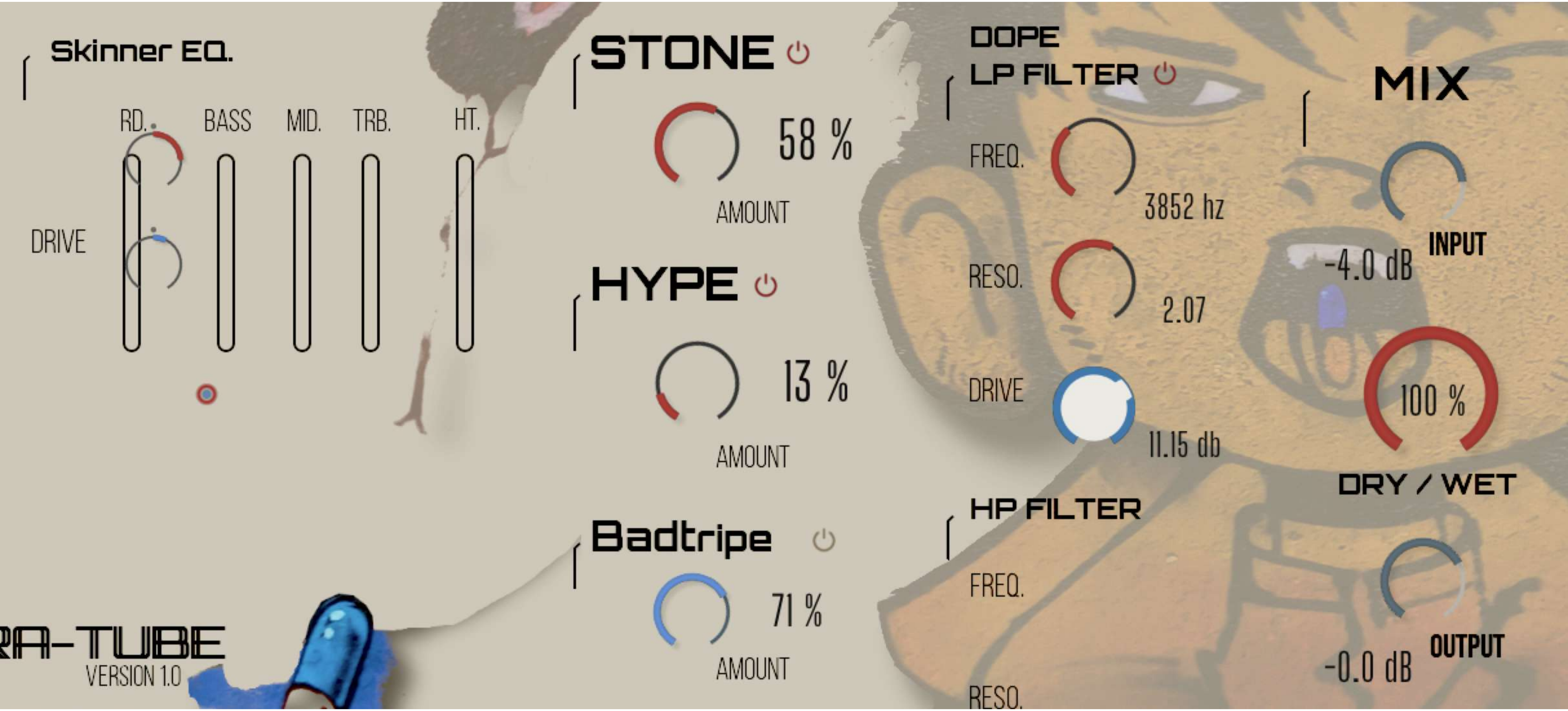
Moog ladder filter

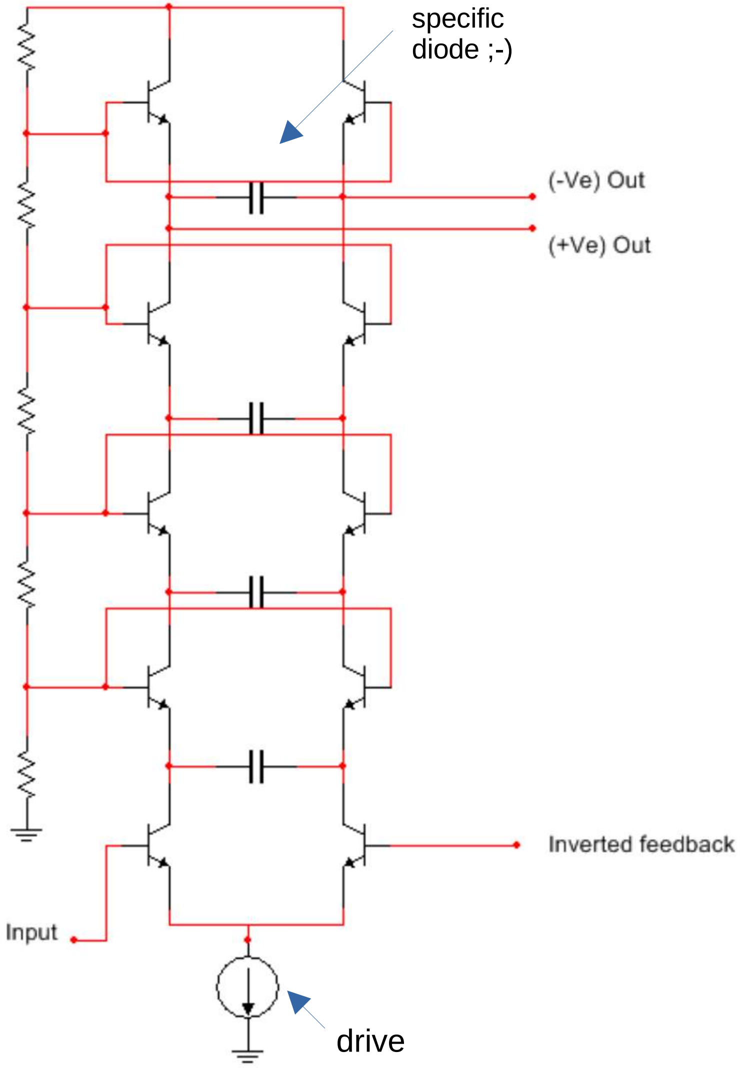
Low Pass Filter explained



Skinner EQ.
RD.
BASS
MID.
TRB.
HT.
DRIVE
STONE
58 %
AMOUNT
HYPE
13 %
AMOUNT
Badtripe
71 %
AMOUNT
RA-TUBE
VERSION 1.0
DOPE
LP FILTER
FREQ.
3852 hz
RESO.
2.07
DRIVE
11.15 db
HP FILTER
FREQ.
RESO.
MIX
INPUT
-4.0 dB
100 %
DRY / WET
OUTPUT
-0.0 dB

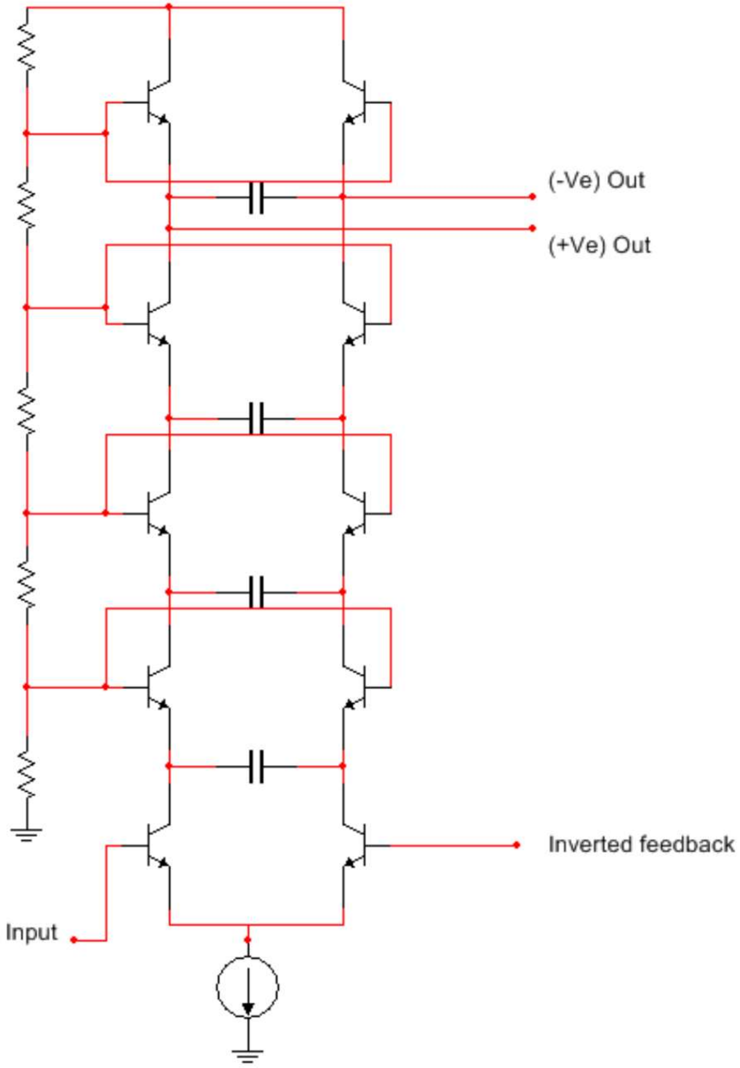
Moog Filter …

- In 1965 Dr. Robert Moog presented a voltage controlled low pass filter
- first of its kind and would go on to become one of the most acclaimed and influential tools in electronic music



Low pass filter

- The core of the Moog VCF is a **driver**; a **transistor** “ladder” of four identical, **buffered stages**
- a variable **gain feedback** path
- Stages ?
  - differential pair of NPN-transistors with a capacitor placed in between them
  - **four one-pole, voltage controlled, low-pass RC filters !**



Low pass filter

- An input signal enters at the bottom of the ladder, passes through each stage (**filter**),
- Each stage causes 3 dB of attenuation
  - complete core causes 12 dB of attenuation
- The output is split between the result .. and an inverting feedback path which leads back to the first stage

Antti Huovilainen method

- In 2004, Antti Huovilainen analysed the Moog VCF with the intent of capturing the inherent non-linearities present in the circuitry and implementing an accurate, **non-linear digital model of the filter.**
- this model is built around **calculating four differential equations** using a forward difference method

Let’s analyse the electronic circuit

- Each filter stage is dependent only upon its current state and the current from the preceding stage.

$$I_1 - I_2 = (I_1 + I_2)\tanh\left(\frac{V_1 - V_2}{2V_t}\right)$$

Vt =28,5mV is the moog transistor implementation

Let's analyse the electronic circuit

$$I_1 + I_2 = I_{ctl}$$

$$I_1 - I_2 = I_s - 2I_c$$

$$2I_c = I_s - I_{ctl} \tanh\left(\frac{V_c}{2V_t}\right)$$

$$I_c = C\frac{\mathrm{d}V_c}{\mathrm{d}t}$$

$$2C\frac{\mathrm{d}V_c}{\mathrm{d}t} = I_s - I_{ctl} \tanh\left(\frac{V_c}{2V_t}\right)$$

Vt =28,5mV is the moog transistor implementaion

Let's analyse the electronic circuit

As each stage is driven by the preceding one, it is possible to replace signal current Is to create a general relation

$$\frac{\mathrm{d}V_c}{\mathrm{d}t} = \frac{I_{ctl}}{2C}\left(\tanh\left(\frac{V_{in}}{2V_t}\right) - \tanh\left(\frac{V_c}{2V_t}\right)\right)$$

Antti Huovilainen method

Euler's introduced a method (that Huovilainen's will improve) -> the result is equivalent to a **cascade of first order IIR filters** with embedded non-linear **tanh functions…**

He solves the differential equation at time step using Euler's method !

$$v_i^n = v_i^{n-1} + \frac{\omega_c}{F_s}(\tanh v_{i-1}^n - \tanh v_i^{n-1})$$

Antti Huovilainen method

He noticed that at small signal levels, where the **tanh** function is approximately linear, the above difference equation is that of a digital **one pole low-pass filter**

$$y^n = y^{n-1} + g(x^n - y^{n-1})$$

$$g = 1 - e^{-\omega_c/F_s}$$

Antti Huovilainen method

$$y_a^n = y_a^{n-1} + g(x^n - ky_d^{n-1} - y_a^{n-1})$$
$$y_b^n = y_b^{n-1} + g(y_a^n - y_b^{n-1})$$
$$y_c^n = y_c^{n-1} + g(y_b^n - y_c^{n-1})$$
$$y_d^n = y_d^{n-1} + g(y_c^n - y_d^{n-1})$$

where $g = 1 - e^{-\omega_{c,a}/Fs}$, $\omega_{c,a} = \frac{2}{T_s}\tan(T_s\omega_{c,d}/2)$ is the "analogue" cut-off frequency and $\omega_{c,d}$ is the "digital" cut-off frequency set by the user. The subscript notation a, b, c, d refers to the output of the first, second, third and fourth stages of the ladder rising from the bottom. It is essential that the calculations are carried out in the order that they are presented above. This implementation of Huovilainen's model is referred to as the "unit-delay model".

Code example ?

```
in = tanh(inputsample * drive);
y_a = y_a + g * (tanh(in) - resonance * (y_d_1 - tanh(y_a)));
y_b = y_b + g * (tanh(y_a) - tanh(y_b));
y_c = y_c + g * (tanh(y_b) - tanh(y_c));
y_d_1 = y_d;
y_d = y_d + g * (tanh(y_c) - tanh(y_d));
out = y_d;  // Impulse Filter response
```

Where g = 1 - polyexp(-2 * polytan(2 * PIINV * CUTOFF/sampleRate))) AND **k represent the resonance** :-)

Method can be improved...

Antti proposes that a “unit- and-a-half-delay” feedback signal, that is the **average of two previous output values**, will improve the model’s frequency response.

Oversampling is necessary

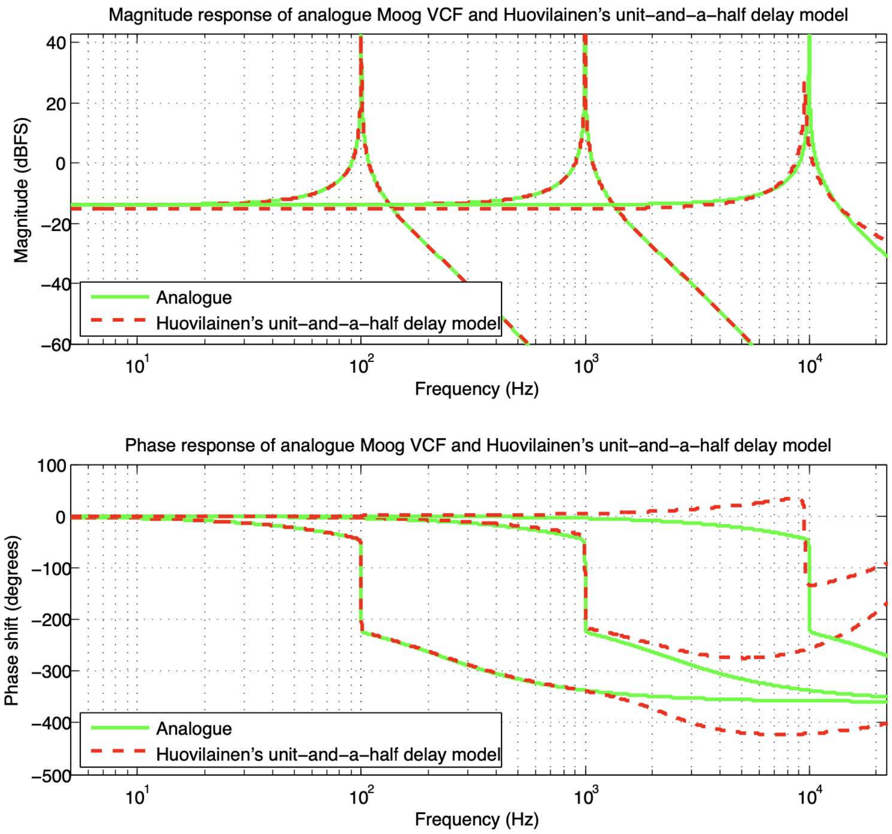
Code example ?

for i = .. to Oversampling ...

```
in = tanh(inputsample * drive);
y_a = y_a + g * (tanh(in) - resonance * ((y_d_1 + y_d)/2) - tanh(y_a)));
y_b = y_b + g * (tanh(y_a) - tanh(y_b));
y_c = y_c + g * (tanh(y_b) - tanh(y_c));
y_d_1 = y_d;
y_d = y_d + g * (tanh(y_c) - tanh(y_d));
out = y_d;  // Impulse Filter response
```

where g = 1 - polyexp(-2 * polytan(2 * PIINV * CUTOFF/sampleRate))) AND **k represent the resonance** :-)

Results


Magnitude response of analogue Moog VCF and Huovilainen's unit–and–a–half delay model
Magnitude (dBFS)
40
20
0
−20
−40
−60
Analogue
Huovilainen's unit–and–a–half delay model
10^1
10^2
10^3
10^4
Frequency (Hz)
Phase response of analogue Moog VCF and Huovilainen's unit–and–a–half delay model
Phase shift (degrees)
100
0
−100
−200
−300
−400
−500
Analogue
Huovilainen's unit–and–a–half delay model
10^1
10^2
10^3
10^4
Frequency (Hz)

Good & bad things ?

CPU friendly for an excellent result !

Code can be easily improved

But … **issue #1** of the **DB drop**

And … **issue #2** with **Q variation** hype when above 18khz…

You have much better simulation (backwards difference method for instance BUT **quid of cpu usage** & understanding the formula to code them !)

Future Works

Solve the issues #1 and #2 in a simple and naive manner ?

Create a high pass filter ?

- using spectral inversion method ?
- HPF = Original signal – Low pass Signal ?
- …

References

A comparison of virtual analogue Moog VCF models

By Paul Daly

http://www.acoustics.ed.ac.uk/wp-content/uploads/AMT_MSc_FinalProjects/2012__Daly__AMT_MSc_FinalProject_MoogVCF.pdf

+

Analysis of the Moog Transistor Ladder and Derivative Filters

Timothy E. Stinchcombe † 25 Oct 2008

+

https://www.allaboutcircuits.com/technical-articles/analyzing-the-moog-filter/

+

Code sources : https://github.com/dshr/ + https://github.com/dshr/tinySynth