ECE 220

introduction to `spice` source files

The Input (*.cir) File

In `pspice`, this file must be named `<filename>.cir`. The source file for any version of `spice` has the following format.

```
TITLE
ELEMENT DESCRIPTIONS
.MODEL STATEMENTS
ANALYSIS COMMANDS
OUTPUT COMMANDS
.END
```

Important points:

- The first line of this file is used as a title on output files. It is not included in the circuit description. A common and frustrating error is caused by omitting the title.
- The file must end with the command `.END` followed by a newline.
- Comment entire lines by beginning them with '*' or the rest of a line by using a ';'
- Continue a statement on a new line with '+'

The Circuit Description

A circuit description in `spice`, which is frequently called a **netlist**, consists of a statement defining each circuit element.

Connections are described by naming nodes. (The usual names are actually numbers.) One node name has a defined meaning. **Node 0 is ground!**

The format of an element description is

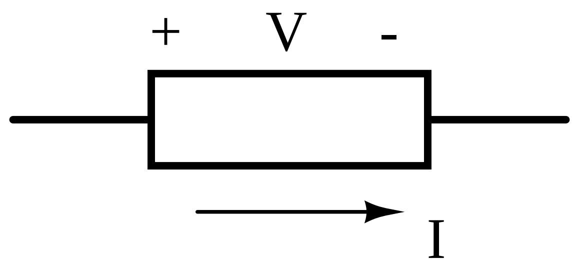
```
<letter><name> <n1> <n2> ...[mname] [parvals]
```

where `<...>` must be present and `[...]` is optional.

- `<letter>` is a single letter denoting the component type
- `<name>` is a unique alpha-num combination describing the particular instance of this component
- `<ni>` is the name of a node
- `[mname]` is the (optional) model name
- `[parvals]` are (sometimes optional) parameter values

Sign Conventions

Two-terminal elements (including sources!) assume the “load” sign convention



The power $P = IV$ associated with an element is positive when the element **absorbs** power.

`spice` shows sources delivering negative power.

Passive Elements

The `<letter>` that begins an element instance denotes the circuit element. The passive elements are

`R` or `r` for resistors, `L` or `l` for inductors, and `C` or `c` for capacitors.

This `<letter>` is followed by a unique instance name and then (in order) the nodes associated with + and - voltage and the value of the associated parameter (R, L, or C).

Examples:

- `R1 5 0 20k`
- `cload nIN GND 250pF`
- `L4 122 21 4mH`

Powers of Ten

The following abbreviations for powers of ten are recognized by `spice`.

F	P	N	U	M	K	MEG	G	T	MIL
femto	pico	nano	micro	milli	kilo	mega	giga	tera	mil (10^{-3} inch)
10^{-15}	10^{-12}	10^{-9}	10^{-6}	10^{-3}	10^{+3}	10^{+6}	10^{+9}	10^{+12}	25.4×10^{-6}

Once a valid suffix is read, `spice` ignores following letters. Thus `10pF`, `10pAmps`, and `10psec` all simply represent the value 10^{-12}.

`MIL` is used to convert distances in thousands of an inch, since `spice` uses metric lengths.

Independent Sources

```
V<name> <n+> <n-> [type] <val>
```

defines an independent voltage source with its + terminal at node `n+` and its - node at node `n-`.

```
I<name> <n+> <n-> [type] <val>
```

defines an independent current source whose current flows through the source from node `n+` to node `n-`.

Examples:

- `Vdd 4 0 5` defines a 5 V source with the + terminal connected at node 4 and the - terminal connected at node 0 (ground)
- `ibias 18 4 DC 15m`
- `V2 3 0 25V` (`spice` recognizes the common abbreviations for units, which helps to make source files more easily understood by humans.)

The optional `type` argument will be described in the analysis commands.

Voltage-Controlled Dependent Sources

The voltage-controlled dependent sources are defined using statements of the form

```
<letter><name> <nout+> <nout-> <nc+> <nc-> <gain>
```

or

```
<letter><name> <nout+> <nout-> (<nc+>,<nc->) <gain>
```

where `E` is a voltage-controlled voltage source, `G` is a voltage-controlled current source, the output voltage is connected between nodes `nout+` and `nout-`, and the control voltage is measured at node `nc+` with respect to node `nc-`.

Examples:

- `Ex 5 1 4 3 10` defines a voltage source that makes node 5 a voltage $10(v_4 - v_3)$ above the voltage at node 1
- `G1 2 1 (5,8) 50m` defines a current source connected between node 2 (the + node) and node 1 and supplying a current $50\text{m}\mho(v_5 - v_8)$

Current-Controlled Dependent Sources

The current-controlled dependent sources are defined by statements of the form

```
<letter><name> <nout+> <nout-> <vcontrol> <gain>
```

where `F` is a current-controlled current source, `H` is a current-controlled voltage source, and the output current source is connected between nodes `nout+` and `nout-`, with positive current flowing through the source from node `nout+` to `nout-`. The control current flows from the positive node of the source `vcontrol` through the source and out the negative node.

Examples:

- `Fds 11 9 Vsens 1.25` defines a current source connected from node 11 to node 9 that generates a current 1.25 times the current flowing through the source `Vsens`.
- `H1 30 20 v5 100k` defines a voltage source connected from node 30 to node 20 and supplying a voltage $100\,\mathrm{k}\Omega$ times the current through the source `v5`.

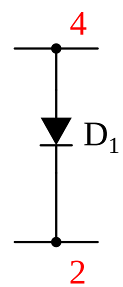
It is frequently necessary to add a voltage source with value 0 V to the circuit to sense the control current for these sources!

Diodes

Diodes are defined using two statements. The netlist definition is of the form

```
D<name> <n+> <n-> <model-name>
```

Example: `D1 4 2 my-diode` corresponds to



The `<model-name>` must be defined using a `.MODEL` statement

```
.MODEL <model-name> D ( [parameter = value] ...)
```

Continuing the above example,

```
.MODEL my-diode D ( IS=1.4e-18 N=1.2 )
```

where `IS` is the saturation current and `N` is the ideality factor (sometimes called the emission coefficient). There are approximately 30 parameters which can be specified for diodes. Those not explicitly specified have default values.

Useful Diode Model parameters

Parameter	Description	Units	Default
IS	Reverse saturation current	Amp	10^{-14}
XTI	Temperature exponent of IS		3
N	Ideality factor (emission coefficient)		1
BV	Reverse breakdown “knee” voltage	Volt	∞
RS	Parasitic series resistance	Ohm	0
CJO	Zero-bias junction capacitance	Farad	0

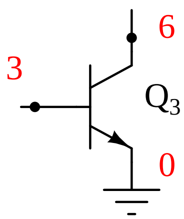
Bipolar Junction Transistors

The BJT also requires both a netlist statement and a `.MODEL`. A BJT is included in the netlist with a statement of the form

```
Q<name> <nc> <nb> <ne> <model-name>
```

where the collector is connected at node `nc`, the base at node `nb`, and the emitter at node `ne`.

Example: `Q3 6 3 0 my-npn` corresponds to



The `model-name` is defined as

```
.MODEL <model-name> <npn | pnp> ( [parameter = value] ...)
```

Continuing the example,

```
.MODEL my-npn npn ( BF = 175 IS=1e-17 VA=75 BR=2 )
```

where BF is the forward β, `IS` is the saturation current, VA is the Early voltage, and BR is the reverse β. There are approximately 60 parameters for BJTs.

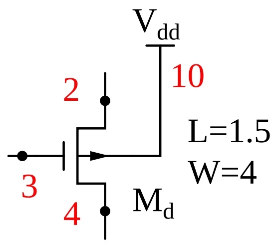
MOSFETs

The BJT again requires both a netlist statement and a `.MODEL`. A MOSFET is included in the netlist with a statement of the form

```
M<name> <nd> <ng> <ns> <nb> <model-name> [L=value] [W=value]
```

where the drain, gate, source, and body are connected at nodes `nd`, `ng`, `ns`, and `nb` respectively. The length `L` and width `W` are optional.

Example: `Md 4 3 2 10 my-pmos L=1.5u W=4u` corresponds to



The `model-name` is defined as

```
.MODEL <model-name> <nmos | pmos> ( [parameter = value] ...)
```

Continuing the example,

```
.MODEL my-pmos pmos ( VTO=-0.8V KP=5e-4 LAMBDA=0.01)
```

where `VTO` is the threshold voltage, `KP` is the transconductance parameter, and `LAMBDA` is the channel-length modulation coefficient.

Subcircuits

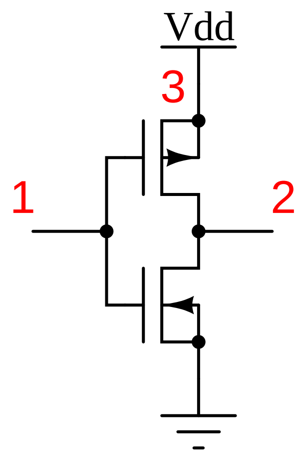
A subcircuit simplifies `spice` netlists by allowing re-use of a set of circuit elements. The syntax is

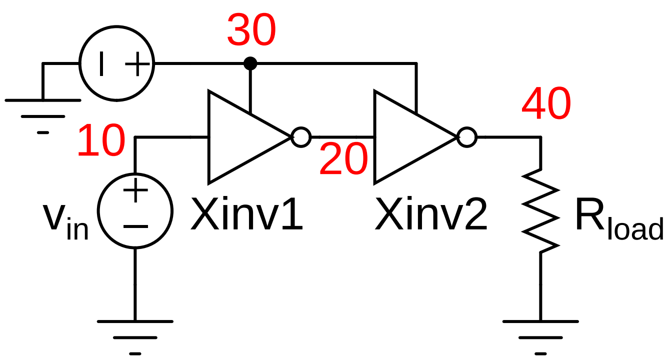
```
SUBCKT <SubName> <N1> <N2> ...

...

.ENDS
```

The `SubName` is the name used to reference the subcircuit, and the nodes are the internal node numbers used to connect to the subcircuit.

A subcircuit can contain any `spice` netlist statements, including `.model` statements and other subcircuits.

Any elements, nodes, models, subcircuits, or definitions in the subcircuit are completely local to the subcircuit.






```
* subcircuit example

.subckt inverter 1 2 3
*              input | |
*               output |
*              dc supply
mp 2 1 3 3 mypmos
mn 2 1 0 0 mynmos
.model mypmos pmos( ... )
.model mynmos nmos( ... )
.ends

Vdd 30 0 dc 3.6V
vin 10 0 dc 0V

Xinv1 10 20 30 inverter
Xinv2 20 40 30 inverter
Rload 40  0 100k

.dc vin 0 3.6 0.1

.end
```

.AC Small-Signal Analysis

.AC calculates the small-signal response as a function of frequency. The command is

```
.AC <type> <npts> <f-start> <f-end>
```

where `<type>` is one of

LIN (linear sweep) The analysis is repeated at `<npts>` linearly-spaced frequencies starting at `<f-start>` and ending at `<f-end>`.

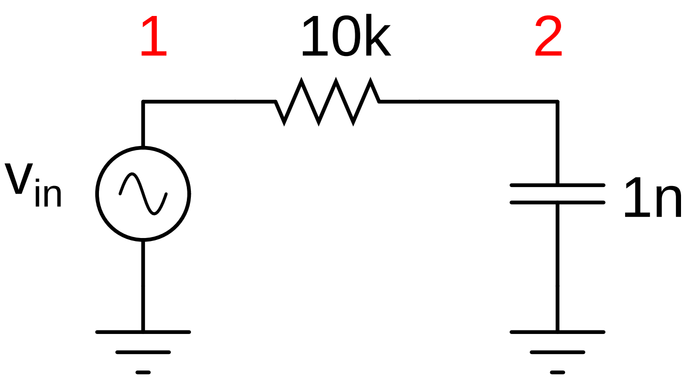
DEC (log sweep by decades) The analysis is repeated at frequencies starting with `<f-start>` and ending with `<f-end>`. The frequencies are equally-spaced on a $\log_{10}$ scale with `<npts>` per decade.

OCT (log sweep by octaves) The analysis frequencies start at `<f-start>` and end with `<f-end>`, with `<npts>` points per octave.

A source whose frequency is swept has a `type` designation AC included in its element description.

.AC Example

```
RC circuit
r1 1 2 10k
c1 2 0 1n
vin 1 0 ac 1 dc 0
.ac dec 10 .01 10
.probe
.end
```



.DC Sweep

.DC calculates the DC voltages and currents in a circuit for a range of values of a chosen variable or variables. The .DC command has three forms.

```
.DC [LIN] <var1> <s1> <e1> <d1> [<var2> <s2> <e2> <d2>]
```

sweeps var1 from s1 to e1 with a linear increment d1. If the second set of values is present, the entire first analysis will be done for each value of var2.

```
.DC <DEC | OCT>] <var1> <s1> <e1> <np1> [<var2> ...]
```

does a log sweep, adjusting var1 from s1 to e1 in decades (DEC) or octaves (OCT) with np1 points per interval. If the second set of values is present, the entire first analysis will be done for each value of var2.

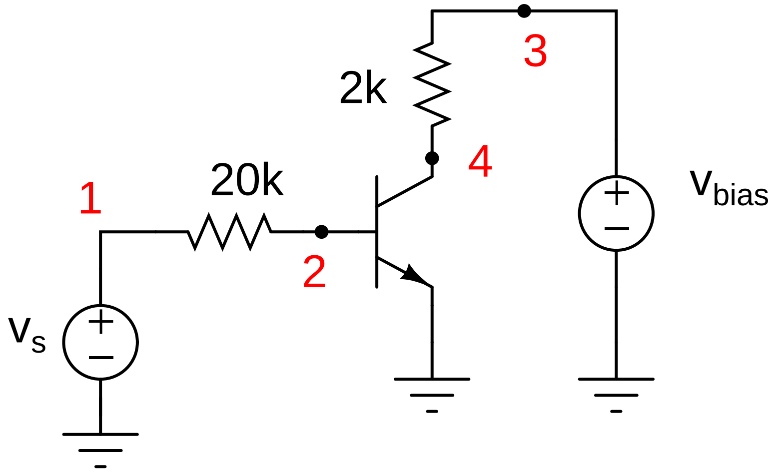
```
.DC <var1> LIST <val1> <val2> [...]  [<var2> ...]
```

performs the analysis for a list of values. Again, the entire first analysis is performed for every value in the second list.

A source whose value is used in a .DC sweep has a type designation DC included in its element description.

.DC Example

```
1 transistor inverter
vbias 3 0 5V
vs 1 0 dc 0V
rb 1 2 20k
rc 3 4 2k
q 4 2 0 gennpn
...
.dc vs 0 5 0.05
...
.end
```



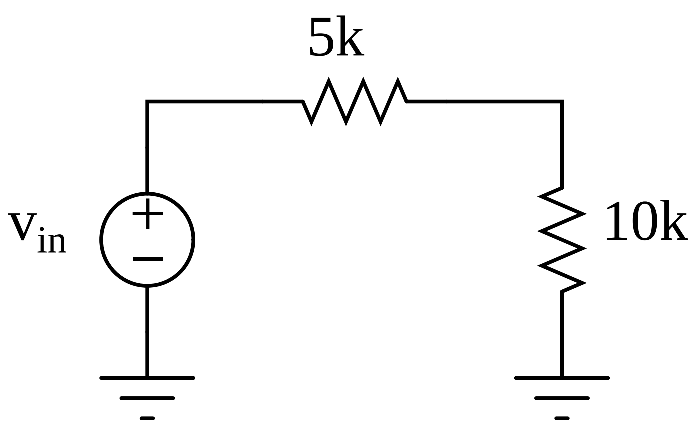
.TF - Transfer Function

```
.TF <var-out> <source-in>
```

calculates the small-signal gain from `source-in` to `var-out`, as well as the input and output resistances. If `var-out` is a current, it must be the current through a voltage source. (You may have to add a $0V$ source to sense this current.)

Example:

```
tf example
vin 1 0 dc 0.5Vac 0.1V
rin 1 2 5k
ro 2 0 10k
...
tf v(2) vin
...
.end
```



.TRAN - Transient Analysis

.TRAN calculates the voltages and currents in a circuit as a function of time. The form of the command is

```
.TRAN[/OP] <print-inc> <t-end> [print-start] [UIC]
```

The operating point for the conditions at the start of the transient analysis is calculated, but will not be displayed unless the optional .TRAN/OP version of the command is used.

print-inc is the time step for the output (**not** the time step for the calculation!), and t-end is the end of the simulation time period.

The optional argument print-start only reports results after that time, and UIC indicates that the simulation should use initial conditions. (Initial conditions are set either with a .IC command or by following the element definition of a capacitor with IC=<voltage> and that for an inductor with IC=<current>.

Input Sources for .TRAN

The source types available include EXP for exponential waveforms, PULSE, PWL for piecewise-linear waveforms, and SIN.

```
EXP( <v1> <v2> [Td1 [Tau1 [Td2 [Tau2]]]])
```

defines an exponential pulse which has an initial value of v1, starts at time Td1, rises with a time constant Tau1 until time Td2, then falls with a time constant Tau2.

```
PULSE( <v1> <v2> [Td [Tr [Tf [pw [tau]]]]])
```

represents a pulse train with low voltage v1 and high voltage v2. The first pulse starts at time T1, has rise time Tr and fall time Tf, holds v2 for a time pw, and has a period tau.

```
PWL ( <t1> <v1> [t2 v2 [t3 v3 ...]]  )
```

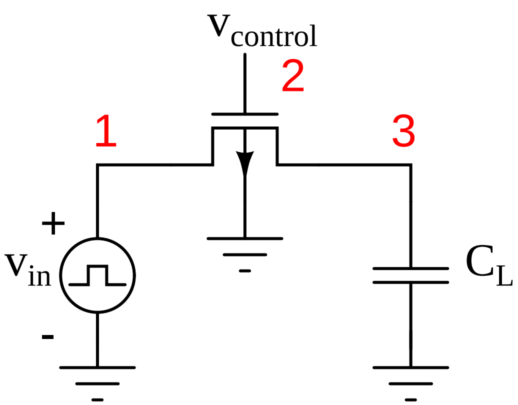
describes a piecewise-linear waveform with each t v defining a “corner.”

```
SIN ( <v0> <va> [f [Td [df [phi]]]])
```

defines an exponentially-damped sine waveform with peak amplitude va, frequency f, damping factor df, and phase phi in degrees. The waveform has the offset voltage v0, and holds that value for a time Td before the time-dependence begins.

.TRAN Example

```
tran example
vin 1 0 pulse(0V 5V 0 0.1n 0.1n
+ 1u 2u)
vcontrol 2 0 dc 5V
mpass 1 2 3 0 mynmos L=5u W=5u
cl 3 0 10p
...
.model mynmos nmos(kp=0.01m
+ vto-0.8)
...
.tran 0.1u 2.5u
...
.end
```



.OP - Operating Point

The .OP command writes detailed information about the default dc operating point to be written in the *.out file. (This information is actually calculated by spice.)

The current and power of all voltage sources is given, as well as the small-signal model parameters for all semiconductor devices.

The .TRAN/OP command supplies operating point information for the bias point of a transient analysis.

Writing results to a file

The results of an analysis request can be written to a file using the `.PRINT` command. The form of the command is

```
.PRINT <type> <OV1> <OV2> <Ov3> ...
```

The output variables are `OV1, OV2,` .... Node voltages and branch currents can be specified as magnitude (`M`), phase (`P`), real (`R`), or imaginary (`I`) by adding the appropriate suffix to `V` pr `I` using the following designations:

M: Magnitude

DB: Magnitude in dB

P: Phase

R: Real part

I: Imaginary part

Example: `.print dc vm(4,0) vdb(4,2) ip(3,2)`