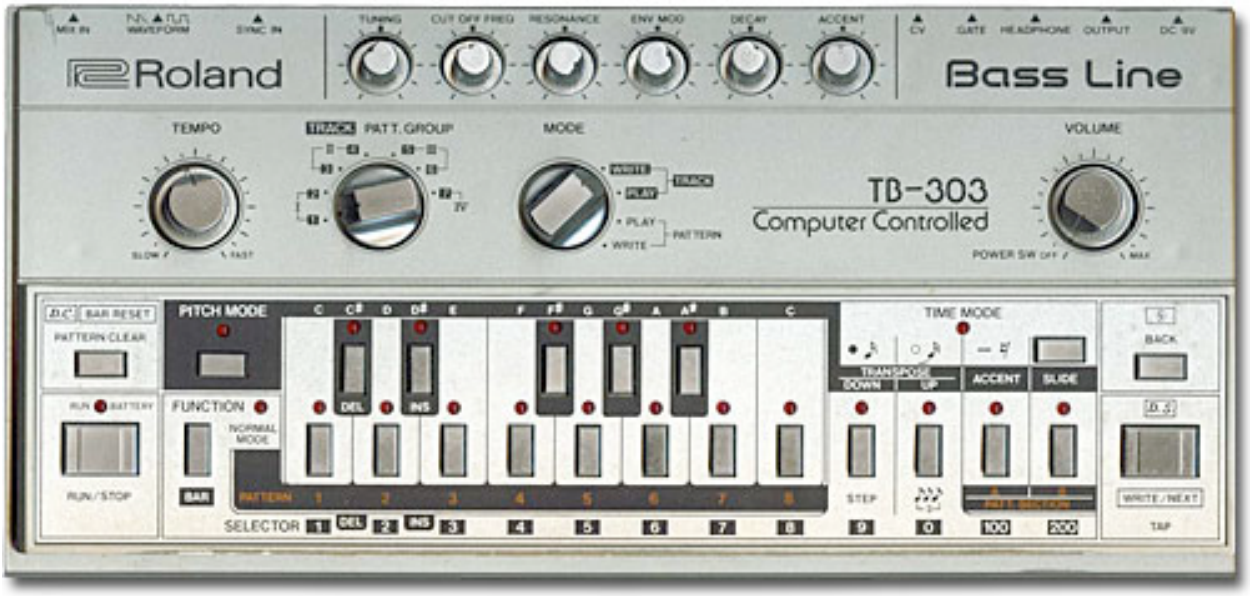
Oscillator Project: TB-303 Saw and Square Waves



In this project you will try to replicate the sawtooth and square waves from the Roland TB-303 BassLine synth. This uber-famous synth had only two waveforms; *saw* and *square*. Both waveforms are very non-textbook and mathematically incorrect. But they sound great.

The goals of the project are:

- create a trivial waveform mathematically (saw)
- understand polyBLEP by actually using it (saw and square) rather than copying my code
- use creativity, intelligence and imagination to implement something you have never seen (saw, square) - you have to be an engineer/problem solver

Your project consists of 4 parts:
1. trivial TB-303 sawtooth
2. TB-303 sawtooth with polyBLEP
3. trivial TB-303 square wave
4. fully shaped TB-303 square wave with polyBLEP

I will give you the project template so we will all have the same starting point for the plug-in - this will let you concentrate on the modeling part.

The TB-303 VCO generates a sawtooth wave with a “smashed” top. The square wave is very non square. This is shown in Figure 1 which is taken from the TB-303 Service Manual; the waveforms are drawn by hand and are there for a technician to refer to during repairs.

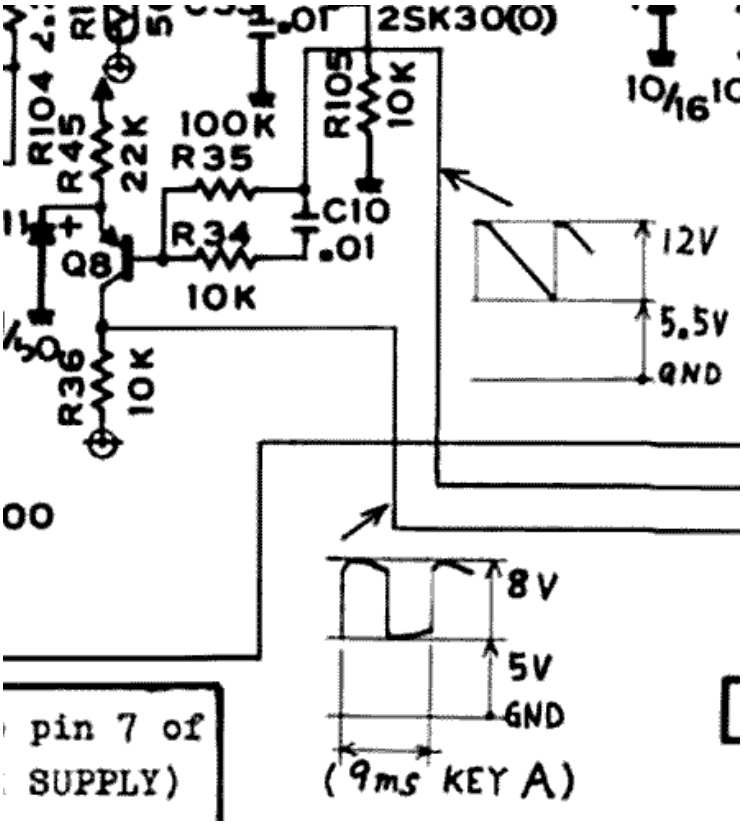


Figure 1: saw and square waves from the TB-303 service manual

Part 1: Sawtooth

The sawtooth wave in the TB-303 has several distinguishing characteristics:

- it is a “down” sawtooth
- the top part of it is nearly smashed (clipped) while the rest is perfectly linear

Let’s shoot for this (actual scope shot of the TB-303 saw) in Figure 2:

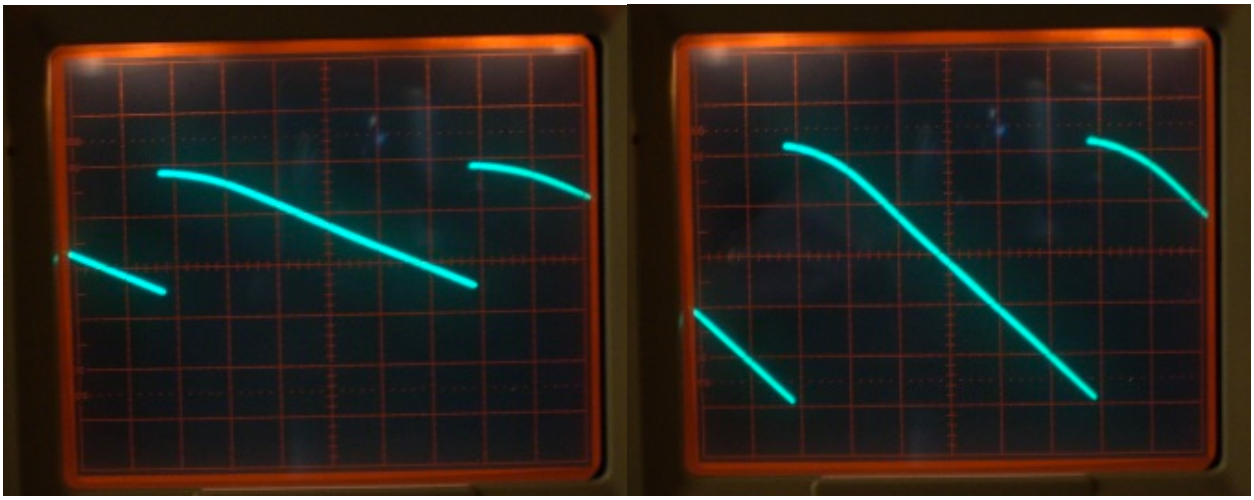


Figure 2: TB-303 Sawtooth at two different amplitudes

1. **Create a trivial version of this waveform**, no polyBLEP; for this, I estimated the waveform to look *something* like this two-section down-saw - In RackAFX, I produced what you see in Figure 3. In my version you can adjust the slope of the smashed part as well as the duration of it to get a range of clipped down-saw waveforms. I expect your versions to be better than this, with the more curved appearance in Figure 2. You can start with this approach (piecewise linear) and then go from there. Remember, there will be multiple ways to approach and solve this problem.

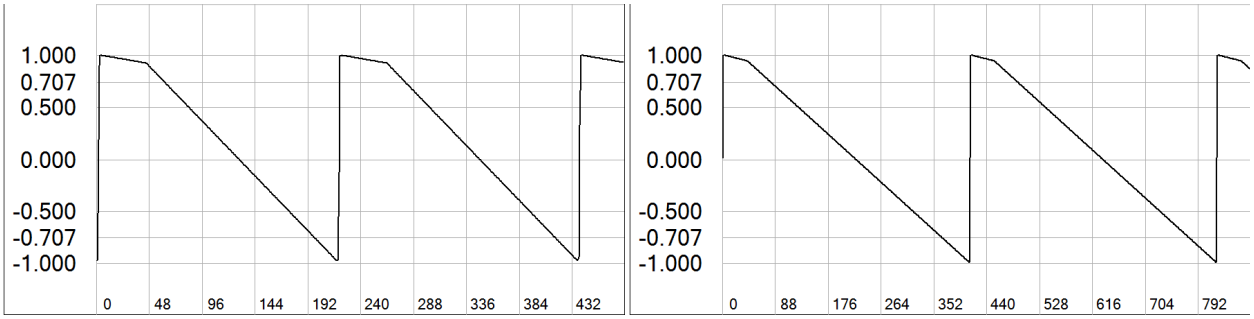


Figure 3: two versions of the TB-303 Saw

2. **Apply polyBLEP to reduce aliasing**; in the template plug-in I will provide, there is a polyBLEP button so you can make sure it works. You will need to implement the function calls correctly.

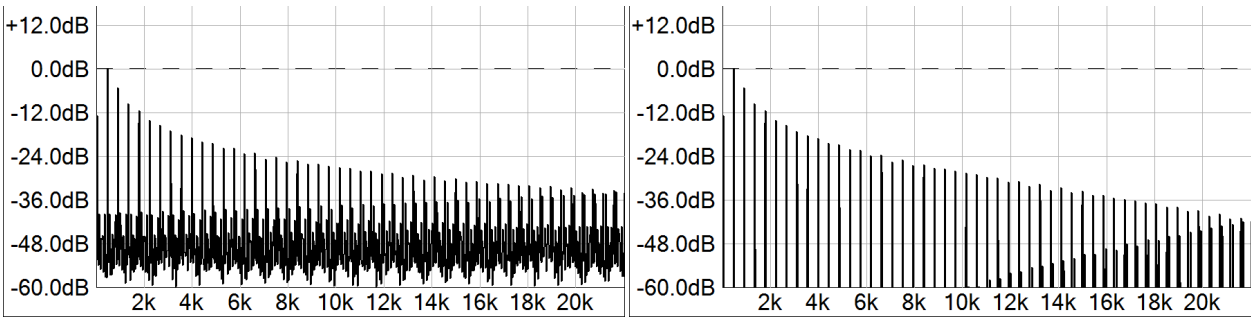


Figure 4: my (left) trivial TB-303 saw, no polyBLEP and (right) trivial TB-303 saw with polyBLEP

Part 2: Square

This is going to be the most challenging and fun component; there are many many many ways to emulate this waveform. I expect many different versions of this - it would be unlikely that any two projects would be identical unless you use the barely acceptable piecewise-linear approximation.

The TB-303 square wave was made by running the sawtooth through a transistor switcher/amp. It is incorrectly referred to as a “waveshaper” on the internet, but we can certainly use waveshaping techniques to mimic it.

1. **Create a trivial version of this waveform**, no polyBLEP

Refer to Figure 1 and transistor Q8; this is a PNP transistor so it will only conduct when the base is 0.6V *lower* than the emitter. The emitter voltage is established with the collector and emitter resistors which set up the bias current. This voltage is about 8V on the TB-303, the CircuitMaker circuit is in Figure 5.

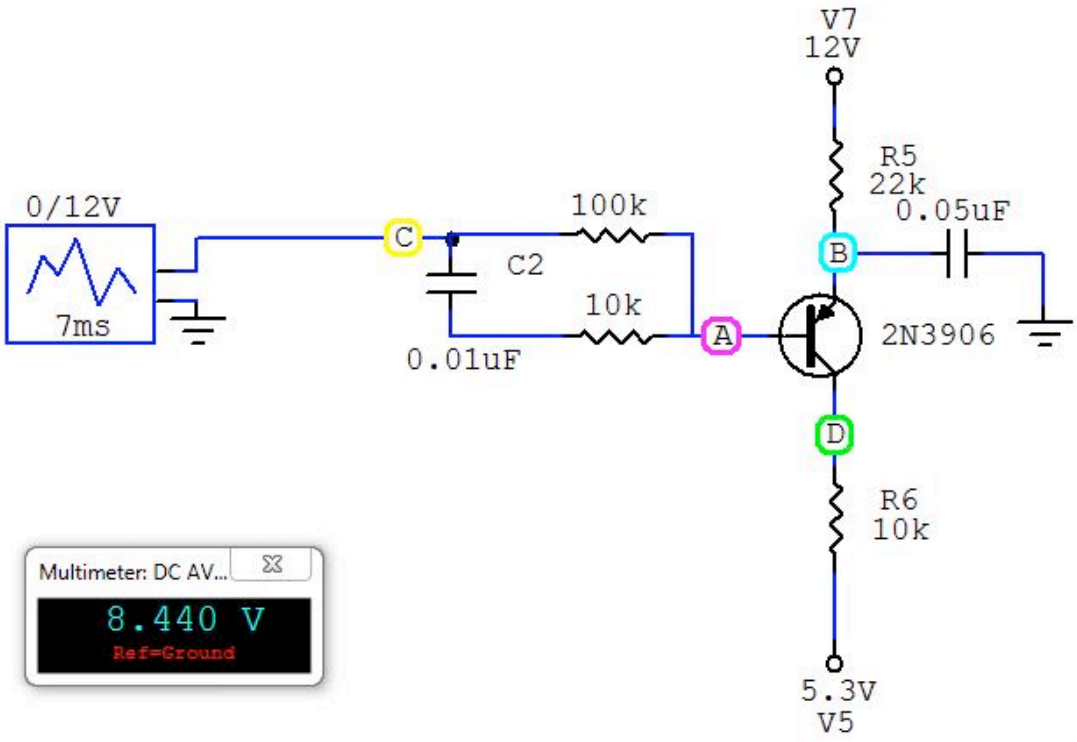


Figure 5: CircuitMaker simulation of the saw to square circuit

Suppose the sawtooth (driving the base) is at a maximum 12V (see Figure 1) then the transistor Q8 is off and the output (collector) sits at its negative rail, 5.3V. The sawtooth drops and the output stays clamped at 5.3V until the base voltage drops 0.6V below the emitter; at this point the transistor conducts and produces an *inverted* version of the input ramp at the collector. I am calling this the **trivial version** of the waveform; you need to generate this waveform without polyBLEP for Part 2.1 of the project. In Figure 6 you can see this as it relates to the sawtooth. Using CircuitMaker I found that the Q8 turn-on point is at the 30% duty cycle point for a 100Hz waveform, corresponding to a modulo counter value of 0.3 and also shown in Figure 6.

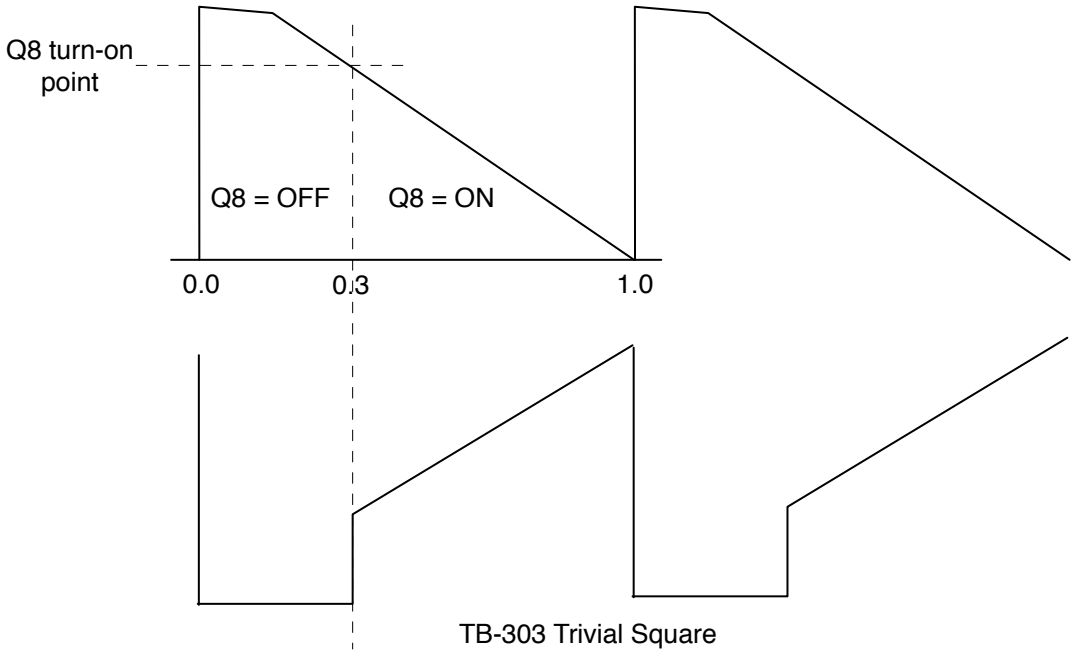


Figure 6: trivial square wave formed by Q8 waveshaper

Figure 7 shows my trivial waveform:

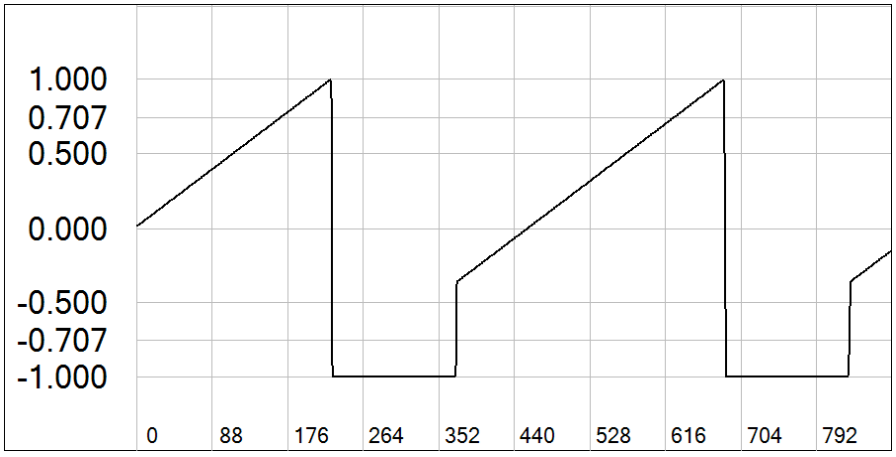


Figure 7: my trivial “square” wave

2. **Mimic the waveform with a custom waveshape of your design and apply polyBLEP to reduce aliasing**; in the template plug-in I will provide, there is a polyBLEP button so you can make sure it works. You will need to implement the function calls correctly. Be careful - remember the rules about polyBLEP - you need to know the discontinuity height (unipolar, I will explain in class). Figure 8 shows the difference in my TB-303 trivial saw with and without polyBLEP.

Figure 6 doesn't tell the whole story behind the waveform. The reason is that as the current increases through the emitter resistor, the voltage on the emitter side drops. When it hits the the same value as the collector voltage, it begins to squash it as shown in Figures 8, 9, 10 and 11 from my CircuitMaker simulation. (Blue is emitter, Green is output)

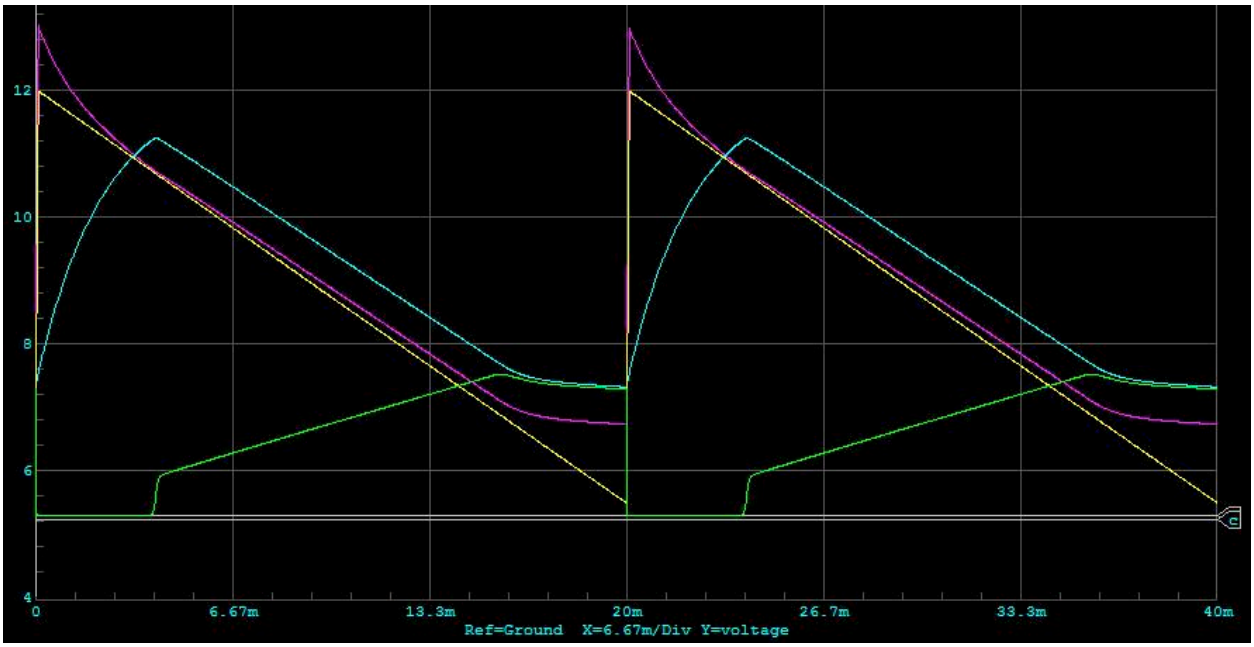


Figure 8: two sawtooth cycles applied to Q8 for f = 50 Hz

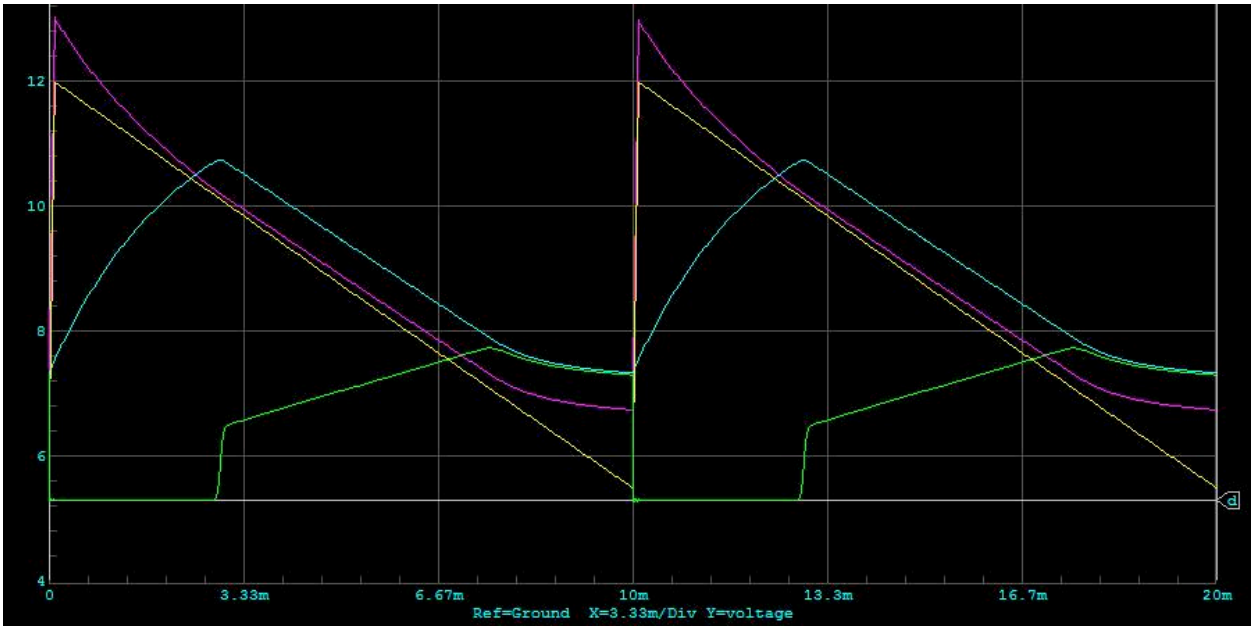


Figure 9: two sawtooth cycles applied to Q8 for f = 100 Hz

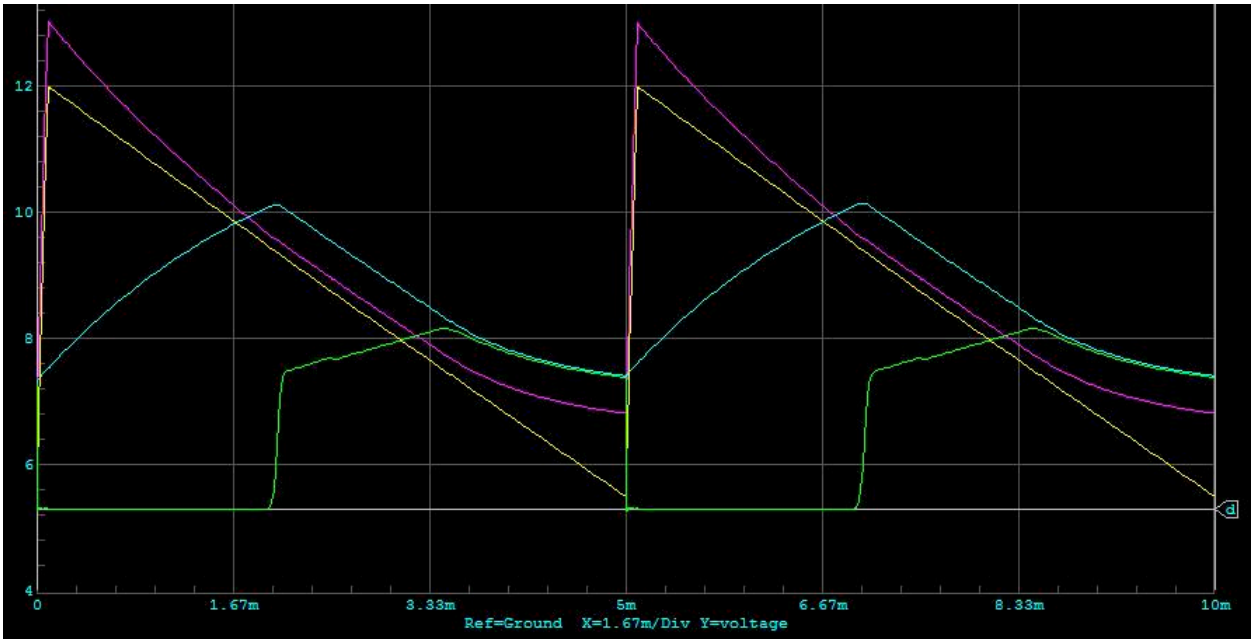


Figure 10: two sawtooth cycles applied to Q8 for f = 200 Hz

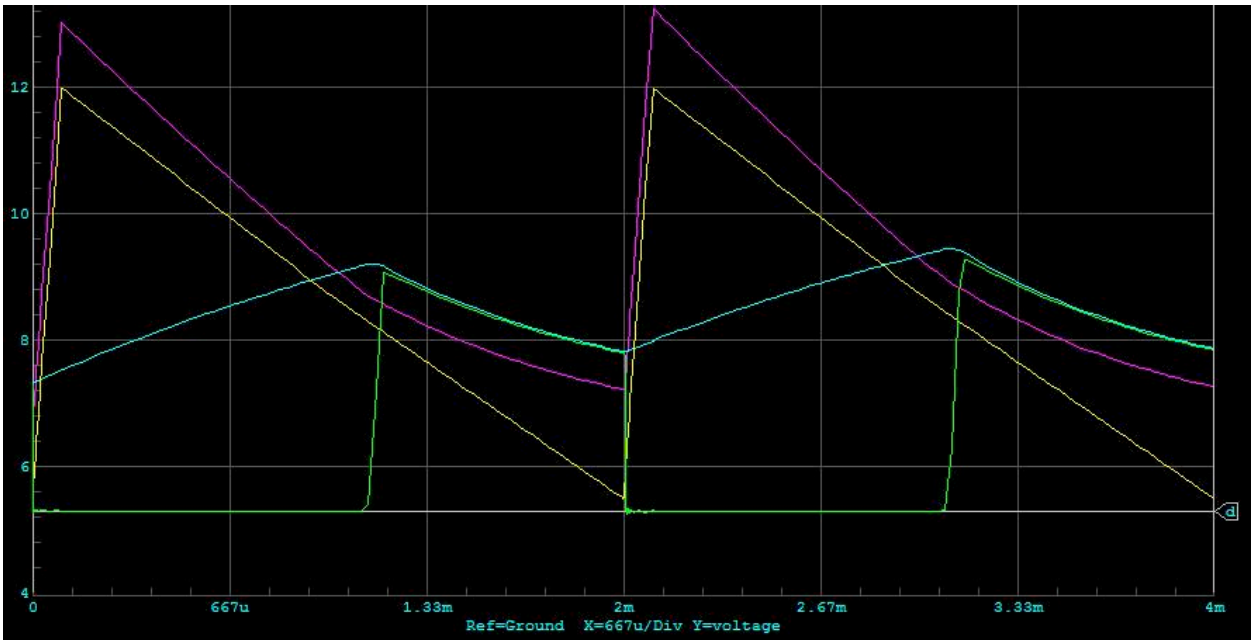


Figure 11: two sawtooth cycles applied to Q8 for f = 500 Hz

In Figures 8, 9, 10, 11 the color codes are:

Yellow: the sawtooth output of the voltage source
Magenta: the output of the quasi-shelving filter formed by the input network (100k,10k, 0.01uF)
Blue: the voltage at the emitter of Q8
Green: the “square wave” output at the collector of Q8

What happens is that the duty cycle is changing with input (sawtooth) frequency. Also, the roof-top shape changes with frequency. This is confirmed by another source I found on the internet (link below).

Also: actual transistors do not turn on instantaneously; there is a soft knee that you might want to also try to model/mimic on the corners during the off-to-on transition.

You do **not** have to implement the change in duty cycle or change in waveshape with change in frequency. However, your model will need to be able to have the duty cycle adjusted. In future projects, you may decide to investigate this further and model it more completely (perhaps for a final project?) but **for this project, use the f = 100Hz plot in Figure 8 to base your calculations on.**

You can also get more plots/info here:

http://forums.adafruit.com/viewtopic.php?f=12&t=17694&start=90

I have posted the **TB-303 Service Manual** and **CircuitMaker simulation** on the Wiki. Mimic this waveform with polyBLEP. Here are four different versions I produced using the 100Hz simulation only and with polyBLEP applied properly shown in Figure 12.

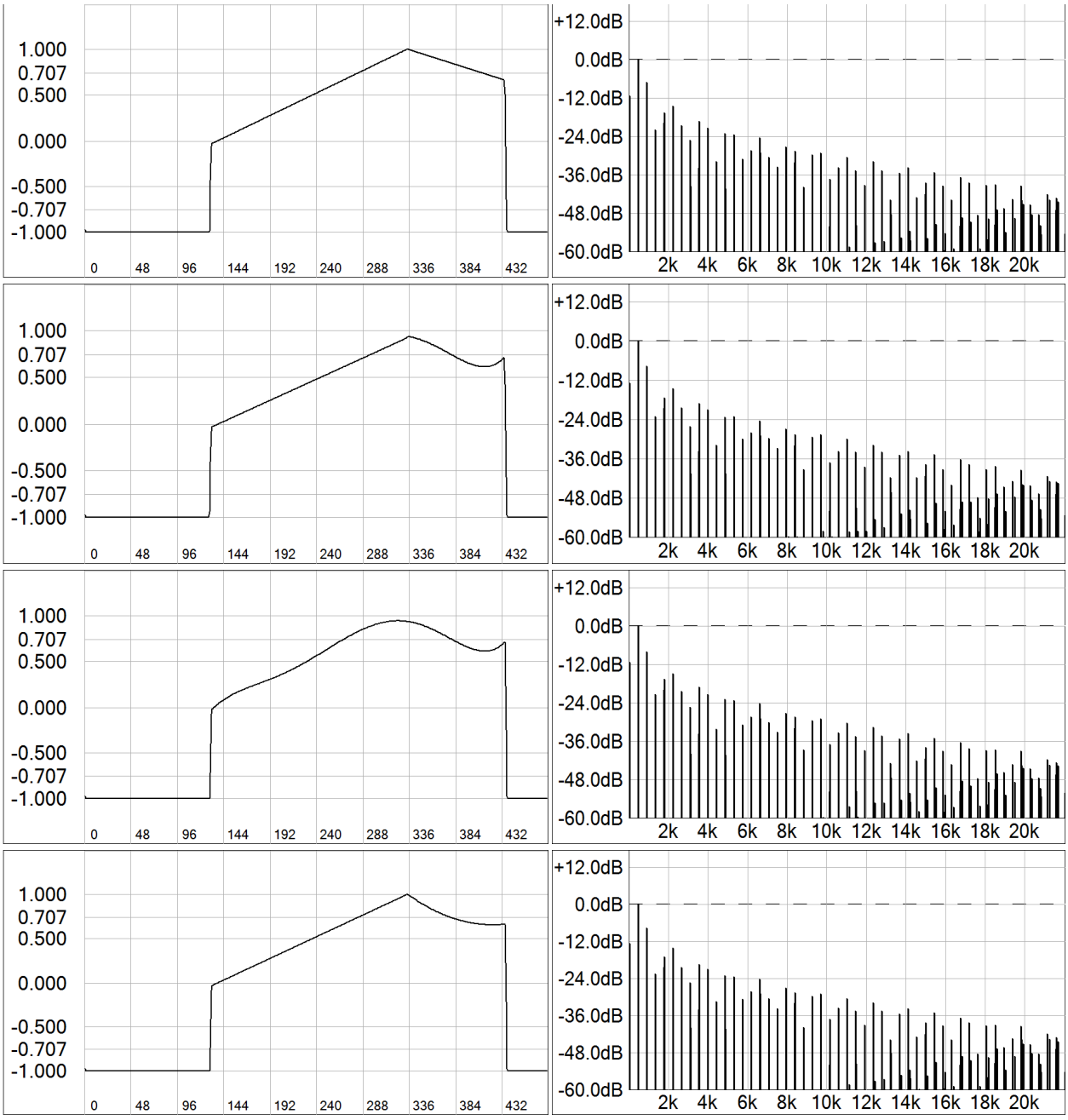


Figure 12: four versions of the TB-303 square waveform (left) 100Hz and (right) spectra at 440Hz

You can see a lot of variation in the waveshapes of my four versions but they all capture the essence of the waveform shape. The last one matches the simulation best, BUT the third one is closer to the actual output I have seen. I am hoping you will produce much better waveforms. For example, none of mine have smooth (sigmoid) transitions on the band corners. And, the top plot (piecewise linear) is ***barely*** acceptable .

I will definitely give you some leeway here but I won't accept something that isn't even close. You will also have to describe for me (and the class) how you implemented the shaper system.

Advice: use Wolfram Alpha or other math sites/tools.

Submit:

1. your plug-in DLL
2. your BaseLine.h and BaseLine.cpp files that has your plug-in code
3. you will also have to present your plug-in in class and explain how you implemented the square wave and suggest improvements to your methods

Extra Credit:
You will need to spend time with CircuitMaker characterizing the waveform over a 3-octave range of frequencies; A0 (27.5Hz) to A3 (220Hz) and please plan on sharing your results! Then, implement the change in waveshape vs. change in frequency (somehow). This EC will add 25% to your score on this project. That means I'll be picky about your waveforms!