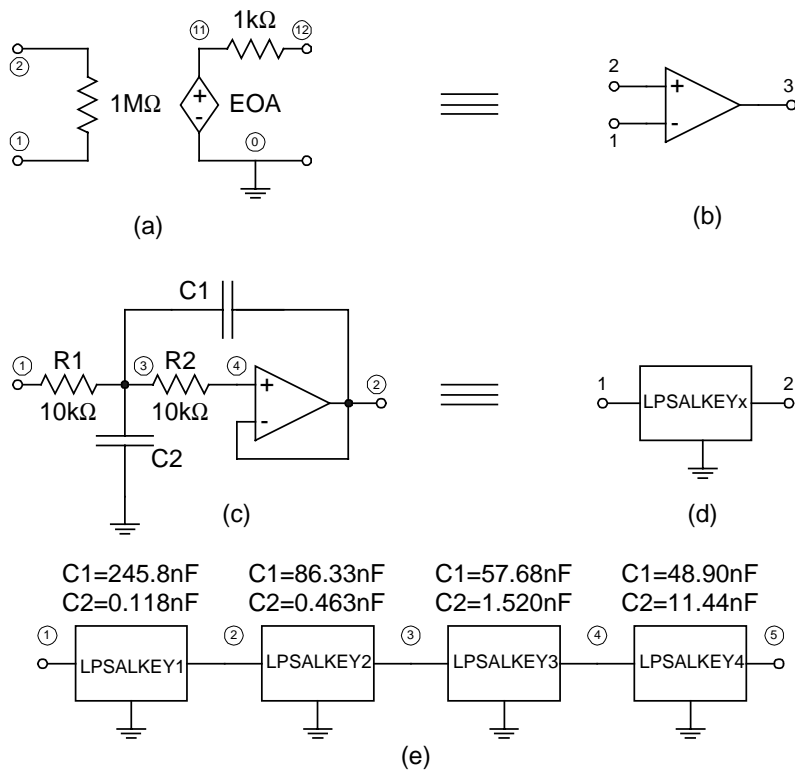


Fig. 1.6.1. 8th order Chebyshev active filter: (a) internal circuitry of the subcircuit IOPAMP, (b) macro diagram of the subcircuit IOPAMP, (c) internal circuitry of subcircuits LPSALKEYx, (d) macro diagram of subcircuits LPSALKEYx, and (e) circuit of the 8th order active filter using subcircuits LPSALKEYx.

```

Low Pass Filter Circuit ; title line
* This file is generated by the FILTER program ; comment line
* see IEEE Trans. on Education. vol E-35, ; comment line
* no 4, pp. 351-361, 1992 ; comment line
.SUBCKT IOPAMS 1 2 12 ; subcircuit declaration of ideal operational amplifier
  RIN 1 2 1MEG ; input resistance 1MΩ
  ROUT 11 12 1K ; output resistance 1kΩ
  EOA 11 0 2 1 1.0E+6 ; voltage source controlled by voltage
.ENDS IOPAMS ; end of OPAMP subcircuit definition
.SUBCKT LPSALKEY1 1 2 ; subcircuit of second order low pass Sallen-Key filter
* K = 1 Q =22.87040 wo =0.987002 ; comment line: gain=1, Q=22.87, and ω0=0.987
  R1 1 3 RMOD 10 ; resistor R1 uses RMOD with 10 multiplication factor
  R2 3 4 RMOD 10 ; resistor R2 uses RMOD with 10 multiplication factor
  C1 2 3 CMOD 245.8 ; capacitor C1 uses CMOD with 245.8 multiplication factor
  C2 4 0 CMOD 0.118 ; capacitor C2 uses CMOD with 0.118 multiplication factor
  XA1 2 4 2 IOPAMS ; call to IOPAMPS subcircuit
.ENDS LPSALKEY1 ; end of second order low pass filter definition
.SUBCKT LPSALKEY2 1 2 ; subcircuit of second order low pass Sallen-Key filter
* K = 1 Q =6.825080 wo =0.838794 ; comment line: gain=1, Q=6.825, and ω0=0.8388
  R1 1 3 RMOD 10 ; resistor R1 uses RMOD with 10 multiplication factor
  R2 3 4 RMOD 10 ; resistor R2 uses RMOD with 10 multiplication factor
  C1 2 3 CMOD 86.33 ; capacitor C1 uses CMOD with 86.33 multiplication factor
  C2 4 0 CMOD 0.463 ; capacitor C2 uses CMOD with 0.463 multiplication factor
  XA1 2 4 2 IOPAMS ; call to IOPAMPS subcircuit
.ENDS LPSALKEY2 ; end of second order low pass filter definition
.SUBCKT LPSALKEY3 1 2 ; subcircuit of second order low pass Sallen-Key filter
* K = 1 Q =3.079813 wo =0.566473 ; comment line: gain=1, Q=3.0798, and ω0=0.566
  R1 1 3 RMOD 10 ; resistor R1 uses RMOD with 10 multiplication factor
  R2 3 4 RMOD 10 ; resistor R2 uses RMOD with 10 multiplication factor
  C1 2 3 CMOD 57.68 ; capacitor C1 uses CMOD with 57.68 multiplication factor
  C2 4 0 CMOD 1.520 ; capacitor C2 uses CMOD with 1.520 multiplication factor
  XA1 2 4 2 IOPAMS ; call to IOPAMPS subcircuit
.ENDS LPSALKEY3 ; end of second order low pass filter definition
.SUBCKT LPSALKEY4 1 2 ; subcircuit of second order low pass Sallen-Key filter
* K = 1 Q =1.033654 wo =0.224263 ; comment line: gain=1, Q=1.0336, and ω0=0.224
  R1 1 3 RMOD 10 ; resistor R1 uses RMOD with 10 multiplication factor
  R2 3 4 RMOD 10 ; resistor R2 uses RMOD with 10 multiplication factor
  C1 2 3 CMOD 48.90 ; capacitor C1 uses CMOD with 48.90 multiplication factor
  C2 4 0 CMOD 11.44 ; capacitor C2 uses CMOD with 11.44 multiplication factor
  XA1 2 4 2 IOPAMS ; call to IOPAMPS subcircuit
.ENDS LPSALKEY4 ; end of second order low pass filter definition
*** main circuit begins *** ; comment line
.MODEL RMOD RES (R = 1k DEV = 5% ) ; resistor model RMOD with 10kW resistance and 5% DEV
.MODEL CMOD CAP (C = 1n DEV = 5% ) ; capacitor model RMOD with 10kW resistance and 5% DEV
V1 1 0 AC 1 ; AC voltage source between nodes 1 and 0 with 1V magnitude
X1 1 2 LPSALKEY1 ; call to subcircuit LPSALKEY connected between nodes 1, 2
X2 2 3 LPSALKEY2 ; call to subcircuit LPSALKEY connected between nodes 2, 3
X3 3 4 LPSALKEY3 ; call to subcircuit LPSALKEY connected between nodes 3, 4
X4 4 5 LPSALKEY4 ; call to subcircuit LPSALKEY connected between nodes 4, 5
.AC DEC 100 100 10k ; AC analysis 100 points from 100Hz to 10kHz
.PRINT AC V(5) ; print AC voltages on node 5 to the output file
.PROBE V(5) ; save voltages on node 5 for the PROBE program
.MC 10 AC V(5) YMAX LIST OUTPUT ALL ; set Monte Carlo analysis with 10 runs and save voltage V(5)
.END ; end of circuit description

```

Fig. 1.6.2. PSPICE input file MC1.CIR for the frequency analysis of the 8th order Chebyshev active filter from Fig. 1.6.1.

The frequency characteristics of the 8th order active filter obtained for nominal parameters, without the `.MC` statement, are shown in Fig. 1.6.3. The same characteristics using the `.MC` statement are shown in Fig. 1.6.4. Note the significant differences from nominal characteristics even through only 5% variation of circuit elements was allowed.

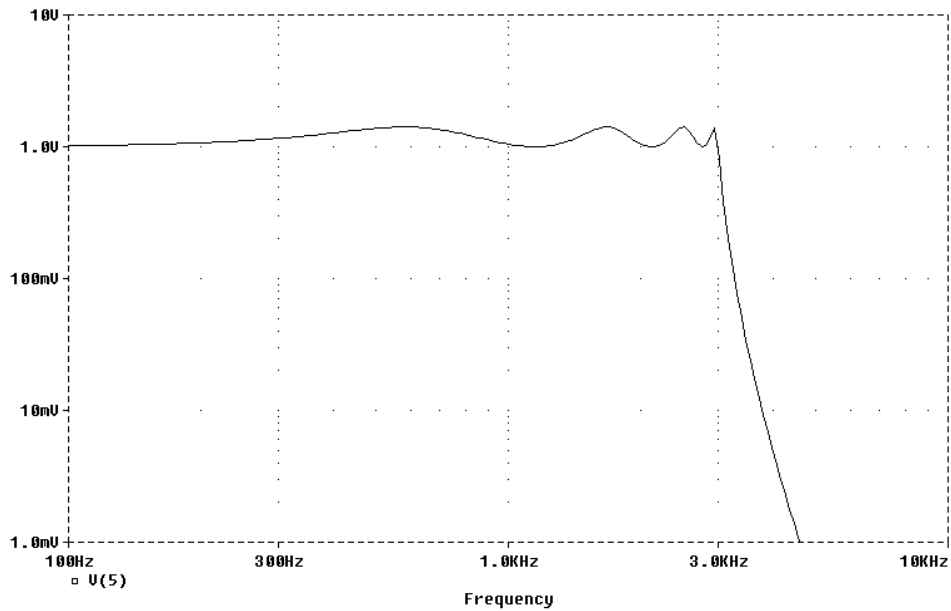


Fig. 1.6.3. Frequency response of the 8th order low-pass filter from Fig. 1.6.1 with the MC1.CIR input file, without `.MC` statement, obtained using PSPICE and PROBE graphical postprocessor.

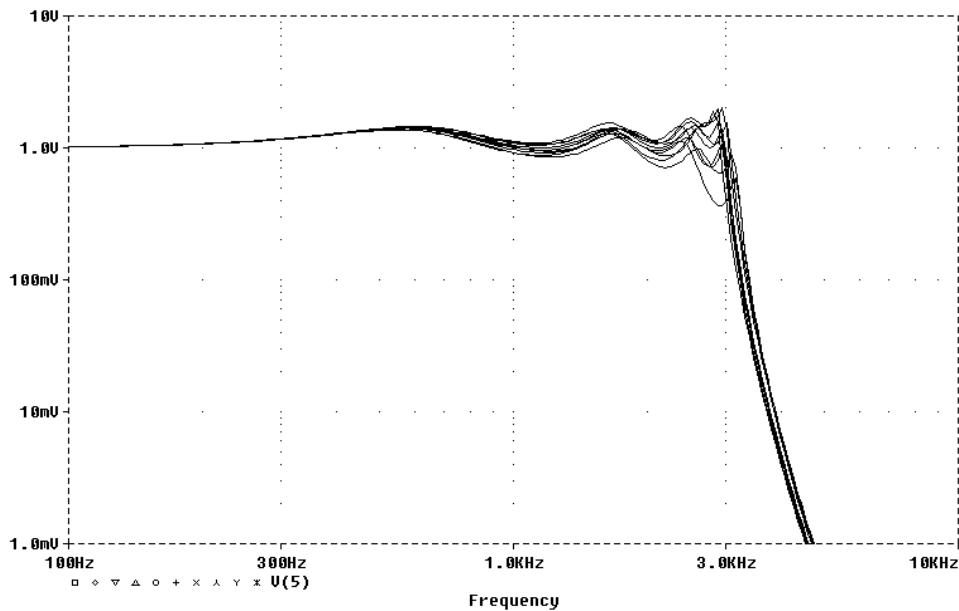


Fig. 1.6.4. Frequency response of the 8th order low-pass filter from Fig. 1.6.1 with the MC1.CIR input file, with `.MC` statement, obtained using PSPICE and PROBE graphical postprocessor.

This chapter has presented a “fast start” introduction to SPICE analysis with examples of the most important features of SPICE including circuit description techniques and analysis models. More detail and many further examples on all the capacity of SPICE appear in Chapter 2.

### 1.7. References

1. Banzhaf, Walter, *Computer-Aided Circuit Analysis Using PSpice*, Prentice Hall, Englewood Cliffs, NJ: 1992.
2. Brumgnach, Edward, *PSpice for Windows*, Delmar, Albany NY: 1995.
3. Conant, Roger, *Engineering Circuit Analysis with PSpice and Probe* McGraw-Hill, New York: 1993.
4. Ferris, Clifford D., and Jerry C. Hamann, *SPICE for Electronics*, West, St. Paul, MN: 1995.
5. Goody, Roy W., *PSpice for Windows - A Circuit Simulation Primer* Prentice Hall, Englewood Cliffs, NJ: 1995.
6. Intusoft, *ICAP/4 IsSpice4 User's guide*, Intusoft, 1996.
7. Jaeger, Richard, *Microelectronic Circuit Design*, McGraw-Hill, New York: 1997.
8. Keown, John, *PSpice and Circuit Analysis*, (2nd ed.) Macmillan, New York: 1993.
9. Kielkowski, Ron M., *Inside SPICE - Overcoming the Obstacles of Circuit Simulation*, McGraw-Hill, New York: 1994.
10. Lamey, Robert, *The Illustrated Guide to PSpice* Delmar, Albany NY: 1995.
11. MicroSim Corporation, *Circuit Analysis Reference for PSPICE*, version 6.2 April 1995.
12. Monssen, Franz, *PSpice with Circuit Analysis*, Macmillan, New York: 1993.
13. Morris, E. F., *Introduction to PSpice with Student Exercise Disk*, Houghton Mifflin, Boston: 1991.
14. Quarles, T., A. R. Newton, D. O. Pederson, and A. Sangiovanni-Vincentelli, *SPICE3 Version 3f3 User's Manual*, Department of Electrical and Computer Sciences, University of California, Berkeley, May 1993.
15. Rashid, Muhammad H., *SPICE for Circuit and Electronics using PSPICE*, (2nd ed.) Prentice Hall, Englewood Cliffs, NJ: 1995.
16. Roberts, Gordon W. and Adel S. Sedra, *SPICE for Microelectronic Circuits*, (3d ed.), Oxford University Press, New York: 1992.
17. Roberts, Gordon W. and Adel S. Sedra, *SPICE for Microelectronic Circuits*, (4th ed.), Oxford University Press, New York: 1997.
18. Rohrer, R. H., “Circuit Simulation - The Early Years,” *IEEE Circuit and Devices Magazine*, May 1992.
19. Thrope, Thomas W., *Computerized Circuit Analysis with SPICE - A Complete Guide to SPICE with Applications*, Wiley, New York: 1991.
20. Tuinega, Paul W., *SPICE - A Guide to Circuit Simulation & Analysis Using Pspice*, (3d ed.) Prentice Hall, Englewood Cliffs, NJ: 1995.
21. Vladimirescu, Andrei, *The SPICE Book* Wiley, New York: 1994.
22. Wilamowski, Bogdan, *FILTER program*, software available at <http://nn.uwyo.edu/sp/>.