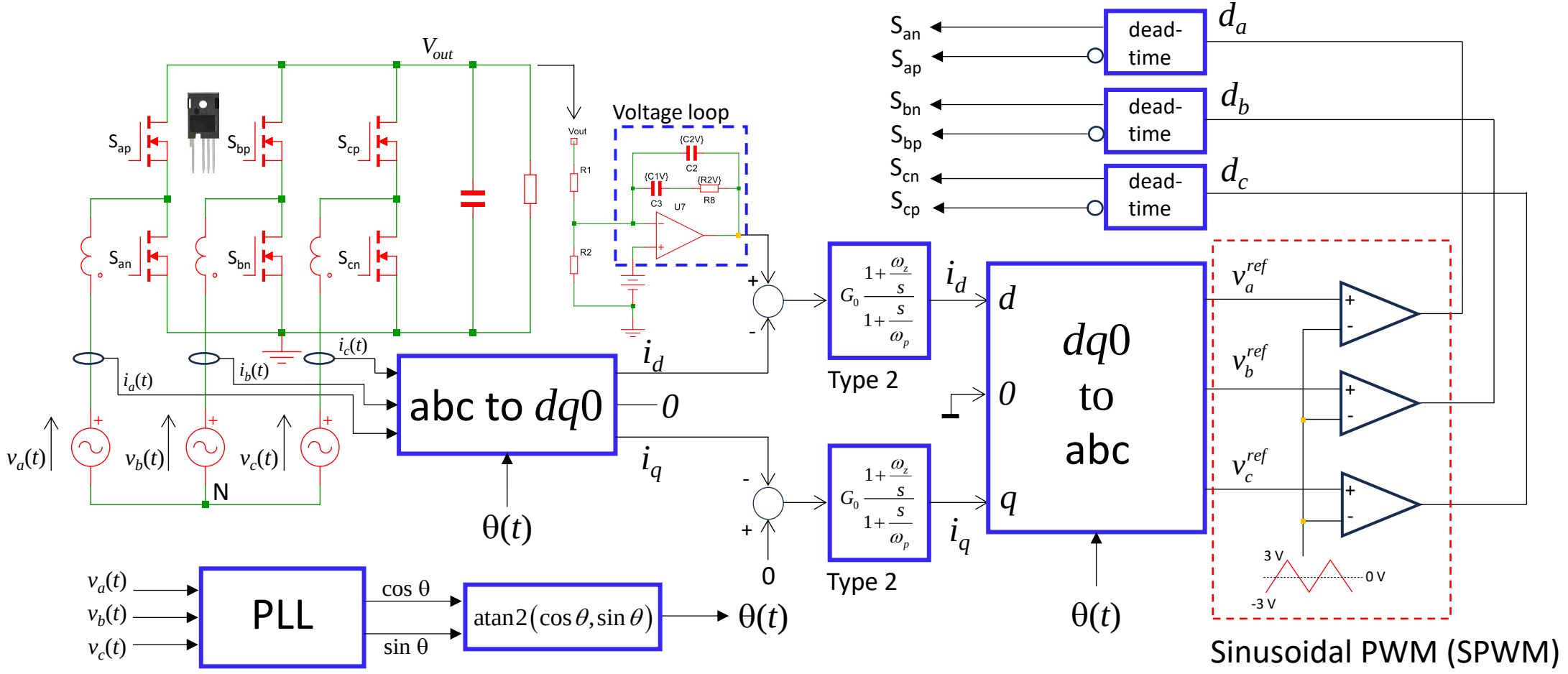
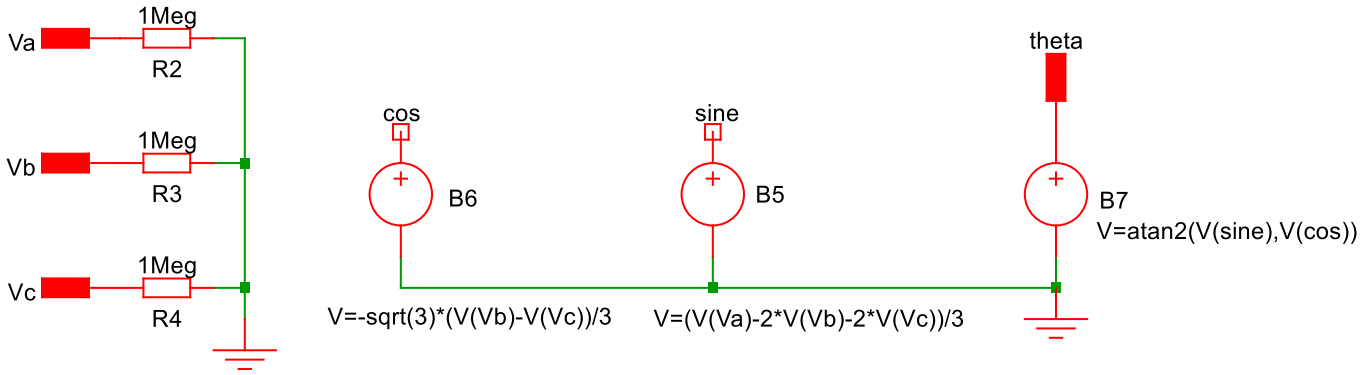
This is a simplified 3-phase PFC implementing dq0 modulation. There is no cross-coupling compensation in this application circuit. A module reconstructs the angle which is going to feed the abc-to-dq0 and dq0-to-abc blocks. The dc values d and q are respectively scaled as a function of P_{out} and kept to 0. The last block reconstructs the current sinewaves further modulated via a 30-kHz triangle carrier.



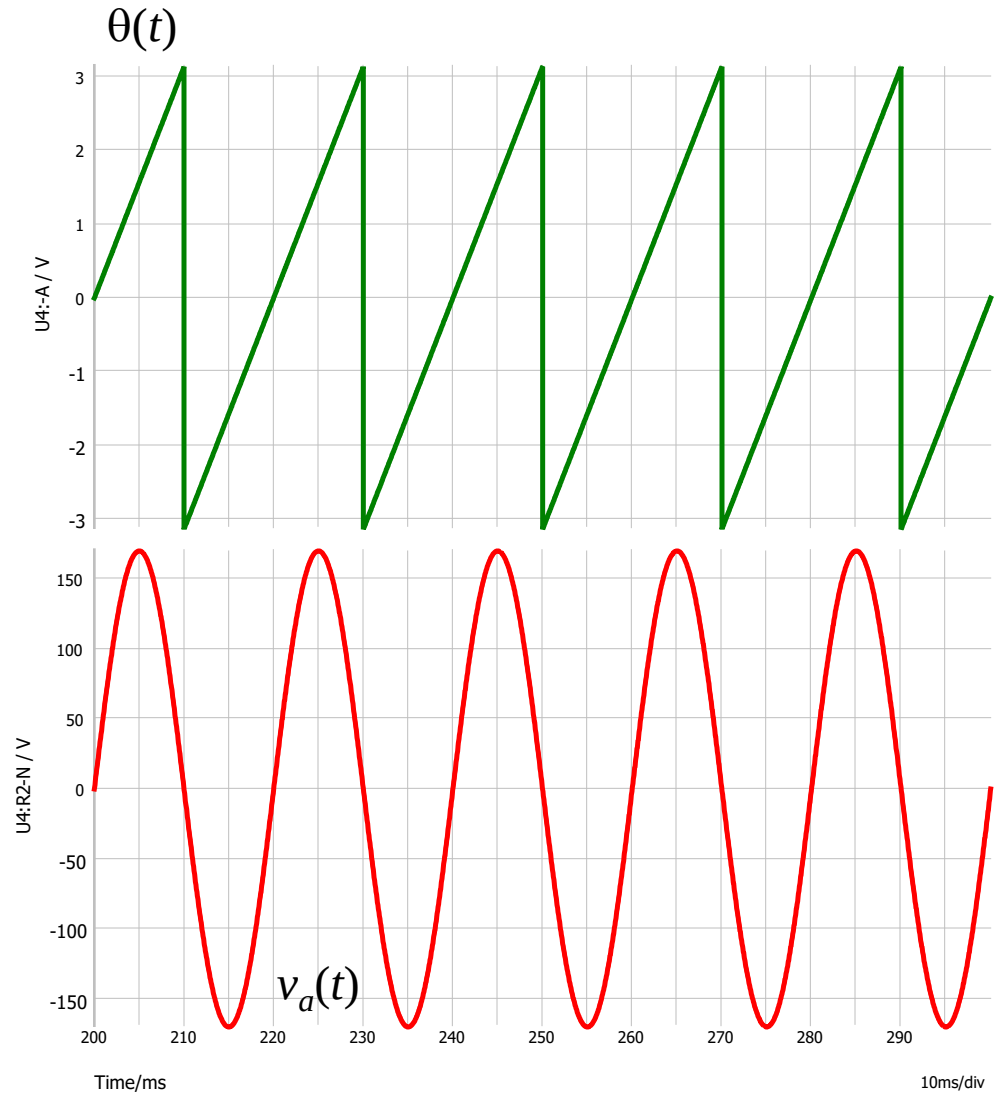
D. Boroyevich, *Modeling and Control of Three-Phase PWM Converters*, PECon 2008, Malaysia

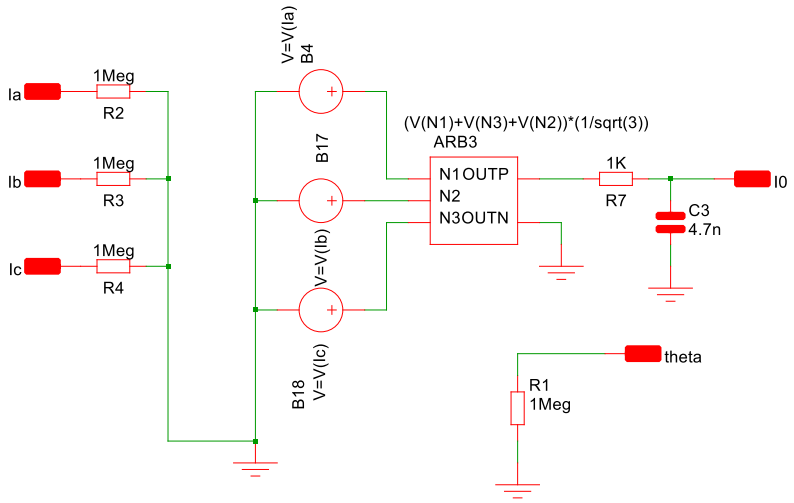
Christophe Basso –January 2025 – rev 0.21

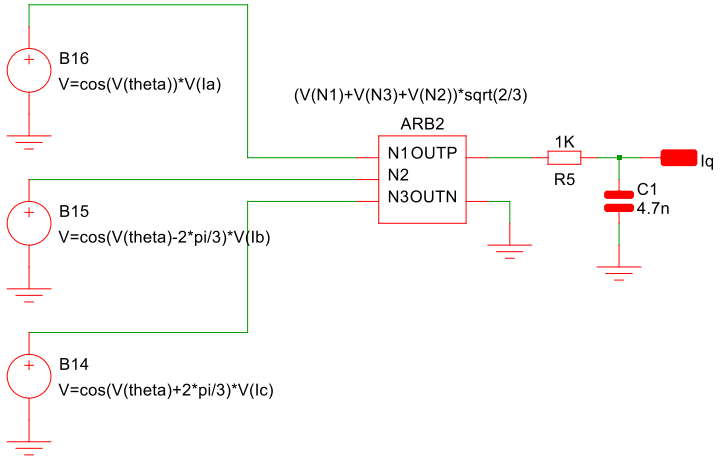
The blocks have first been individually tested and it is a mandatory step before running a complete simulation with all active subcircuits.

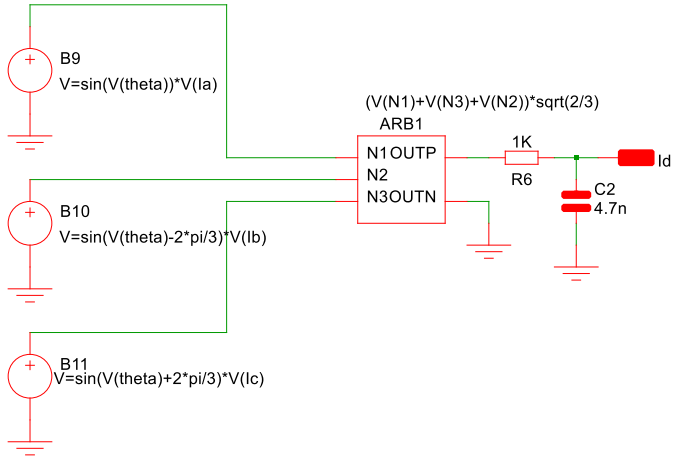


This is the angle generator and it works as expected.



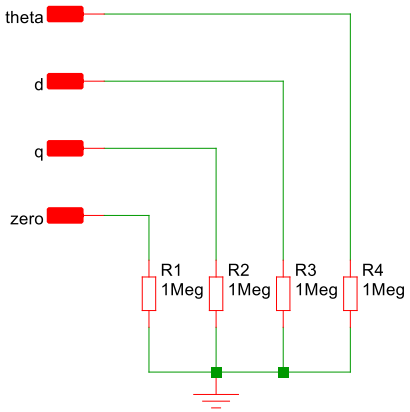


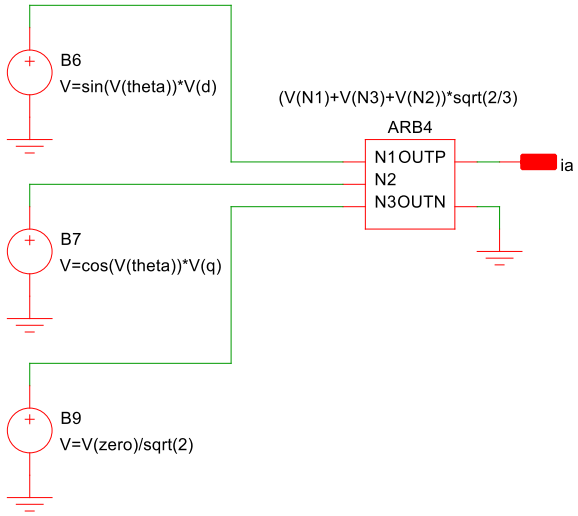


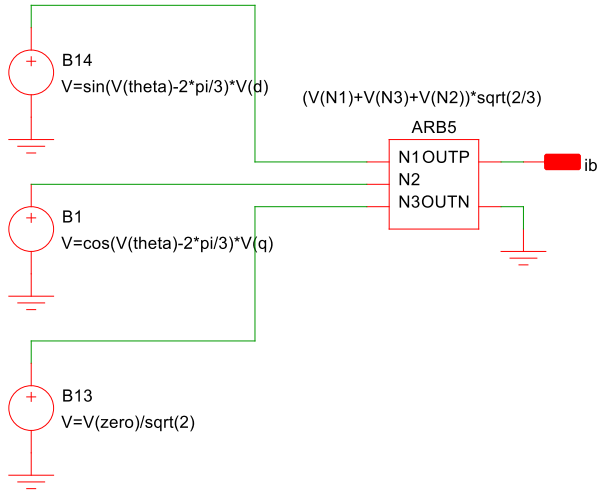


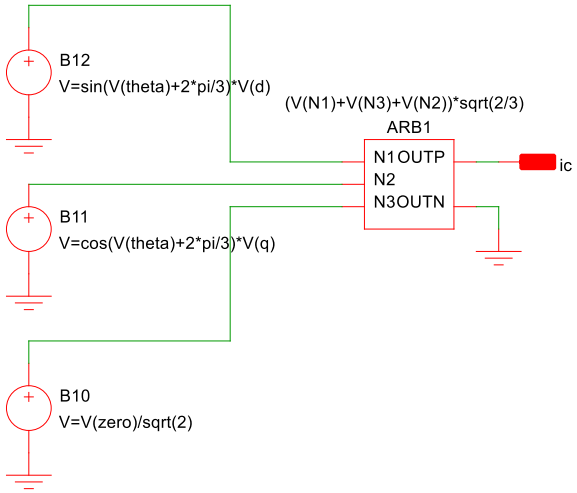
This is the abc-to-dq0 transform block which takes the input currents and transforms them into dc quantities, d, q plus the zero-sequence component I_0.

This is the dq0-to-abc transform circuitry which builds the currents setpoints sent to the modulators.

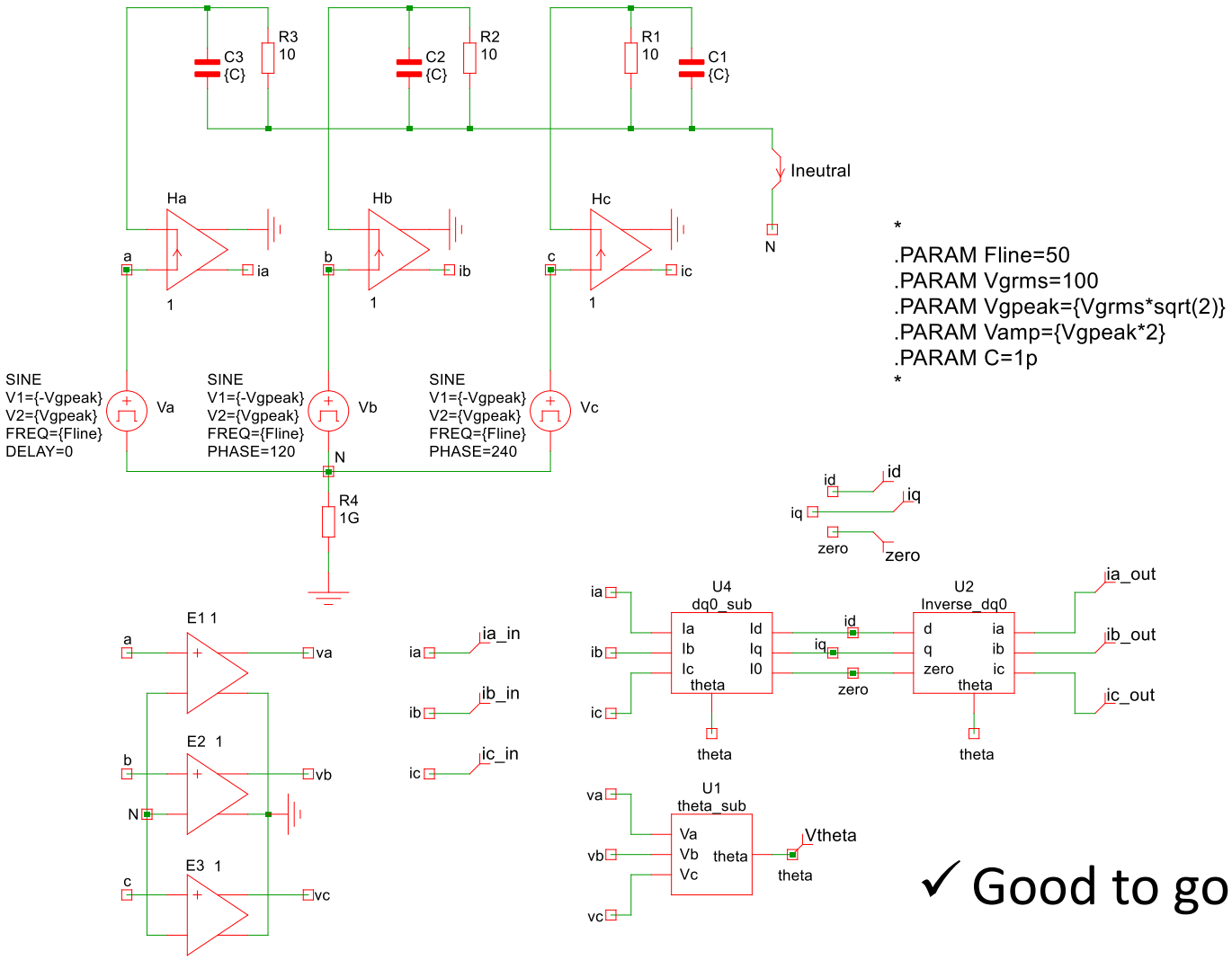






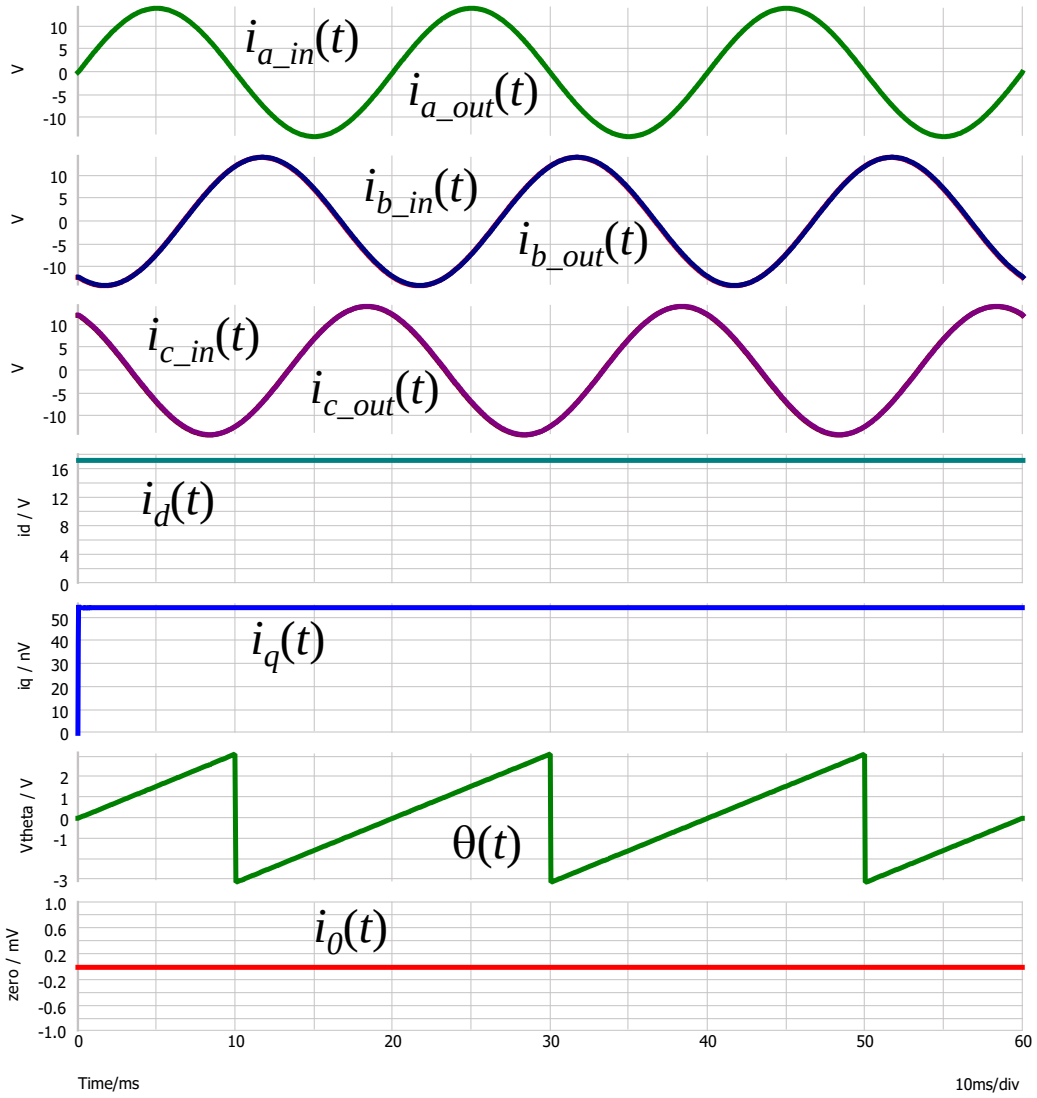


A small setup then helps verify that all calculations are ok. I have superimposed the input currents with their reconstructed versions via subcircuit U2. They are perfectly identical as well as the d value which confirms the active power value.

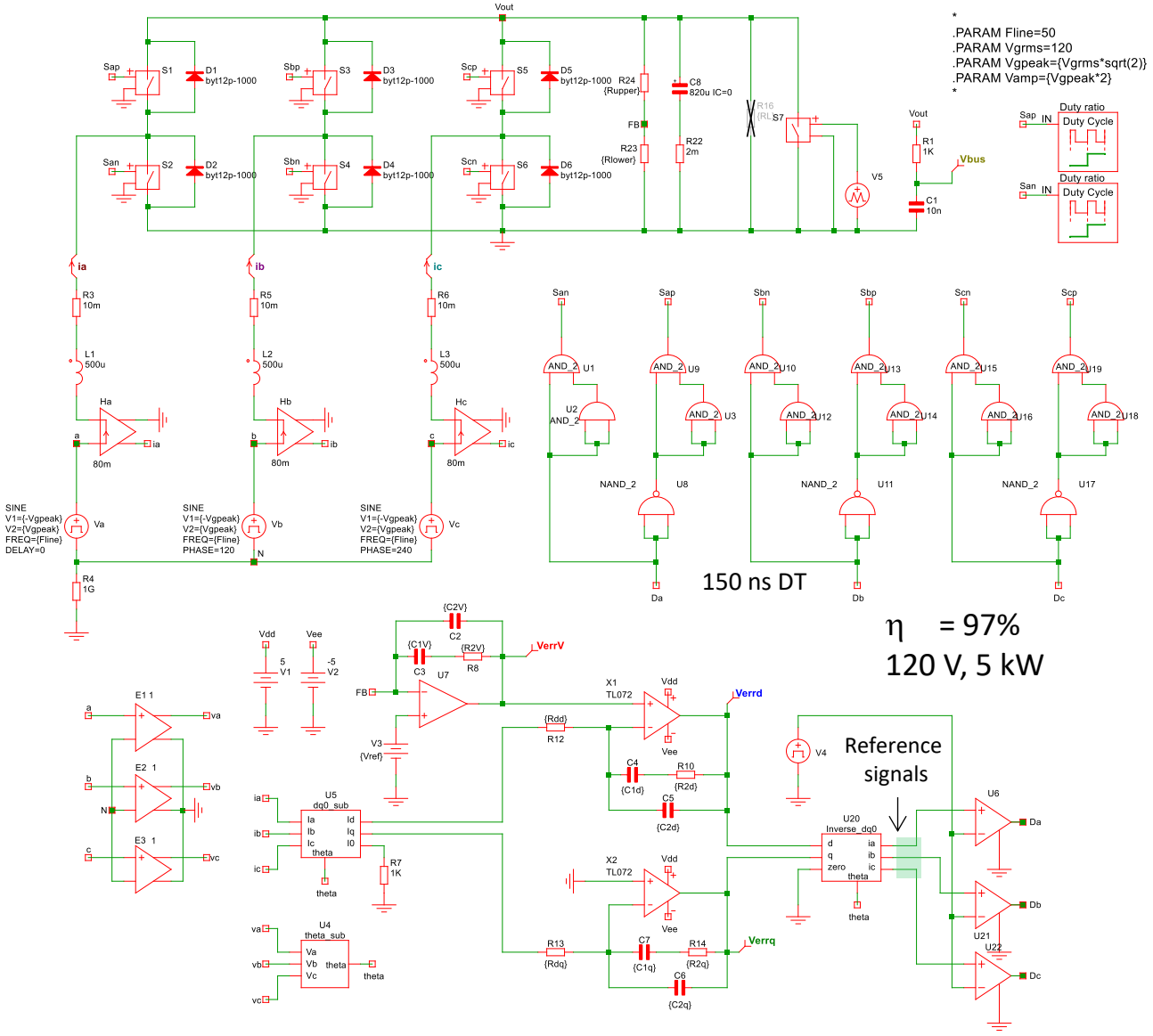



```
*
.PARAM Fline=50
.PARAM Vgrms=100
.PARAM Vgpeak={Vgrms*sqrt(2)}
.PARAM Vamp={Vgpeak*2}
.PARAM C=1p
*
```

✓ Good to go!



This is the complete simulation setup. I have added macros to automate the type 2 compensators. PFCs usually have an excess of gain at low frequencies so I put 20 and shot for a phase margin of 60° at 20-Hz crossover. The d and q loops must be faster. I arbitrarily set the d and q plant gains to 1 and asked for a 2-kHz crossover. I stabilized the loops in the next slides.



This first attempt is conclusive and the converter is stable, delivering the wanted power level with a THD below 5% at 5 kW and below 2.5% at 10 kW:

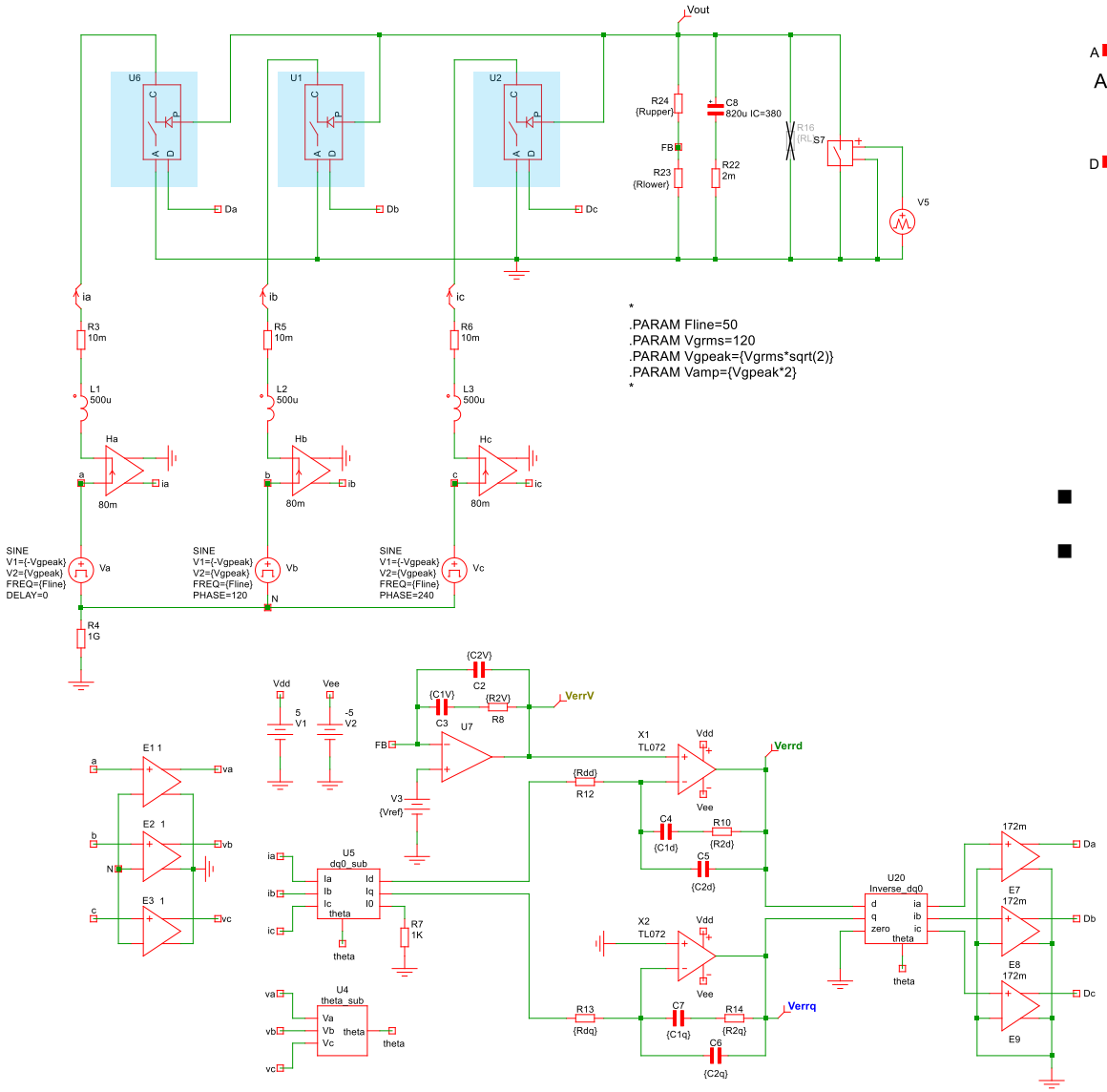

```
.param GfcV=20 ; magnitude at crossover *
.param PSV=-90 ; phase lag at crossover *
*
* Enter Design Goals Information Here *
*
.param fcV=20 ; targeted crossover *
.param PMV=60 ; choose phase margin at crossover *
*
* Enter the Values for Vout and Bridge Bias Current *
*
.param Vout=400
.param Pout=5k
.param RL={(Vout)^2/(Pout)}
.param Ibias=100u
.param Vref=2.5
.param Rlower={Vref/Ibias}
.param Rupper={(Vout-Vref)/Ibias}
```

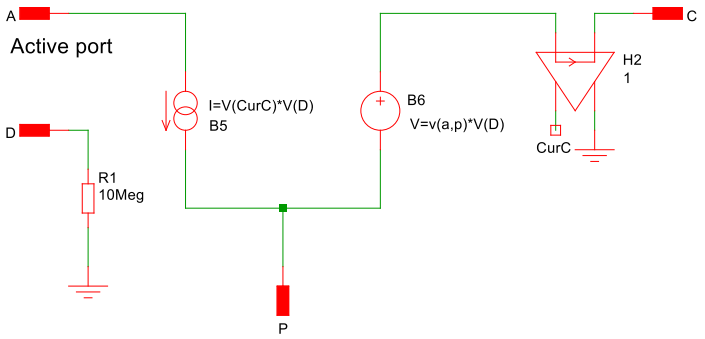
```
*
* Do not edit the below lines *
.param boostV={PMd-PSd-90}
.param GV={10^(-GfcV/20)}
.param kV={tan((boostV/2+45)*pi/180)}
.param fpV={fcV*kV}
.param fzV={fcV/kV}
.param
C2V={1/(2*pi*fcV*GV*kV*Rupper)}
.param C1V={C2V*(kV^2-1)}
.param R2V={kV/(C1V*2*pi*fcV)}
```

Type 2 voltage loop compensation

Simulation time for 200 ms: 5 mn

Now, to rigorously compensate these loops, you can either a) determine the small-signal response of this entire circuit - "good luck", as Marko from Tropoja would say ^_^ b) measure it in the lab with a working prototype or c), resort to an averaged model as I did below:



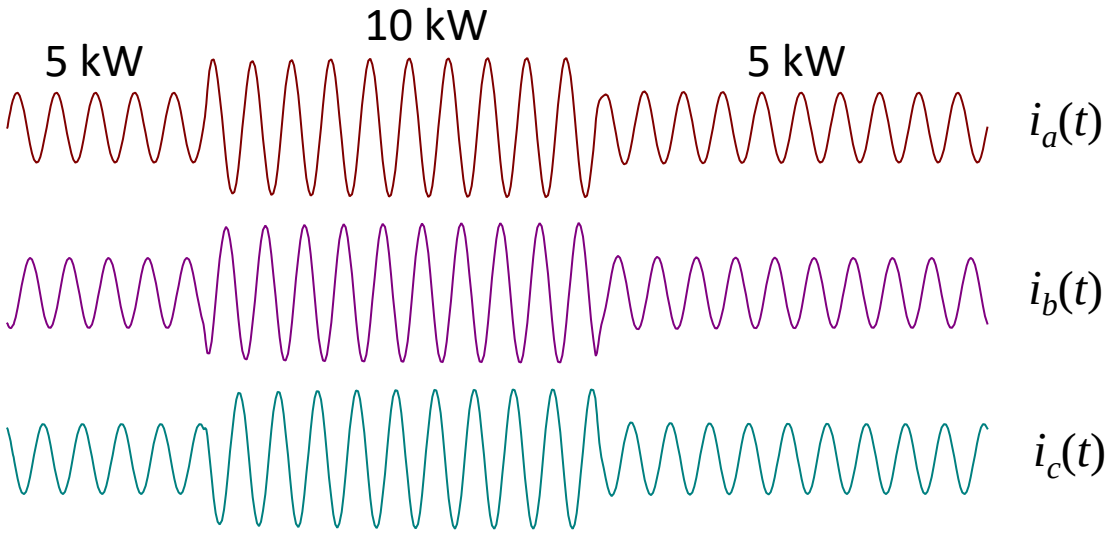


Simplified Analysis of PWM Converters Using Model of PWM Switch Part I: Continuous Conduction Mode

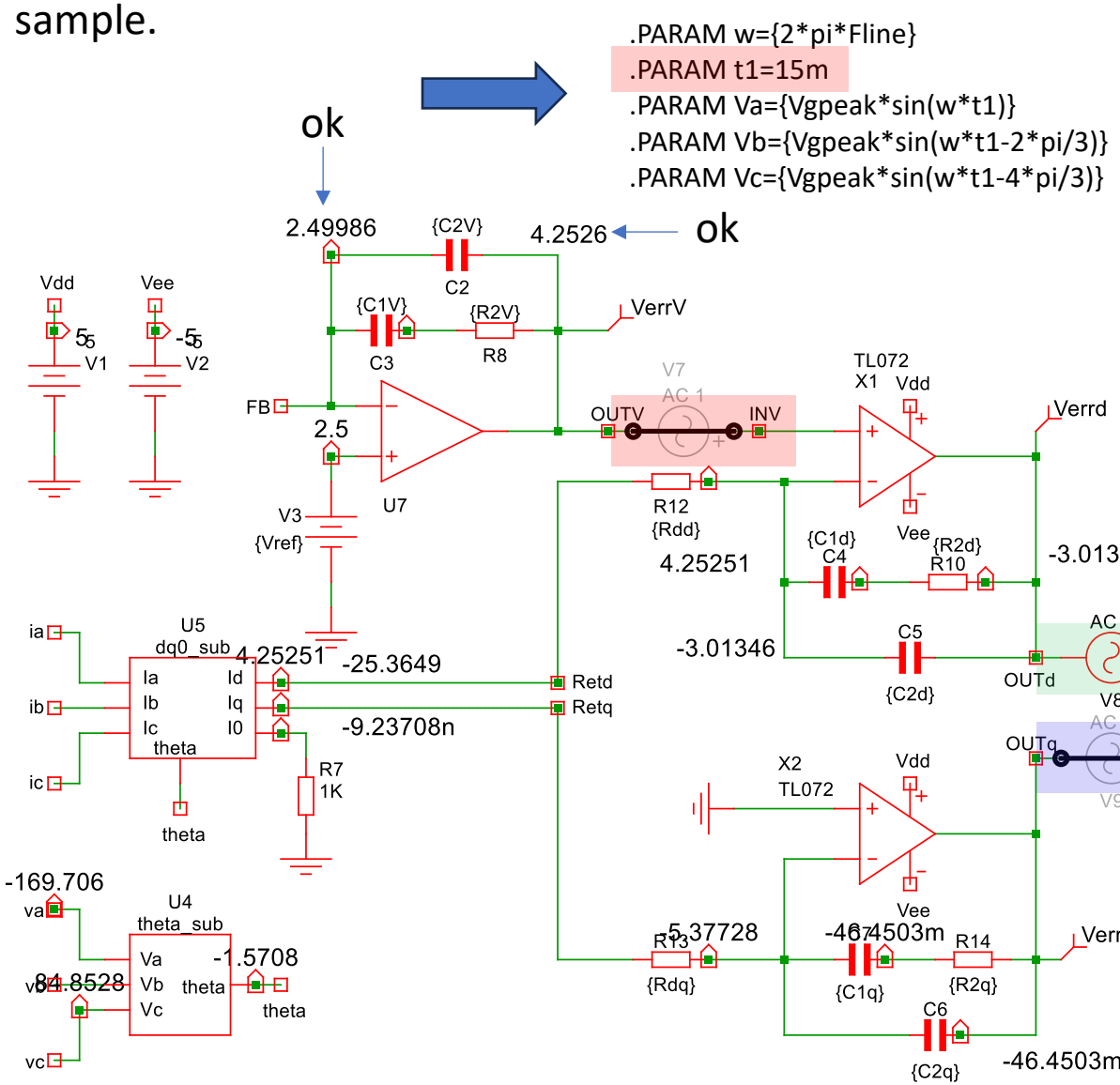
VATCHÉ VORPÉRIAN
Virginia Polytechnic Institute and State University

V. Vorpérian, *Simplified analysis of PWM converters using model of PWM switch. Continuous conduction mode*, in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 3,, May 1990

- I have replaced the switching cell by the VM PWM switch model
- Pulse-width modulators are replaced by small-signal gains

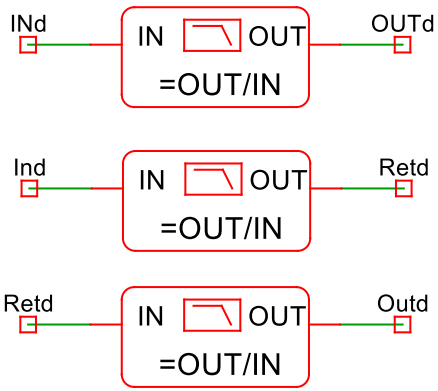


It then becomes possible to ac-sweep each loop individually while the others perform their duty. In the below illustration, the d loop is actively swept while the two others are still closed but ac-silent. One important thing here, you have to replace the power input sources by dc values. I have arbitrarily set these sources to the dc level corresponding to a 15 ms time sample.

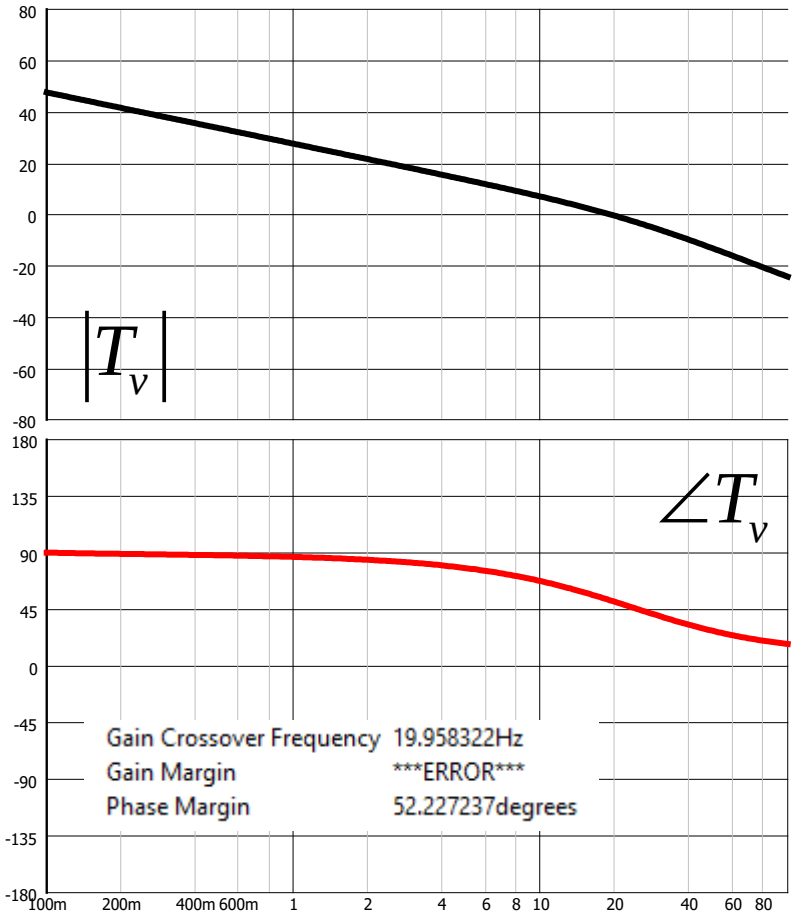


Given the complexity of the setup, convergence can be at stake and you must verify that the dc operating points are correct, e.g. V_{out} regulates and the error amps are not saturated. You may need to try different timing values for t_1 (15 ms and 35 ms worked for me)

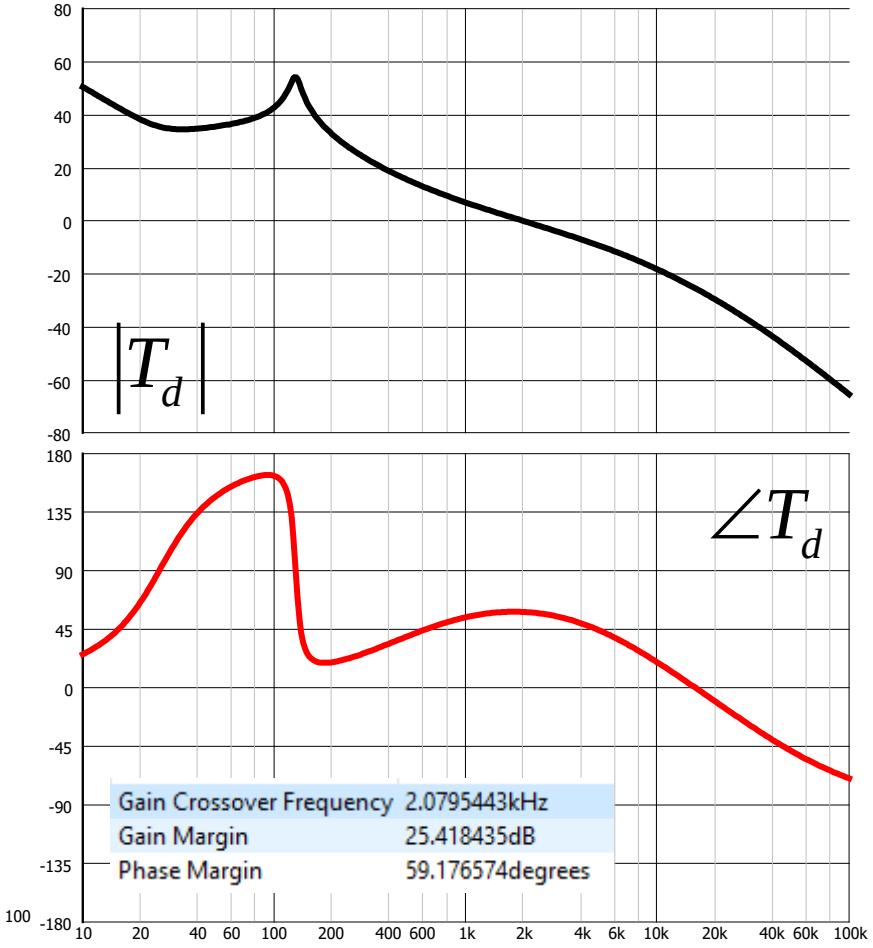
- ✓ One ac source is active at a time
- ✓ First the slow V-loop is compensated
- ✓ Then the d and q loops



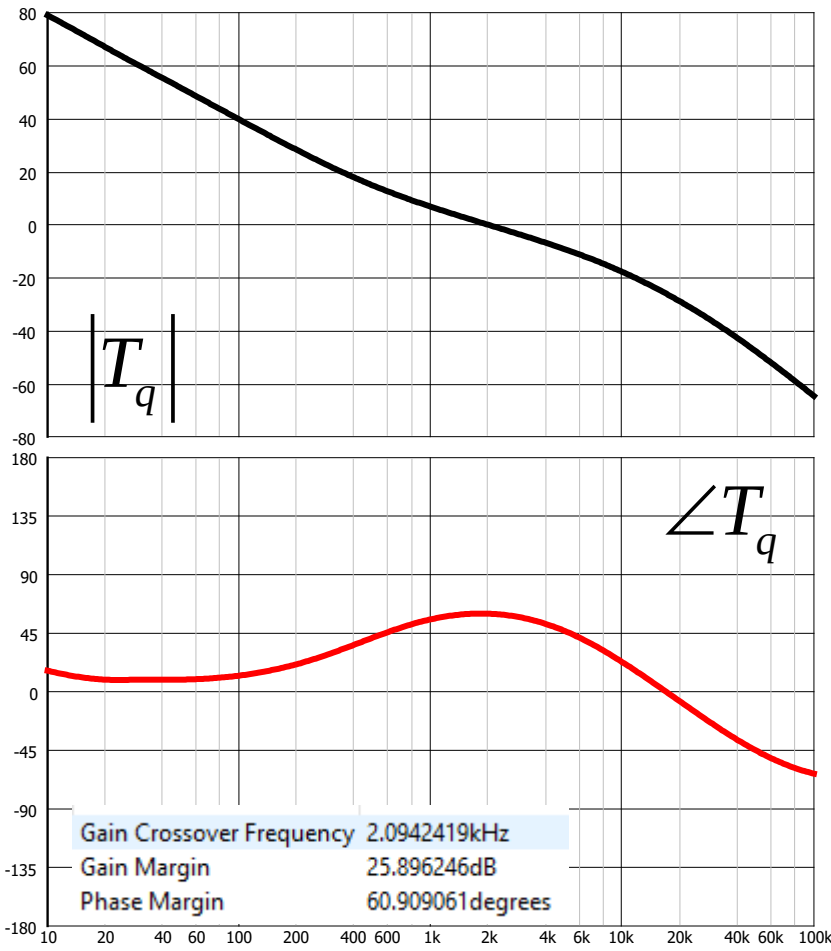
With the averaged model, I can now extract the control-to-output transfer functions of each loop and feed the macro to automate the compensators components values calculations. Once done, I obtain the wanted crossover frequencies with a decent phase margin. The d and q loops are compensated for a 2-kHz crossover in this example.



Voltage loop – f_c = 20 Hz

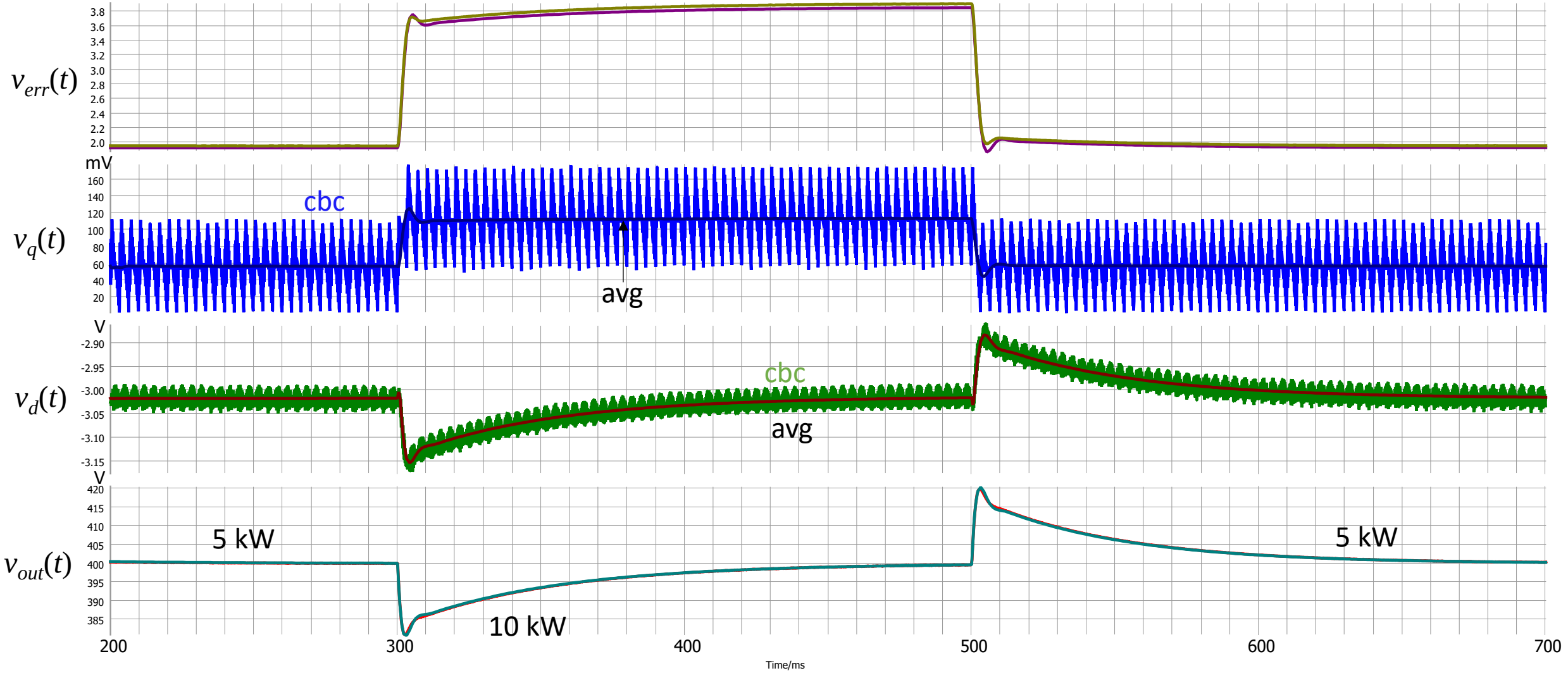


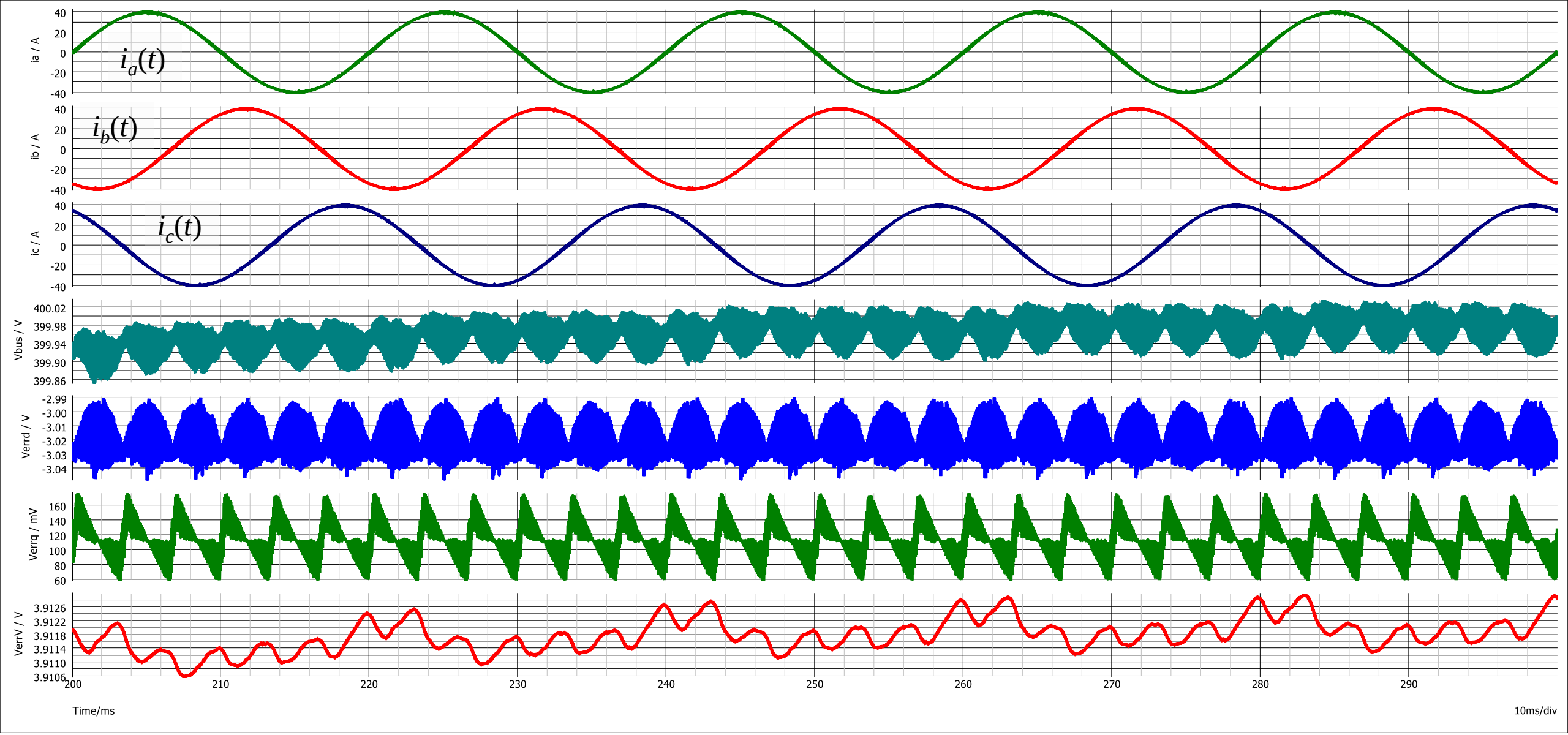
d loop – f_c = 2 kHz



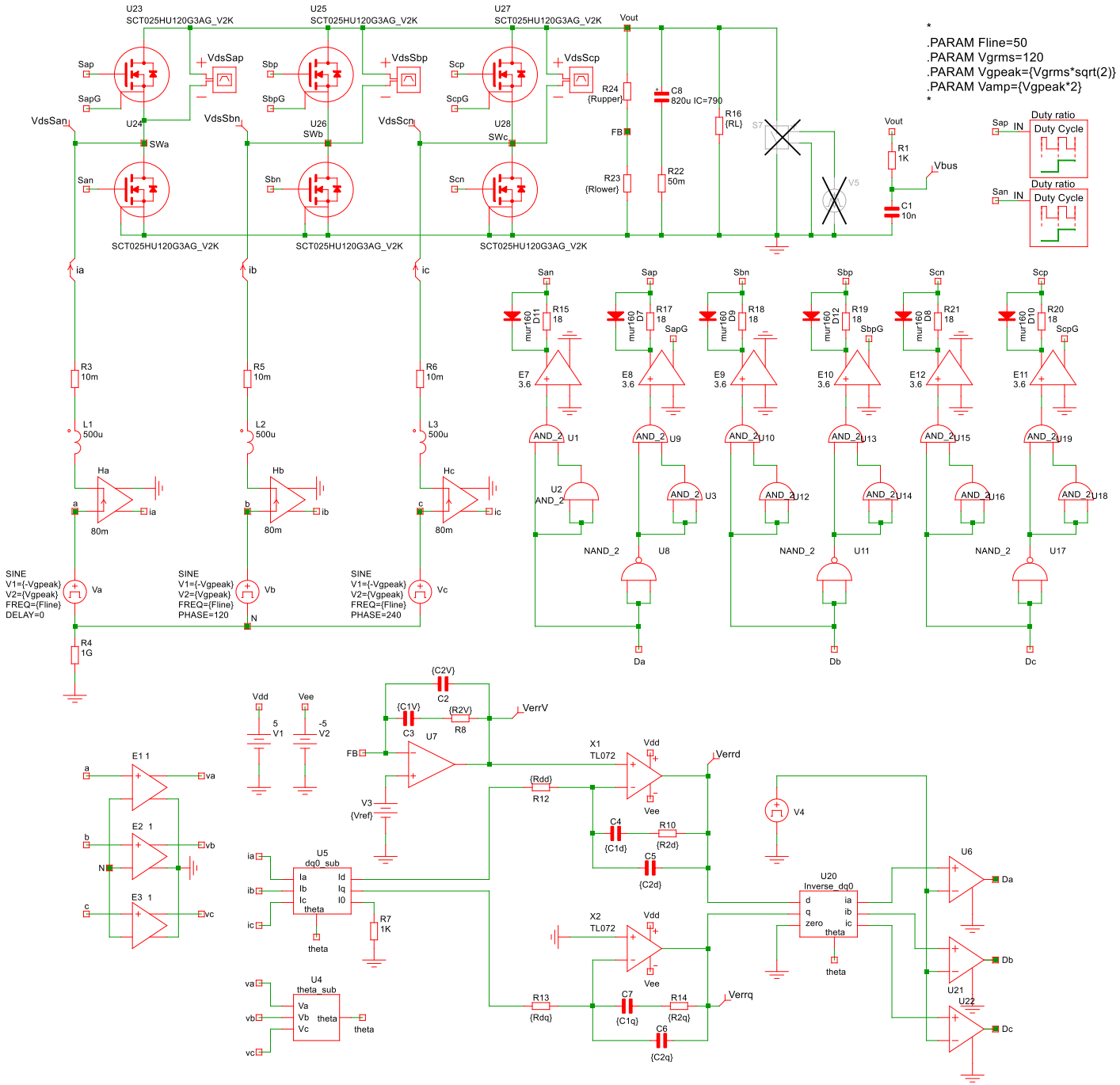
q loop – f_c = 2 kHz

This the real test where you check the averaged model response and the cycle-by-cycle version. They are compensated with the exact same components and show almost no difference during a transient load step, naturally validating the large-signal averaged model of this 3-phase PFC.





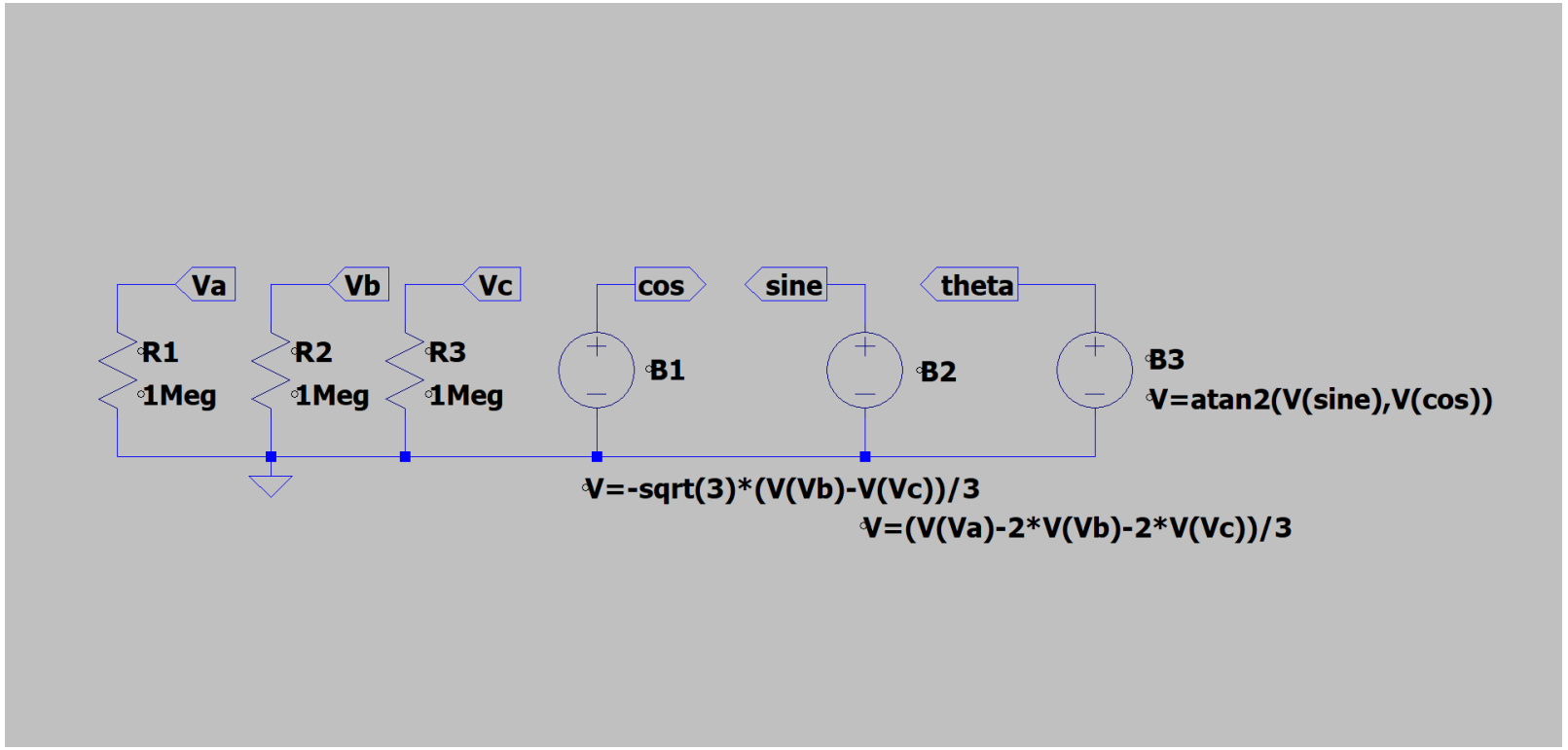
The steady-state simulation shows nice input currents with a rail regulated at 400 V dc.



You now have the same 3-phase 10-kW PFC equipped with SiC transistors. The drivers have been adapted for providing a drive voltage from 0 to 18 V. I purposely kept a simplified version of this driver stage but it obviously affects the switching characteristics of these components so a more thorough modeling is necessary here. Keep in mind that the applied bias, a turn off, modifies the dynamic resistance offered by the transistor when conducting in the third quadrant. The power distribution network (PDN) around each transistor shall also be thoroughly modeled but that is another story :)

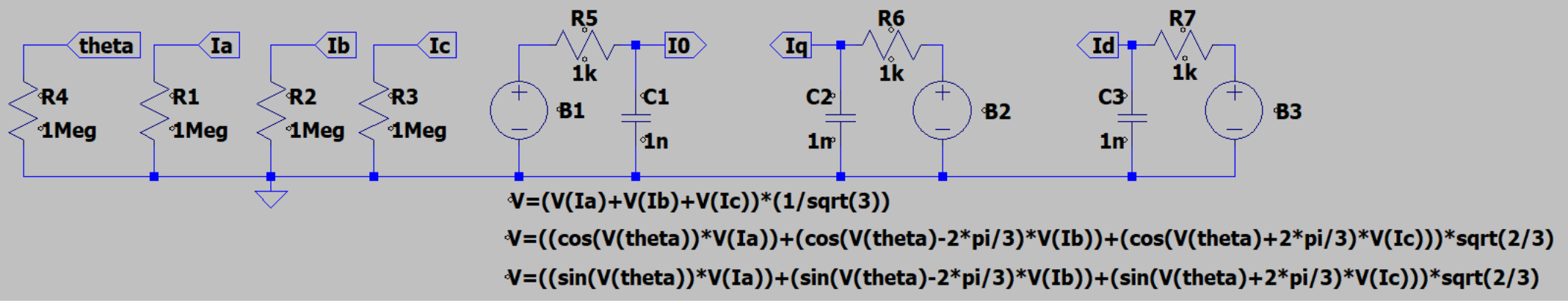
The simulation time is quite long here and approaches an hour, which is still reasonable. The setup with the perfect switches simulates in 10 mn while the averaged model delivers results in a few tens of seconds.

This 3-phase PFC has successfully been simulated in SIMetrix but I wanted a version running on LTspice also, considering the popularity of the program. I thus did duplicate and individually test all the subcircuits already running in SIMetrix. We start with the angle calculation:

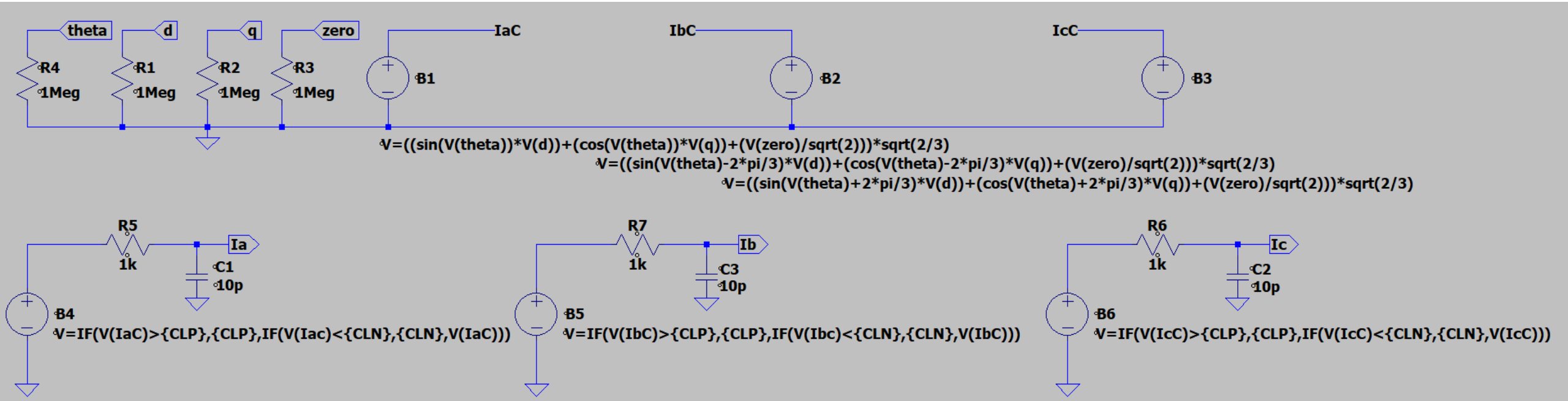


This circuit is a classic and it reconstructs the angle based on the three input voltages referenced to ground.

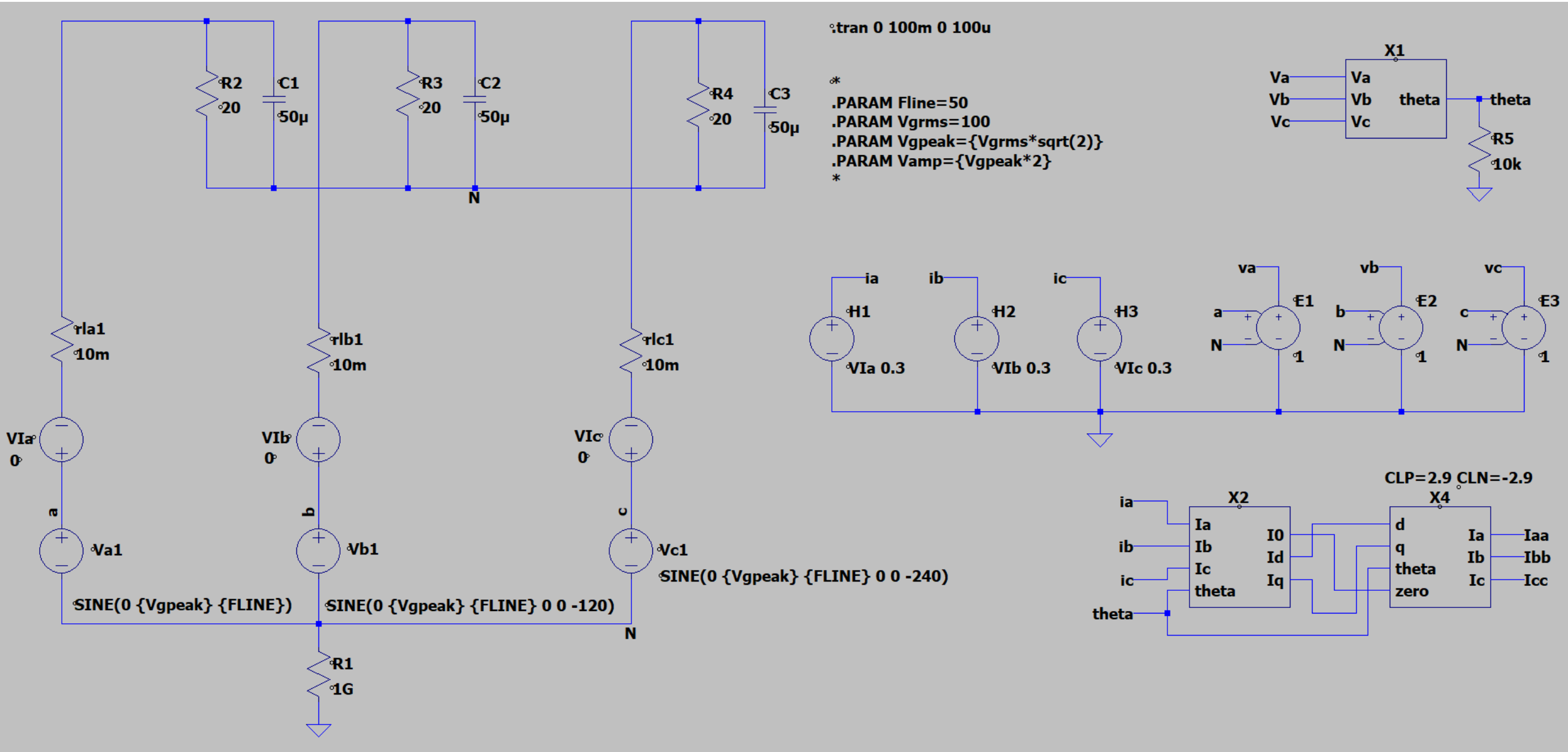
The abc-to-dq0 block is simple to model with behavioral voltage sources. Output filters are added for noise filtering purposes. The pole is located at 160 kHz, not a problem for a 2-kHz crossover. It can be easily altered if necessary.

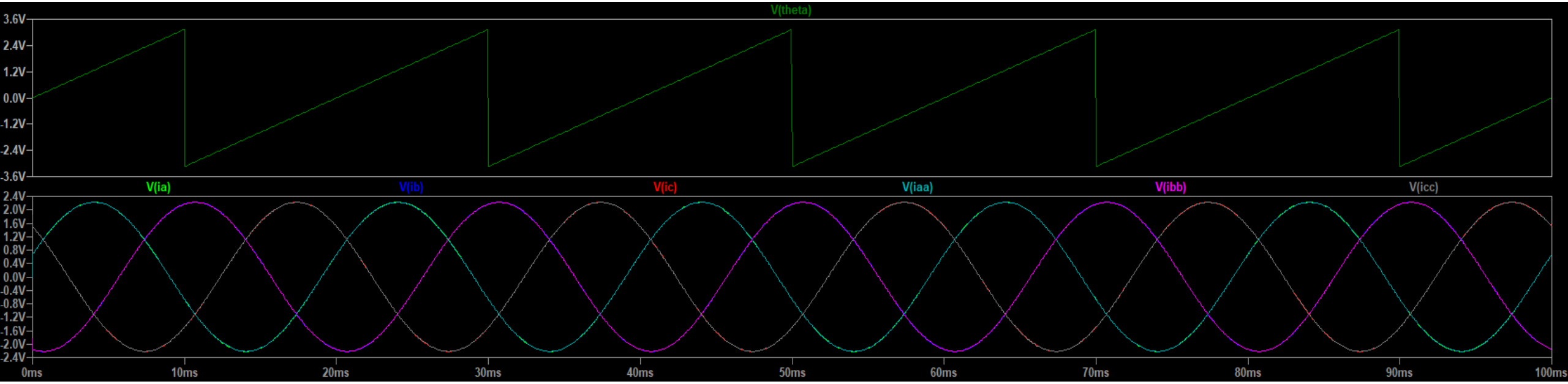


The dq0-to-abc required a bit more care and I have added output clamps, mainly for the debugging sessions. I used three if-then-else statements with parameters passing. Considering a 6-V peak-to-peak triangle carrier, each output is clamped between -2.9 V and 2.9 V.



The validation of the whole set is made by loading the 3-phase network and injecting the input currents into the first subcircuit X2 which computes d and q. The resulting values are feeding the inverse-transform block and I compared the original currents versus the reconstructed ones.

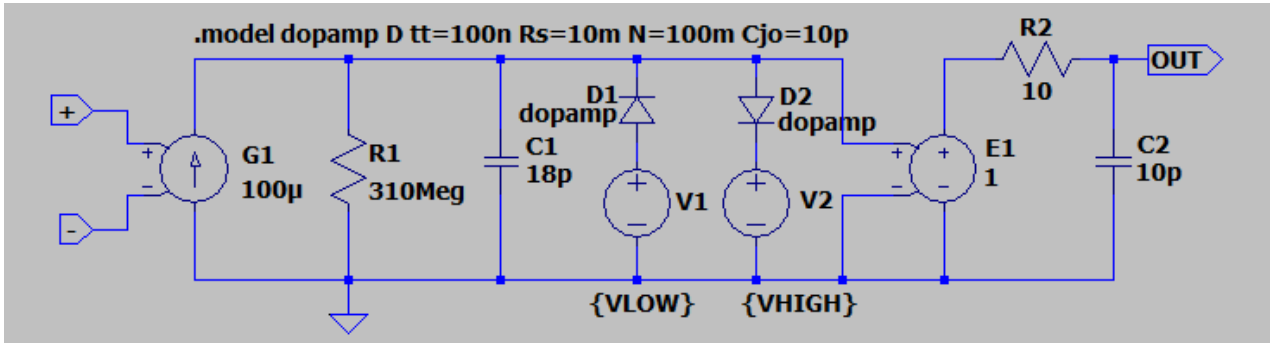


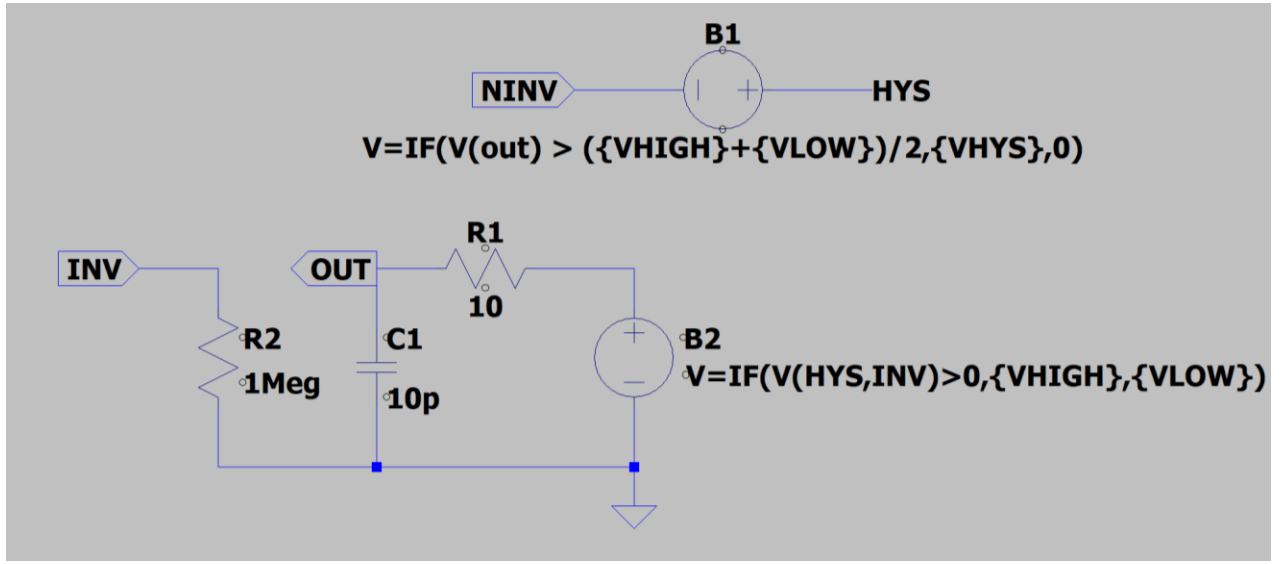


The angle calculation is shown in the top, the three input and output currents are perfectly superimposed in the bottom. The blocks are good to go! I also built other basic subs to really simplify the simulation setup.

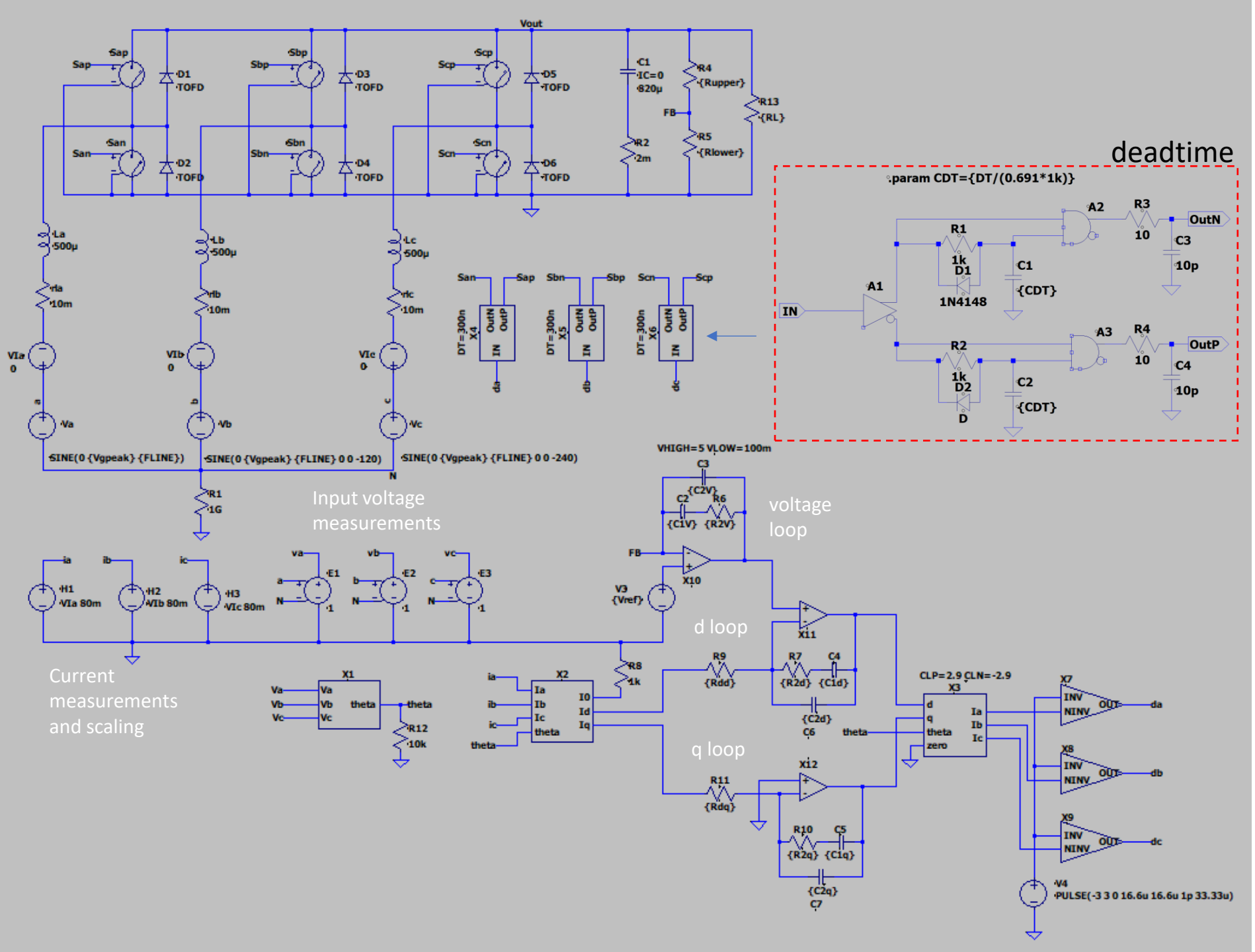
Please note that the above curves were obtained with angles v_a, v_b and v_c respectively of 0°, -120° and -240°.

A simple op-amp with clamps:





A simple comparator with hysteresis:



This is the complete setup of the 3-phase PFC in LTspice. The output voltage is 400 V and the input sources are 120 V rms. The switches are simple voltage-controlled types but the diode gave me hard times to find a suitable type, leading to acceptable switching transitions (read less noisy).

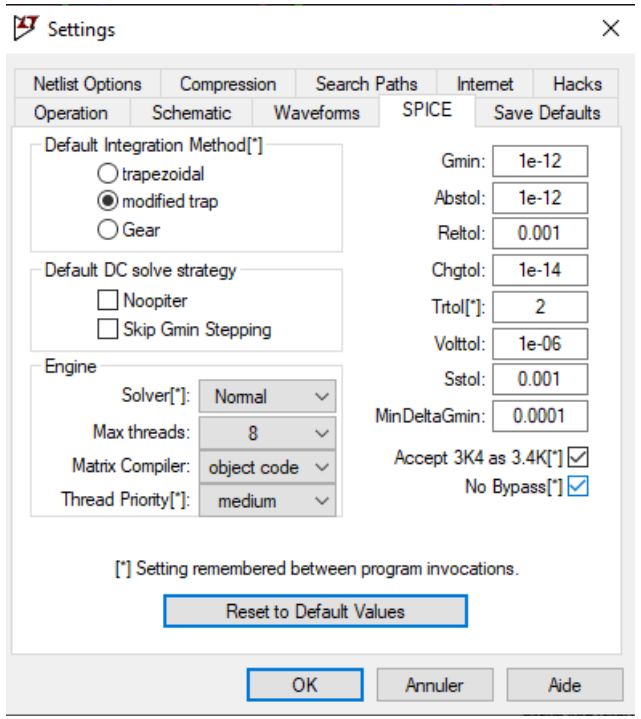
```
.param GfcV=20 ; magnitude at crossover *
.param PSV=-90 ; phase lag at crossover *
*
* Enter Design Goals Information Here *
*
.param fcV=20 ; targeted crossover *
.param PMV=60 ; choose phase margin at crossover *
*
* Enter the Values for Vout and Bridge Bias Current *
*
.param Vout=400
.param Pout=10k ; nominal is 5 kW
.param RL={(Vout)**2/(Pout)}
.param Ibias=100u
.param Vref=2.5
.param Rlower={Vref/Ibias}
.param Rupper={(Vout-Vref)/Ibias}
*
* Do not edit the below lines *
.param boostV={PMV-PSV-90}
.param GV={10**(-GfcV/20)}
.param kV={tan((boostV/2+45)*pi/180)}
.param fpV={fcV*kV}
.param fzV={fcV/kV}
.param C2V={1/(2*pi*fcV*GV*kV*Rupper)}
.param C1V={C2V*(kV**2-1)}
.param R2V={kV/(C1V*2*pi*fcV)}
*
```

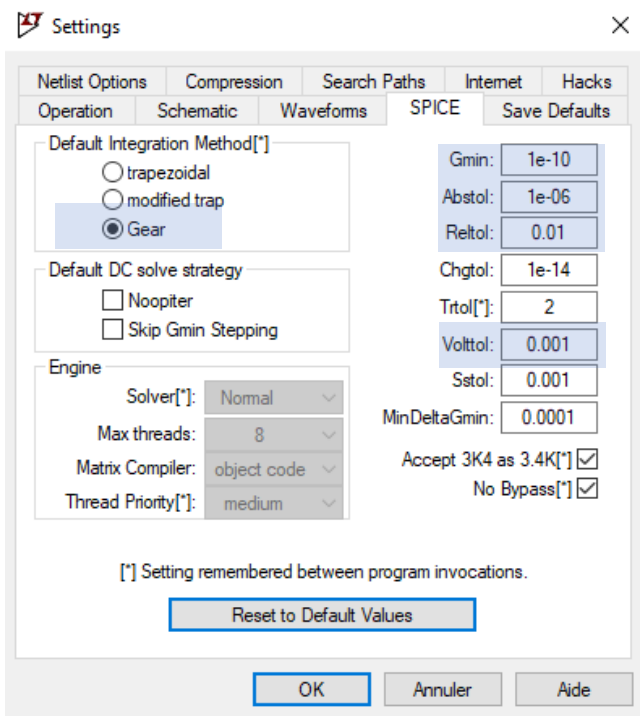
```
* Components for the d loop *
.param Gfcd=0 ; magnitude at crossover *
.param PSd=-100 ; phase lag at crossover *
*
* Enter Design Goals Information Here *
*
.param fcd=2k ; targeted crossover *
.param PMd=60 ; choose phase margin at crossover *
*
* Enter the Values for Vout and Bridge Bias Current *
*
.param Rdd=100k
*
* Do not edit the below lines *
.param boostd={PMd-PSd-90}
.param Gd={10**(-Gfcd/20)}
.param kd={tan((boostd/2+45)*pi/180)}
.param fpd={fcd*kd}
.param fzd={fcd/kd}
.param C2d={1/(2*pi*fcd*Gd*kd*Rdd)}
.param C1d={C2d*(kd**2-1)}
.param R2d={kd/(C1d*2*pi*fcd)}
*
```

```
* Components for the q loop *
.param Gfcq=0 ; magnitude at crossover *
.param PSq=-100 ; phase lag at crossover *
*
* Enter Design Goals Information Here *
*
.param fcq=2k ; targeted crossover *
.param PMq=60 ; choose phase margin at crossover *
*
.param Rdq=100k
*
* Do not edit the below lines *
.param boostq={PMq-PSq-90}
.param Gq={10**(-Gfcq/20)}
.param kq={tan((boostq/2+45)*pi/180)}
.param fpq={fcq*kq}
.param fzq={fcq/kq}
.param C2q={1/(2*pi*fcq*Gq*kq*Rdq)}
.param C1q={C2d*(kq**2-1)}
.param R2q={kq/(C1q*2*pi*fcq)}
*
```

These are the automated macros for calculating the compensators components values based on data extracted from the plant transfer functions like GfcV and PSV for instance. Some adjustments were necessary as the LTspice syntax differs slightly from SIMetrix's.

To speed up the simulation in LTspice, I usually relax some parameters in the control window:



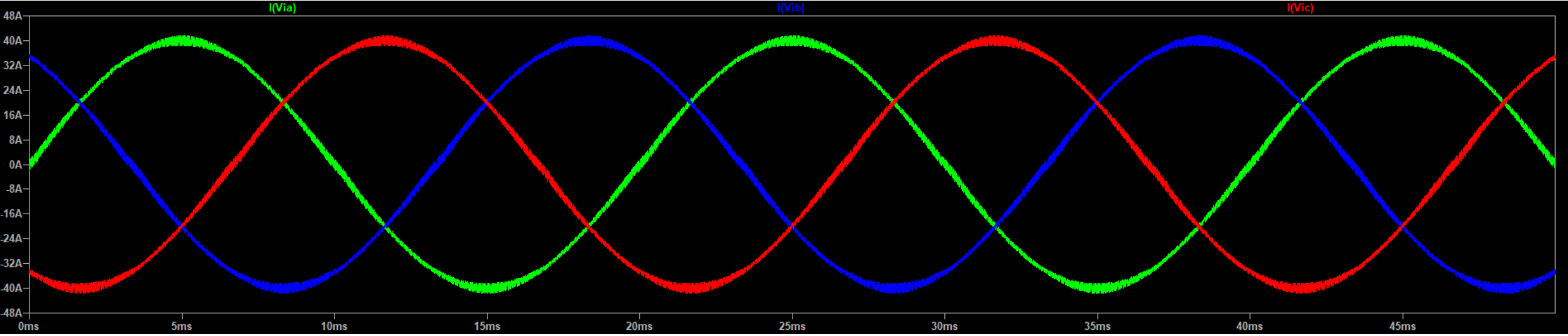


The integration is set to gear which usually brings more damped oscillations than trapezoidal (see this article from Mike Engelhardt). The absolute tolerances for currents and voltages are increased to 1 µA and 1 mV while the relative tolerance, reltol, is set to 1%. Finally, to improve convergence, the g_{min} conductance is set to 100 pS.

The simulation time for a 200-ms run with the default values is 417 s
With the modified setup it is 191 s

For a reason that I ignore, the settings are not saved in the .asc and are lost when you close LTspice.
For this reason, it is best to save them in a .options instruction placed in the schematic:

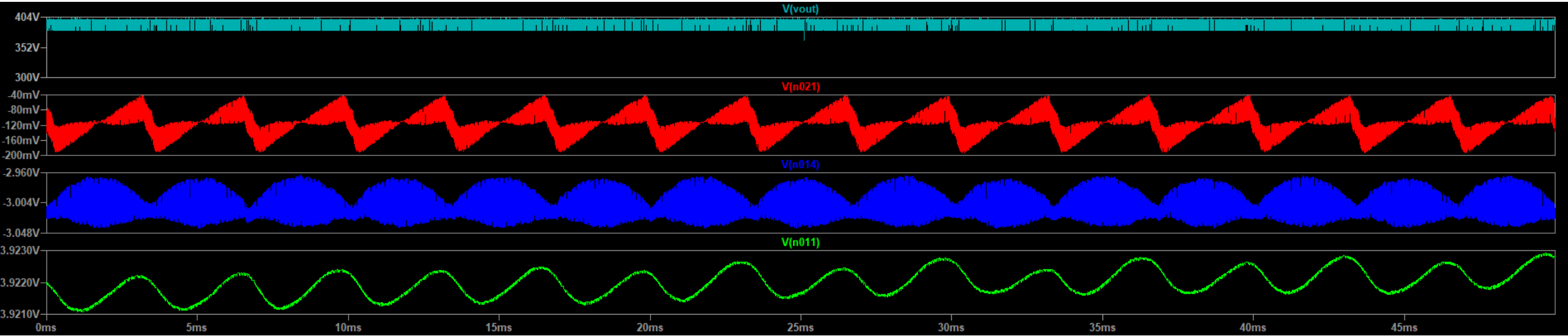
```
.options abstol=1u vntol=1m reltol=0.01
+gmin=100p method=gear
```



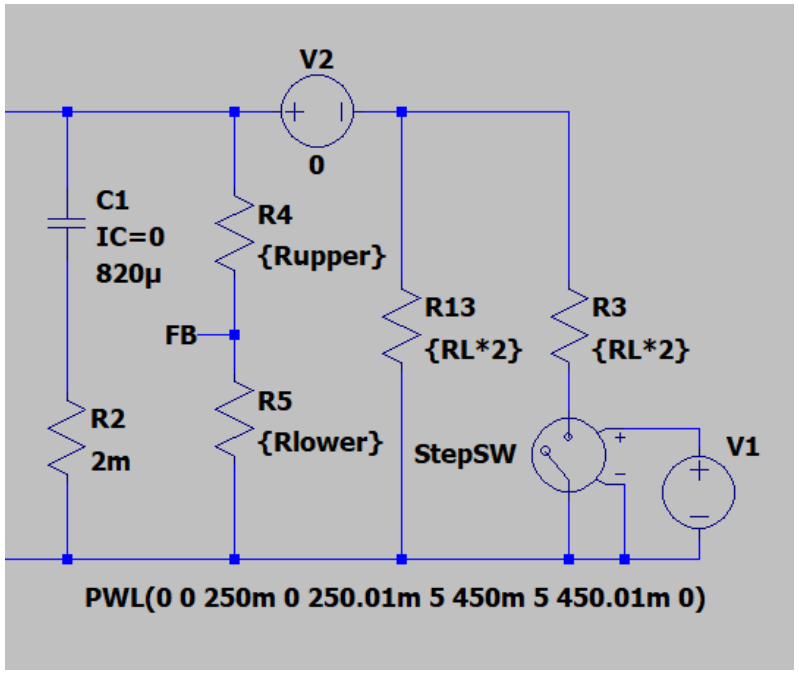
The THD measured after a longer run, is around 2%.

```
Total Harmonic Distortion:   2.230186%
```

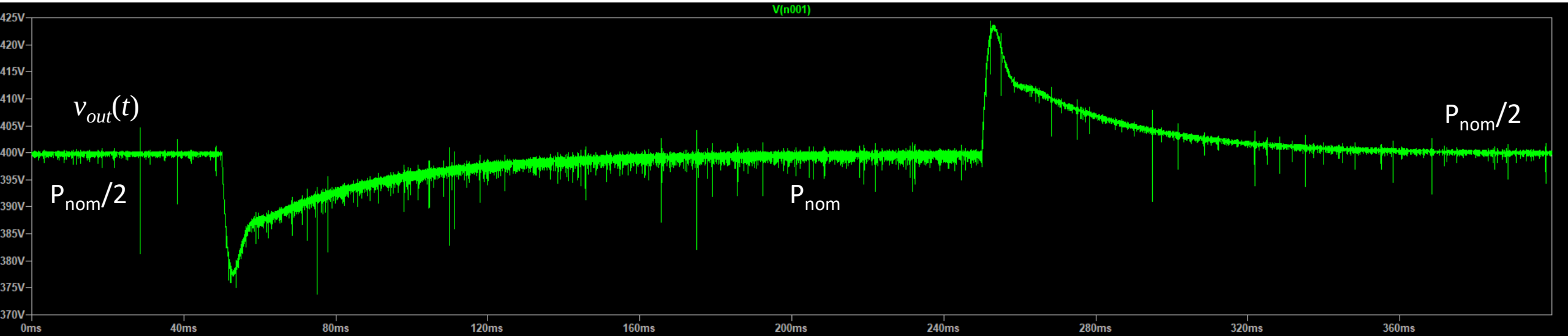
You can see below the output voltage with quite some switching noise and the computed values for the q component (almost 0) and the d value. The error amplifier for the voltage loop is slightly below 4 V.



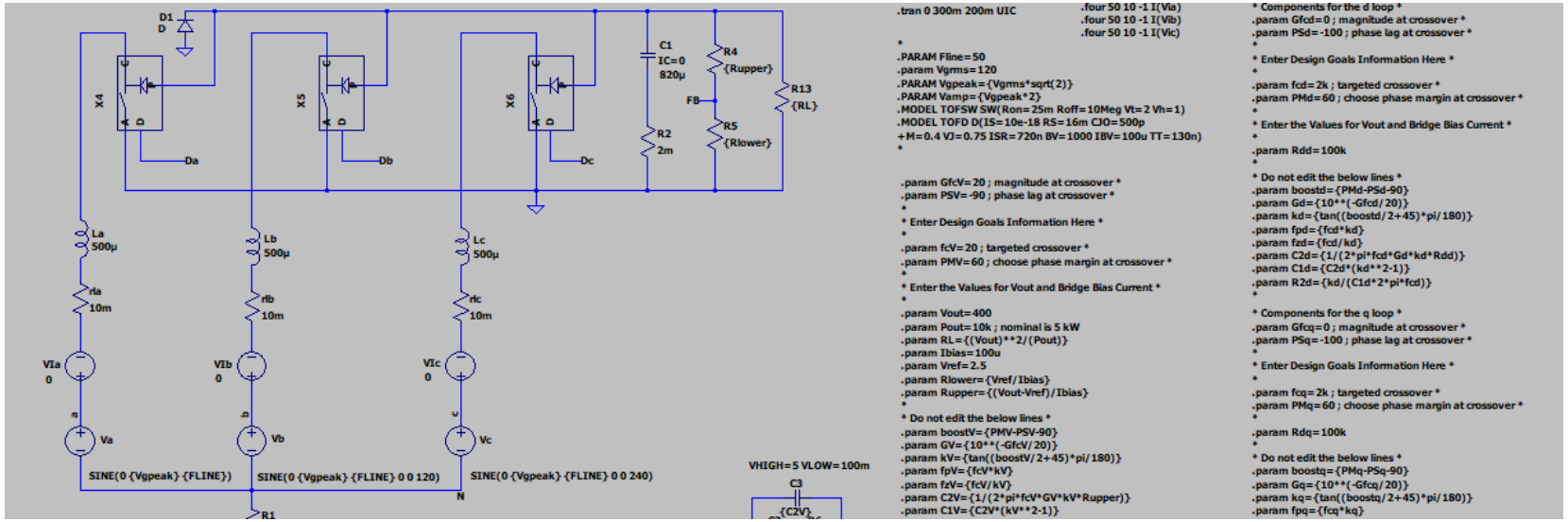
It is then possible to test the transient response by adding a switch on the output rail.



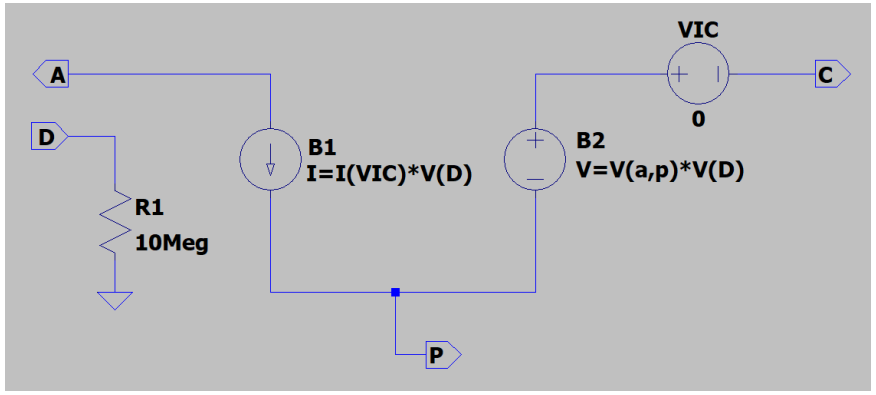
In this plot, the power is stepped from 5 to 10 kW and the response is stable, with a good recovery. Simulation time is 511 s.

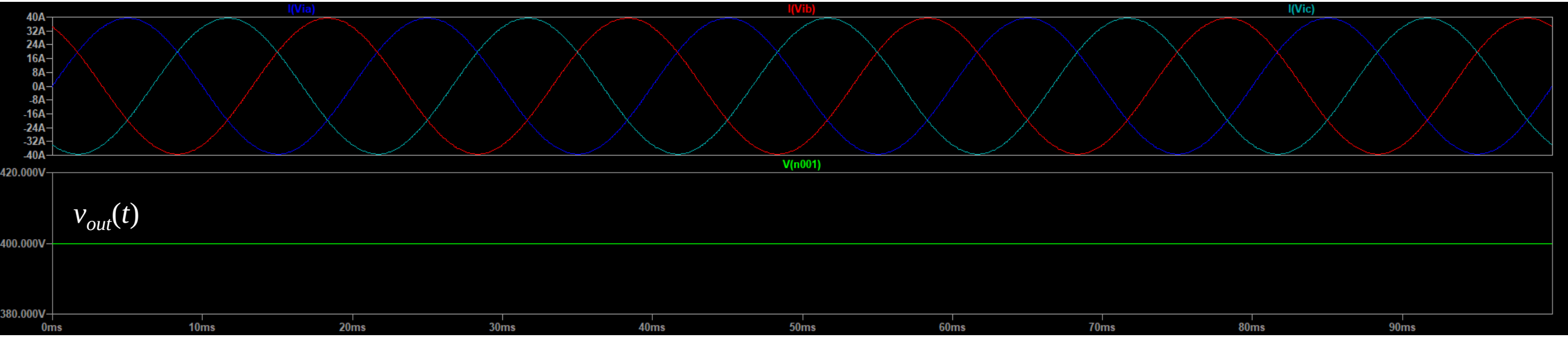


I have now inserted the voltage-mode PWM switch model operated in CCM. Diode D_1 is here to improve the operating bias point calculation.

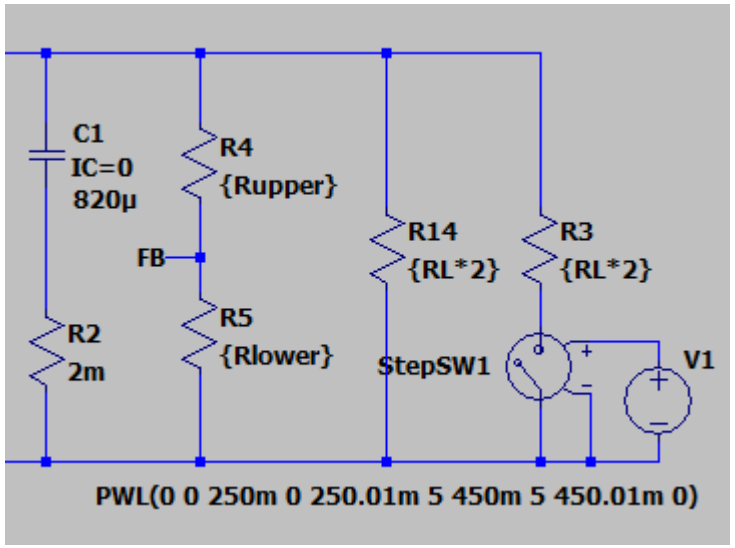


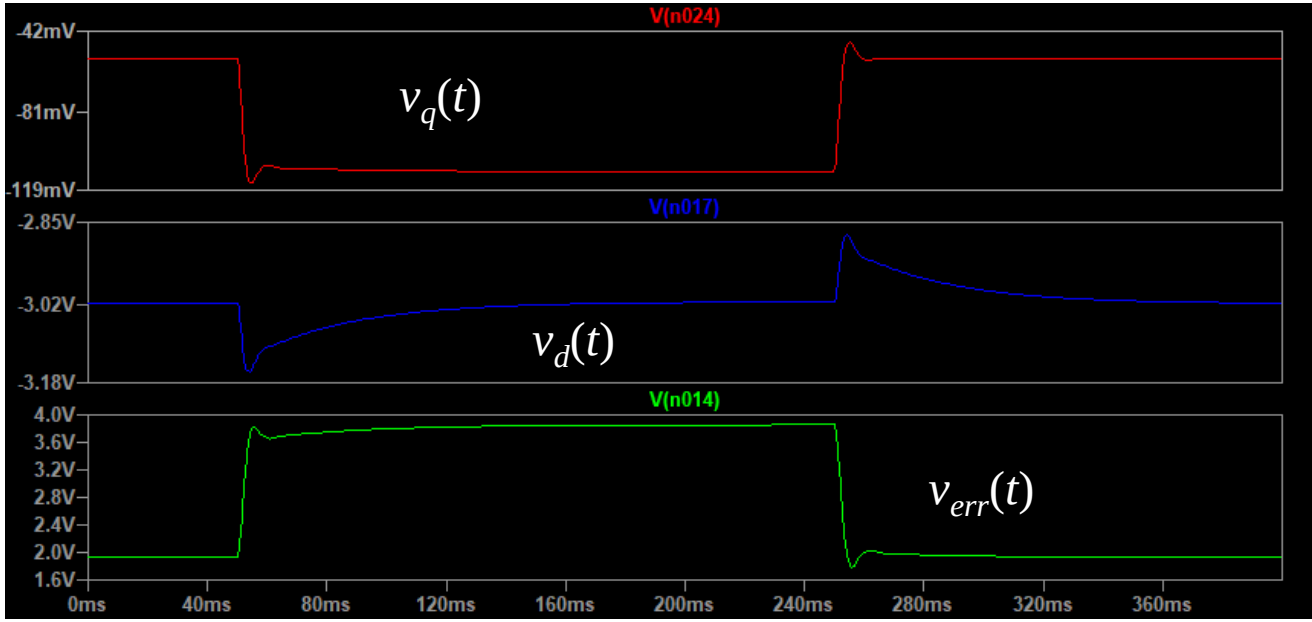

```
.tran 0 300m 200m UIC
*
.PARAM Fline=50
.param Vgrms=120
.PARAM Vgpeak={Vgrms*sqrt(2)}
.PARAM Vamp={Vgpeak*2}
.MODEL TOFSW SW(Ron=25m Roff=10Meg Vt=2 Vh=1)
.MODEL TOFD D(IS=10e-18 RS=16m CJO=500p
+M=0.4 VJ=0.75 ISR=720n BV=1000 IBV=100u TT=130n)
*

.param GfcV=20 ; magnitude at crossover *
.param PSV=-90 ; phase lag at crossover *
*
* Enter Design Goals Information Here *
*
.param fcV=20 ; targeted crossover *
.param PMV=60 ; choose phase margin at crossover *
*
* Enter the Values for Vout and Bridge Bias Current *
*
.param Vout=400
.param Pout=10k ; nominal is 5 kW
.param RL={(Vout)**2/(Pout)}
.param Ibias=100u
.param Vref=2.5
.param Rlower={Vref/Ibias}
.param Rupper={(Vout-Vref)/Ibias}
*
* Do not edit the below lines *
.param boostV={PMV-PSV-90}
.param GV={10**(-GfcV/20)}
.param kV={tan((boostV/2+45)*pi/180)}
.param fpV={fcV*kV}
.param fzV={fcV/kV}
.param C2V={1/(2*pi*fcV*GV*kV*Rupper)}
.param C1V={C2V*(kV**2-1)}
.four 50 10 -1 I(Via)
.four 50 10 -1 I(Vib)
.four 50 10 -1 I(Vic)
*
* Components for the d loop *
.param Gfcd=0 ; magnitude at crossover *
.param PSd=-100 ; phase lag at crossover *
*
* Enter Design Goals Information Here *
*
.param fcd=2k ; targeted crossover *
.param PMd=60 ; choose phase margin at crossover *
*
* Enter the Values for Vout and Bridge Bias Current *
*
.param Rdd=100k
*
* Do not edit the below lines *
.param boostd={PMd-PSd-90}
.param Gd={10**(-Gfcd/20)}
.param kd={tan((boostd/2+45)*pi/180)}
.param fpd={fcd*kd}
.param fzd={fcd/kd}
.param C2d={1/(2*pi*fcd*Gd*kd*Rdd)}
.param C1d={C2d*(kd**2-1)}
.param R2d={kd/(C1d*2*pi*fcd)}
*
* Components for the q loop *
.param Gfcq=0 ; magnitude at crossover *
.param PSq=-100 ; phase lag at crossover *
*
* Enter Design Goals Information Here *
*
.param fcq=2k ; targeted crossover *
.param PMq=60 ; choose phase margin at crossover *
*
.param Rdq=100k
*
* Do not edit the below lines *
.param boostq={PMq-PSq-90}
.param Gq={10**(-Gfcq/20)}
.param kq={tan((boostq/2+45)*pi/180)}
.param fpq={fcq*kq}
```

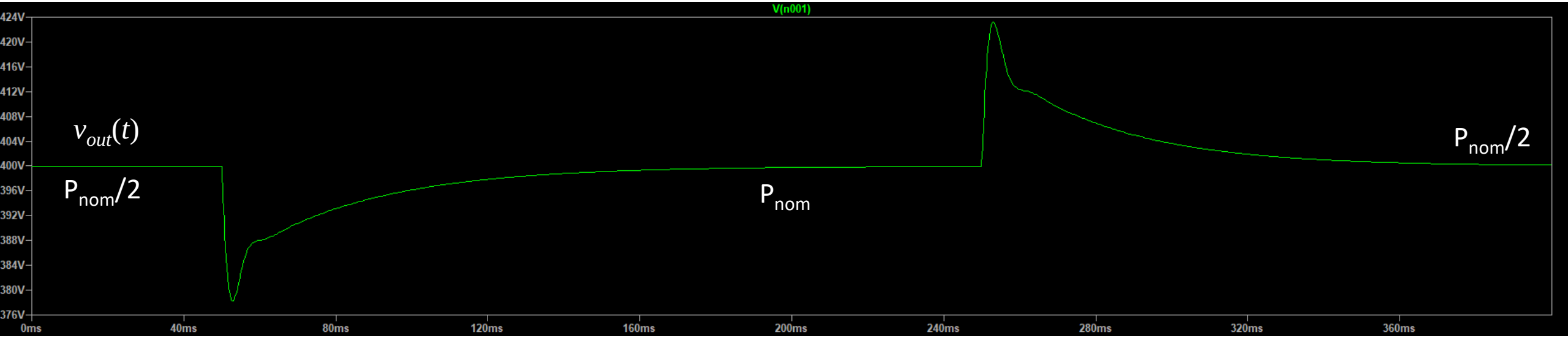




The averaged transient simulation shows the exact same response as with the cycle-by-cycle exercise:

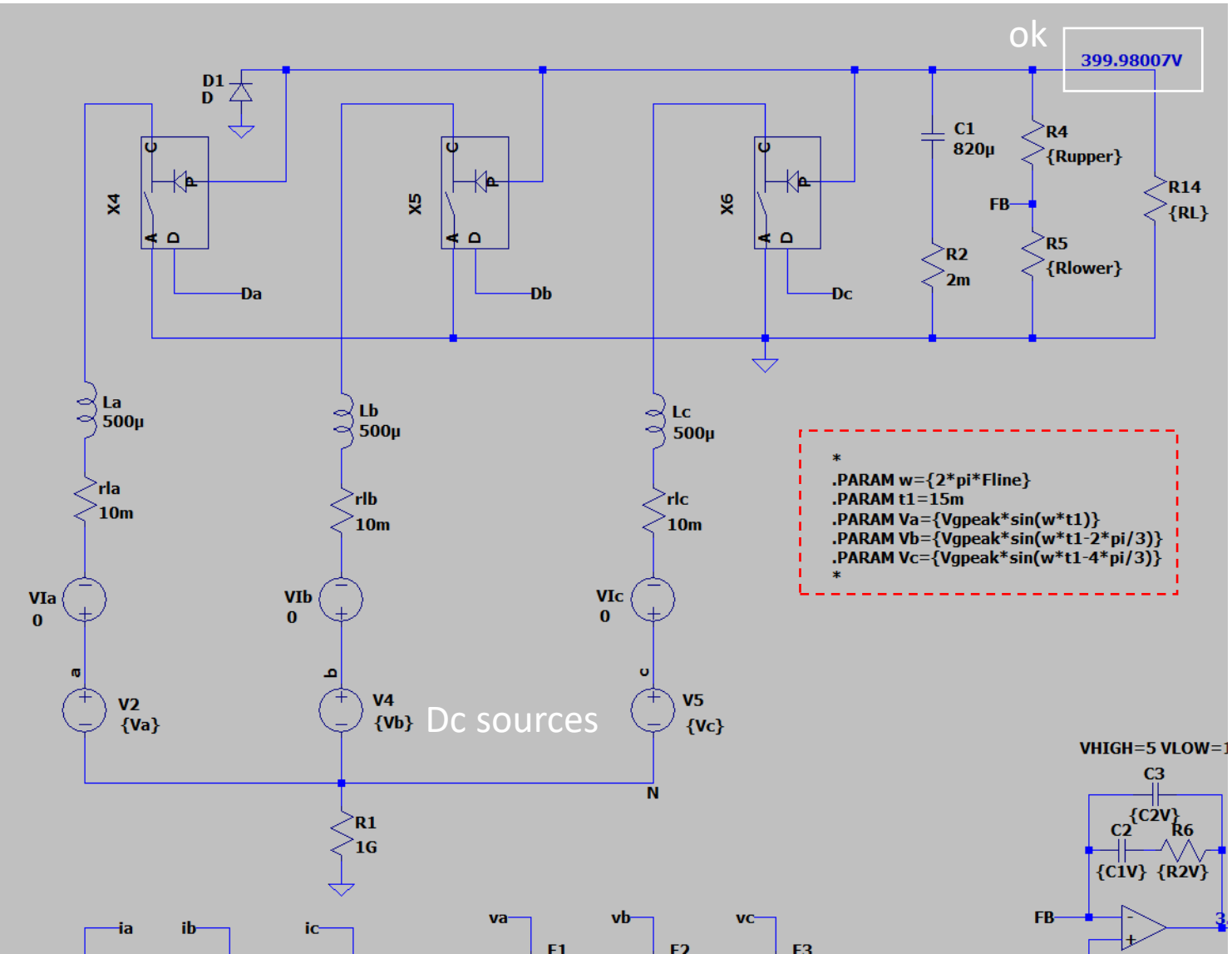


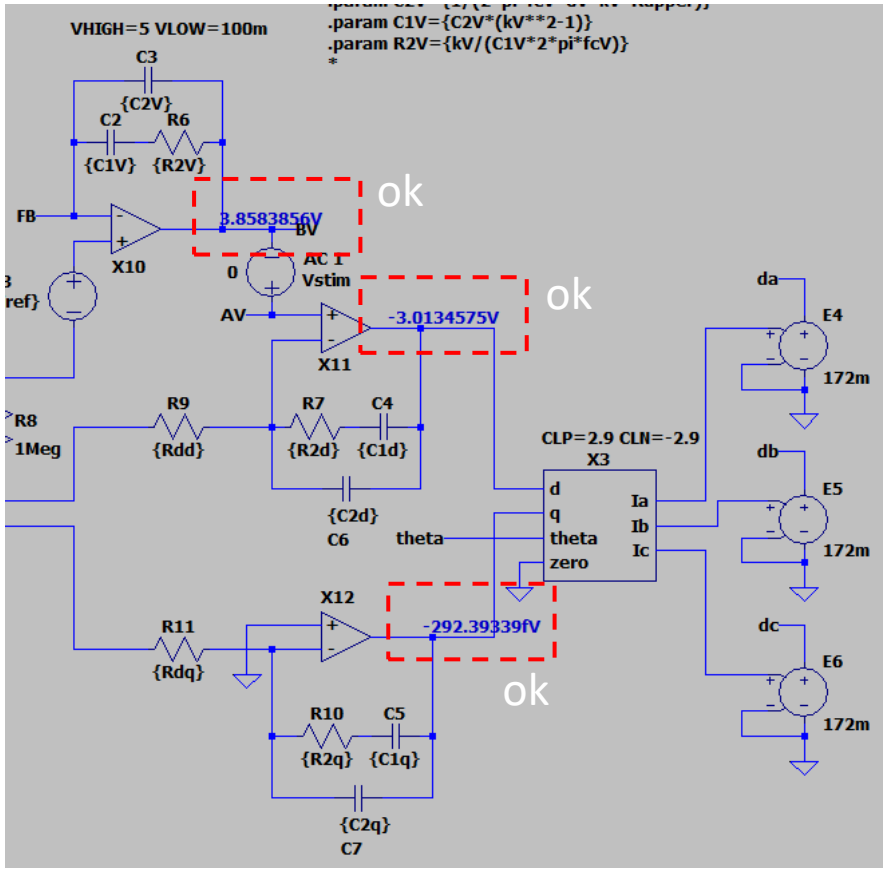




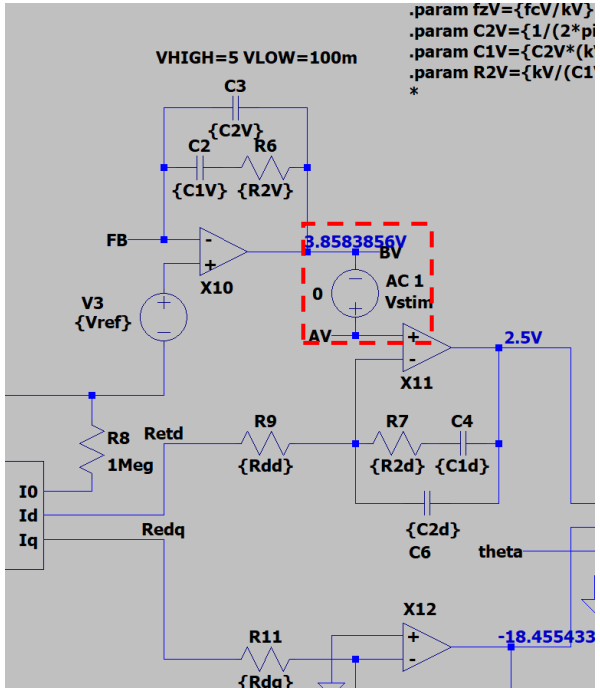
Simulation time is 9 s

It is now important to analyze all three loops in ac and feed the macro with the open-loop values. Before this step, the averaged models are biased by fixed dc sources. I have arbitrarily set a time at which these sources are frozen, e.g. 15 ms in this example. Before considering the ac results, it is important to verify that the bias points are correct, e.g. the output is well regulated and the error, d and q voltages from the op-amps are meaningful. The .IC is gone on the output capacitor.

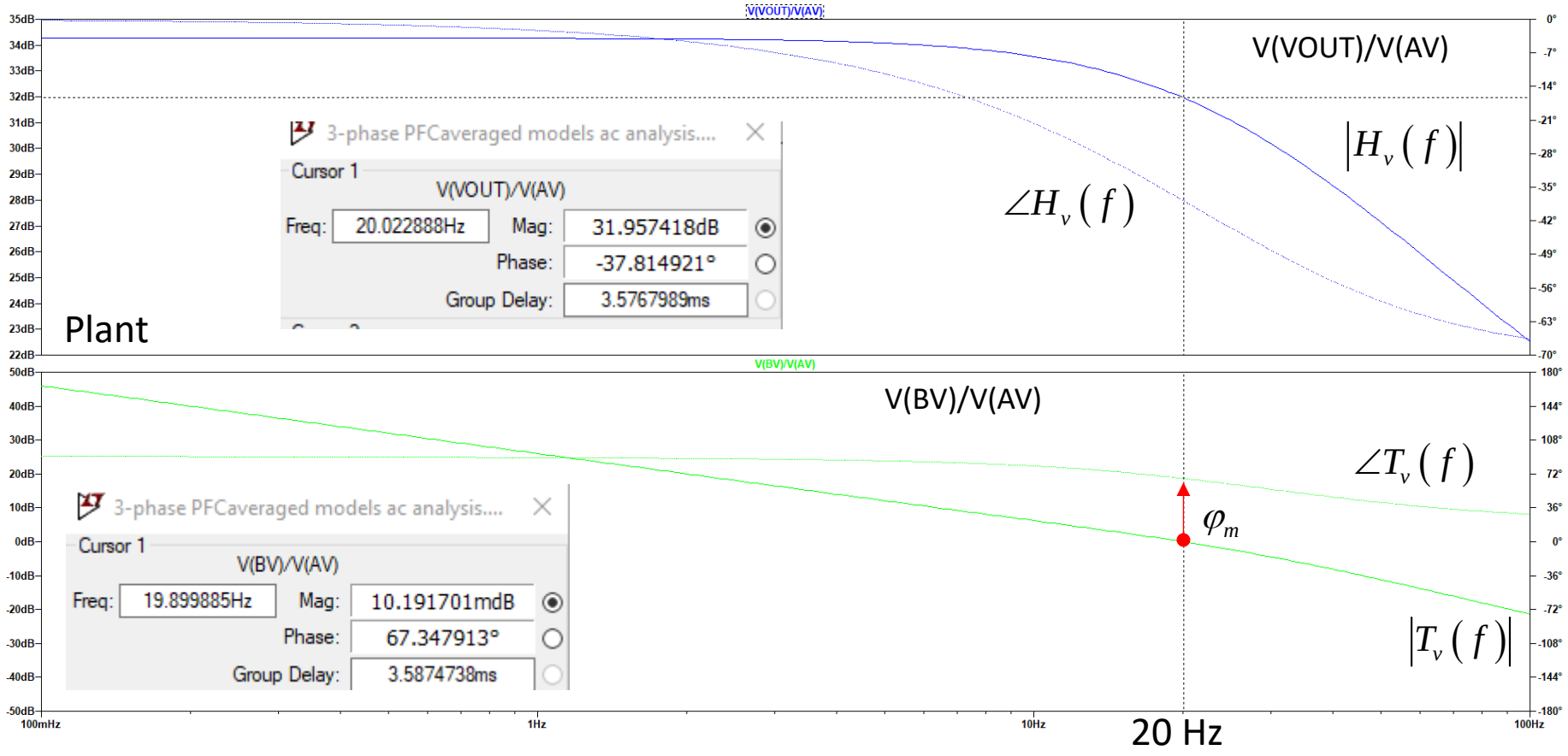




The loops have to be individually swept. First, extract the control-to-output transfer function for the voltage loop.



Plotting V(BV)/V(AV) gives the compensated loop gain. The control-to-output transfer function is obtained by plotting V(VOUT)/V(AV).



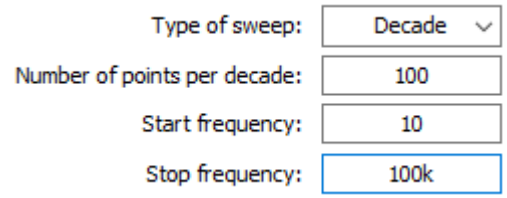
Type of sweep:	Decade
Number of points per decade:	100
Start frequency:	100m
Stop frequency:	100

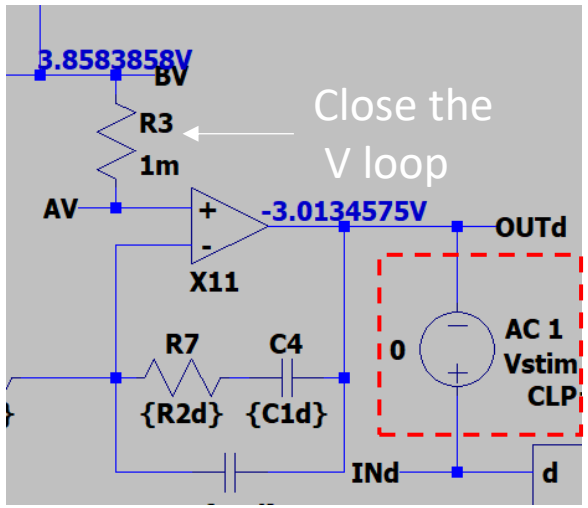
```
.param GfcV=32 ; magnitude at crossover *
.param PSV=-45 ; phase lag at crossover *
*
* Enter Design Goals Information Here *
*
```

Simulation shows a gain excess of 32 dB and I purposely set the phase shift to -45° (and not -32°) to force a type 1 response from the type 2 compensator. Phase margin is good at 67° in the end.

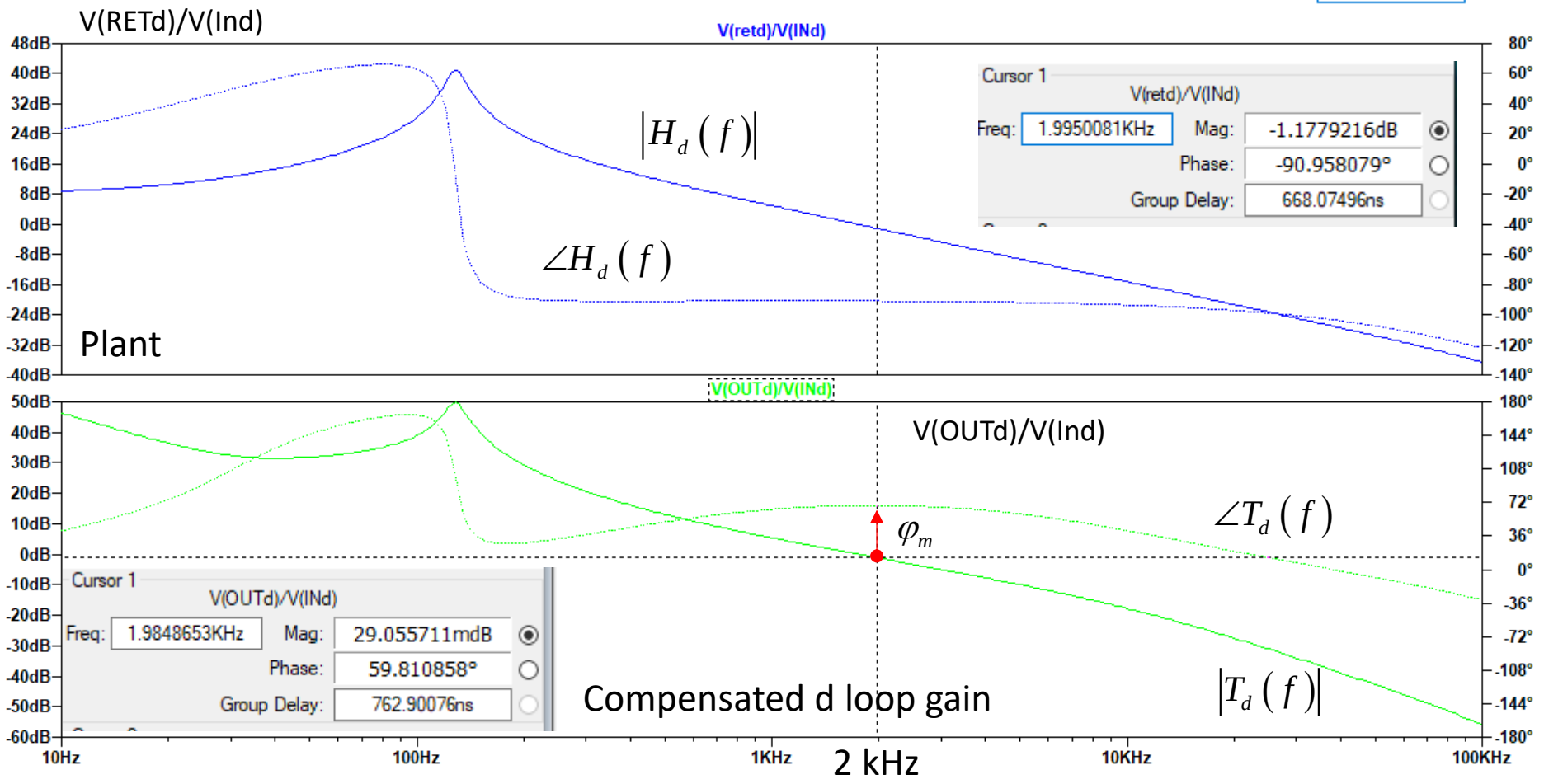
I am not that happy with the background and trace colors but this site helped for the selection.

For the d loop, change the ac sweep range from 10 to 100 kHz in the LTspice control menu:

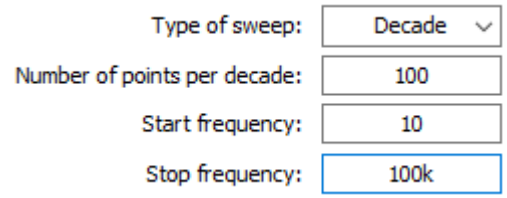


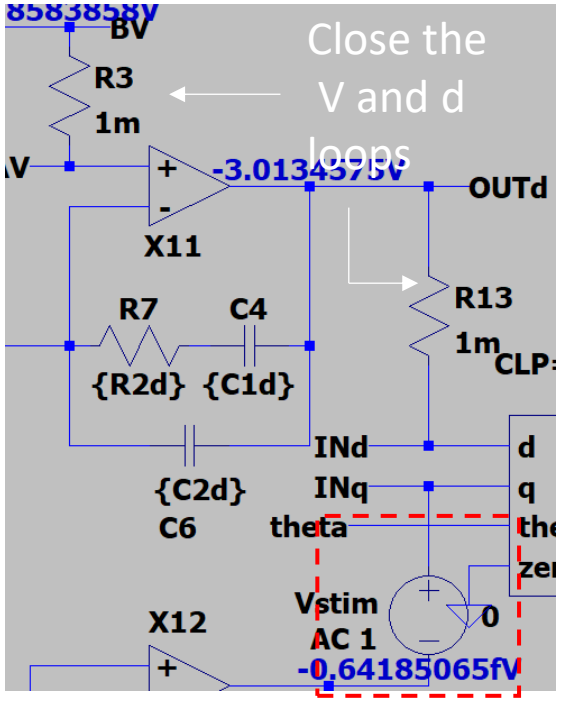


Insert the ac source in the d loop, keep the V loop closed for regulation purposes.

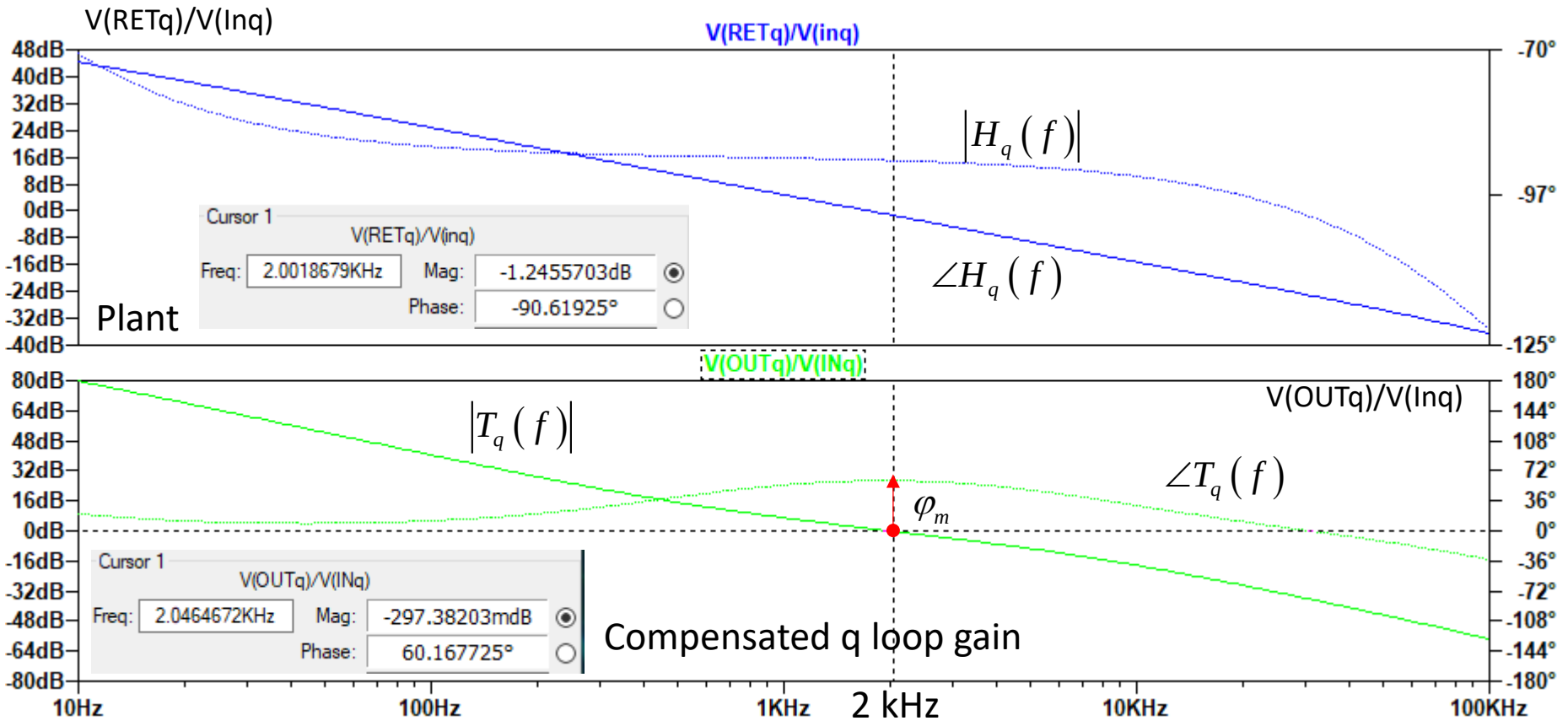


For the q loop, keep the ac sweep range from 10 to 100 kHz in the LTspice control menu:

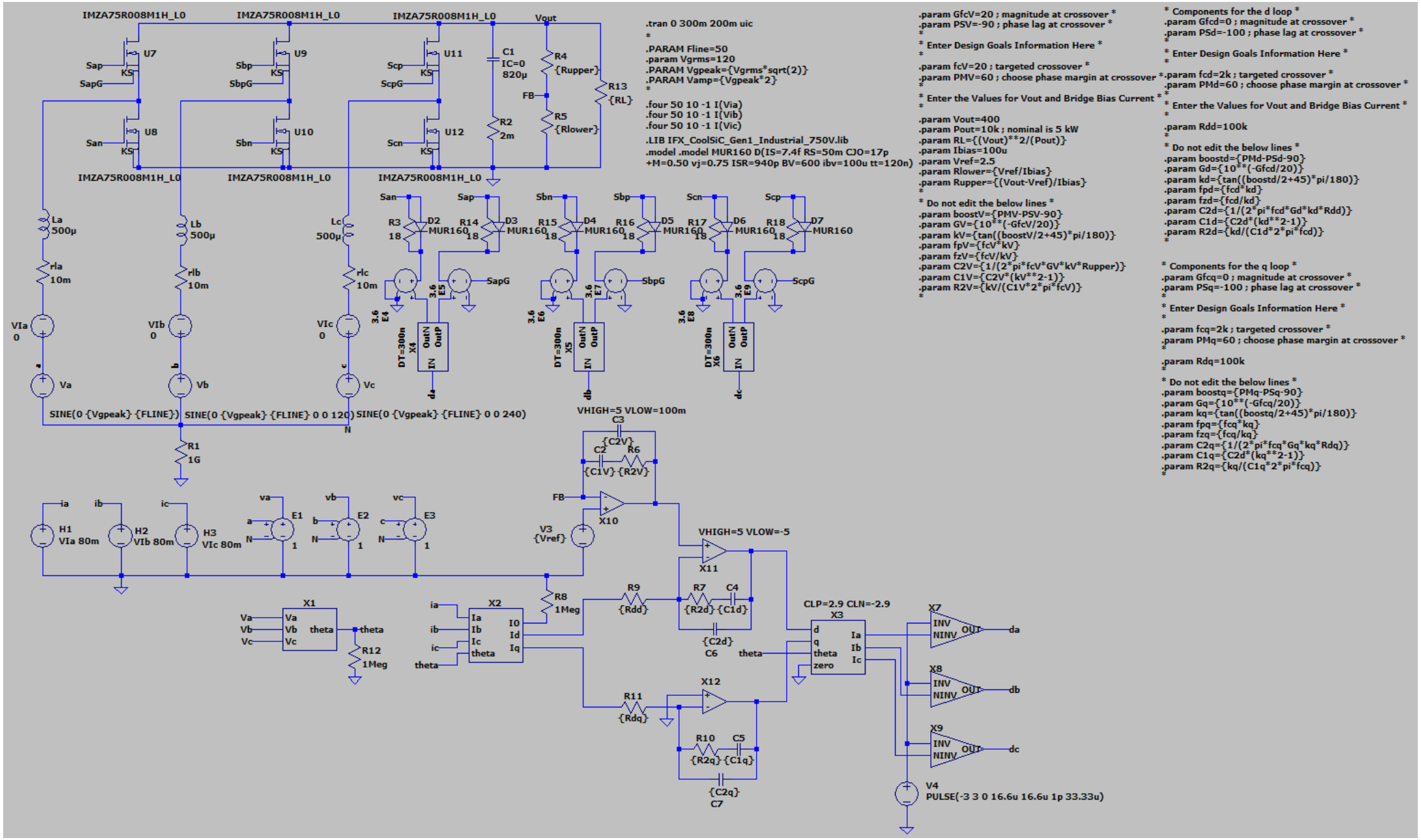




Insert the ac source in the q loop, keep the V and d loops closed for regulation purposes.



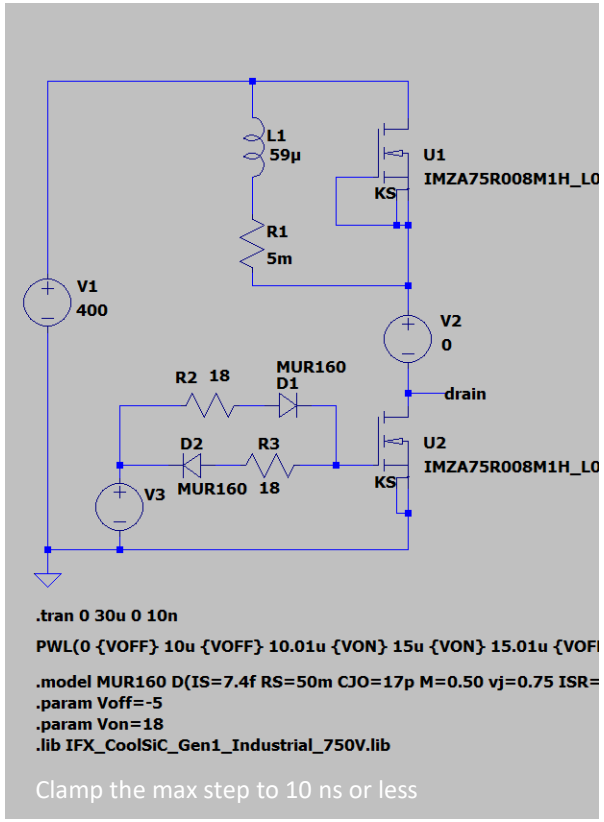
I have then imported SiC transistors from Infineon and, as expected, the simulation time is longer (25 mn)

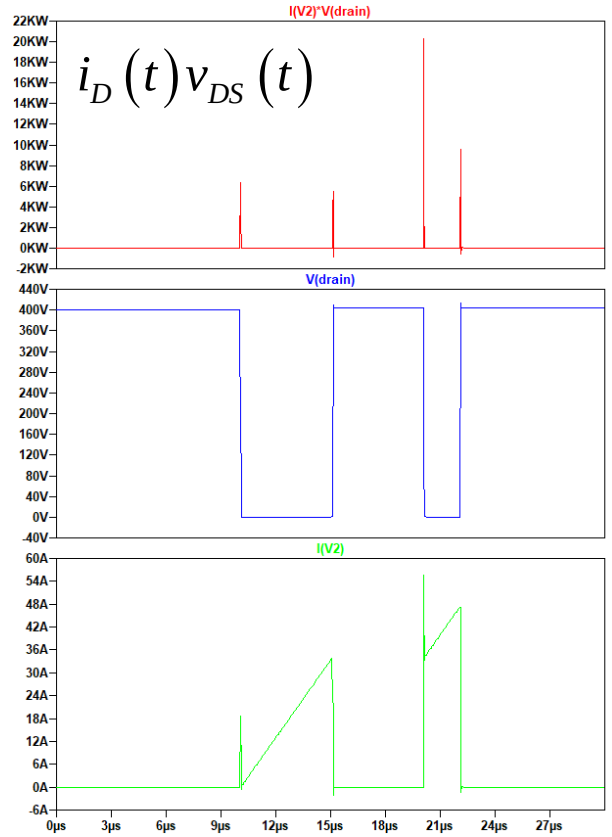


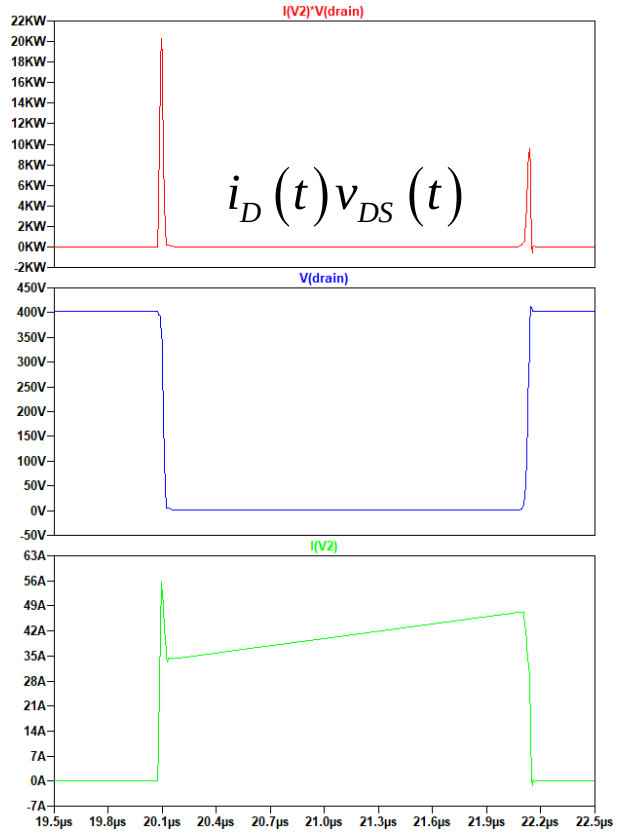
Before picking up a MOSFET or a SiC model, it is crucial to check its accuracy, in particular its switching characteristics. It is important to note that the switching characteristics of a transistor like a MOSFET or a SiC heavily depend on the below elements:

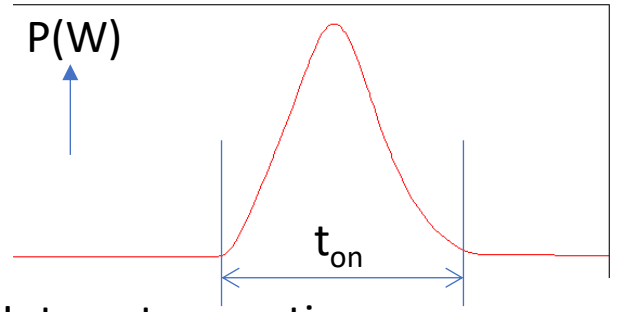
- Junction temperature
- Driver: output impedance, on and off drive levels, network in series with DRV output (diode/resistance), distance to gate etc.
- Power distribution network: what are the stray elements characterizing the traces/wires bringing the voltage/current variables to the connection pins of the transistor (resistances, inductances and capacitances)

So before running a long simulation, make sure the adopted model is of good quality. One way to check its basic switching properties, is to characterize it with a double-pulse test. Extract the energy at turn on and turn off and check with the data-sheet (reproduce the same test setup as in the document of course, current and voltage).





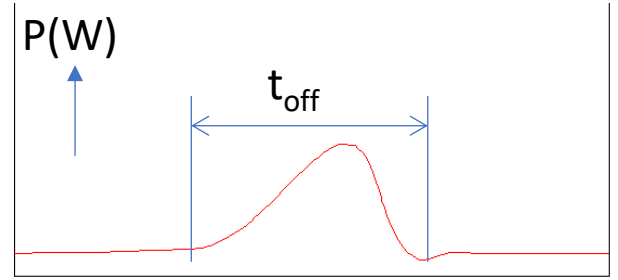




Integrate over time

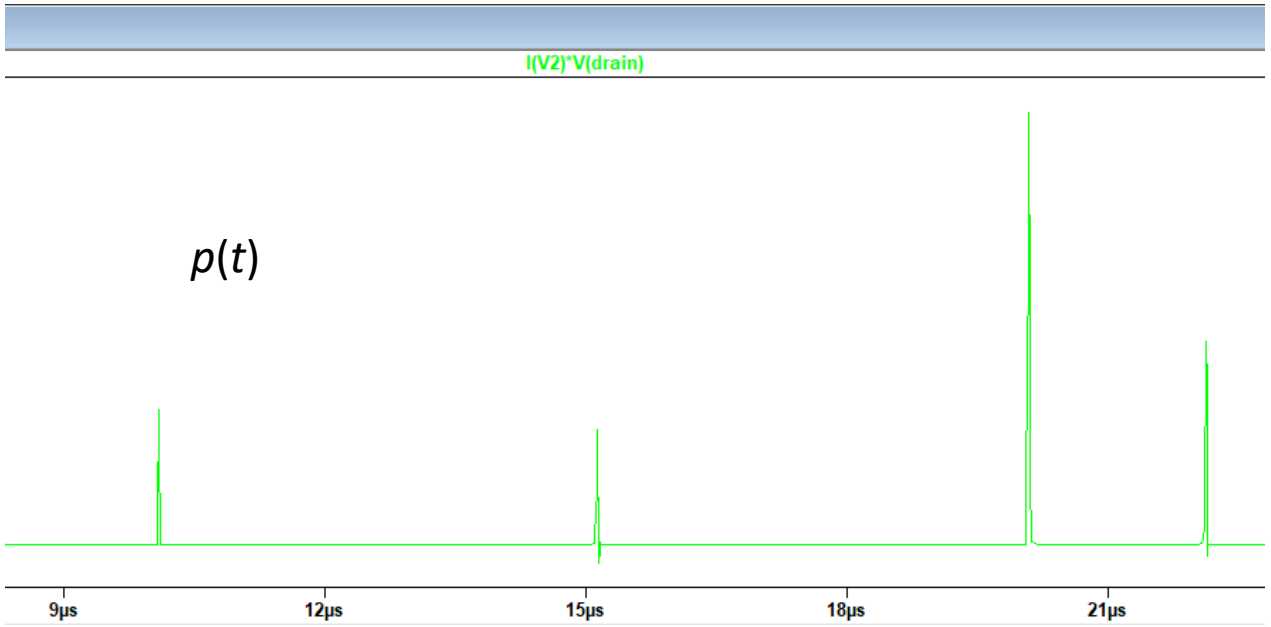
$$E_{on} = P_{avg}\Big|_{\Delta t = t_{on}} t_{on}\ [J]$$

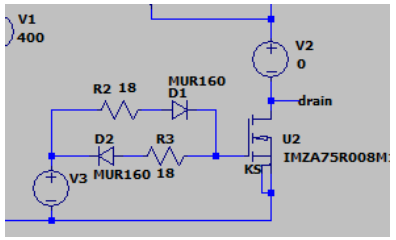
$$E_{off} = P_{avg}\Big|_{\Delta t = t_{off}} t_{off}\ [J]$$



PWL(0 {VOFF} 10u {VOFF} 10.01u {VON} 15u {VON} 15.01u {VOFF} 20u {VOFF} 20.01u {VON} 22u {VON} 22.01u {VOFF})

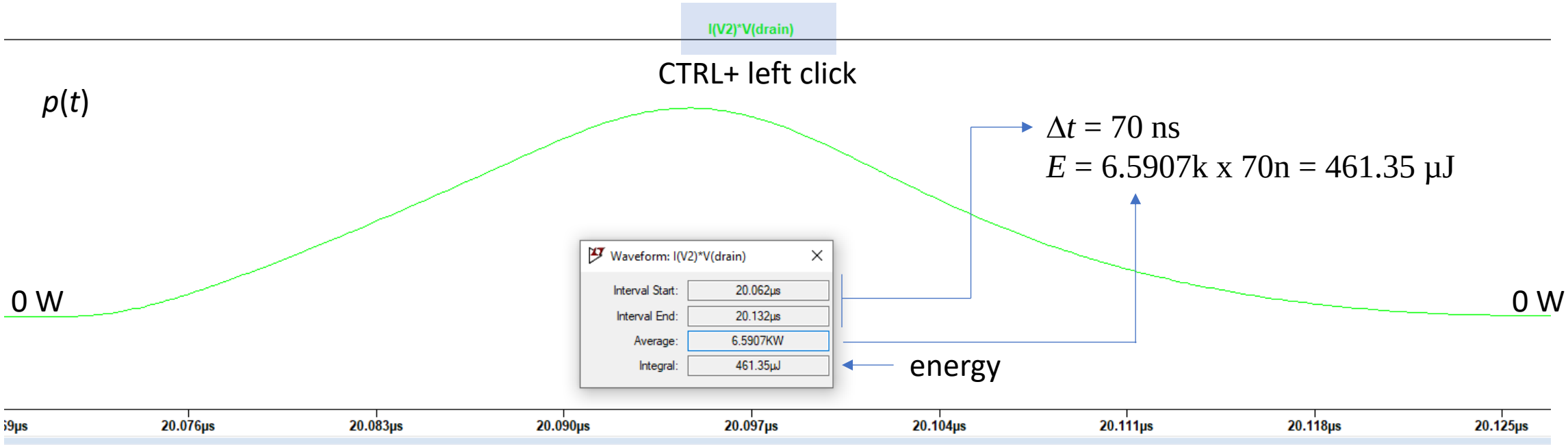
The energy can be measured by zooming-in the instantaneous power waveform:



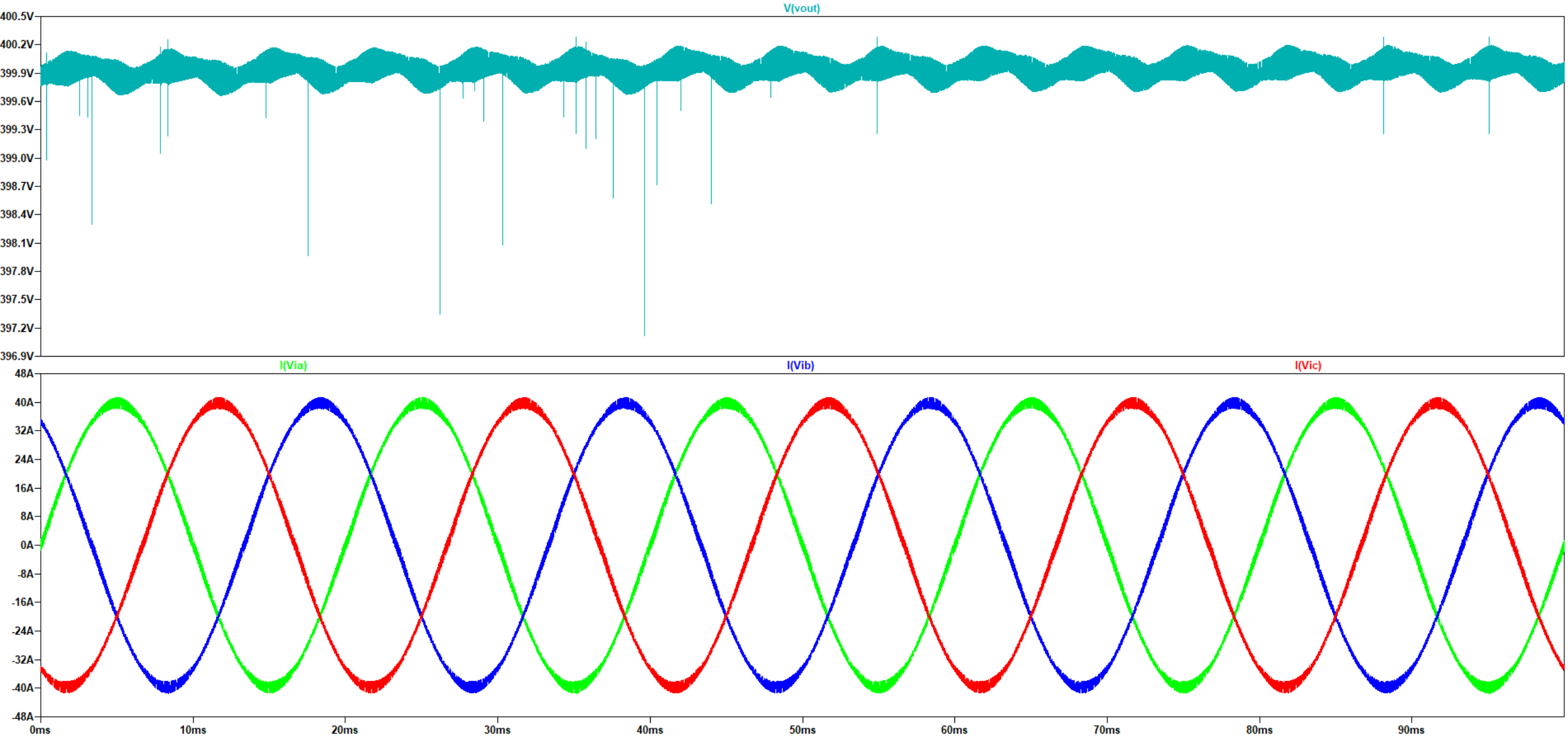


For power measurement, hover the cursor over the MOSFET or diode and press ALT, a thermometer is shown: click and $p(t)$ is displayed

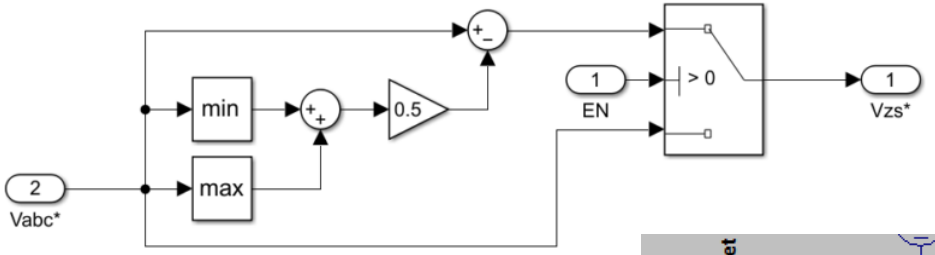
Once the on-time or off-time instantaneous power is exactly zoomed-in (almost 0 W on every side), press CTRL + left click on the mouse while pointing on the waveform label. You have the energy integrated along the displayed curve. I did not find a way to average between the cursors only.



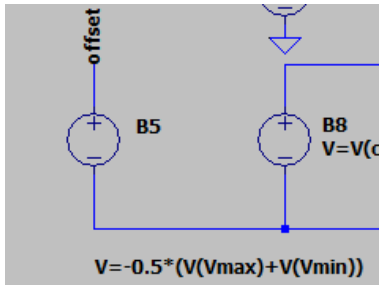
Excellent results, with a THD at 2.3%.

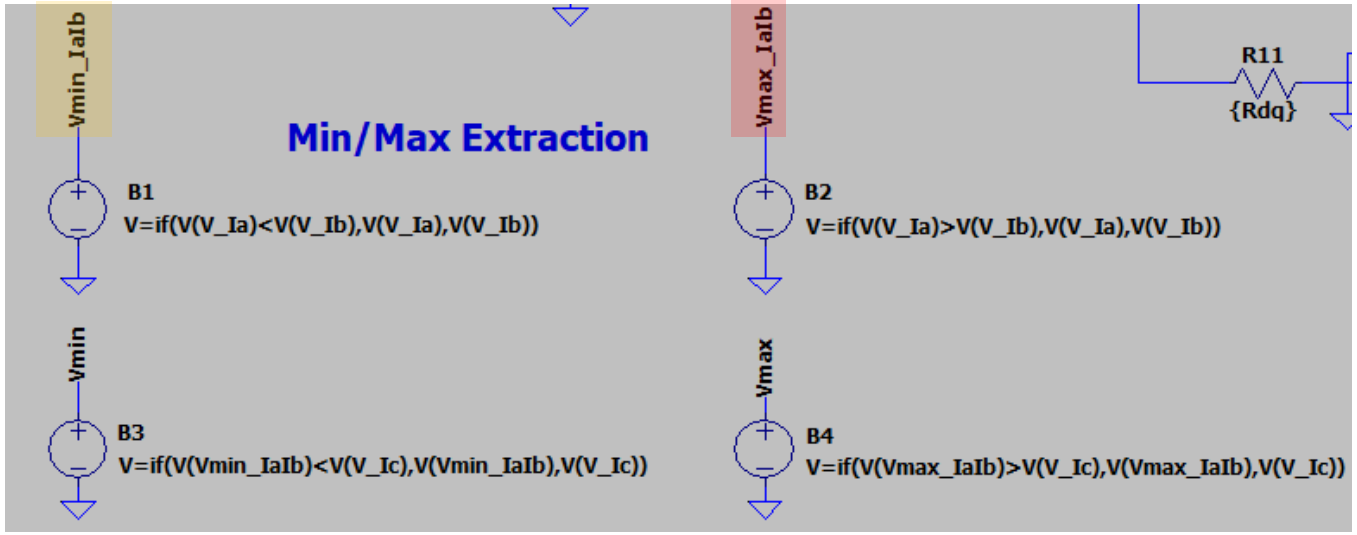


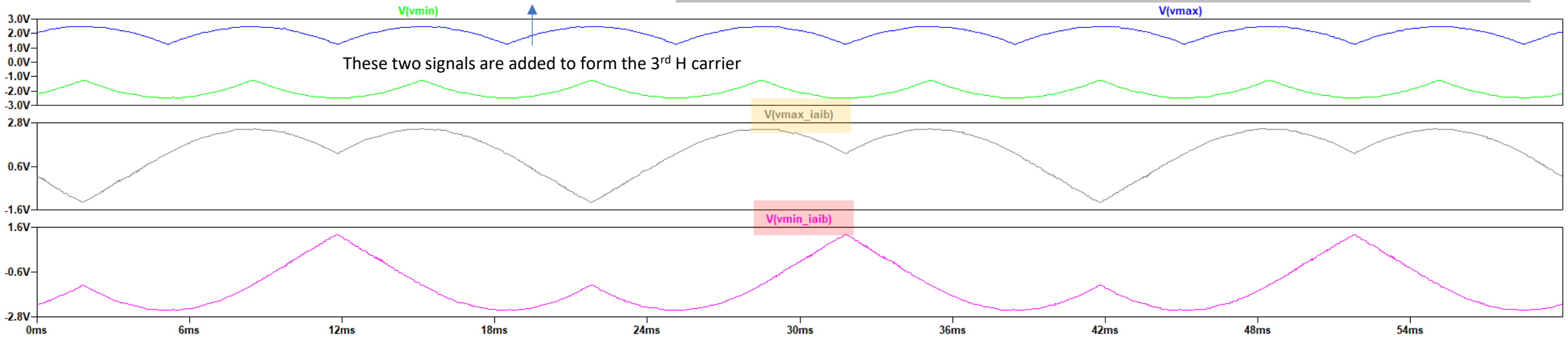
It is also possible to add a 3rd-harmonic modulation to improve the current distortion. In motor control, adding a third harmonic helps increasing the modulation index up to 115% as documented in this article. Adding this offset in LTspice requires the implementation of the so-called min/max injection technique. The idea is to build a 150-Hz (for a 50-Hz line) carrier that will be added to the original signals computed by the dq0 block.



SVPWM vs SPWM modulation technique
Imperix – Simon Strobl – August 2021

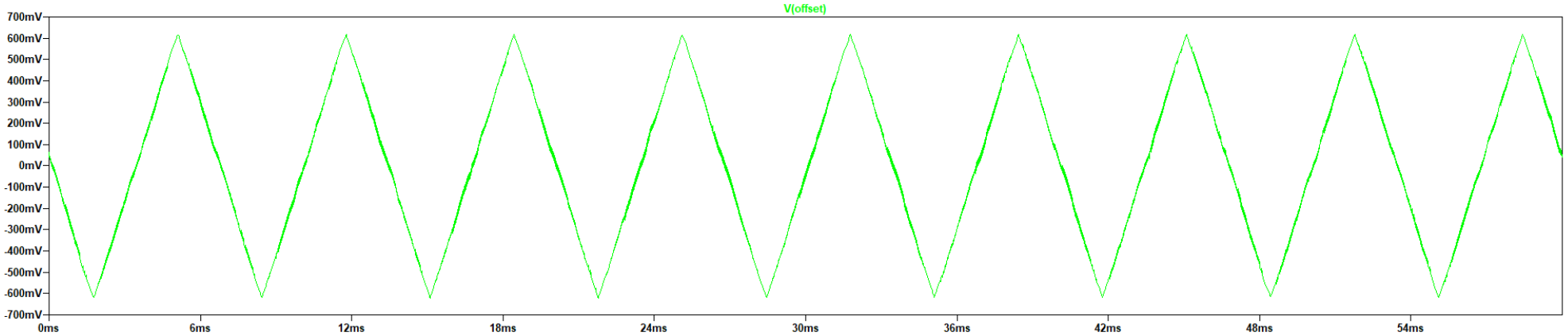


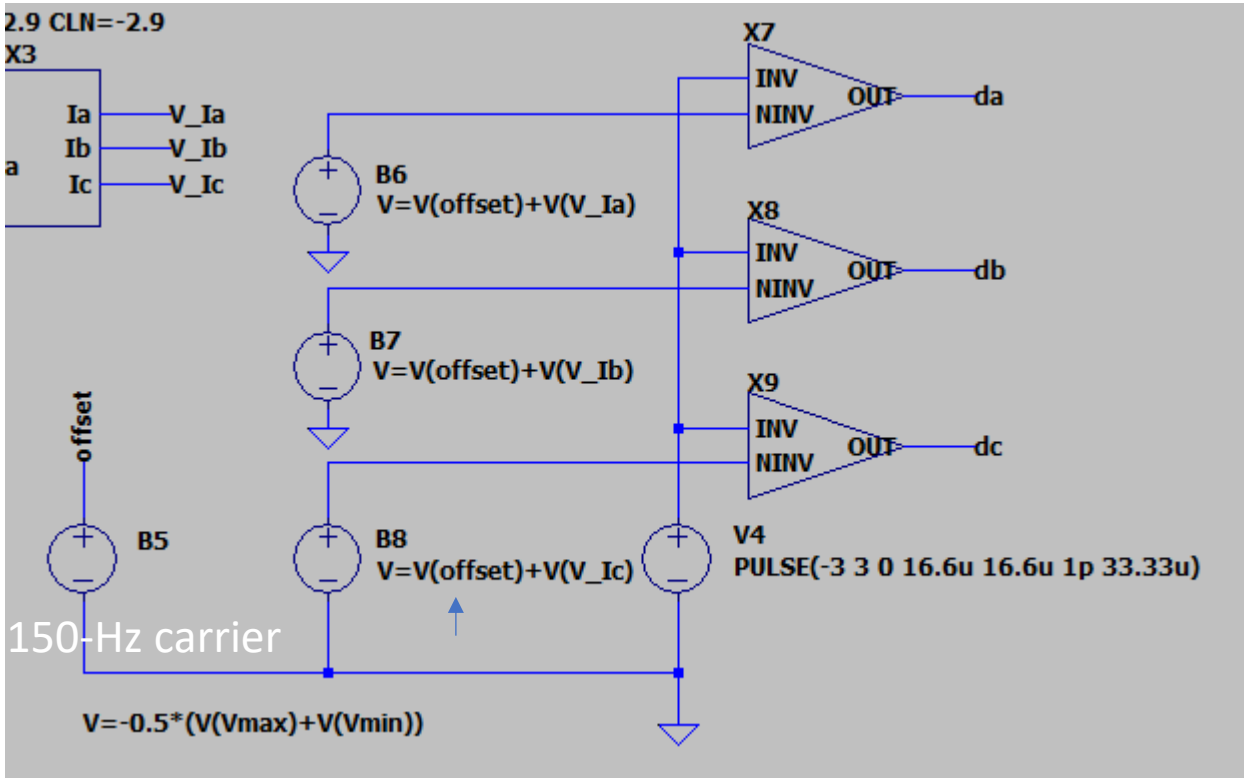


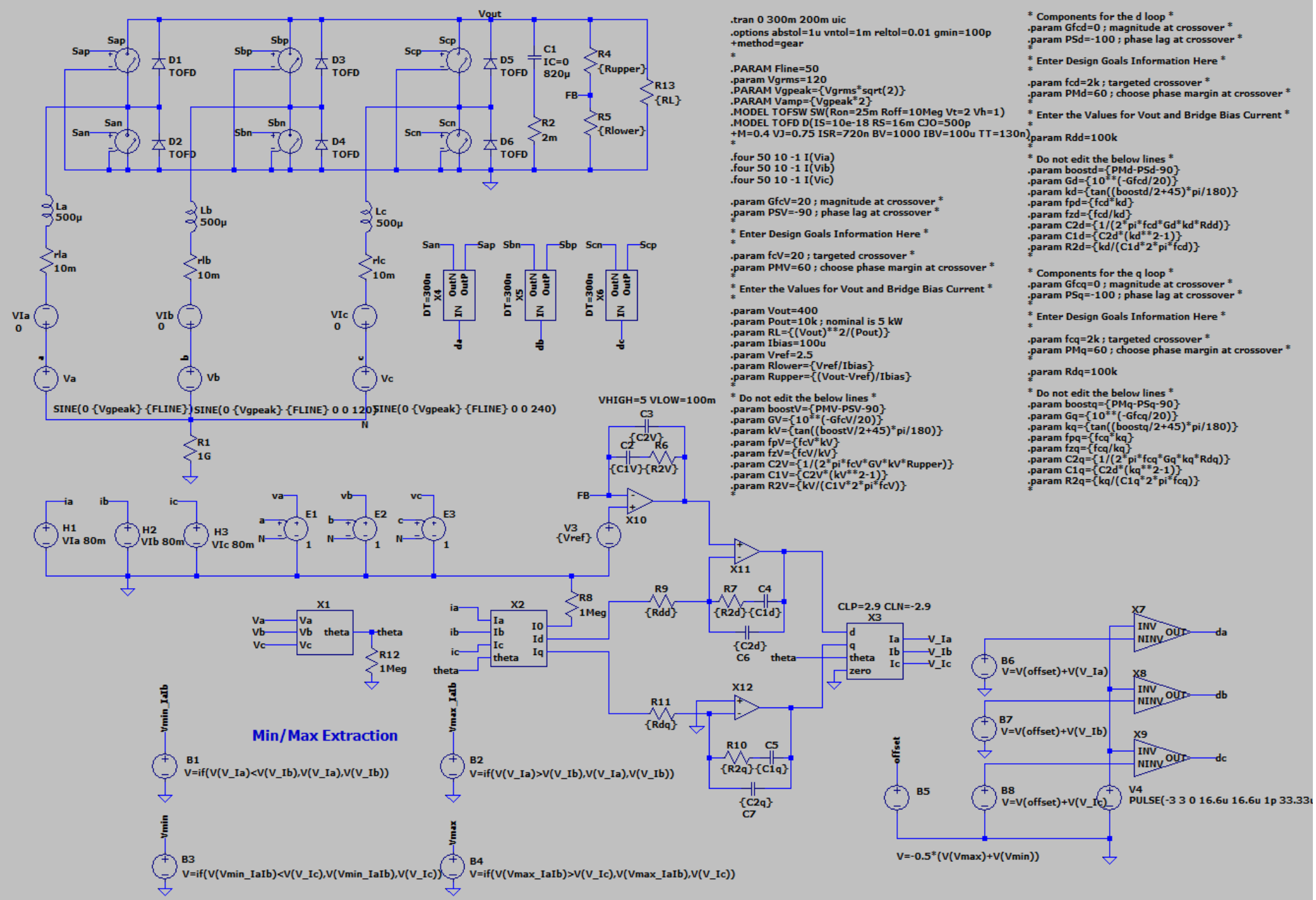


This idea has been kindly contributed by Teddy Bonnin from Cissoid – Merci !

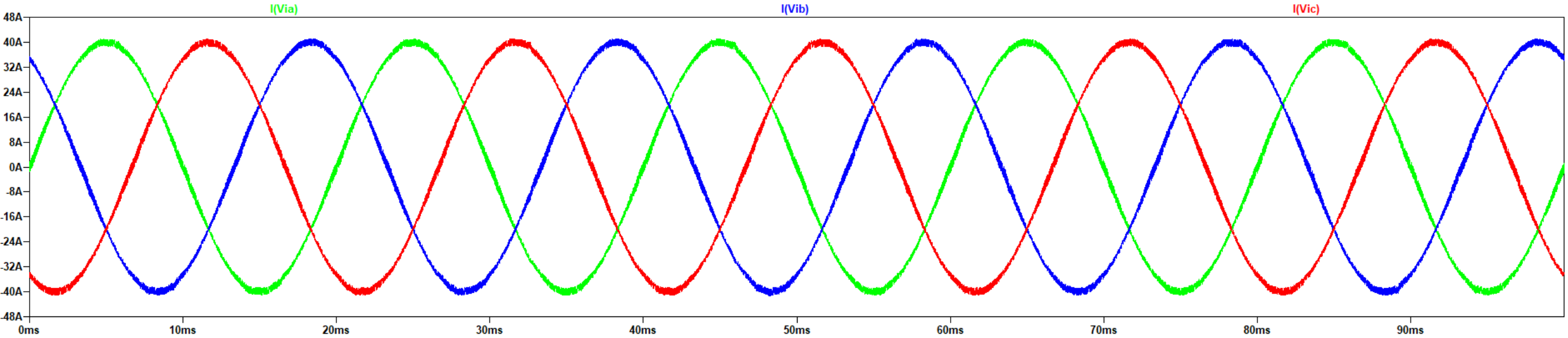
The resulting 150-Hz triangular carrier is then added as an offset to the dq0 signals before entering the PWM comparators:





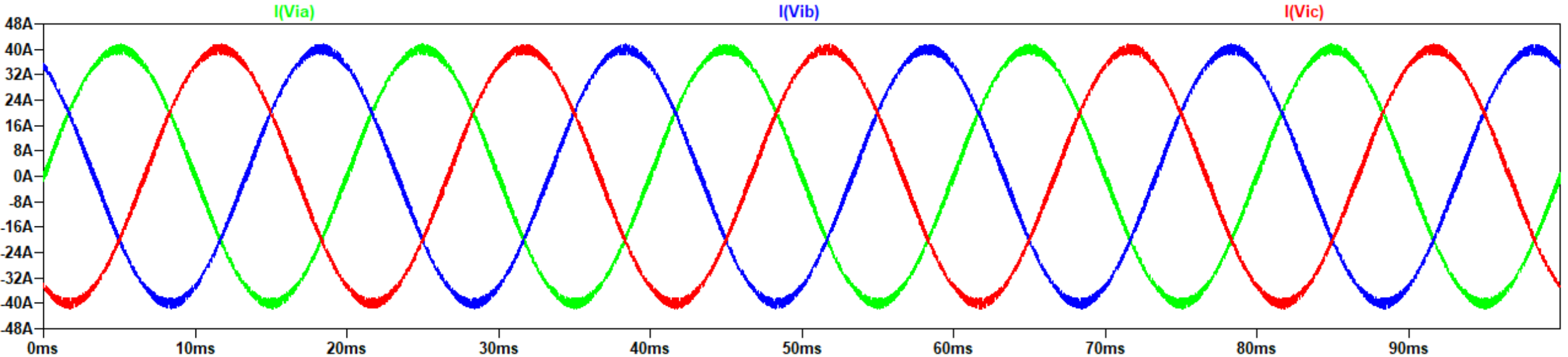


This is the complete circuit featuring the 3rd-harmonic injection.



Total Harmonic Distortion: 2.005869%

with 3rd harmonic injection



Total Harmonic Distortion: 2.230810%

without 3rd harmonic injection