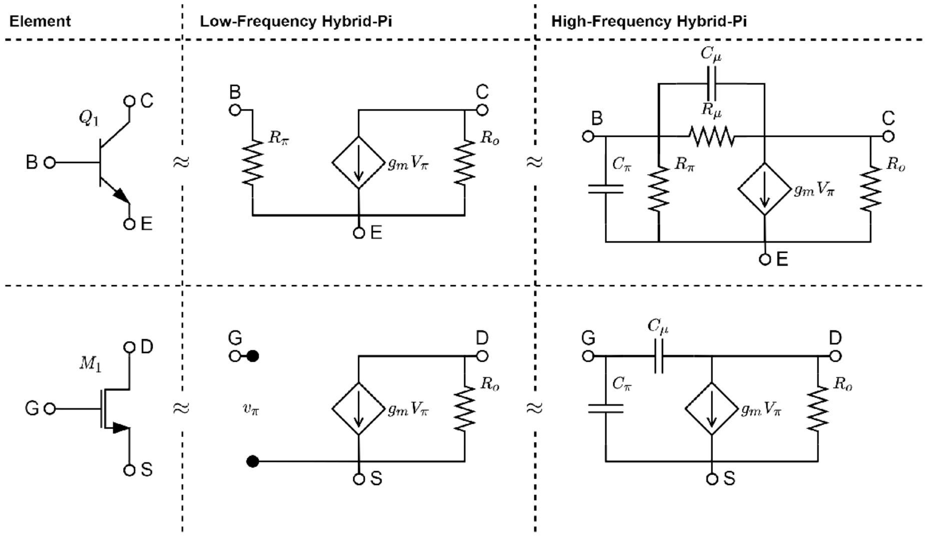
Transistor circuits

ELABorate can handle transistors by modelling them using an appropriate set of linear elements.

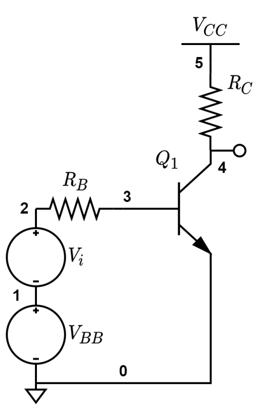


1. BJTs

We start by loading in a circuit containing a BJT. In this case, it's a simple common-source amplifier with biasing. The circuit is from Microelectronic Circuits 7th edition, page 410.

```
circuit = Circuit('circuits/bjt_cs_amp.txt');
circuit.list
```

```
ans =
    'V_BB 1 0 DC V_BB
     V_CC 5 0 DC V_CC
     V_i 2 1 AC V_i
     R_B 2 3 R_B
     R_C 4 5 R_C
     Q_1 3 4 0 beta_Q_1
     '
```

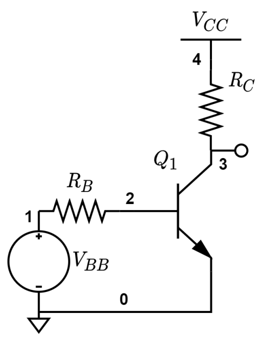


Typically, when analyzing such circuits, we split up the analysis into a DC-part and an AC-part. To find the DC-equivalent of the circuit above, one can simply call the `dc_eq` function (short for *direct-current-equivalent*).

This function uses lower-level functions such as `short`, `open` and `clean` to modify the given circuit into its DC-equivalent.

```
ELAB.dc_eq(circuit);
circuit.list
```

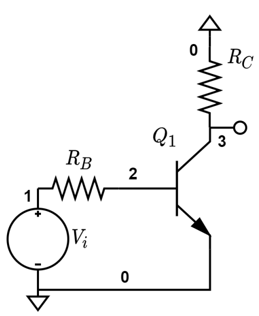
```
ans =
    'V_BB 1 0 DC V_BB
     V_CC 4 0 DC V_CC
     R_B 1 2 R_B
     R_C 3 4 R_C
     Q_1 2 3 0 beta_Q_1
     '
```



Similarly, the program can convert the given circuit into its AC-equivalent.

```
circuit = Circuit('circuits/bjt_cs_amp.txt');
ELAB.ac_eq(circuit);
circuit.list
```

```
ans =
    'V_i 1 0 AC V_i
     R_B 1 2 R_B
     R_C 3 0 R_C
     Q_1 2 3 0 beta_Q_1
     '
```

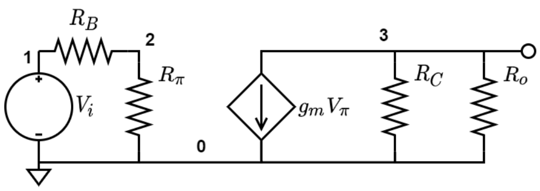


From the AC-equivalent, you can model the transistor as linear elements, using the `hpi` (hybrid-pi) function, which can either assume the circuit will operate in low-frequency "`lf`" or high-frequency "`hf`". We can use the `clone` function to preserve the AC-equivalent, if needed later. The third parameter determines whether the modeller will account for the early effect.

```
circuit = Circuit('circuits/bjt_cs_amp.txt');
ELAB.hybrid_pi(circuit,'lf', true);
```

```
circuit.list
```

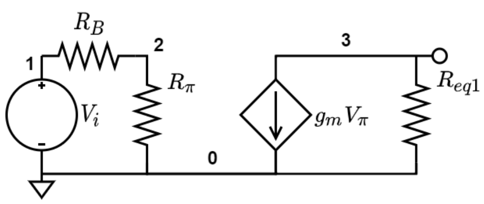
```
ans =
    'V_i 1 0 AC V_i
     R_B 1 2 R_B
     R_C 3 0 R_C
     R_pi_Q_1 2 0 R_pi_Q_1
     R_o_Q_1 3 0 R_o_Q_1
     G_Q_1 3 0 2 0 G_Q_1
     '
```



Of course we may also simplify this circuit. The simplifier takes into account that a dependent source relies on the voltage across R_π and so does not simplify the series $R_B + R_\pi$.

```
ELAB.simplify(circuit);
circuit.list
```

```
ans =
    'V_i 1 0 AC V_i
     R_B 1 2 R_B
     R_pi_Q_1 2 0 R_pi_Q_1
     R_eq1 3 0 (R_C*R_o_Q_1)/(R_C+R_o_Q_1)
     G_Q_1 3 0 2 0 G_Q_1
     '
```




```
ELAB.analyze(circuit)
```

```
Symbolic analysis successful (0.225215 sec).
```

```
circuit.symbolic_node_voltages
```

ans =

$$\begin{pmatrix} v_1 = V_i \\ v_2 = \dfrac{R_{\pi,Q,1}\, V_i}{R_B + R_{\pi,Q,1}} \\ v_3 = -\dfrac{G_{Q,1}\, R_{\mathrm{eq1}}\, R_{\pi,Q,1}\, V_i}{R_B + R_{\pi,Q,1}} \end{pmatrix}$$

```
ELAB.ec2sd(circuit,1,3)
```

```
Symbolic transfer function calculated successfully (3.164900e-03 sec).
ans =
```

$$\frac{v_3}{v_1} = -\frac{G_{Q,1}\, R_{\mathrm{eq}1}\, R_{\pi,Q,1}}{R_B + R_{\pi,Q,1}}$$

```
circuit = Circuit('circuits/bjt_cs_amp.txt');
ELAB.hybrid_pi(circuit,'hf', true);
circuit.list
```

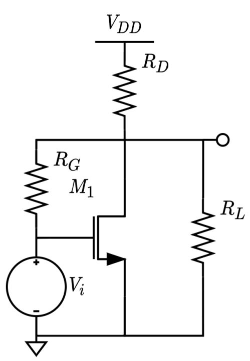
```
ans =
    'V_i 1 0 AC V_i
     R_B 1 2 R_B
     R_C 3 0 R_C
     R_pi_Q_1 2 0 R_pi_Q_1
     R_mu_Q_1 2 3 R_mu_Q_1
     R_o_Q_1 3 0 R_o_Q_1
     C_pi_Q_1 2 0 C_pi_Q_1
     C_mu_Q_1 2 3 C_mu_Q_1
     G_Q_1 3 0 2 0 G_Q_1
     '
```

2. MOSFETs

All the same functionality exists for circuits containing MOSFET transistors. To illustrate this, we load in this circuit.

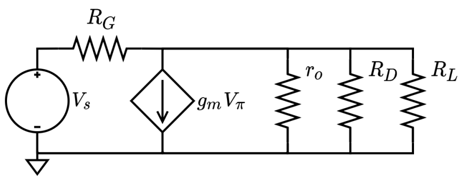
```
circuit = Circuit('circuits/mos_cs_amp.txt');
circuit.list
```

```
ans =
    'V_i 1 0 AC V_i
     V_DD 3 0 DC V_DD
     R_D 2 3 R_D
     R_G 1 2 R_G
     R_L 2 0 R_L
     M_1 1 2 0 100
     '
```




```
ELAB.hybrid_pi(circuit,'lf', true);
circuit.list
```

```
ans =
    'V_i 1 0 AC V_i
     R_D 2 0 R_D
     R_G 1 2 R_G
     R_L 2 0 R_L
     R_o_M_1 2 0 R_o_M_1
     G_M_1 2 0 0 1 G_M_1
     '
```



As this circuit contains three resistors in parallel, we may want to simplify the circuit for further analysis.

```
ELAB.simplify(circuit);
circuit.list
```

```
ans =
    'V_i 1 0 AC V_i
     R_G 1 2 R_G
     R_eq1 2 0 (R_D*R_L*R_o_M_1)/(R_D*R_L+R_D*R_o_M_1+R_L*R_o_M_1)
     G_M_1 2 0 0 1 G_M_1
     '
```

```
ELAB.analyze(circuit)
```

```
Symbolic analysis successful (0.174017 sec).
```

```
ELAB.ec2sd(circuit,1,2)
```

```
Symbolic transfer function calculated successfully (3.558700e-03 sec).
ans =
```

$$\frac{v_2}{v_1} = \frac{R_{\mathrm{eq1}}\ (G_{M,1}\, R_G + 1)}{R_G + R_{\mathrm{eq1}}}$$

3. Numerical evaluation

If there are numerical values available, the program is also able to incorporate these into the modelling of the transistor. We reload the the common-source amplifier with biasing, but this time with numerical values.

```
circuit = Circuit('circuits/bjt_cs_amp_num.txt');
circuit.list
```

```
ans =
    'V_BB 1 0 DC 3
     V_CC 5 0 DC 10
     V_i 2 1 AC V_i
     R_B 2 3 100000
     R_C 4 5 3000
     Q_1 3 4 0 100
     '
```

```
circuit = ELAB.biasing(circuit);
```

```
Symbolic analysis successful (0.307091 sec).

Numerical evaluation successful (0.0570216 sec).
```
$I_{V,\mathrm{BE},Q,1} = 2.3\text{e-}5$

In this case, the biasing/base current is $I_B = 23\mu A$, so the collector current is $I_C = \beta I_B = 2.3mA$. These values are stored in their corresponding transistors, and can be utilized by the hybrid-pi function. This time, we neglect the early effect.

```
ELAB.hybrid_pi(circuit,'lf', false);
circuit.list
```

```
ans = 
    'V_i 1 0 AC V_i
     R_B 1 2 100000
     R_C 3 0 3000
     R_pi_Q_1 2 0 1130.4347826086958178137709958459
     G_Q_1 3 0 2 0 0.088461538461538448576407565561762
     '
```

As can be seen from the netlist, the elements used for the hybrid-pi model has been given numerical values. The voltage-controlled-current-source transconductance is $g_m = 88.5\ mA/V$, and the resistance over which the controlling voltage is, has been calculated to be $R_\pi = 1130\,\Omega$.

We can now treat is as any other linear circuit, like finding the symbolic transfer function.

```
ELAB.ec2sd(circuit,1,3)
```

```
Symbolic analysis successful (0.219763 sec).
Symbolic transfer function calculated successfully (2.268344e-01 sec).
ans = 
```

$$\frac{v_3}{v_1} = -\frac{G_{Q,1}\,R_C\,R_{\pi,Q,1}}{R_B + R_{\pi,Q,1}}$$

Or evaluating the transfer function to check the gain.

```
ELAB.ec2tf(circuit,1,3)
```

```
Numerical evaluation successful (0.0642584 sec).
Transfer function object created successfully (7.878930e-02 sec).

ans =
 
  -2.966
 
Static gain.
```

More examples

```
circuit = Circuit('circuits/bjt_complex_amp.txt');
circuit.list
```

```
ans = 
    'V_cc 4 0 DC V_cc
     V_s 1 0 AC V_s
```

```
     R_s 1 2 R_s
     R_1 3 4 R_1
     R_2 3 0 R_2
     R_C 4 5 R_C
     R_L 6 0 R_L
     C_1 2 3 C_1
     C_2 5 6 C_2
     Q_1 3 5 0 beta_Q_1
     '
```

```
ELAB.hybrid_pi(circuit, 'lf', true);
circuit.list
```

```
ans =
    'V_s 1 0 AC V_s
     R_s 1 2 R_s
     R_1 3 0 R_1
     R_2 3 0 R_2
     R_C 0 4 R_C
     R_L 5 0 R_L
     R_pi_Q_1 3 0 R_pi_Q_1
     R_o_Q_1 4 0 R_o_Q_1
     C_1 2 3 C_1
     C_2 4 5 C_2
     G_Q_1 4 0 3 0 G_Q_1
     '
```

```
ELAB.simplify(circuit);
circuit.list
```

```
ans =
    'V_s 1 0 AC V_s
     R_s 1 2 R_s
     R_L 5 0 R_L
     R_eq1 0 4 (R_C*R_o_Q_1)/(R_C+R_o_Q_1)
     R_eq2 3 0 (R_1*R_2*R_pi_Q_1)/(R_1*R_2+R_1*R_pi_Q_1+R_2*R_pi_Q_1)
     C_1 2 3 C_1
     C_2 4 5 C_2
     G_Q_1 4 0 3 0 G_Q_1
     '
```

```
circuit.Resistors(3).resistance
```

ans =

$$\frac{R_C\, R_{o,Q,1}}{R_C + R_{o,Q,1}}$$

```
circuit.Resistors(4).resistance
```

ans =

$$\frac{R_1\, R_2\, R_{\pi,Q,1}}{R_1\, R_2 + R_1\, R_{\pi,Q,1} + R_2\, R_{\pi,Q,1}}$$