

Operating Point

The first SPICE simulation analysis we'll look at is the .op or operating point analysis. An operating point simulation's output data is not graphical but rather simply a list of node voltages, loop currents, and, when active elements are used, small-signal AC parameters. Consider the schematic seen in Fig. 1.10. The SPICE netlist used to simulate this circuit may look like the following (again, remember, that all of these simulation examples are available for download at CMOSedu.com):

```
*** Figure 1.10 CMOS: Circuit Design, Layout, and Simulation ***

*#destroy all
*#run
*#print all

.op

Vin 1 0 DC 1
R1 1 2 1k
R2 2 0 2k

.end
```

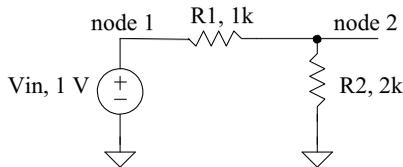


Figure 1.10 Operation point simulation for a resistive divider.

The first line in a netlist is a title line. SPICE ignores the first line (important to avoid frustration!). A comment line starts with an asterisk. SPICE ignores lines that start with a * (in most cases). In the netlist above, however, the lines that start with *# are command lines. These command lines are used for control in some SPICE simulation programs. In other SPICE programs, these lines are simply ignored. The commands in this netlist destroy previous simulation data (so we don't view the old data), run the simulation, and then print the simulation output data. SPICE analysis commands start with a period. Here we are performing an operating point analysis. Following the .op, we've specified an input voltage source called Vin (voltage source names must start with a V, resistor names must start with an R, etc.) connected from node 1 to ground (ground always has a node name of 0 [zero]). We then have a 1k resistor from node 1 to node 2 and a 2k resistor from node 2 to ground. Running the simulation gives the following output:

```
v(1) = 1.000000e+00
v(2) = 6.666667e-01
vin#branch = -3.33333e-04
```

The node voltages, as we would expect, are 1 V and 667 mV, respectively. The current flowing through Vin is 333 μ A. Note that SPICE defines positive current flow as from the + terminal of the voltage source to the - terminal (hence, the current above is negative).

It's often useful to use names for nodes that have meaning. In Fig. 1.11, we replaced the names node 1 and 2 with *Vin* and *Vout*. *Vin* corresponds to the input voltage source's name. This is useful when looking at a large amount of data. Also seen in Fig. 1.11 is the modified netlist.

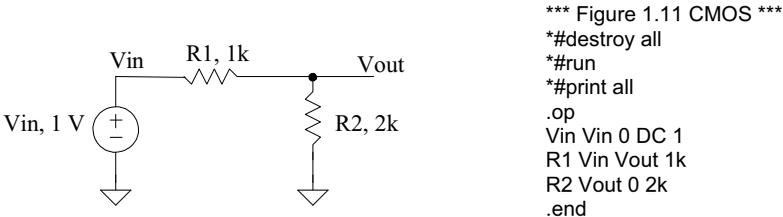


Figure 1.11 Operation point simulation for a resistive divider.

Transfer Function Analysis

The transfer function analysis can be used to find the DC input and output resistances of a circuit as well as the DC transfer characteristics. To give an example, let's replace, in the netlist seen above, `.op` with

```
.TF V(Vout,0) Vin
```

The output is defined as the voltage between nodes *Vout* and 0 (ground). The input is a source (here a voltage source). When we run the simulation with this command line, we get an output of

```
transfer_function = 6.666667e-01
output_impedance_at_v(vout,0) = 6.666667e+02
vin#input_impedance = 3.000000e+03
```

As expected, the "gain" of this voltage divider is $2/3$, the input resistance is $3k$ ($1k + 2k$), and the output resistance is 667Ω ($1k \parallel 2k$).

As another example of the use of the `.tf` command consider adding the $0 V$ voltage source to Fig. 1.11, as seen in Fig. 1.12. Adding a $0 V$ source to a circuit is a common method to measure the current in an element (we plot or print $I(V_{meas})$ for example).

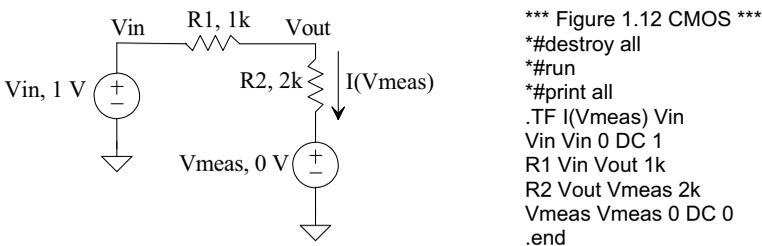


Figure 1.12 Measuring the transfer function in a resistive divider when the output variable is the current through *R2* and the input is *Vin*.

Here, in the .tf analysis, we have defined the output variable as a current, $I(V_{meas})$ and the input as the voltage, V_{in} . Running the simulation, we get an output of

```
transfer_function = 3.333333e-04
vin#input_impedance = 3.000000e+03
vmeas#output_impedance = 1.000000e+20
```

The gain is $I(V_{meas})/V_{in}$ or $1/3k$ ($= 333 \mu\text{mhos}$), the input resistance is still $3k$, and the output resistance is now an open (V_{meas} is removed from the circuit).

The Voltage-Controlled Voltage Source

SPICE can be used to model voltage-controlled voltage sources (VCVS). Consider the circuit seen in Fig. 1.13. The specification for a VCVS starts with an E in SPICE. The netlist for this circuit is

```
*** Figure 1.13 CMOS: Circuit Design, Layout, and Simulation ***
```

```
*#destroy all
*#run
*#print all

.TF    V(Vout,0) Vin

Vin    Vin    0    DC    1
R1     Vb     0    3k
R2     Vt     Vout 1k
R3     Vout   0    2k
E1     Vt     Vb   Vin    0    23

.end
```

The first two nodes (V_t and V_b), following the VCVS name $E1$, are the VCVS outputs (the first node is the + output). The second two nodes (V_{in} and ground) are the controlling nodes. The gain of the VCVS is, in this example, 23. The voltage between V_t and V_b is $23 \cdot V_{in}$. Running this simulation gives an output of

```
transfer_function = 7.666667e+00
output_impedance_at_v(vout,0) = 1.333333e+03
vin#input_impedance = 1.000000e+20
```

Notice that the input resistance is infinite.

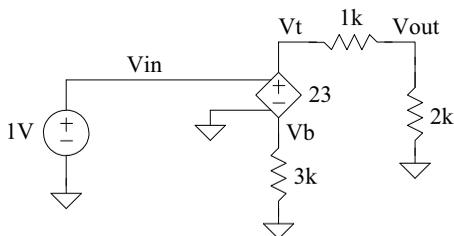
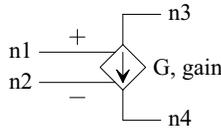


Figure 1.13 Example using a voltage-controlled voltage source.

An Ideal Op-Amp

We can implement a (near) ideal op-amp in SPICE with a VCVS or with a voltage-controlled current source (VCCS), Fig. 1.14. It turns out that using a VCCS to implement an op-amp in SPICE results, in general, in better simulation convergence. The input voltage, the difference between nodes n1 and n2 in Fig. 1.14, is multiplied by the transconductance G (units of amps/volts or mhos) to cause a current to flow between n3 and n4. Note that the input resistance of the VCCS, the resistance seen at n1 and n2, is infinite.



Voltage-Controlled Current Source (VCCS)
G1 n3 n4 n1 n2 G

Figure 1.14 Voltage-controlled current source in SPICE.

Figure 1.15 shows the implementation of an ideal op-amp in SPICE along with an example circuit. The open-loop gain of the op-amp is a million (the product of the VCCS's transconductance with the 1-ohm resistor). Note how we've flipped the polarity of the (SPICE model of the) op-amp's input to ensure a rising voltage on the noninverting input (+ input) causes V_{out} to increase. The closed-loop gain is -3 (if this isn't obvious then the reader should revisit sophomore circuits before going too much further in the book).

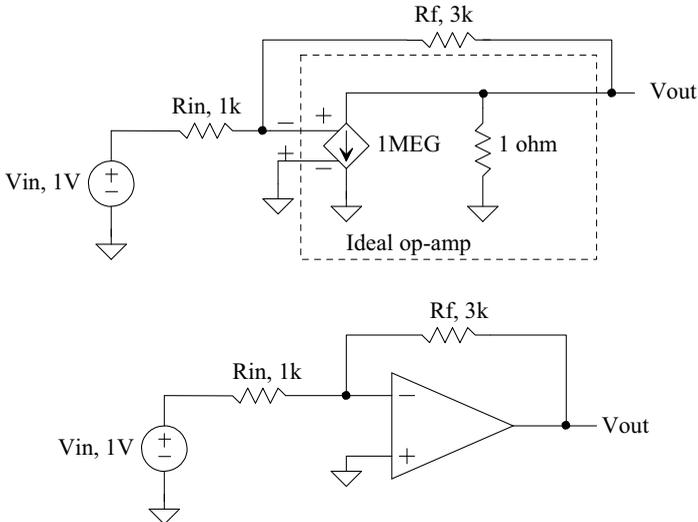


Figure 1.15 An op-amp simulation example.

The Subcircuit

In a simulation we may want to use a circuit, like an op-amp, more than once. In these situations we can generate a subcircuit and then, in the main part of the netlist, call the circuit as needed. Below is the netlist for simulating, using a transfer function analysis, the circuit in Fig. 1.15 where the op-amp is specified using a subcircuit call.

*** Figure 1.15 CMOS: Circuit Design, Layout, and Simulation ***

```

*#destroy all
*#run
*#print all

.TF    V(Vout,0) Vin

Vin    Vin    0    DC    1
Rin    Vin    Vm    1k
Rf     Vout    Vm    3k

X1     Vout    0     vm    Ideal_op_amp

.subckt Ideal_op_amp Vout Vp Vm
G1     Vout    0     Vm    Vp    1MEG
RL     Vout    0     1
.ends

.end

```

Notice that a subcircuit call begins with the letter X. Note also how we've called the noninverting input (the + input) Vp and not V+ or +. Some SPICE simulators don't like + or - symbols used in a node's name. Further note that a subcircuit ends with .ends (end subckt). Care must be exercised with using either .end or .ends. If, for example, a .end is placed in the middle of the netlist all of the SPICE netlist information following this .end is ignored.

The output results for this simulation are seen below. Note how the ideal gain is -3 where the simulated gain is -2.99999. Our near-ideal op-amp has an open-loop gain of one million and thus the reason for the slight discrepancy between the simulated and calculated gains. Also note how the input resistance is 1k, and the output resistance, because of the feedback, is essentially zero.

```

transfer_function = -2.99999e+00
output_impedance_at_v(vout,0) = 3.999984e-06
vin#input_impedance = 1.000003e+03

```

DC Analysis

In both the operating point and transfer function analyses, the input to the circuit was constant. In a DC analysis, the input is varied and the circuit's node voltages and currents (through voltage sources) are simulated. A simple example is seen in Fig. 1.16. Note how we are now plotting, instead of printing, the node voltages. We could also plot the current through Vin (plot Vin#branch). The .dc command specifies that the input source, Vin, should be varied from 0 to 1 V in 1 mV steps. The x-axis of the simulation results seen in the figure is the variable we are sweeping, here Vin. Note that, as expected, the slope of the Vin curve is one (of course) and the slope of Vout is 2/3 (= Vout/Vin).

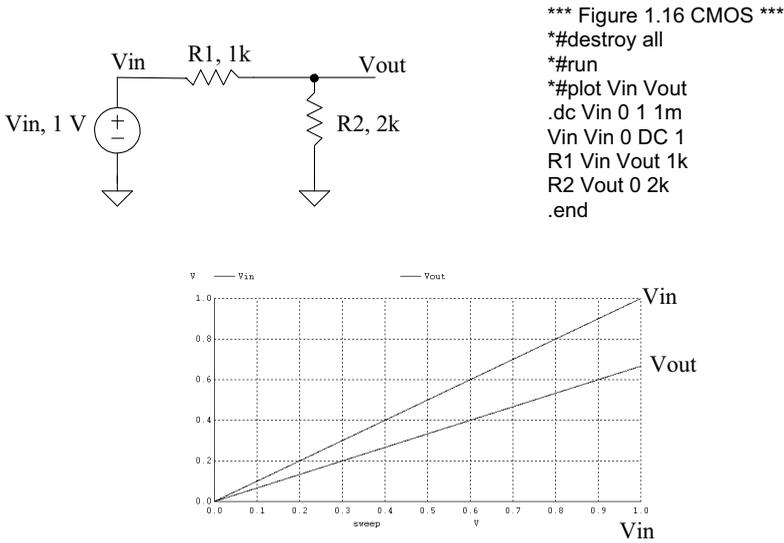


Figure 1.16 DC analysis simulation for a resistive divider.

Plotting IV Curves

One of the simulations that is commonly performed using a DC analysis is plotting the current-voltage (IV) curves for an active device (e.g., diode or transistor). Examine the simulation seen in Fig. 1.17. The diode is named D1. (Diodes must have names that start with a D.) The diode's anode is connected to node Vd, while its cathode is connected to

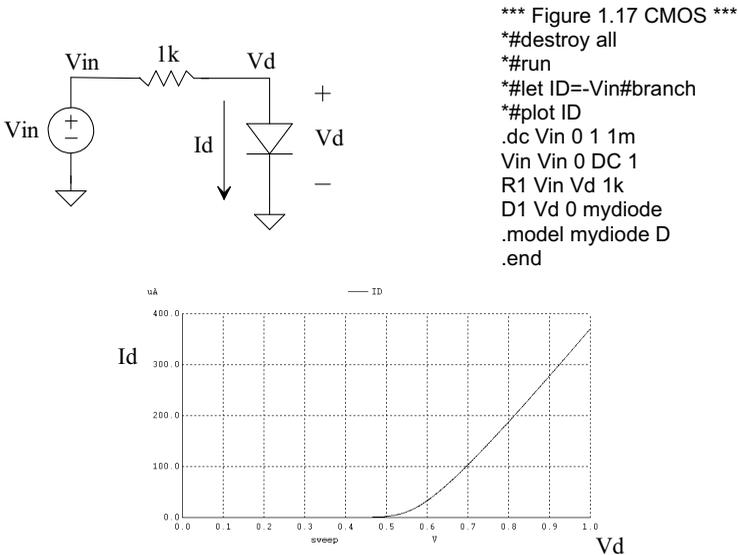


Figure 1.17 Plotting the current-voltage curve for a diode.

ground. This is our first introduction to the .model specification. Here our diode's model name is mydiode. The .model parameter D seen in the netlist simply indicates a diode model. We don't have any parameters after the D in this simulation, so SPICE uses default parameters. The interested reader is referred to Table 2.1 on page 47 for additional information concerning modeling diodes in SPICE. Note, again, that SPICE defines positive current through a voltage source as flowing from the + terminal to the - terminal (hence why we defined the diode current the way we did in the netlist).

Dual Loop DC Analysis

An outer loop can be added to a DC analysis, Fig. 1.18. In this simulation we start out by setting the base current to 5 μA and sweeping the collector-emitter voltage from 0 to 5 V in 1 mV steps. The output data for this particular simulation is the trace, seen in Fig. 1.18, with a label of "Ib=5u." The base current is then increased by 5 μA to 10 μA , and the collector-emitter voltage is stepped again (resulting in the trace labeled "Ib=10u". This continues until the final iteration when Ib is 25 μA . Other examples of using a dual-loop DC analysis for MOSFET IV curves are found in Figs. 6.11, 6.12, and 6.13.

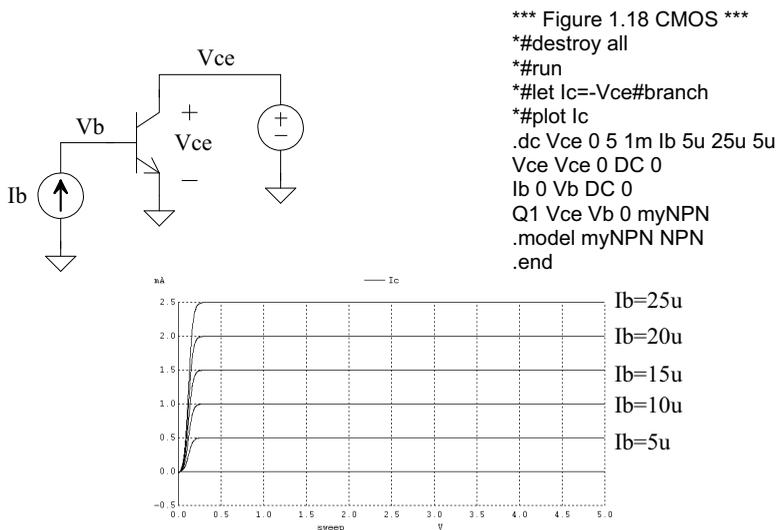


Figure 1.18 Plotting the current-voltage curves for an NPN BJT.

Transient Analysis

The form of the transient analysis statement is

```
.tran tstep tstop <tstart> <tmax> <uic>
```

where the terms in $\langle \rangle$ are optional. The tstep term indicates the (suggested) time step to be used in the simulation. The parameter tstop indicates the simulation's stop time. The starting time of a simulation is always time equals zero. However, for very large (data) simulations, we can specify a time to start saving data, tstart. The tmax parameter is used to specify the maximum step size. If the plots start to look jagged (like a sinewave that isn't smooth), then tmax should be reduced.

A SPICE transient analysis simulates circuits in the time domain (as in an oscilloscope, the x-axis is time). Let's simulate, using a transient analysis, the simple circuit seen back in Fig. 1.11. A simulation netlist may look like (see output in Fig. 1.19):

*** Figure 1.19 CMOS: Circuit Design, Layout, and Simulation ***

```

*#destroy all
*#run
*#plot vin vout

.tran 100p 100n

Vin  Vin  0    DC    1
R1   Vin  Vout 1k
R2   Vout 0    2k

.end

```

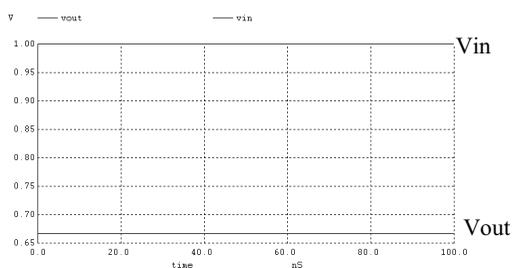


Figure 1.19 Transient simulation for the circuit in Fig. 1.11.

The SIN Source

To illustrate a simulation using a sinewave, examine the schematic in Fig 1.20. The statement for a sinewave in SPICE is

```
SIN Vo Va freq <td> <theta>
```

The parameter V_o is the sinusoid's offset (the DC voltage in series with the sinewave). The parameter V_a is the peak amplitude of the sinewave. Freq is the frequency of the sinewave, while t_d is the delay before the sinewave starts in the simulation. Finally, θ is used if the amplitude of the sinusoid has a damped nature. Figure 1.20 shows the netlist corresponding to the circuit seen in this figure and the simulation results.

Some key things to note in this simulation: (1) MEG is used to specify 10^6 . Using “m” or “M” indicates milli or 10^{-3} . The parameter 1MHz indicates 1 milliHertz. Also, f indicates femto or 10^{-15} . A capacitor value of 1f doesn't indicate one Farad but rather 1 femto Farad. (2) Note how we increased the simulation time to 3 μ s. If we had a simulation time of 100 ns (as in the previous simulation), we wouldn't see much of the sinewave (one-tenth of the sinewave's period). (3) The “SIN” statement is used in a transient simulation analysis. The SIN specification is **not** used in an AC analysis (discussed later).

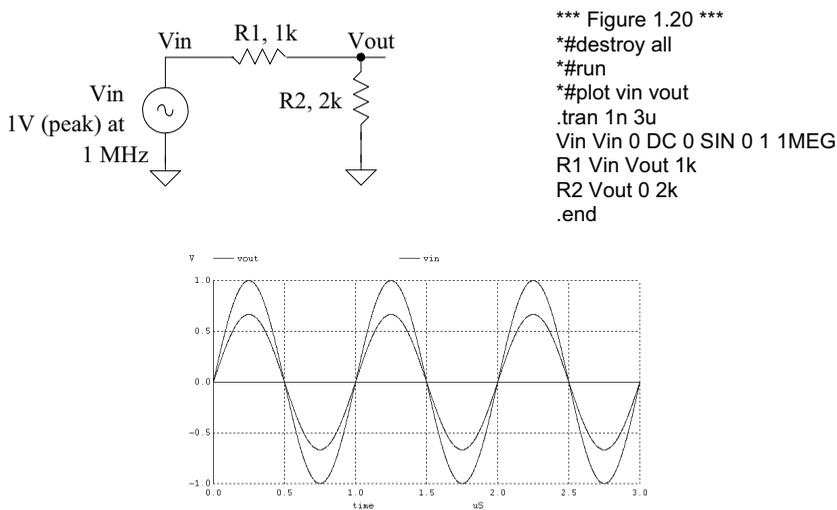


Figure 1.20 Simulating a resistive divider with a sinusoidal input.

An RC Circuit Example

To illustrate the use of a .tran simulation let's determine the output of the RC circuit seen in Fig. 1.21 and compare our hand calculations to simulation results. The output voltage can be written in terms of the input voltage by

$$V_{out} = V_{in} \cdot \frac{1/j\omega C}{1/j\omega C + R} \text{ or } \frac{V_{out}}{V_{in}} = \frac{1}{1 + j\omega RC} \tag{1.1}$$

Taking the magnitude of this equation gives

$$\left| \frac{V_{out}}{V_{in}} \right| = \frac{1}{\sqrt{1 + (2\pi f RC)^2}} \tag{1.2}$$

and taking the phase gives

$$\angle \frac{V_{out}}{V_{in}} = -\tan^{-1} \frac{2\pi f RC}{1} \tag{1.3}$$

From the schematic the resistance is 1k, the capacitance is 1 μF, and the frequency is 200 Hz. Plugging these numbers into Eqs. (1.1) – (1.3) gives $\left| \frac{V_{out}}{V_{in}} \right| = 0.623$ and $\angle \frac{V_{out}}{V_{in}} = -0.898$ radians or -51.5 degrees. With a 1 V peak input then our output voltage is 623 mV (and as seen in Fig. 1.21, it is). Remembering that phase shift is simply an indication of time delay at a particular frequency,

$$\angle \text{ (radians) } = \frac{t_d}{T} \cdot 2\pi \text{ or } \angle \text{ (degrees) } = \frac{t_d}{T} \cdot 360 = t_d \cdot f \cdot 360 \tag{1.4}$$

The way to remember this equation is that the time delay, t_d , is a percentage of the period (T), t_d/T , multiplied by either 2π (radians) or 360 (degrees). For the present example, the time delay is 715 μs (again, see Fig. 1.21). Note that the minus sign indicates that the output is lagging (occurring later in time) the input (the input leads the output).

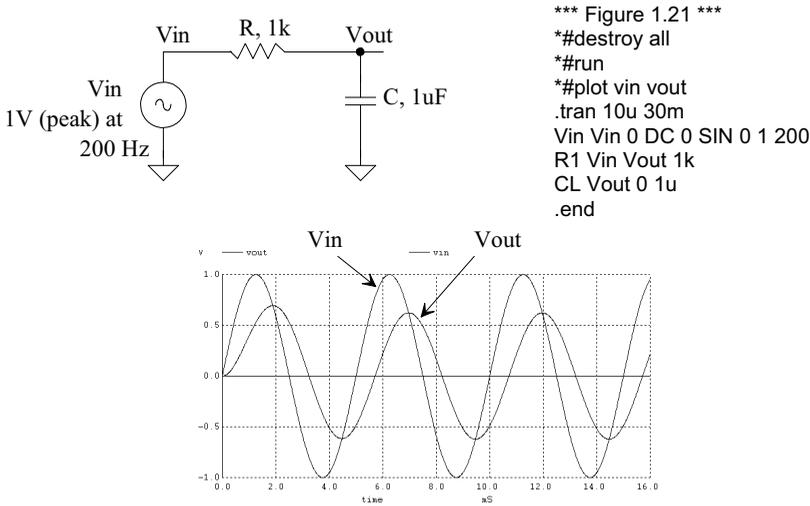


Figure 1.21 Simulating the operation of an RC circuit using a .tran analysis.

Another RC Circuit Example

As one more example of simulating the operation of an RC circuit consider the circuit seen in Fig. 1.22. Combining the impedances of C_1 and R , we get

$$Z = \frac{R/j\omega C_1}{R + 1/j\omega C_1} = \frac{R}{1 + j\omega RC_1} \quad (1.5)$$

The transfer function for this circuit is then

$$\frac{V_{out}}{V_{in}} = \frac{1/j\omega C_2}{1/j\omega C_2 + Z} = \frac{1 + j\omega RC_1}{1 + j\omega R(C_1 + C_2)} \quad (1.6)$$

The magnitude of this transfer function is

$$\left| \frac{V_{out}}{V_{in}} \right| = \frac{\sqrt{1 + (2\pi f RC_1)^2}}{\sqrt{1 + (2\pi f R \cdot (C_1 + C_2))^2}} \quad (1.7)$$

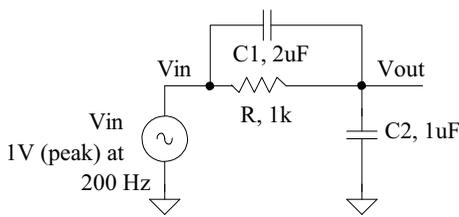
and the phase response is

$$\angle \frac{V_{out}}{V_{in}} = \tan^{-1} \frac{2\pi f RC_1}{1} - \tan^{-1} \frac{2\pi f R(C_1 + C_2)}{1} \quad (1.8)$$

Plugging in the numbers from the schematic gives a magnitude response of 0.6 (which matches the simulation results) and a phase shift of -0.119 radians or -6.82 degrees. The amount of time the output is lagging the input is then

$$t_d = \frac{T \cdot \angle}{360} = \frac{\angle}{f \cdot 360} = \frac{-6.82}{200 \cdot 360} = -95 \mu\text{s} \quad (1.9)$$

which is confirmed with the simulation results seen in Fig. 1.22.



```

*** Figure 1.22 ***
*#destroy all
*#run
*#plot vin vout
.tran 10u 30m
Vin Vin 0 DC 0 SIN 0 1 200
R1 Vin Vout 1k
C1 Vin Vout 2u
C2 Vout 0 1u
.end

```

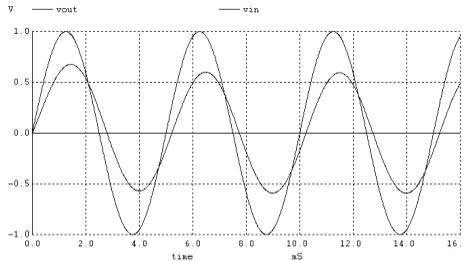


Figure 1.22 Another RC circuit example.

AC Analysis

When performing a transient analysis (.tran) the x-axis is time. We can determine the frequency response of a circuit (the x-axis is frequency) using an AC analysis (.ac). An AC analysis is specified in SPICE using

```
.ac dec nd fstart fstop
```

The dec indicates that the x-axis should be plotted in decades. We could replace dec with lin (linear plot on the x-axis) or oct (octave). The term nd indicates the number of points per decade (say 100), while fstart and fstop indicate the start and stop frequencies (note that fstart cannot be zero, or DC, since this isn't an AC signal). The netlist used to simulate the AC response of the circuit in Fig. 1.21 follows. The simulation output is seen in Fig. 1.23, where we've pointed out the response at 200 Hz (the frequency used in Fig. 1.21 and used for calculations on page 17).

```
*** Figure 1.23 CMOS: Circuit Design, Layout, and Simulation ***
```

```

*#destroy all
*#run
*#plot db(vout/vin)
*#set units=degrees
*#plot ph(vout/vin)

.ac dec 100 1 10k

Vin  Vin  0  DC  0  SIN 0 1 200  AC 1
R1   Vin  Vout 1k
CL   Vout  0  1u

.end

```

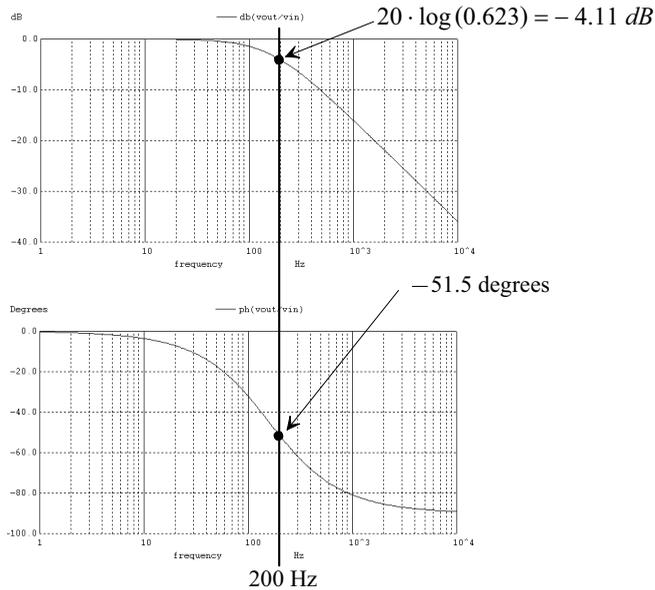


Figure 1.23 AC simulation for the RC circuit in Fig. 1.21.

Note in this netlist that the SIN specification in Vin has nothing to do with an AC analysis (it's ignored for an AC analysis). For the AC analysis, we added, to the statement for Vin, the term AC 1 (specifying that the magnitude or peak of the AC signal is 1). We can add a phase shift of 45 degrees by using AC 1 45 in the statement.

Decades and Octaves

In the simulation results seen in Fig. 1.23 we used decades. When we talk about decades we either are multiplying or dividing by 10. One decade above 23 MHz is 230 MHz, while one decade below 1.2 kHz is 120 Hz.

When we talk about octaves, we talk about either multiplying or dividing by 2. One octave above 23 MHz is 46 MHz while one octave below 1.2 kHz is 600 Hz. Two octaves above 23 MHz is (multiply by 4) 92 MHz.

Decibels

When the magnitude response of a transfer function decreases by 10, it is said it goes down by -20 dB (divide by 10, $20 \cdot \log(0.1) = -20 \text{ dB}$). When the magnitude response increases by 10, it goes up by 20 dB (multiply by 10). For the frequency response in Fig. 1.23 (above 159 Hz, the -3 dB frequency, or here when the magnitude response is 0.707), the response is rolling off at -20 dB/decade . What this means is that if we increase the frequency by 10 the magnitude response decreases by 10. We could also say the response is rolling off at -6 dB/octave above 159 Hz (for every increase in frequency by 2 the magnitude response drops by a factor of 2). If a magnitude response is rolling off at -40 dB/decade , then for every increase in frequency by 10 the magnitude drops by 100. Similarly if a response rolls off at -12 dB/octave , for every doubling in frequency our response drops by 4. Note that -6 dB/octave is the same rate as -20 dB/decade .

Pulse Statement

The SPICE pulse statement is used in transient simulations to specify pulses or clock signals. This statement has a format given by

```
pulse vinit vfinal td tr tf pw per
```

The pulse's initial voltage is *vinit* while *vfinal* is the pulse's final (or pulsed) value, *td* is the delay before the pulse starts, *tr* and *tf* are the rise and fall times, respectively, of the pulse (noting that when these are set to zero the step size used in the transient simulation is used), *pw* is the pulse's width; and *per* is the period of the pulse. Figure 1.24 provides an example of a simulation that uses the pulse statement. A section of the netlist used to generate the waveforms in this figure follows.

```
.tran 100p 30n
```

```
Vin   Vin   0   DC   0   pulse 0 1 6n 0 0 3n 10n
R1    Vin   Vout  1k
C1    Vout  0   1p
```

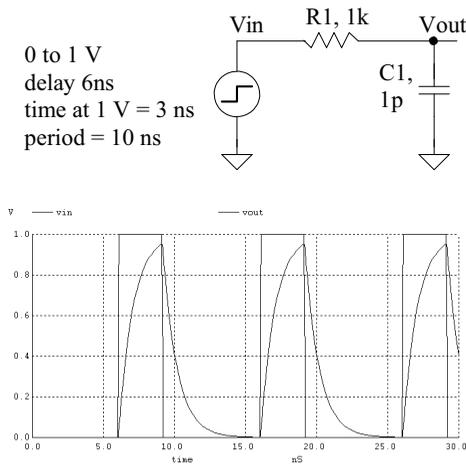


Figure 1.24 Simulating the step response of an RC circuit using a pulsed source voltage.

Finite Pulse Rise Time

Notice, in the simulation results seen in Fig. 1.24, that the rise and fall times of the input pulse are not 0 as specified in the pulse statement but rather 100 ps as specified by the suggested maximum step size in the `.tran` statement. Figure 1.25 shows the simulation results if we change the pulse statement to

```
Vin   Vin   0   DC   0   pulse 0 1 6n 10p 10p 3n 10n
```

where we've specified 10 ps rise and fall times. Note that in some SPICE simulators you must specify a maximum step size in the `.tran` statement. You could do this in the `.tran` statement above by using `.tran 10p 30n 0 10p` (where the 10p is the maximum step size and the simulation starts saving data at 0.)

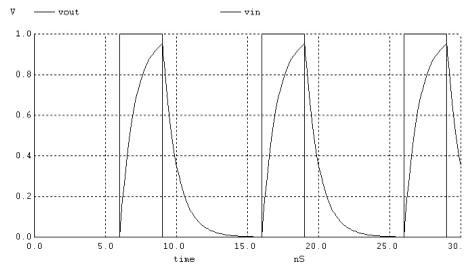


Figure 1.25 Specifying a rise time in the pulse statement to avoid slow rise times (rise times set by the maximum step size in the .tran statement.)

Step Response

The pulse statement can also be used to generate a step function

```
Vin Vin 0 DC 0 pulse 0 1 2n 10p
```

We've reduced the delay to 2n and have specified (only) a rise time for the pulse. Since the pulse width isn't specified, the pulse transitions and then stays high for the extent of the simulation. Figure 1.26 shows the step response for the RC circuit seen in Fig. 1.24.

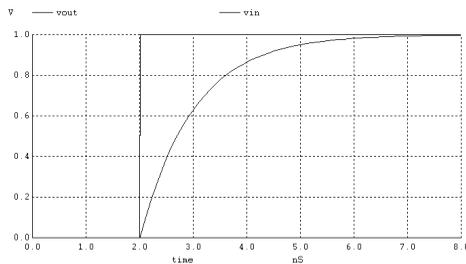


Figure 1.26 Step response of an RC circuit.

Delay and Rise Time in RC Circuits

From the RC circuit review on page 50 we can write the delay time, the time it takes the pulse to reach 50% of its final value in an RC circuit, using

$$t_d \approx 0.7RC \quad (1.10)$$

and the rise time (or fall time) as

$$t_r \approx 2.2RC \quad (1.11)$$

Using the RC in Fig. 1.24 (1 ns), we get a (calculated) delay time of 700 ps and a rise time of 2.2 ns. These numbers are verified in Fig. 1.26. To show that the pulse statement can be used for other amplitude steps consider resimulating the circuit in Fig. 1.24 (see Fig. 1.27) with an input pulse that transitions from -1 to -2 V (note how the delay and transition times remain unchanged. The SPICE pulse statement is now

```
Vin Vin 0 DC 0 pulse -1 -2 2n 10p
```

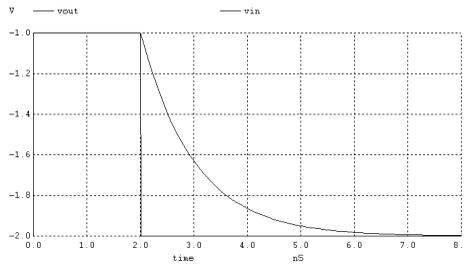


Figure 1.27 Another step response (negative going) of an RC circuit.

Piece-Wise Linear (PWL) Source

The piece-wise linear (PWL) source specifies arbitrary waveform shapes. The SPICE statement for a PWL source is

```
pwl t1 v1 t2 v2 t3 v3 ... <rep>
```

To provide an example using a PWL voltage source, examine Fig. 1.28. The input waveform in this simulation is specified using

```
pwl 0 0.5 3n 1 5n 1 5.5n 0 7n 0
```

At 0 ns, the input voltage is 0.5 V. At 3 ns the input voltage is 1 V. Note the linear change between 0 and 3 ns. Each pair of numbers, the first the time and the second the voltage (or current if a current source is used) represent a point on the PWL waveform. Note that in some simulators the specification for a PWL source may be quite long. In these situations a text file is specified that contains the PWL for the simulation.

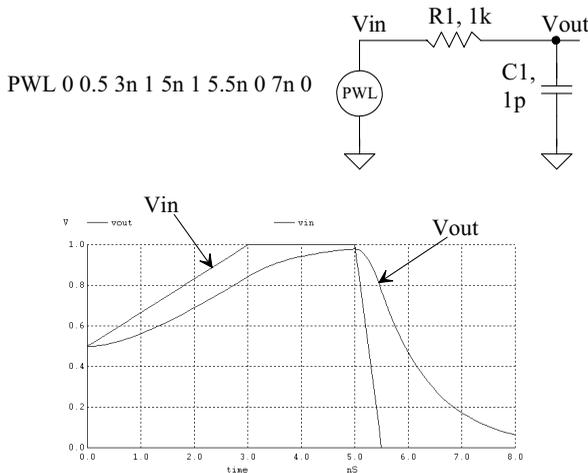


Figure 1.28 Using a PWL source to drive an RC circuit.

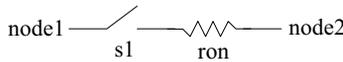
Simulating Switches

A switch can be simulated in SPICE using the following (for example) syntax

```
s1 node1 node2 controlp controlm switmod
.model switmod sw ron=1k
```

The name of a switch must start with an s. The switch is connected between node1 and node2, as seen in Fig. 1.29. When the voltage on node controlp is greater than the voltage on node controlm, the switch closes. The switch is modeled using the .model statement. As seen above, we are setting the series resistance of the switch to 1k.

```
s1 node1 node2 controlp controlm switmod
```



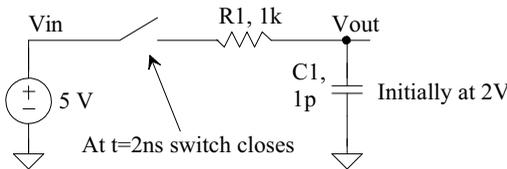
The switch is closed when the node voltage controlp is greater than the node voltage controlm

Figure 1.29 Modeling a switch in SPICE.

Initial Conditions on a Capacitor

An example of a circuit that uses both a switch and an initial voltage on a capacitor is seen in Fig. 1.30. Notice, in the netlist, that we have added UIC to the end of the .tran statement. This addition makes SPICE "use initial conditions" or skip an initial operating point calculation. Also note that to set the initial voltage across the capacitor we simply added IC=2 to the end of the statement for a capacitor. To set a node to a voltage (that may have a capacitor connected to it or not), we can add, for example,

```
.ic v(vout)=2
```



*** Figure 1.30 ***

```

#destroy all
#run
#plot vout

.tran 100p 8n UIC

Vclk clk 0 pulse -1 1 2n
Vin Vin 0 DC 5
S1 Vin Vouts clk 0 switmodel
R1 Vouts Vout 1k
C1 Vout 0 1p IC=2

.model switmodel sw ron=0.1

.end
```

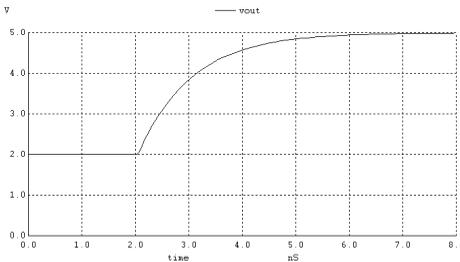
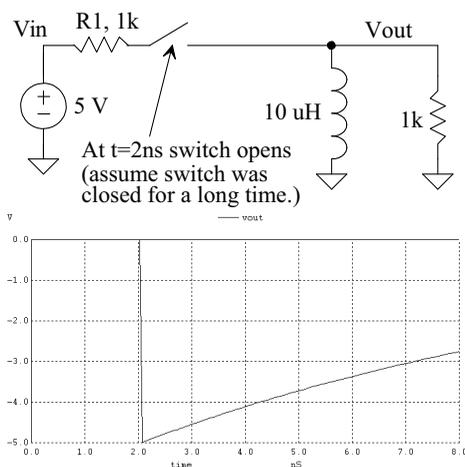


Figure 1.30 Using initial conditions and a switch in an RC circuit simulation.

Initial Conditions in an Inductor

Consider the circuit seen in Fig. 1.31. Here we assume that the switch has been closed for a long period of time so that the circuit reaches steady-state. The inductor shorts the output to ground and the current flowing in the inductor is 5 mA. To simulate this initial condition, we set the current in the inductor using the IC statement as seen in the netlist (remembering to include the UIC in the .tran statement). At 2 ns after the simulation starts, we open the switch (the control voltage connections are switched from the previous simulations). Since we know we can't change the current through an inductor instantaneously (the inductor wants to keep pulling 5 mA), the voltage across the inductor will go from 0 to -5 V. The inductor will pull the 5 mA of current through the 1k resistor connected to the output node. Note that we select the transient simulation time by looking at the time constant, L/R , of the circuit (here 10 ns).



*** Figure 1.31 ***

```

*#destroy all
*#run
*#plot vout

.tran 100p 8n UIC

Vclk clk 0 pulse -1 1 2n
Vin Vin 0 DC 5
S1 Vin Vouts 0 clk switmodel
R1 Vouts Vout 1k
R2 Vout 0 1k
L1 Vout 0 10u IC=5m

.model switmodel sw ron=0.1

.end
    
```

Figure 1.31 Using initial conditions in an inductive circuit.

Q of an LC Tank

Figure 1.32 shows a simulation useful in determining the quality factor or Q of a parallel LC circuit (a tank, used in communication circuits among others). The current source and resistor may model a transistor. The resistor can also be used to model the losses in the capacitor or inductor. Quality factor for a resonant circuit is defined as the ratio of the energy stored in the tank to the energy lost. Our circuit definition for Q is the ratio of the center (resonant) frequency to the bandwidth of the response at the 3 dB points. We can write an equation for this circuit definition of Q as

$$Q = \frac{f_{center}}{BW} = \frac{f_{center}}{f_{3dBhigh} - f_{3DBlow}} \tag{1.12}$$

The center frequency of the circuit in Fig. 1.32 is roughly 503 MHz, while the upper 3 dB frequency is 511.2 MHz and the lower 3 dB frequency is 494.8 MHz. The Q is roughly 30. Note the use of linear plotting in the ac analysis statement.

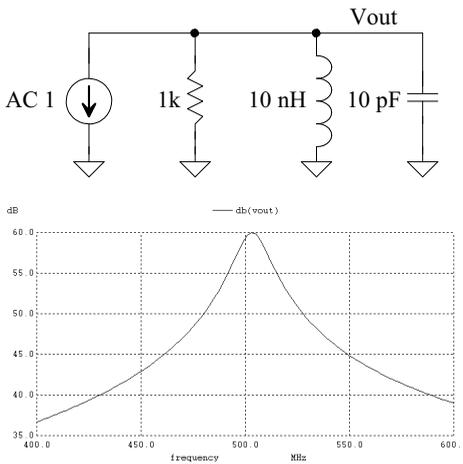


Figure 1.32 Determining the Q, or quality factor, of an LC tank.

*** Figure 1.32 ***

```

*#destroy all
*#run
*#plot db(vout)

.AC lin 100 400MEG 600MEG

lin Vout 0 DC 0 AC 1
R1 Vout 0 1k
L1 Vout 0 10n
C1 Vout 0 10p

.end

```

Frequency Response of an Ideal Integrator

The frequency response of the integrator seen in Fig. 1.33 can be determined knowing the op-amp keeps the inverting input terminal at the same potential as the non-inverting input (here ground). The current through the resistor must equal the current through the capacitor so

$$\frac{V_{in}}{R} + \frac{V_{out}}{1/j\omega C} = 0 \quad (1.13)$$

or

$$\frac{V_{in}}{V_{out}} = \frac{-1}{j\omega RC} = \frac{-(1+j \cdot 0)}{0+j\omega RC} \quad (1.14)$$

The magnitude of the integrator's transfer function is

$$\left| \frac{V_{in}}{V_{out}} \right| = \frac{\sqrt{(-1)^2 + (-0)^2}}{\sqrt{(0)^2 + (2\pi f RC)^2}} = \frac{1}{2\pi RCf} \quad (1.15)$$

while the phase shift through the integrator is

$$\angle \frac{V_{out}}{V_{in}} = \tan^{-1} \frac{0}{-1} - \tan^{-1} \frac{2\pi RCf}{0} = -90^\circ \quad (1.16)$$

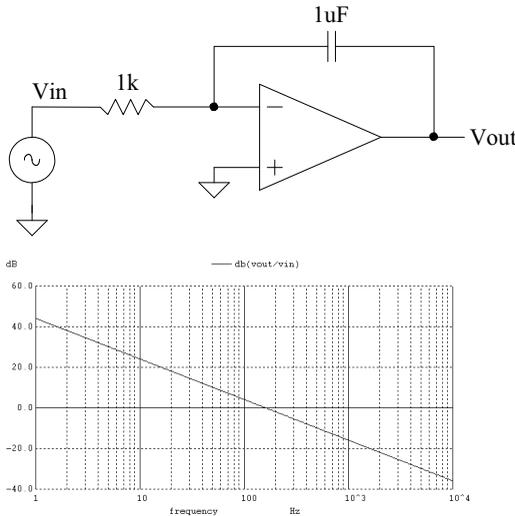
Note that the gain of the integrator approaches infinity as the frequency decreases towards DC while the phase shift is constant.

Unity-Gain Frequency

It's of interest to determine the frequency where the magnitude of the transfer function is unity (called the unity-gain frequency, f_{ur}). Using Eq. (1.15), we can write

$$\left| \frac{V_{out}}{V_{in}} \right| = 1 = \frac{1}{2\pi RCf_{un}} \rightarrow f_{un} = \frac{1}{2\pi RC} \quad (1.17)$$

Using the values seen in the schematic, the unity-gain frequency is 159 Hz (as verified in the SPICE simulation seen in Fig. 1.33).



*** Figure 1.33 ***

```

*#destroy all
*#run
*#plot db(vout/vin)
*#set units=degrees
*#plot ph(vout/vin)

.ac dec 100 1 10k

Vin Vin 0 DC 1 AC 1
Rin Vin vm 1k
Cf Vout vm 1u

X1 Vout 0 vm Ideal_op_amp
.subckt Ideal_op_amp Vout Vp Vm
G1 Vout 0 Vm Vp 1MEG
RL Vout 0 1
.ends
.end

```

Figure 1.33 An integrator example.

Time-Domain Behavior of the Integrator

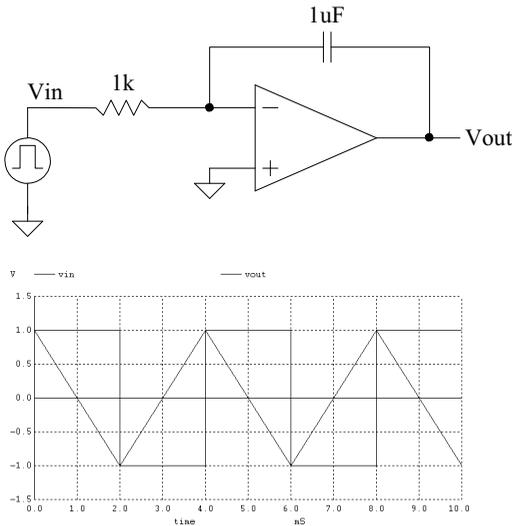
The time-domain behavior of the integrator can be characterized, again, by equating the current in the resistor with the current in the capacitor

$$V_{out} = \frac{1}{C} \int \frac{V_{in}}{R} \cdot dt \quad (1.18)$$

If our input is a constant voltage, then the output is a linear ramp increasing (if the input is negative) or decreasing (if the input is positive) with time. If the input is a squarewave, with zero mean then the output will look like a triangle wave. Using the values seen in Fig. 1.33 for the time-domain simulation seen in Fig. 1.34, we can estimate that if a 1 V signal is applied to the integrator the output voltage will have a slope of

$$V_{out}(t) = \frac{V_{in}}{RC} = \frac{1}{1 \text{ ms}} \quad (1.19)$$

or 1 V/ms slope. This equation can be used to design a sawtooth waveform generator from an input squarewave. Note, however, there are several practical concerns. To begin, we set the output of the integrator, using the `.ic` statement, to ground at the beginning of the simulation. In a real circuit this may be challenging (one method is to add a reset switch across the capacitor). Another issue, discussed later in the book, is the op-amp's offset voltage. This will cause the outputs to move towards the power supply rails even with no input applied. Finally, notice that putting a `+` in the first column treats the SPICE code as if it were continued from the previous line.



*** Figure 1.34 ***

```

*#destroy all
*#run
*#plot vout vin

.tran 10u 10m
.ic v(vout)=0

Vin Vin 0 DC 1
+ pulse -1 1 0 1u 1u 2m 4m
Rin Vin vm 1k
Cf Vout vm 1u

X1 Vout 0 vm Ideal_op_amp
.subckt Ideal_op_amp Vout Vp Vm
G1 Vout 0 Vm Vp 1MEG
RL Vout 0 1
.ends
.end

```

Figure 1.34 Time-domain integrator example.

Convergence

A netlist that doesn't simulate isn't converging numerically. *Assuming* that the circuit contains no connection errors, there are basically three parameters that can be adjusted to help convergence: ABSTOL, VNTOL, and RELTOL.

ABSTOL is the absolute current tolerance. Its default value is 1 pA. This means that when a simulated circuit gets within 1 pA of its "actual" value, SPICE assumes that the current has converged and moves onto the next time step or AC/DC value. VNTOL is the node voltage tolerance, default value of 1 μ V. RELTOL is the relative tolerance parameter, default value of 0.001 (0.1 percent). RELTOL is used to avoid problems with simulating large and small electrical values in the same circuit. For example, suppose the default value of RELTOL and VNTOL were used in a simulation where the actual node voltage is 1 V. The RELTOL parameter would signify an end to the simulation when the node voltage was within 1 mV of 1 V ($1V \cdot \text{RELTOL}$), while the VNTOL parameter signifies an end when the node voltage is within 1 μ V of 1 V. SPICE uses the larger of the two, in this case the RELTOL parameter results, to signify that the node has converged.

Increasing the value of these three parameters helps speed up the simulation and assists with convergence problems at the price of reduced accuracy. To help with convergence, the following statement can be added to a SPICE netlist:

```
.OPTIONS ABSTOL=1uA VNTOL=1mV RELTOL=0.01
```

To (hopefully) force convergence, these values can be increased to

```
.OPTIONS ABSTOL=1mA VNTOL=100mV RELTOL=0.1
```

Note that in some high-gain circuits with feedback (like the op-amp's designed later in the book) decreasing these values can actually help convergence.

Some Common Mistakes and Helpful Techniques

The following is a list of helpful techniques for simulating circuits using SPICE.

1. The first line in a SPICE netlist must be a comment line. SPICE ignores the first line in a netlist file.
2. One megaohm is specified using 1MEG, not 1M, 1m, or 1 MEG.
3. One farad is specified by 1, not 1f or 1F. 1F means one femto-Farad or 10^{-15} farads.
4. Voltage source names should always be specified with a first letter of V. Current source names should always start with an I.
5. Transient simulations display time data; that is, the x-axis is time. A jagged plot such as a sine wave that looks like a triangle wave or is simply not smooth is the result of not specifying a maximum print step size.
6. Convergence with a transient simulation can usually be helped by adding a UIC (use initial conditions) to the end of a .tran statement.
7. A simulation using MOSFETs must include the scale factor in a .options statement unless the widths and lengths are specified with the actual (final) sizes.
8. In general, the body connection of a PMOS device is connected to V_{DD} , and the body connection of an n-channel MOSFET is connected to ground. This is easily checked in the SPICE netlist.
9. Convergence in a DC sweep can often be helped by avoiding the power supply boundaries. For example, sweeping a circuit from 0 to 1 V may not converge, but sweeping from 0.05 to 0.95 will.
10. In any simulation adding .OPTIONS RSHUNT=1E8 (or some other value of resistor) can be used to help convergence. This statement adds a resistor in parallel with every node in the circuit (see the WinSPICE manual for information concerning the GMIN parameter). Using a value too small affects the simulation results.

ADDITIONAL READING

- [1] Wanlass, F.M. “Low Standby-Power Complementary Field Effect Transistor,” US Patent 3,356,858, filed June 18, 1963, and issued December 5, 1967.

PROBLEMS

- 1.1 What would happen to the transfer function analysis results for the circuit in Fig. 1.11 if a capacitor were added in series with R1? Why? What about adding a capacitor in series with R2?
- 1.2 Resimulate the op-amp circuit in Fig. 1.15 if the open-loop gain is increased to 100 million while, at the same time, the resistor used in the ideal op-amp is increased to $100\ \Omega$. Does the output voltage move closer to the ideal value?
- 1.3 Simulate the op-amp circuit in Fig. 1.15 if V_{in} is varied from -1 to $+1V$. Verify, with hand calculations, that the simulation output is correct.