1 Example Circuits

Lets start with an ngspice example to walk you through the basic features of ngspice using its command line user interface. The operation of ngspice will be illustrated through the example.

The example uses the simple one-transistor amplifier circuit illustrated in Fig. 1. This circuit is constructed entirely with ngspice compatible devices and

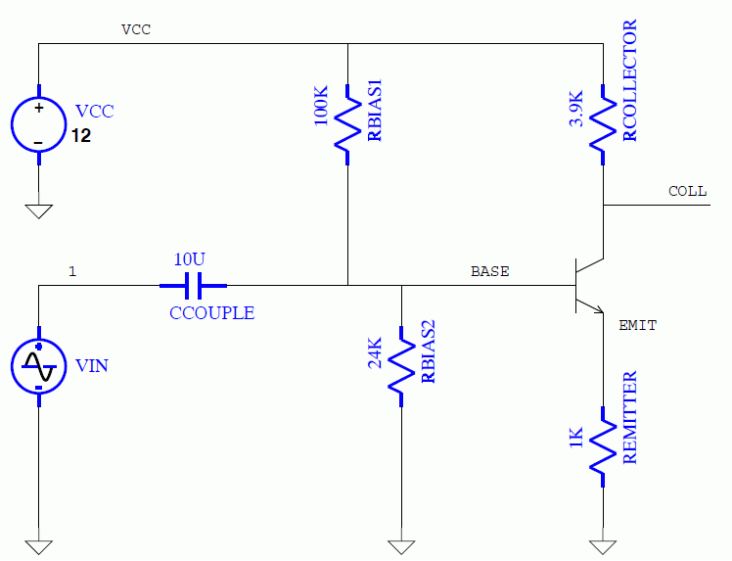

Figure 1: Transistor Amplifier Simulation Example

is used to introduce basic concepts, including:

- Invoking the simulator:
- Running simulations in different analysis modes
- Printing and plotting analog results
- Examining status, including execution time and memory usage
- Exiting the simulator

2 AC coupled transistor amplifier

The circuit shown in Fig. 1 is a simple one-transistor amplifier. The input signal is amplified with a gain of approximately -(Rc/Re) = -(3.9K/1K) = -3.9. The circuit description file for this example is shown below.

Example:

```
A Berkeley SPICE3 compatible circuit
*
* This circuit contains only Berkeley SPICE3 components.
*
* The circuit is an AC coupled transistor amplifier with
* a sinewave input at node "1", a gain of approximately -3.9,
* and output on node "coll".
*
.tran 1e-5 2e-3
*
vcc vcc 0 12.0
vin 1 0 0.0 ac 1.0 sin(0 1 1k)
ccouple 1 base 10uF
rbias1 vcc base 100k
rbias2 base 0 24k
q1 coll base emit generic
rcollector vcc coll 3.9k
remitter emit 0 1k
*
.model generic npn
*
.end
```

To simulate this circuit, invoke the simulator on this circuit as follows:

```
$ ngspice xspice_c1.cir
```

After a few moments, you should see the ngspice prompt:

```
ngspice 1 ->
```

At this point, ngspice has read-in the circuit description and checked it for errors. If any errors had been encountered, messages describing them would have been output to your terminal. Since no messages were printed for this circuit, the syntax of the circuit description was correct.

To see the circuit description read by the simulator you can issue the following command:

```
ngspice 1 -> listing
```

The simulator shows you the circuit description currently in memory:

```
a berkeley spice3 compatible circuit
 1 : a berkeley spice3 compatible circuit
 2 : .global gnd
```

```
10 : .tran 1e-5 2e-3
12 : vcc vcc 0 12.0
13 : vin 1 0 0.0 ac 1.0 sin(0 1 1k)
14 : ccouple 1 base 10uf
15 : rbias1 vcc base 100k
16 : rbias2 base 0 24k
17 : q1 coll base emit generic
18 : rcollector vcc coll 3.9k
19 : remitter emit 0 1k
21 : .model generic npn
24 : .end
```

The title of this circuit is 'A Berkeley SPICE3 compatible circuit'. The circuit description contains a transient analysis control command `.TRAN 1E-5 2E-3` requesting a total simulated time of 2ms with a maximum time-step of 10us. The remainder of the lines in the circuit description describe the circuit of Fig. 1.

Now, execute the simulation by entering the `run` command:

```
ngspice 1 -> run
```

The simulator will run the simulation and when execution is completed, will return with the ngspice prompt. When the prompt returns, issue the rusage command again to see how much time and memory has been used now.

To examine the results of this transient analysis, we can use the plot command. First we will plot the nodes labeled '`1`' and 'base'.

```
ngspice 2 -> plot v(1) base
```

The simulator responds by displaying an X Window System plot similar to that shown in Fig. 2.

Notice that we have named one of the nodes in the *circuit description* with a number ('`1`'), while the others are words ('base'). This was done to illustrate ngspice's special requirements for plotting nodes labeled with numbers. Numeric labels are allowed in ngspice for backwards compatibility with SPICE2. However, they require special treatment in some commands such as `plot`. The `plot` command is designed to allow expressions in its argument list in addition to names of results data to be plotted. For example, the expression `plot (base - 1)` would plot the result of subtracting 1 from the value of node 'base'.

If we had desired to plot the difference between the voltage at node 'base' and node '`1`', we would need to enclose the node name '`1`' in the construction `v( )` producing a command such as `plot (base - v(1))`.

Now, issue the following command to examine the voltages on two of the internal nodes of the transistor amplifier circuit:

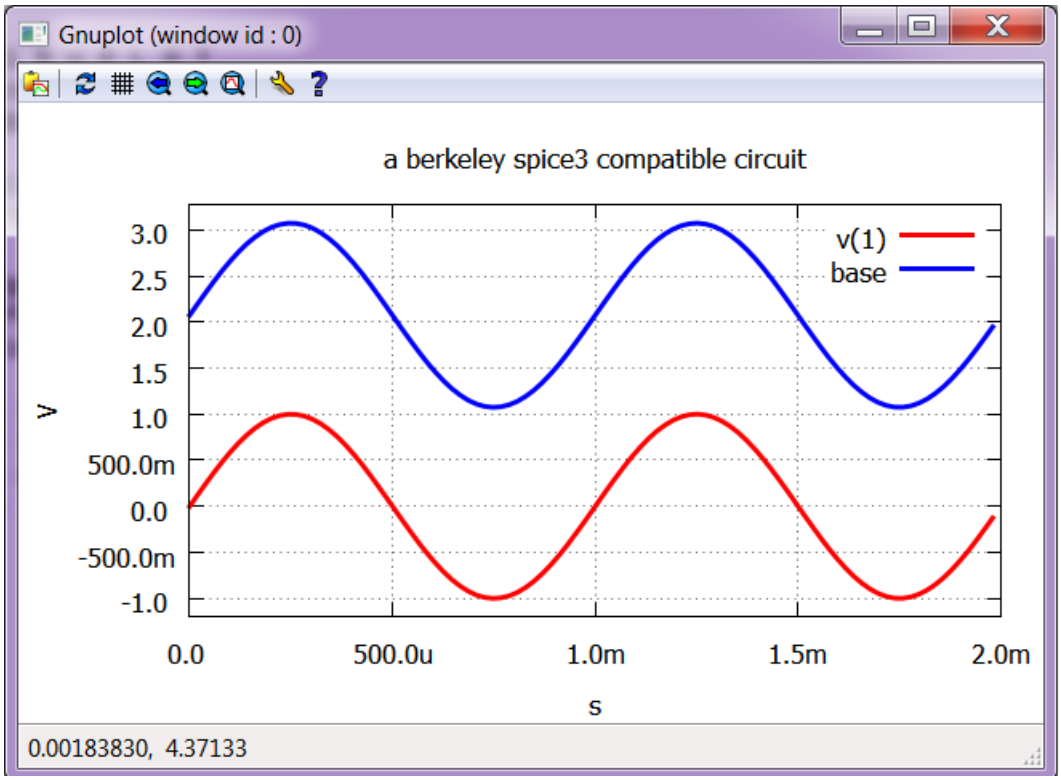

Figure 2: node 1 and node 'base' versus time

```
ngspice 3 -> plot vcc coll emit
```

The plot shown in Fig. 3 should appear. Notice in the circuit description that the power supply voltage source and the node it is connected to both have the name 'vcc'. The `plot` command above has plotted the node voltage 'vcc'. However, it is also possible to plot branch currents through voltage sources in a circuit. ngspice always adds the special suffix `#branch` to voltage source names. Hence, to plot the current into the voltage source named `vcc`, we would use a command such as `plot vcc#branch`.

Now let's run a simple DC simulation of this circuit and examine the bias voltages with the `print` command. One way to do this is to quit the simulator using the `quit` command, edit the input file to change the `.tran` line to `.op` (for 'operating point analysis'), re-invoke the simulator, and then issue the `run` command. However, ngspice allows analysis mode changes directly from the ngspice prompt. All that is required is to enter the control line, e.g. op (without the leading '.'). ngspice will interpret the information on the line and start the new analysis run immediately, without the need to enter a new `run` command.

To run the DC simulation of the transistor amplifier, issue the following

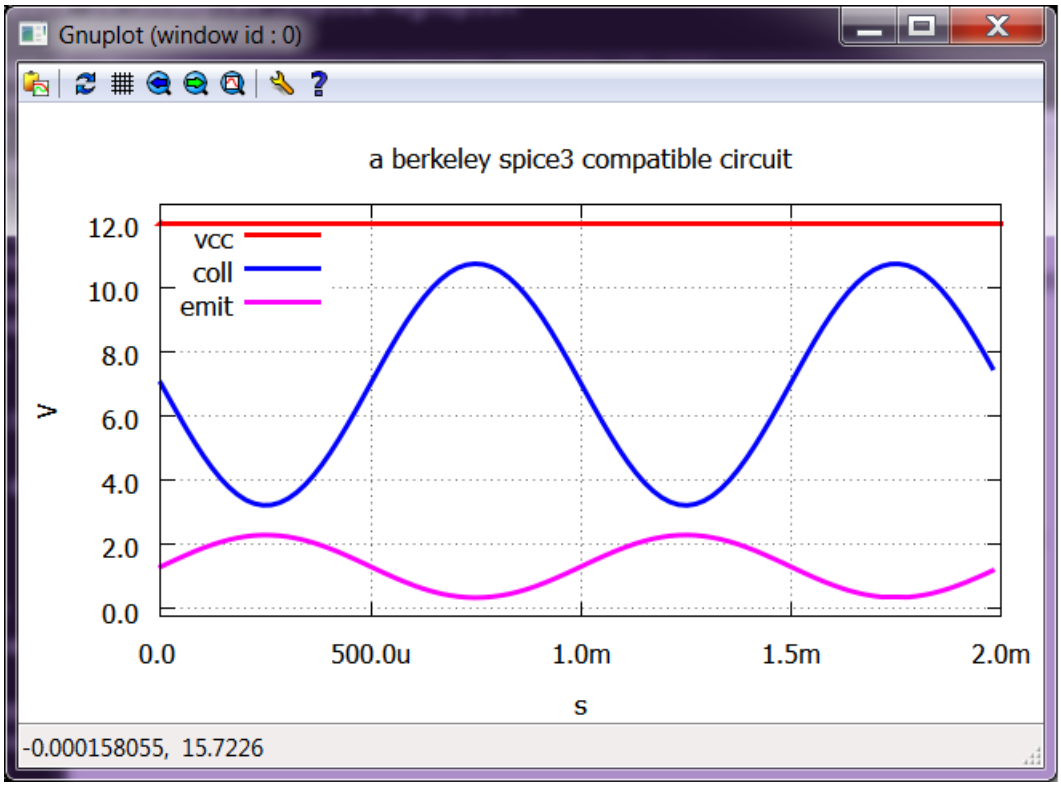

Figure 3: VCC, Collector and Emitter Voltages

command:

```
ngspice 4 -> op
```

After a moment the ngspice prompt returns. Now issue the `print` command to examine the emitter, base, and collector DC bias voltages.

```
ngspice 5 -> print emit base coll
```

ngspice responds with:

```
emit = 1.293993e+00 base = 2.074610e+00 coll = 7.003393e+00
```

To run an AC analysis, enter the following command:

```
ngspice 6 -> ac dec 10 0.01 100
```

This command runs a small-signal swept AC analysis of the circuit to compute the magnitude and phase responses. In this example, the sweep is logarithmic

with 'decade' scaling, 10 points per decade, and lower and upper frequencies of 0.01 Hz and 100 Hz. Since the command sweeps through a range of frequencies, the results are vectors of values and are examined with the `plot` command. Issue to the following command to plot the response curve at node 'coll':

```
ngspice 7 -> plot coll
```

This plot shows the AC gain from input to the collector. (Note that our input source in the circuit description 'vin' contained parameters of the form 'AC 1.0' designating that a unit-amplitude AC signal was applied at this point.) For plotting data from an AC analysis, ngspice chooses automatically a logarithmic scaling for the frequency (x) axis.

To produce a more traditional 'Bode' gain phase plot (again with automatic logarithmic scaling on the frequency axis), we use the expression capability of the `plot` command and the built-in Nutmeg functions db( ) and ph( ):

```
ngspice 8 -> plot db(coll) ph(coll)
```

The last analysis supported by ngspice is a swept DC analysis. To perform this analysis, issue the following command:

```
ngspice 9 -> dc vcc 0 15 0.1
```

This command sweeps the supply voltage 'vcc' from 0 to 15 volts in 0.1 volt increments. To plot the results, issue the command:

```
ngspice 10 -> plot emit base coll
```

Finally, to exit the simulator, use the `quit` command, and you will be returned to the operating system prompt.

```
ngspice 11 -> quit
```

So long.