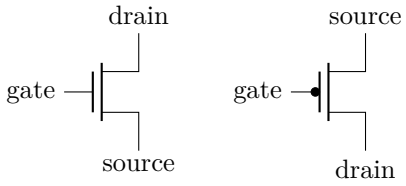
SPICE 4: MOSFETs

ECE 3410, Utah State University

MOSFETs in SPICE

A MOSFET device in SPICE is instantiated by starting a line with the letter `M`:




```
M<name> <drain> <gate> <source> <substrate> <model name> [geometry parameters]
```

In an integrated circuit, MOSFETs can be made with varying sizes, resulting in different channel width and length, and also differing areas and perimeters around their source and drain terminals. These **geometry parameters** can optionally be specified for each individual device. **Although optional, geometry parameters are necessary for accurate simulation.**

MOSFET Substrate Connections

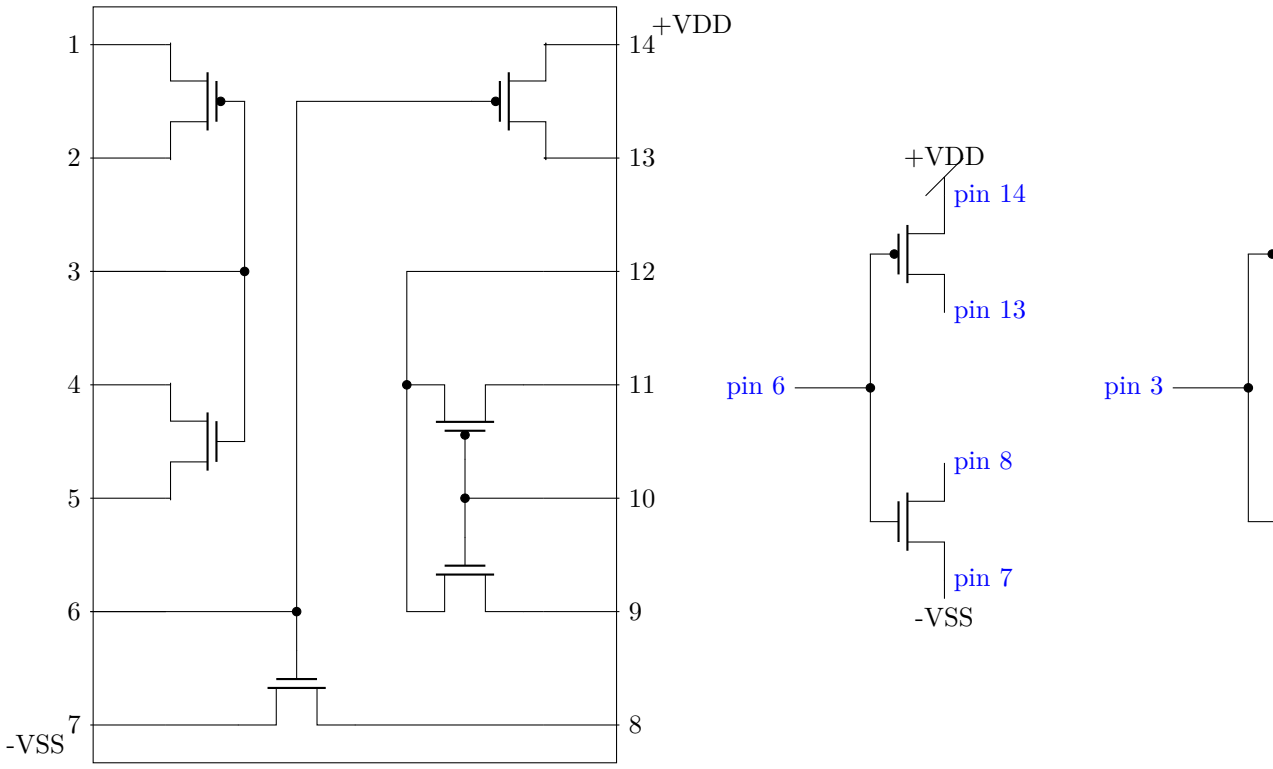
A MOSFET depends physically on the voltage at four nodes: drain, gate, source, and **substrate**. The substrate usually has one of these configurations:

- Discrete MOSFET part: substrate is connected to source.
- Integrated circuit N-type: substrate is connected to **most negative potential** -VSS (or GND if there is a single power supply).
- Integrated circuit P-type: substrate is connected to **most positive potential** +VDD.

In this lab we will use the CD4007 integrated circuit in which all N-type substrates are connected to -VSS, and all P-type substrates are connected to VDD. To ensure proper substrate biasing, it is essential to connect positive and negative power supplies to the indicated pins.

CD4007 MOSFET Array

In this course we are using the **CD4007 MOSFET array chip** which contains a mix of N-type and P-type devices with partially completed wiring as shown below.



Instantiating CD4007 Devices in SPICE

Models for the N-type and P-type models are provided in `models/mosfets.md`.

To use the models, the model file must first be included in the spice netlist. Then the MOSFETs may be placed using statements like this:

```
.include models/mosfets.md

VDD vdd 0 DC 9V
VSS vss 0 DC -9V

* NMOS device:
M1 g1 d1 s1 vss CD4007N W=30u L=10u
```

```
* PMOS device:
M2 g2 d2 s2 vdd CD4007P W=60u L=10u
```

MOSFET Models: CD4007N

MOSFET model parameters describe the device's material composition and physical parameters. SPICE supports many different "levels" of MOS models, from very simple square-law models up to advanced models with dozens of equations and parameters.

The simplest model is **Level 1**. Here are some of the level 1 model parameters for the CD4007 N device:

Parameter	SPICE Symbol	Value
Threshold voltage V_{thN}	VT0	2V
Scale factor $K'_N = \mu_n C_{\text{ox}}$	Kp	$111\mu\text{A/V}^2$
Channel length modulation λ_N	lambda	0.01V^{-1}

MOSFET Models: CD4007P

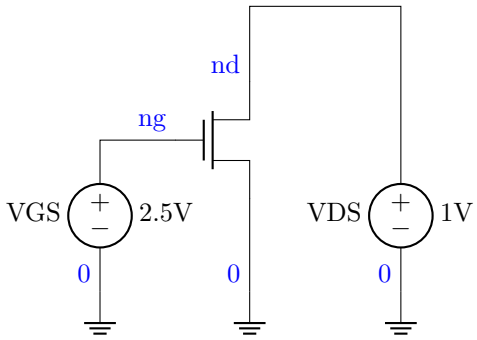
The level-1 parameters for a P-type device:

Parameter	SPICE Symbol	Value
Threshold voltage V_{thP}	VT0	-1.5V
Scale factor $K'_P = \mu_p C_{\text{ox}}$	Kp	$55\mu\text{A/V}^2$
Channel length modulation λ_P	lambda	0.04V^{-1}

The Level 1 model has up to 45 parameters to account for various attributes like terminal capacitances, threshold voltage variation, device noise, and manufacturing imperfections.

Exercise 1: NMOS I/V Transfer Characteristic

First we will simulate an N-type device and compare it to our hand-analysis equations. This schematic is implemented in a file named `netlists/nmos_iv.sp`. Study the each part of the netlist file until you understand each part of it.



NMOS I/V Testbench

To simulate the circuit, a testbench is provided in a file named `tests/nmos_iv_test.sp`. This testbench implements a DC sweep simulation for `vds`, varying it from 0V to 9V in steps of 0.1V. Open the file and study it carefully.

A collection of DC simulations is generated using a `foreach` loop. In each pass through the loop, `vgs` is changed using the `alter` command, so that we repeat the I/V simulation for a sequence of different gate voltages. This allows us to visualize how the current depends on both VDS and VGS.

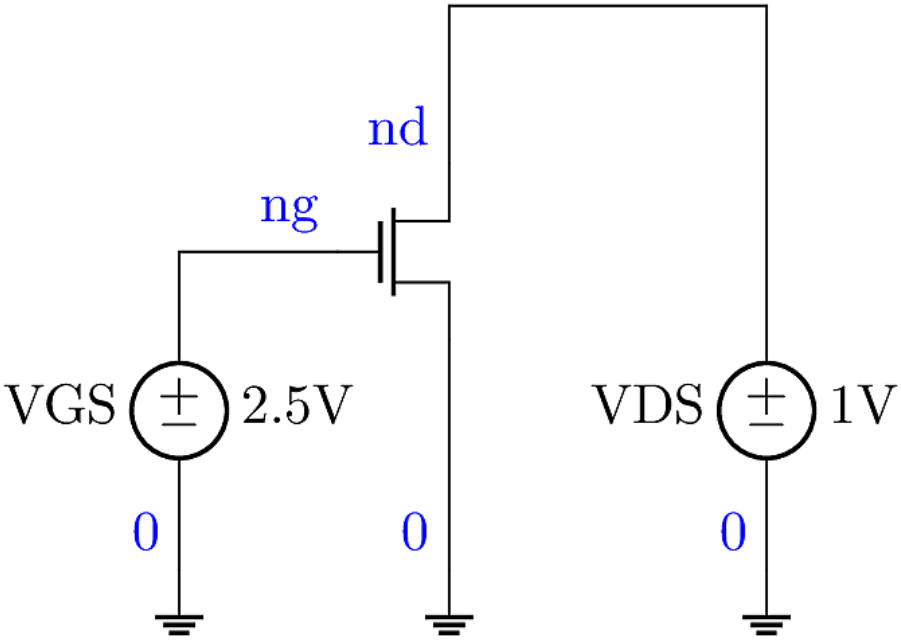
When successive DC simulations are performed, the results are stored under names `dc1.`, `dc2.`, and so on. The control commands shown below will produce the desired simulations and overlay the results in a single plot. The I/V curves are saved in `plots/nmos_iv.svg`.

```
.control

  foreach v 2.5 3 3.5 4 4.5
    alter vgs=$v
    dc vds 0 9 0.1
    let ids=-i(vds)
  end

  plot dc1.ids dc2.ids dc3.ids dc4.ids dc5.ids
  hardcopy plots/nmos_iv.svg dc1.ids dc2.ids dc3.ids dc4.ids dc5.ids

.endc
.end
```

Examine the I/V Curves

Your simulation should produce a family of standard I/V plots like the one shown below. When VDS is increased beyond Vov, the current flattens out (mostly),

indicating the saturation mode. The saturation current level is controlled by the gate voltage. Each curve represents a different gate voltage, therefore a different saturation level.



Comparing to Hand Analysis

Next we will modify the testbench to show the prediction from hand analysis.

Within the `foreach` loop, add the calculations and plot commands shown below. These commands calculate Vov, determine whether the device is in saturation (the `ge` operator is used, meaning "greater than or equal to"), then calculate the saturation and triode current equations, and finally combine them to predict the NMOS current. The prediction is based on the standard MOSFET device equations:

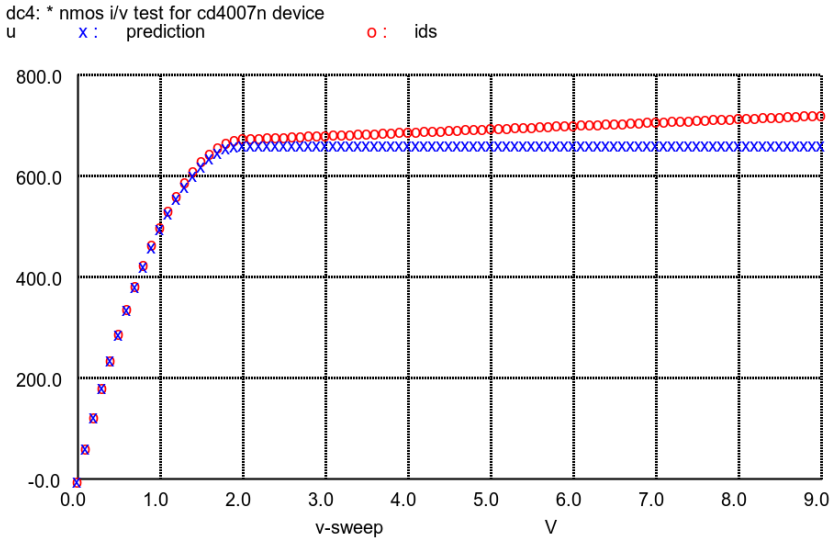
$$i_D = \begin{cases} K'\left(\frac{W}{L}\right)\left(v_{ov}v_{DS} - \frac{1}{2}v_{DS}^2\right), & \text{triode} \\ \frac{1}{2}K'\left(\frac{W}{L}\right)v_{ov}^2, & \text{saturation} \end{cases}$$

```
let vov=ng-2
let saturation=(nd ge vov)
let isat=0.5*(111u)*(3)*(vov^2)
let itri=(111u)*(3)*(vov*nd-0.5*nd^2)
let prediction=saturation*isat + (1-saturation)*itri
plot pointplot ids prediction
```

```
hardcopy plots/nmos_iv_prediction{$v}.svg pointplot ids prediction
```

Understanding the Results

For each value of the gate voltage, you should see a plot like the one shown below. Since the prediction and simulated result are very close, the `pointplot` option was used so they can be distinguished more easily. You should notice that the prediction is close, but not exact.



Improving the Model with Channel Length Modulation

The current is predicted more accurately if we account for **channel length modulation (CLM)**. To do so, we introduce the correction factor (1+lambda*VDS).

$$i_D^\star = \begin{cases} K'\left(\frac{W}{L}\right)\left(v_{ov}v_{DS} - \frac{1}{2}v_{DS}^2\right), & \text{triode} \\ \frac{1}{2}K'\left(\frac{W}{L}\right)v_{ov}^2, & \text{saturation} \end{cases}$$

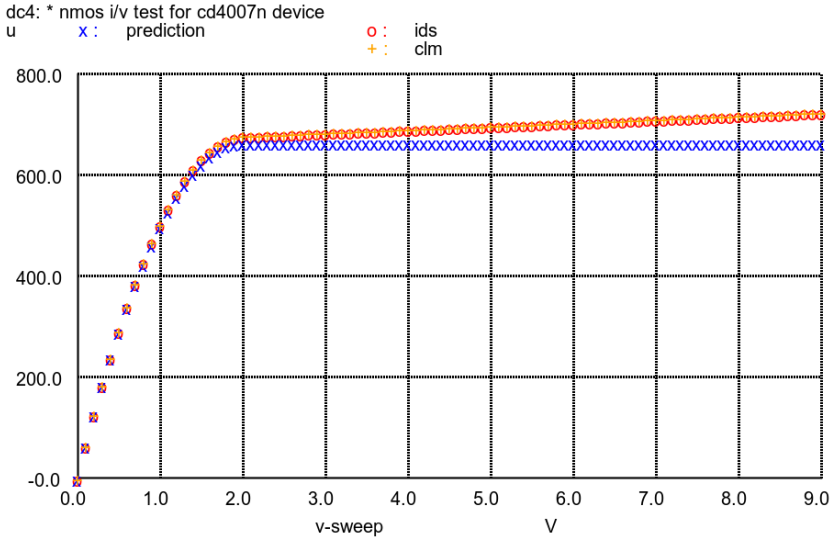
$$i_D = i_D^\star\left(1 + \lambda v_{DS}\right)$$

Add these lines within the `foreach` loop in your testbench:

```
let clm=prediction*(1+0.01*nd)
plot pointplot ids prediction clm
hardcopy plots/nmos_iv_clm{$v}.svg pointplot ids prediction clm
```

Understanding the Results

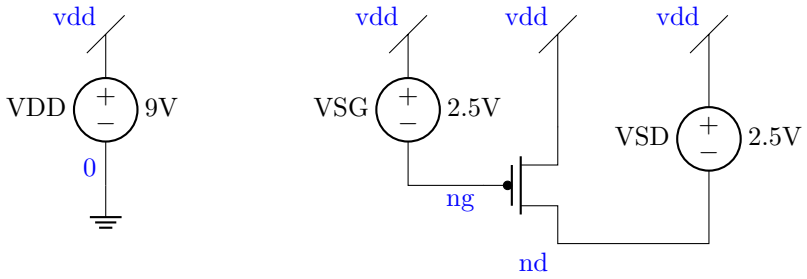
As seen in the plot below, the CLM correction is able to correct for most of the discrepancy between the predicted and simulated currents.



Exercise 2: PMOS I/V Transfer Characteristic

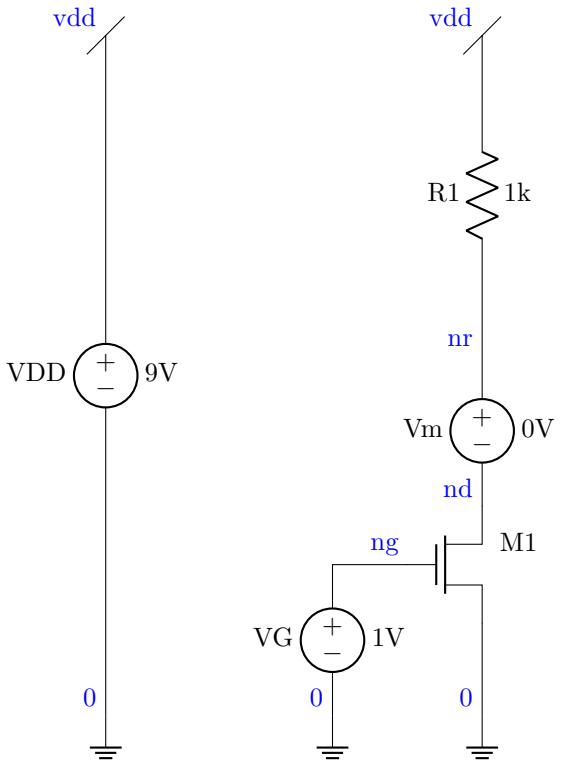
Open the file named `netlists/pmos_iv.sp` which implements the schematic shown below. Study the file carefully until you completely understand it.

For this schematic, **repeat all the simulations and analyses that you performed for the NMOS device.** In all the filenames for netlists, testbenches, and plots, change `nmos` to `pmos`. When calculating predictions, you will need to replace NMOS model parameters with the appropriate PMOS values (see table data provided in the earlier slide).



Exercise 3: NMOS RTL Inverter

Create a file named `netlists/nmos_rtl_inverter.sp` and implement the schematic shown below. Notice that this schematic includes a 0V "meter source" named `Vm`. The meter source is used to observe the branch current, since SPICE saves current data for voltage sources.



NMOS RTL Inverter Testbench

Create a testbench named `tests/nmos_rtl_test.sp` and perform these simulations:

- Repeat for R1 equal to 1k, 10k and 100k Ohms.
  - DC sweep VG from 0V to 9V in steps of 0.1V.
  - Measure the boundary point between triode and saturation, and compare to hand calculation. Name this point `b` with gate/drain voltages `vg_b` and `vd_b`.
  - Measure the final value of VD and compare to hand calculation. Name this point `c` with drain voltage `vd_c`.

Record your hand calculations and comparisons in a text file named `analysis.txt`. Include a separate section for each resistor value.

Perform the simulations using a `foreach` construct like this:

```
.control
  foreach r 1k 10k 100k
     alter R1=$r
     dc vg 0 9 0.1

     let id=i(vm)
     let vov=ng-2

     meas dc vd_b FIND nd WHEN nd=vov
     meas dc vg_b FIND ng WHEN nd=vov
     meas dc vd_c MIN  nd
  end

.endc
.end
```

Exercise 4: Signal Derivatives

Next, alter the testbench so that it computes the **derivative** of ID with respect to VG. You may recall that di_D/dv_G is the **transconductance** of the MOSFET device. Also calculate the derivative of VD with respect to VG, which measures the **voltage gain**. In SPICE, the `deriv` command is used to calculate derivatives. After a DC sweep simulation, all derivatives are calculated with respect to the sweep variable (VG in this case).

```
     let gm   =  deriv(id)
     let gain = -deriv(nd)
```

Measuring the Maximum Gain and its Offset Voltage

The NMOS RTL circuit can be used as an **analog amplifier**. In that application, the **gain** is the derivative dVout/dVin. If VG is the input and VD is the output, then the amplifier's gain is dVD/dVG. Note that the gain varies for different values of VG. To make a high-gain amplifier, we need to know the **offset** VG where the derivative is largest.

SPICE measures the maximum value using the `MAX` measurement. To measure the corresponding voltage at VG, we use the `MAX_AT` measurement and save the result in a variable named `offset`. Also measure the transconductance at that point, using the `FIND/AT` measurement, and save the result in a variable named `maxgm`.

In your `analysis.txt` file, explain how the `offset` relates to the saturation boundary voltages `vg_b`,`vd_b`. Include numerical comparisons.

```
meas dc maxgain MAX    gain
meas dc offset  MAX_AT gain
meas dc maxgm   FIND   gm   AT=offset
```

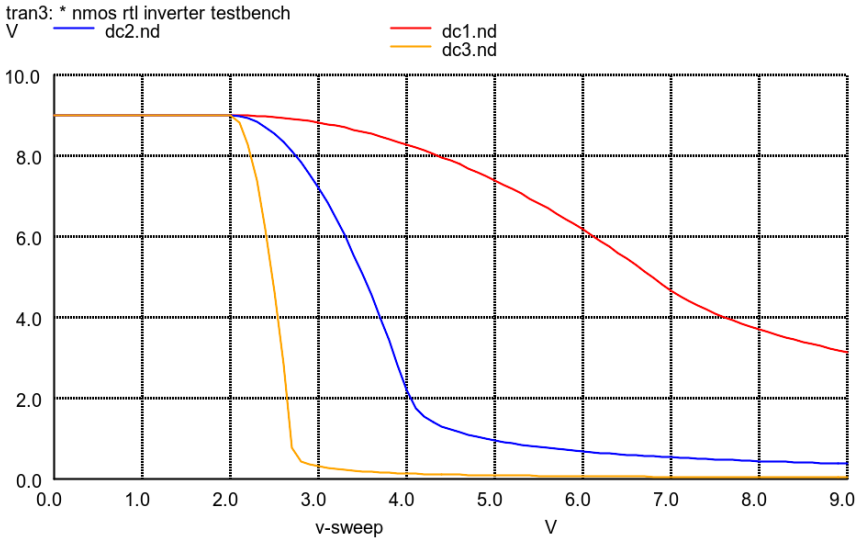
Plotting the Results

There are now three sets of DC simulation results. They can be accessed using prefix `dc1`, `dc2`, and `dc3` as in the lines below.

```
  plot dc1.nd   dc2.nd   dc3.nd
  plot dc1.id   dc2.id   dc3.id
  plot dc1.gain dc2.gain dc3.gain
  plot dc1.gm   dc2.gm   dc3.gm

  hardcopy plots/nmos_rtl_vd.svg   dc1.nd   dc2.nd   dc3.nd
  hardcopy plots/nmos_rtl_id.svg   dc1.id   dc2.id   dc3.id
  hardcopy plots/nmos_rtl_gain.svg dc1.gain dc2.gain dc3.gain
  hardcopy plots/nmos_rtl_gm.svg   dc1.gm   dc2.gm   dc3.gm

end
.endc
.end
```

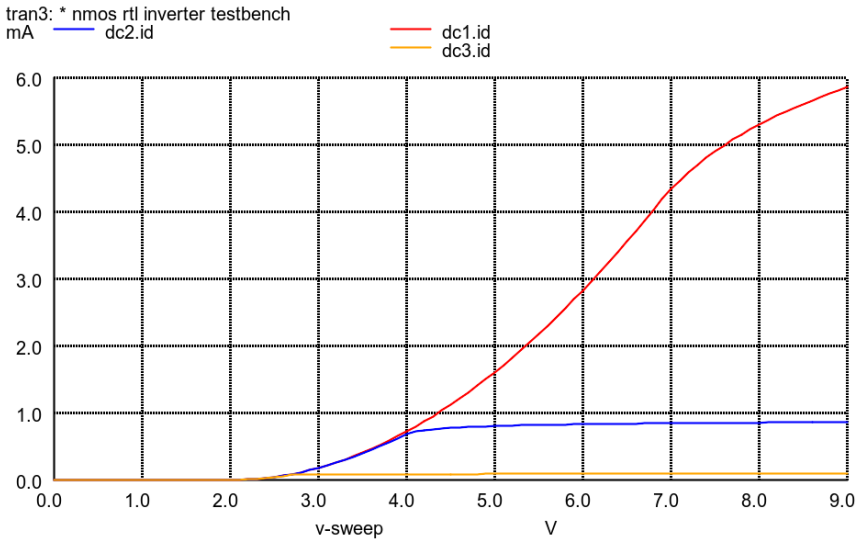
Interpreting the Results: VD vs VG

The transfer characteristic shows a weak logic inversion when R1 is low (case dc1, the red curve), and a better response when R1 is highest (case dc3, the orange curve). A higher resistance also produces a steeper curve – this predicts that the gain should be higher when the resistance is higher.



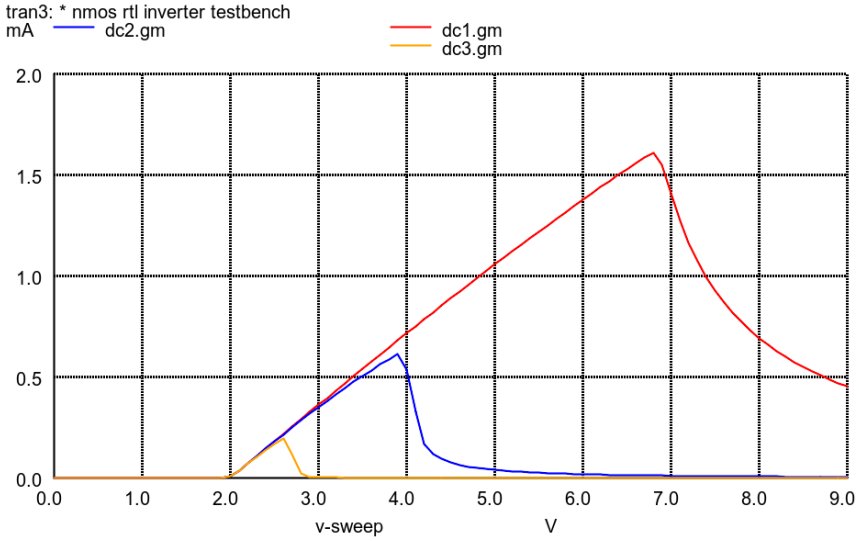
Interpreting the Results: ID vs VG

Since we are sweeping VG, the current follows the square law when VG is small. ID increases parabolically until the device crosses into the triode mode. Since the transconductance is gm=dI/dV, we expect gm to be maximum where the curve is steepest, at the boundary between saturation and triode.



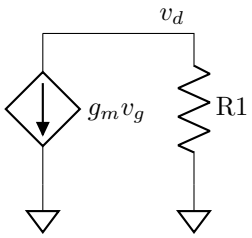
Interpreting the Results: gm vs VG

In saturation mode, the MOSFET can be considered as a **transconductance amplifier** where the input is a voltage (at the gate) and the output is a current (at the drain). Under this interpretation, the transconductance gain is maximized at the boundary between triode and saturation, as seen here:



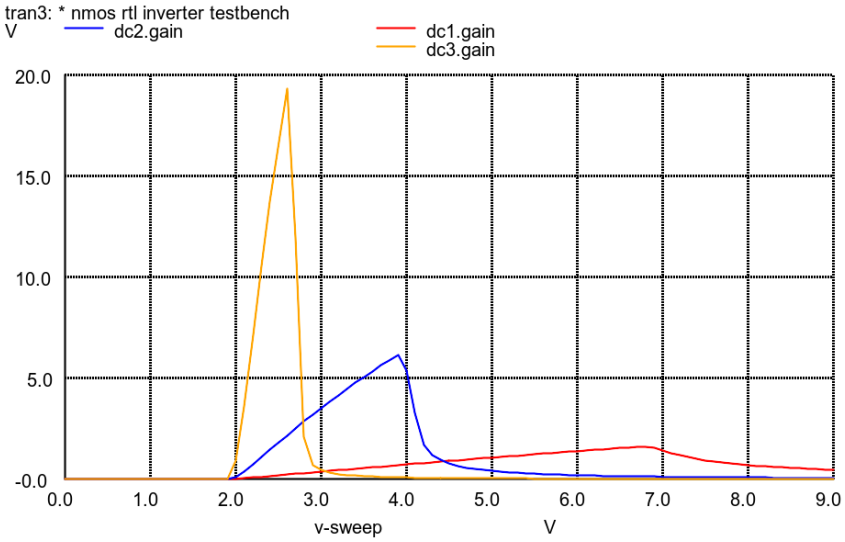
Interpreting the Results: how gm relates to gain

For a small-signal input v_g, the MOSFET (in saturation) acts like a transconductance amplifier with gain g_m and load R1. The output current is $i_d = g_m v_g$, and is scaled by R1 to produce the drain voltage $v_d = -i_d \text{R1} = -g_m v_g \text{R1}$. Therefore the voltage gain is $dv_D/dv_G = v_d/v_g = -g_m \text{R1}$. The gain magnitude is gm*R1.



Interpreting the Results: gain vs VG

Since the gain is gm*R1, the gain curves follow the same shape as the gm curve, but their magnitudes are re-ordered since they are scaled by R1.



Exercise 5: Simulating an Amplifier

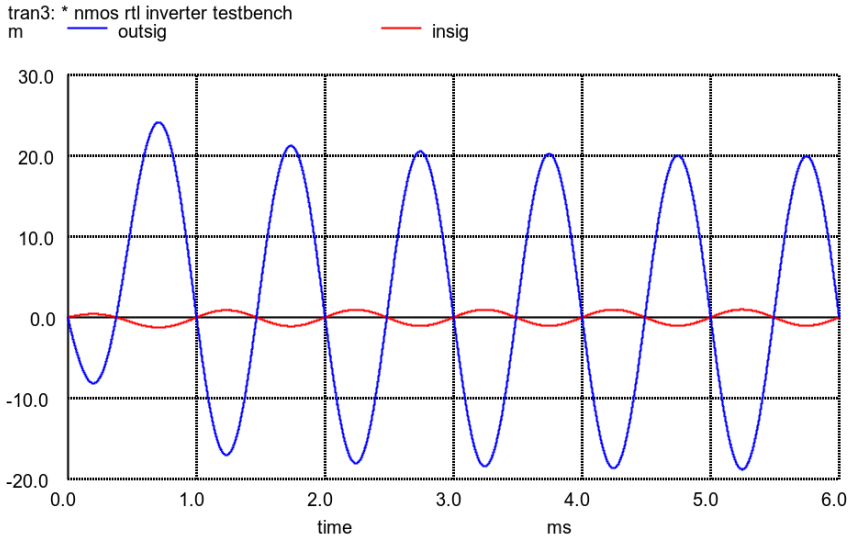
Next, alter the testbench to simulate the RTL inverter as a signal amplifier. Do do so, use the `alter` command to create a sinusoidal input at VG with an offset voltage equal to the measured `offset` value, an amplitude equal to 1mV, and a frequency of 1kHz. Use the `PP` measurement to measure the output peak-to-peak amplitude. The signal gain is the amplitude ratio VD(pp)/VG(pp).

To perform all these tasks, add these lines within the `foreach` loop:

```
alter @vg[sin] = [ $&offset 1m 1k ]
tran 1u 6m

meas tran vopp pp v(nd)
let transient_gain=vopp/2m
echo "Transient gain is " $&transient_gain

* subtract the average signal levels to see the small signal:
let insig  = ng - avg(ng)
let outsig = nd - avg(nd)

plot insig outsig
hardcopy plots/nmos_rtl_tran_{$r}.svg insig outsig
```

Interpreting the Amplifier Results

After subtracting the average signal levels, we see the AC variations in the output (drain) compared to the input (gate). The output amplitude is bigger than the input amplitude, so the signal has been amplified.



Record the Amplifier Measurements

In your `analysis.txt` file, edit the sections for each resistor value and record all SPICE measurement results. The maximum `gain` result from the DC simulation should be close to the transient gain. How do they compare?

The measured gain values should also be close to g_mR1 for each case. Using the measurement `maxgm` as the transconductance, calculate g_mR1 for each case and note how closely it matches the other gain measurements.

Summary of Exercises

1. NMOS I/V DC simulation.
2. PMOS I/V DC simulation.
3. NMOS RTL Inverter DC simulation.
4. Inverter signal derivatives, g_m and gain.
5. Transient amplifier simulation and gain measurements.

Turning in Your Work

The preferred way to turn in your work is to use `git`. From the Linux terminal:

```
git add *
git commit -a -m "Submitting SPICE 4 assignment"
git push origin master
```

Alternatively you can upload a ZIP file to Canvas containing all your assignment files.