ELEC3106

Electronics

Lecture notes: circuit simulation with SPICE

Objective

The objective of these brief notes is to supplement the textbooks used in the course on the topic of circuit simulation; specifically using SPICE-type simulators. The present notes give only a very brief introduction to circuit simulation and should be used in conjunction with the laboratory notes. Look in your circuit simulator manual for complete descriptions of models, syntax, and capabilities.

Circuit simulators

Circuit simulators can be an invaluable tool in both understanding, designing and debugging electrical and electronic circuits. Most circuit simulators come from the common SPICE ancestor which was originally designed to simulate integrated circuits (SPICE stands for Simulation Program with Integrated Circuits Emphasis). While simulations of integrated circuits prior to fabrication remain particularly important (once fabricated, mistakes on a chip cannot be corrected, and the circuit can only be observed through the pins on the chip), circuit simulation in general has become an important engineering tool. With the power computers of today, it is often quicker to set up a simulation than building a physical circuit. Also, it is possible to do parametric investigations (e.g. sweeping component values) and other forms simulations which are impractical (or impossible) in a laboratory.

A very important thing to remember, however, is that any type of simulation can only be a *simplified model* of a real, physical system. It is very easy to set up a simulation that bears no resemblance with any possible physical implementation: “garbage in, garbage out”. Thus in using a simulator, it is necessary to have both qualitative and quantitative expectations of what simulation results to expect – and when discrepancies occur revise simulations and/or analysis to bring expectations and simulations into broad agreement.

There are many different circuit simulators from different companies and organisations – HSpice, PSpice, LTSpice, ngspice, Spectre, Eldo, AFS, Aplac to name a few – “SPICE” is often used as a generic name.

SPICE input file

Originally the input to SPICE simulation programs were ASCII files describing netlists and simulation directives as shown below. This example defines the netlist of a simple voltage driven RC low-pass filter and a directive to simulate the filter gain against frequency. These days, of course, SPICE packages comes with graphical user interfaces: schematics are drawn and simulation directives given in menus. Anyone seriously using circuit simulators, however, will soon make contact with the still used underlying SPICE input files, and some of the idiosyncrasies seen in circuit simulators (and their GUIs) comes from this legacy.

```
V1 1 0 1V AC 1
R1 1 2 1e3
C1 2 0 0.1uF
.AC dec 10 100Hz 100Meg
```

A few key point to note about SPICE input files:

- SPICE files are (usually) case insensitive.
- Unit prefixes may be used with their usual meaning (`u`, `m`, `k`, etc.). Note: `m` or `M` means “milli”; for “mega” use `Meg`.
- Units may be added (optionally).
- SPICE commands start with full-stop as in `.AC`.
- A line beginning with + is an extension of the previous line.
- Spaces and “exotic” characters cannot be used in names (this include filenames) – restrict use to alphanumeric characters (including underscores).

SPICE models

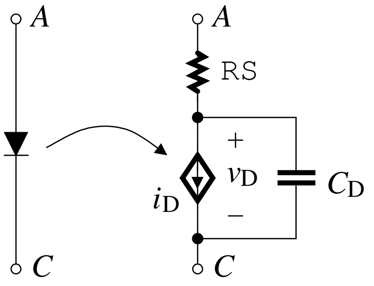
SPICE has built-in parametrisable models for most electronic components: resistors, capacitors, inductors, bipolar transistors, MOS transistors and many others. Many of these models also have various flavours – some with 100s of parameters (consult your manual). The parameters enable the user to build (or import) realistic models for most physical components – from small signal transistors like BC548 to power transistors like MJE13009. Model parameters all have default values, so often it is not necessary to set all parameters in order to define useful models.

A diode model

An example of a component “model deck” is shown below: this defines a new diode model `MYDIODE` and instantiates a diode (`DX`) of this type in the netlist from node `NA` (anode) to node `NK` (cathode).

```
.MODEL MYDIODE D(IS=1E-10 RS=0.1OHM TT=2NS CJO=2P)
DX NA NK MYDIODE
```

A simplified version of the circuit model that SPICE implement for a diode is illustrated in the figure below; here:



$$i_D = \mathtt{IS}\left(e^{v_D/(\mathtt{N}V_T)} - 1\right)$$

$$C_D = \frac{\mathtt{TT}\cdot\mathtt{IS}}{V_T} e^{v_D/(\mathtt{N}V_T)} + \frac{\mathtt{CJO}}{(1 - v_D/\mathtt{VJ})^{\mathtt{M}}},$$

where $V_T = kT/q$ is the thermal voltage and the model parameters are given in the following table:

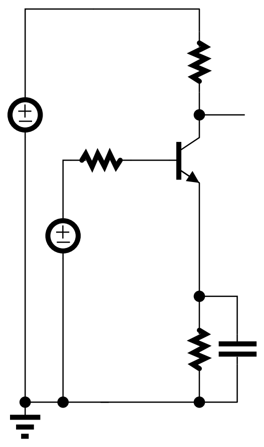
Description	name	default
Series resistance	`RS`	$0\,\Omega$
Saturation current	`IS`	$10\,\text{fA}$
Emission coefficient	`N`	1
Transit time	`TT`	$0\,\text{s}$
Zero-bias junction capacitance	`CJO`	$0\,\text{F}$
Junction potential	`VJ`	$1\,\text{V}$
Grading coefficient	`M`	0.5

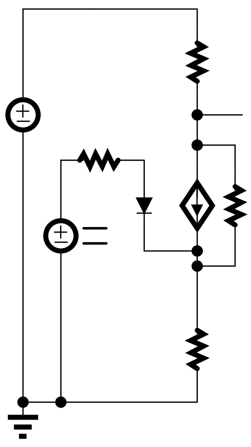
Manufacturer models

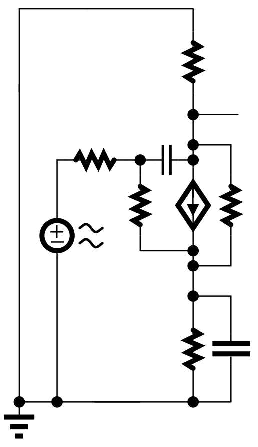
Manufacturers of discrete components normally provide SPICE models for their devices, and many device models may already be included in your simulation software installation from start. Manufacturers of integrated circuits, such as amplifiers or data converters, also usually provide (simplified) SPICE models for their devices. Note, though, that while simple device models are often specified quite accurately, more complex components – such as op-amps – are usually very simplified models and may not include all relevant effects (PSRR, for instance, is often not properly modelled).

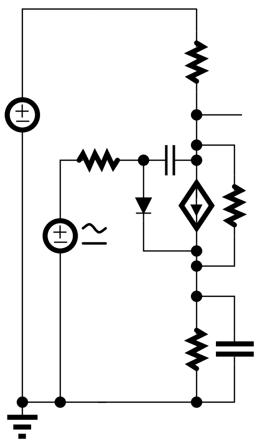
Core SPICE simulations

SPICE has three core types of simulations that it performs: DC simulation, AC simulation, and Transient simulation. It uses different circuit models for each of the simulation types, as described in the following sections. The figure below illustrates the different models used by SPICE for the common-emitter amplifier show to the left in the figure. The behaviour of components, such as diodes and transistors, in the different simulation types are described in their model definition. Independent sources (e.g. voltage sources) have parameters for each type of the three simulations that need to be set appropriately.


Schematic entry

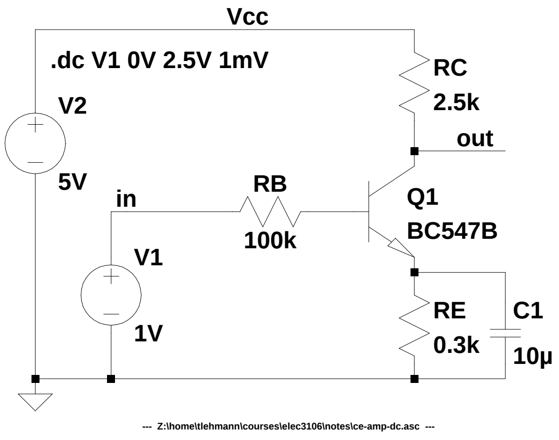

DC model

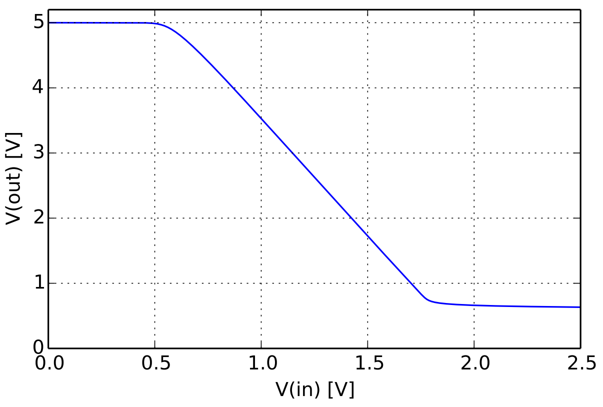

AC model


Transient model

DC simulation

DC simulations model non-linear behaviour of components but do not take into account any time-dependant effects – that is reactive components such as capacitors and inductors are replaced with their zero-frequency equivalents (i.e. open or short circuits). All independent sources take on their DC values. In a DC simulation one usually conducts a sweep of one of the voltage sources such that voltages and currents in the circuit can be observed as a function of this swept voltage – one can sweep the input voltage, for instance, to see what the circuit transfer characteristic or input characteristic looks like. The source swept, will take on the sweep value rather than its ascribed DC value. In most SPICE implementations, you are not limited to sweeping voltages or currents, but can also sweep parameters that you define yourself.





The figure above, shows an example of a common-emitter amplifier simulated in LTSpice (note how the SPICE simulation directive "`.dc V1 0V 2.5V 1mV`" appears in the schematic). What is carried out is a DC simulation where the input voltage (voltage source `V1`) is swept from 0 V to 2.5 V; the resulting output voltage (`V(out)`) as a function of the swept value (`V(in)`) – i.e. the circuit transfer characteristic – is shown in the plot.

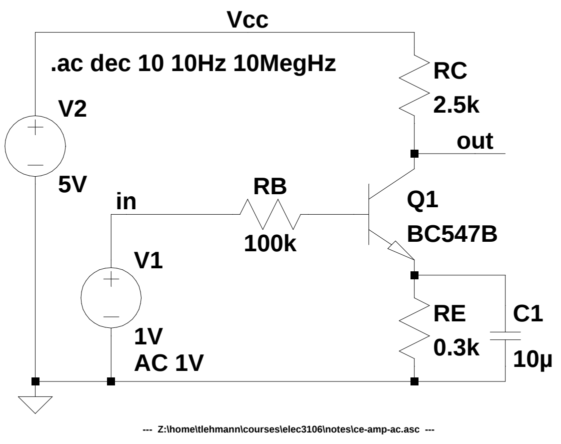
AC simulation

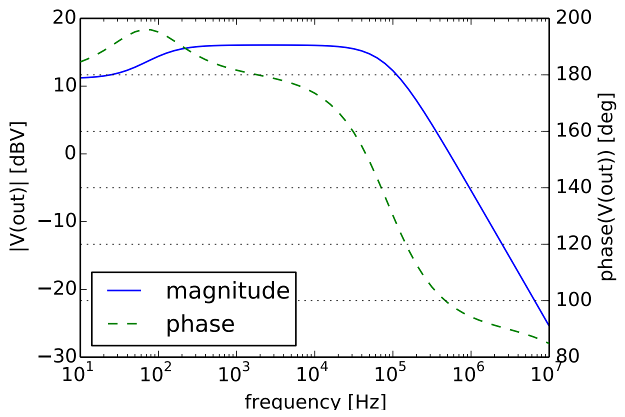
AC simulations are linear, small-signal type of simulations, with frequency being the independent variable. This is usually used for finding (linear) circuit transfer functions or for stability analysis. The DC values of all sources are used to find the circuit operating point, and then the circuit is linearised at this point. At each frequency step, then, complex impedances are found and used to find currents and voltage magnitudes and phases in the circuit. Because the analysis is linear, it is customary to use an AC magnitude of 1 V (0°) for the exciting source (e.g. the input voltage) as gains are then most easily calculated.

The figure above, shows the common-emitter example amplifier simulated in LTSpice using an AC simulation. The input voltage (voltage source `V1`) has a magnitude of 1 V and its frequency is swept from 10 Hz to 10 MHz; the plot shows phase and magnitude of the output voltage (`V(out)`) – i.e. the amplifier transfer function.

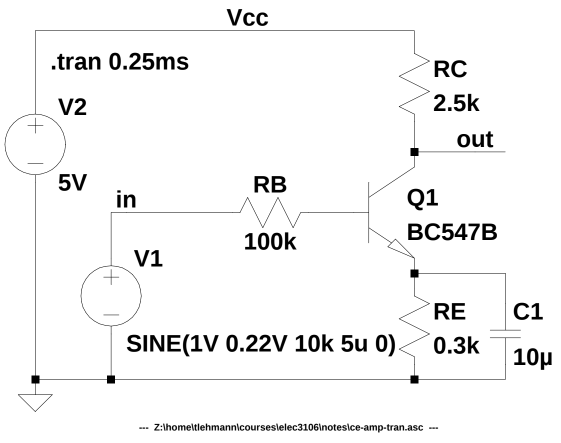
Transient simulation

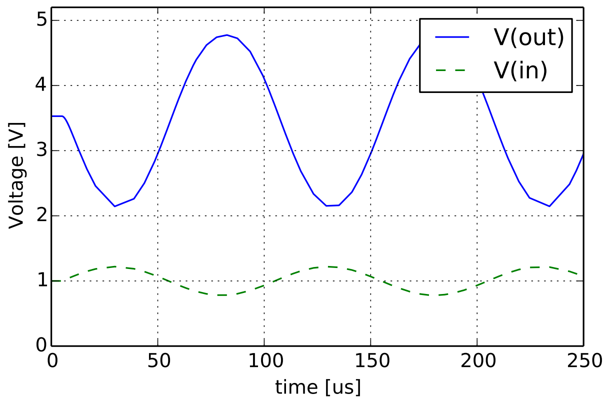
Transient simulations are the most powerful type of simulation. Both the non-linear operation of components and time-varying effects are modelled. Simulation of slew-rate in an amplifier,





for instance, require the use of Transient simulation as this is a non-linear effect where reactive components are involved. The independent variable is time. With this sort of simulation, therefore, it is harder (though possible) to find transfer functions and transfer characteristics which is why the other types of simulations are useful. DC values specified for sources are used when they have no transient-type specification. Transient-type source specifications include sinusoidal (as in the simulation below), exponential, pulse-train, and piece-wise linear functions of time.





The figure above, shows the common-emitter example amplifier simulated in LTSpice using a Transient simulation. The input voltage (voltage source `V1`) is a 10 kHz sinusoid with a DC level of 1 V, and an amplitude of 0.22 V; note also that the sinusoid does not start until 5 μs into the 0.25 ms simulation. The plot of the output voltage (`V(out)`) reveals a slight rounding (distortion) at the positive peaks. The cornered shapes visible at the negative peaks are due to insufficient time resolution used in the simulation.

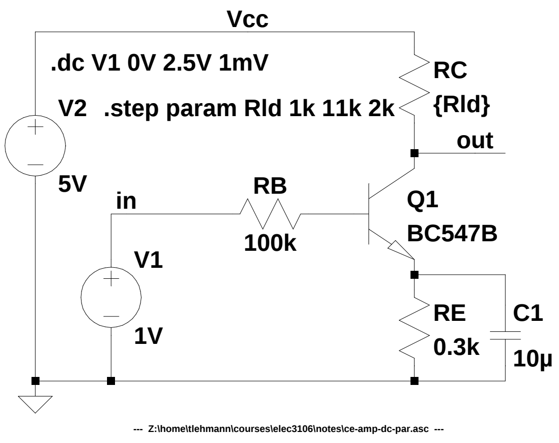
Parameters

Modern SPICE dialects allow extensive use of parameters and expressions used in the netlists. Parameters are defined using the `.param` directive. The LTSpice syntax is to put expressions to be evaluated in braces (`{}`), as shown in the example below that defines a parameter `myr` and assign twice this value to the resistor `Rx`.

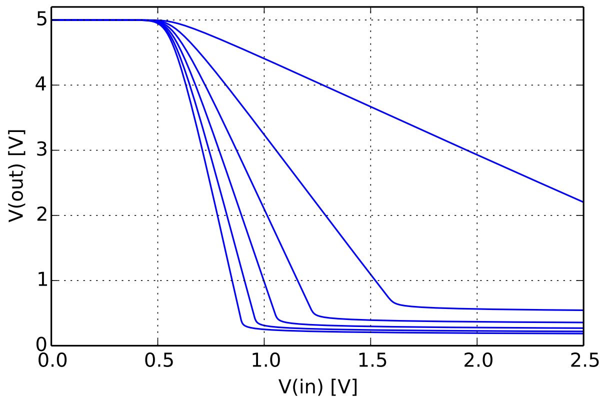
```
.param myr=100k
Rx 3 7 {myr*2}
```

Nested sweeps

Parameterised simulations – or nested sweeps – can be very useful to investigate the influence of, for instance, a component parameter on circuit function. Typically a simulation – for instance a DC simulation – is specified and then ran for a number of different settings of a parameter, resulting in a family of simulation waveforms. Simulations can be nested more than once, although visualising the results can be a bit difficult (unless post-processing software is used).



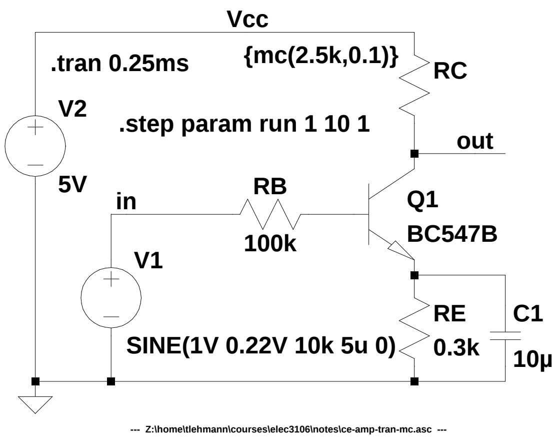
Vcc
.dc V1 0V 2.5V 1mV
V2 .step param Rld 1k 11k 2k
RC
{Rld}
out
5V
in
RB
100k
Q1
BC547B
V1
1V
RE
0.3k
C1
10µ
--- Z:\home\tlehmann\courses\elec3106\notes\ce-amp-dc-par.asc ---

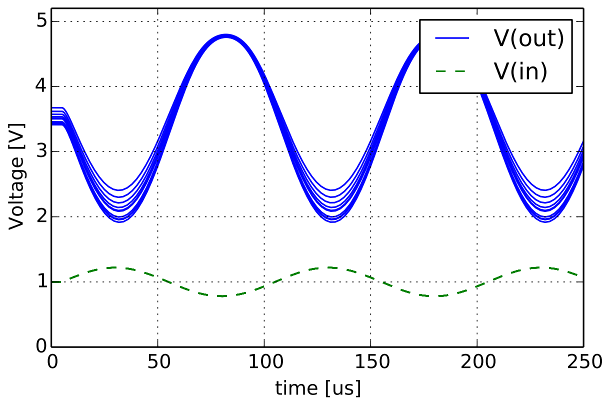


The figure above, shows an example of a parametrically swept DC simulation on the common-emitter example amplifier in LTSpice. Using the `.step` directive, the load resistance (`RC`) is swept between $1\,\mathrm{k}\Omega$ and $11\,\mathrm{k}\Omega$ (in $2\,\mathrm{k}\Omega$ steps); the plot clearly shows the effect on the DC transfer characteristic.

Monte Carlo simulations

A particular type of nested sweeps is used for characterising the effect of component variation: Monte Carlo simulations. Here new random variations are assigned to parameters for each simulation. Using a large number of simulations (typically 30–100 simulations), a good estimate of the effect of component variation and mismatch can be obtained. (Note, LTSpice is not particularly easy to use for characterising mismatch in larger circuits). In LTSpice, the `mc(a,b)` function is used to assign parameters with box distribution in the range $\mathtt{a}(1-\mathtt{b})$ to $\mathtt{a}(1+\mathtt{b})$.





The figure above, shows a 10-run Transient Monte Carlo simulation on the common-emitter example amplifier in LTSpice. Here, the resistor `RC` is given a value of $2.5\,\mathrm{k}\Omega \pm 10\,\%$. The dummy parameter `run` is stepped from 1 to 10 to force the simulation to be carried out 10 times.

Traps and tricks

Be pedantic

One very important thing to remember about computer simulation software is that it will (attempt) to do what you *tell* it to do, not what you *want* it to do. Probably 9/10 problems with simulations are errors in the simulation description – maybe a "`u`" was missing in the description of a transition time or a resistor value was specified as "`1M`" instead of "`1Meg`". The advice here is to be very, *very* pedantic in entering schematics and simulation specifications, and to always double check your entries.

Good practices

Some tricks and good practices in creating circuits:

- Always remember at least one ground connection.
- Remember DC paths to every node in your circuit; if two sub-circuits are galvanically isolated, tie them together with a resistor large enough to prevent any significant current flow between the sub-circuits.
- Put realistic source impedances in power supplies and signals sources.
- Create logical schematic hierarchies and good symbols for sub-circuits.
- Always separate circuits from test benches.
- Never use current sources unless the current is guaranteed to flow without restriction no matter what your circuit condition is (that is: current sources should only be used as inputs to current mirrors).
- In long or complex simulations, only save the signals that you know you want to observe – that can greatly reduce simulation time.

Convergence issues

SPICE simulations in general – and Transient simulations in particular – can be very difficult numerical problems to solve: netlists can result in very large sets of coupled non-linear differential equations that need to be solved repeatedly to high accuracy. For large netlists, therefore, it is common that simulations do not converge. There is no one-size-fits all solution to this problem, unfortunately, but here are some suggestions to try out:

- Reduce simulation accuracy using parameters `abstol`, `reltol`, etc. (go by orders of magnitude).
- Increase simulation accuracy.
- Change integration method (trapezoidal, Gear, etc.).
- Add parasitics to inductors and capacitors.
- If possible, reduce gain in feedback paths. Flip-flops with fast clock edges are notoriously difficult to simulate.
- Make components slightly different rather than identical.
- Make events occur at slightly different times.

- Increase transient transition times.
- Ramp up power supplies from 0 V.
- Use initial conditions in circuits with any form for memory.

Extensions

To get the most out of your circuit simulator, you should, of course, read its manual. SPICE has many more functions than what is described in this short document. But you can also take the use of SPICE further than what is described in the manual: You can, for instance interface to SPICE (either pre- or post simulation) using your favourite programming language – Matlab or Python are good candidates here. You can, for instance, use Python to generate a complicated voltage waveform (LTSpice allows you to specify a PWL waveform from a file), or even generate parts of your netlist. With the power of a high-level programming language, the possibilities here are endless. You can likewise use Python to read in the results from a SPICE simulation (download `ltspy.py` from the course website for this), and do advanced post-processing on the simulation data – for instance doing 3D plots of parameterised simulations or finding propagation delays or bandwidths using your own code.