AUDIO MOSTLY 2023 WORKSHOP ON SUPERCOLLIDER

SUBTRACTIVE SYNTHESIS

NIKLAS RÖNNBERG

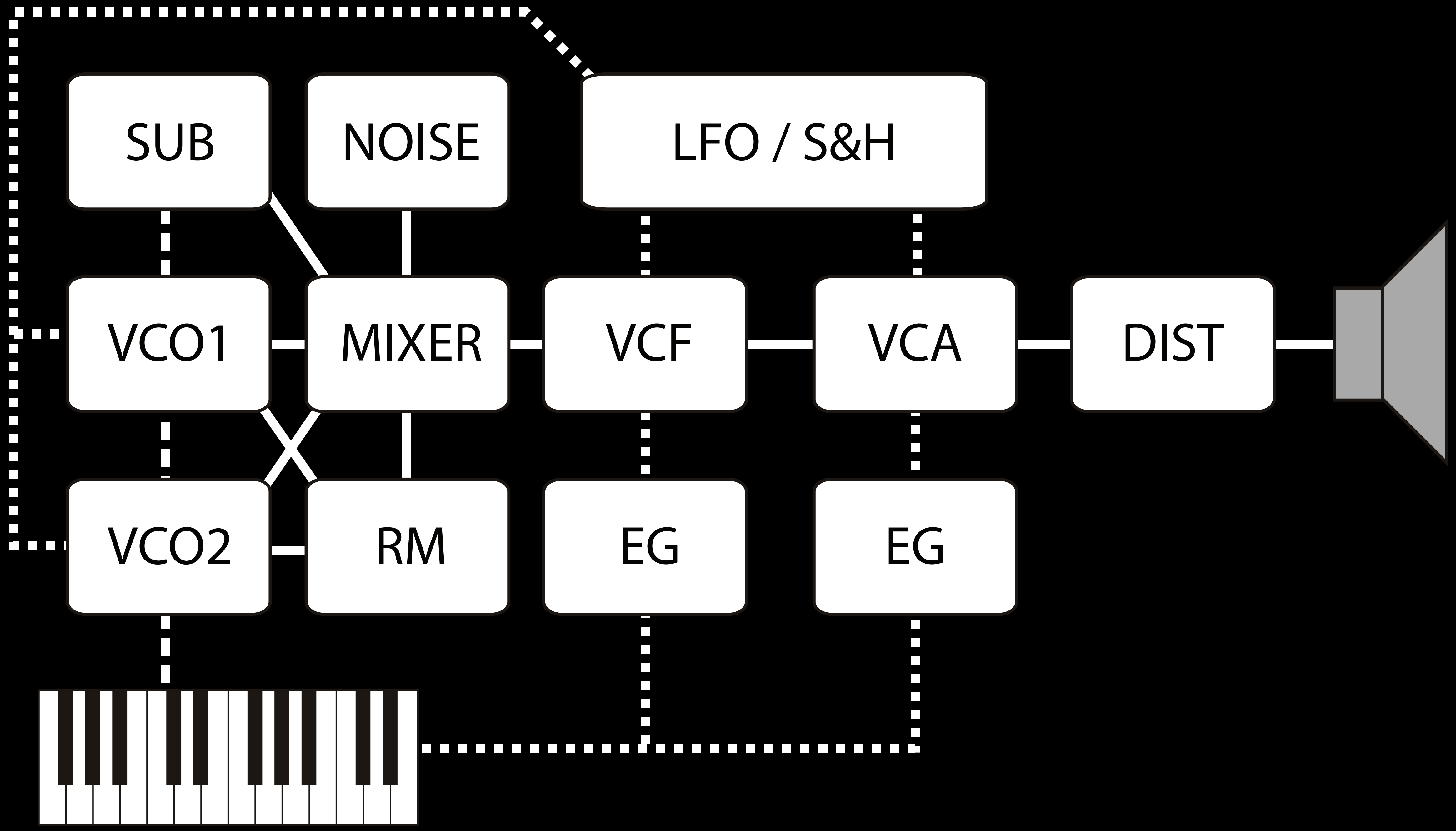
LINKÖPING UNIVERSITY
niklas.ronnberg@liu.se

SUBTRACTIVE SYNTHESIS

- Easier to make with analogue electronics compared to additive synthesis.

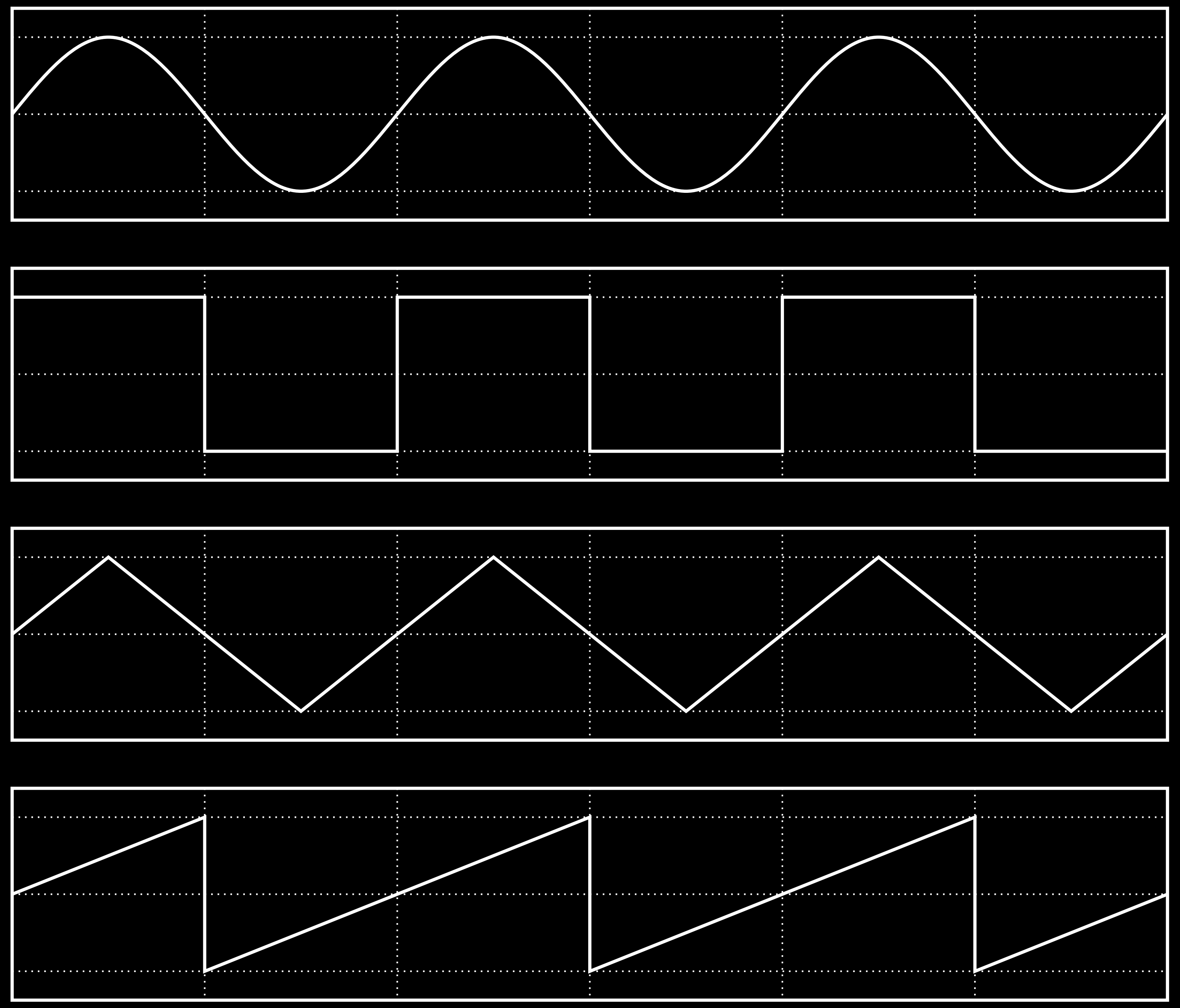
- The basic idea is to take a quite complex sound wave and then remove harmonics to get the desired sound.

- An analog synth consists of several modules (VCO, VCF, VCA, LFO)



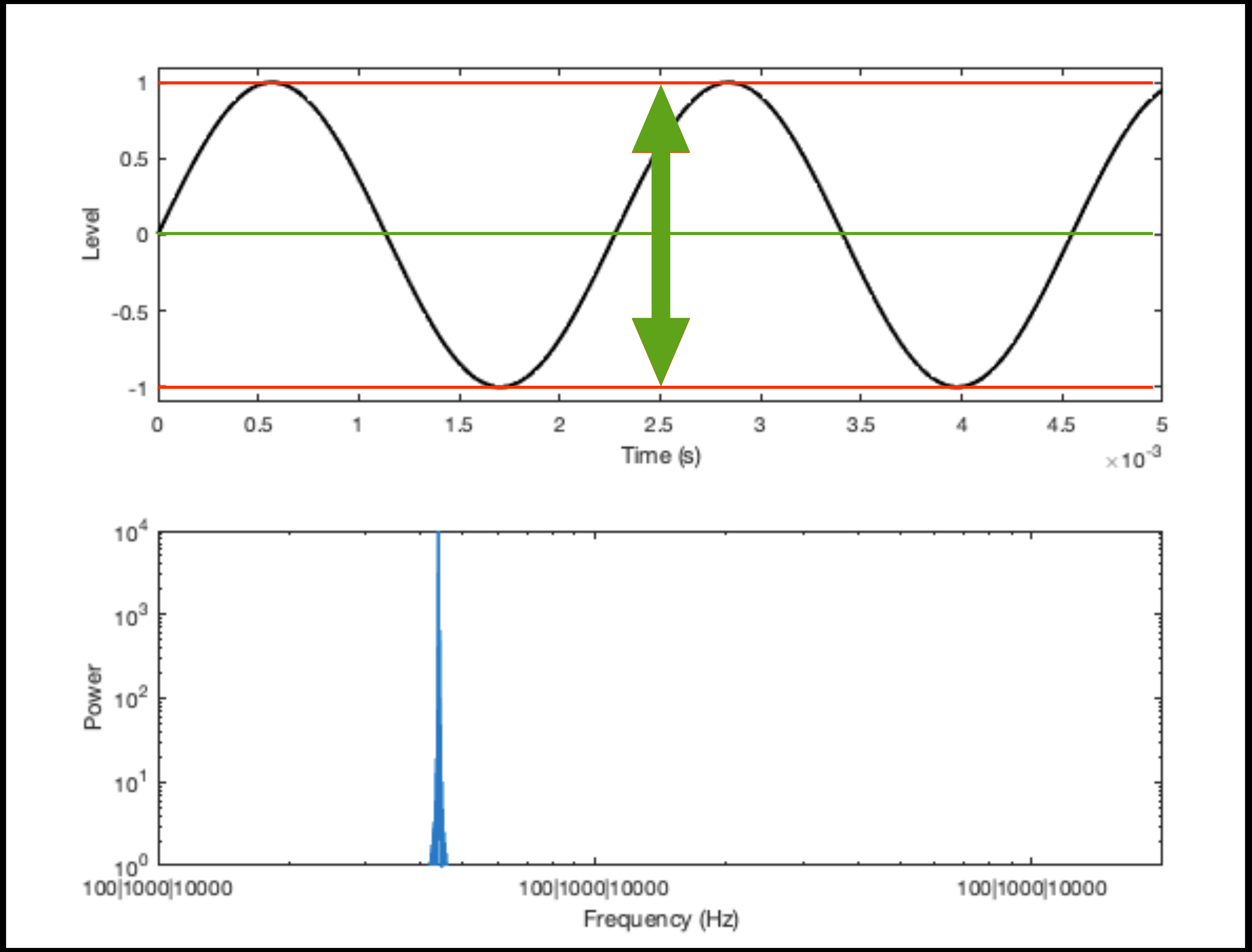
BASIC WAVEFORMS

- Sine wave
- Square (pulse) wave
- Triangle wave
- Sawtooth (ramp) wave
- Noise



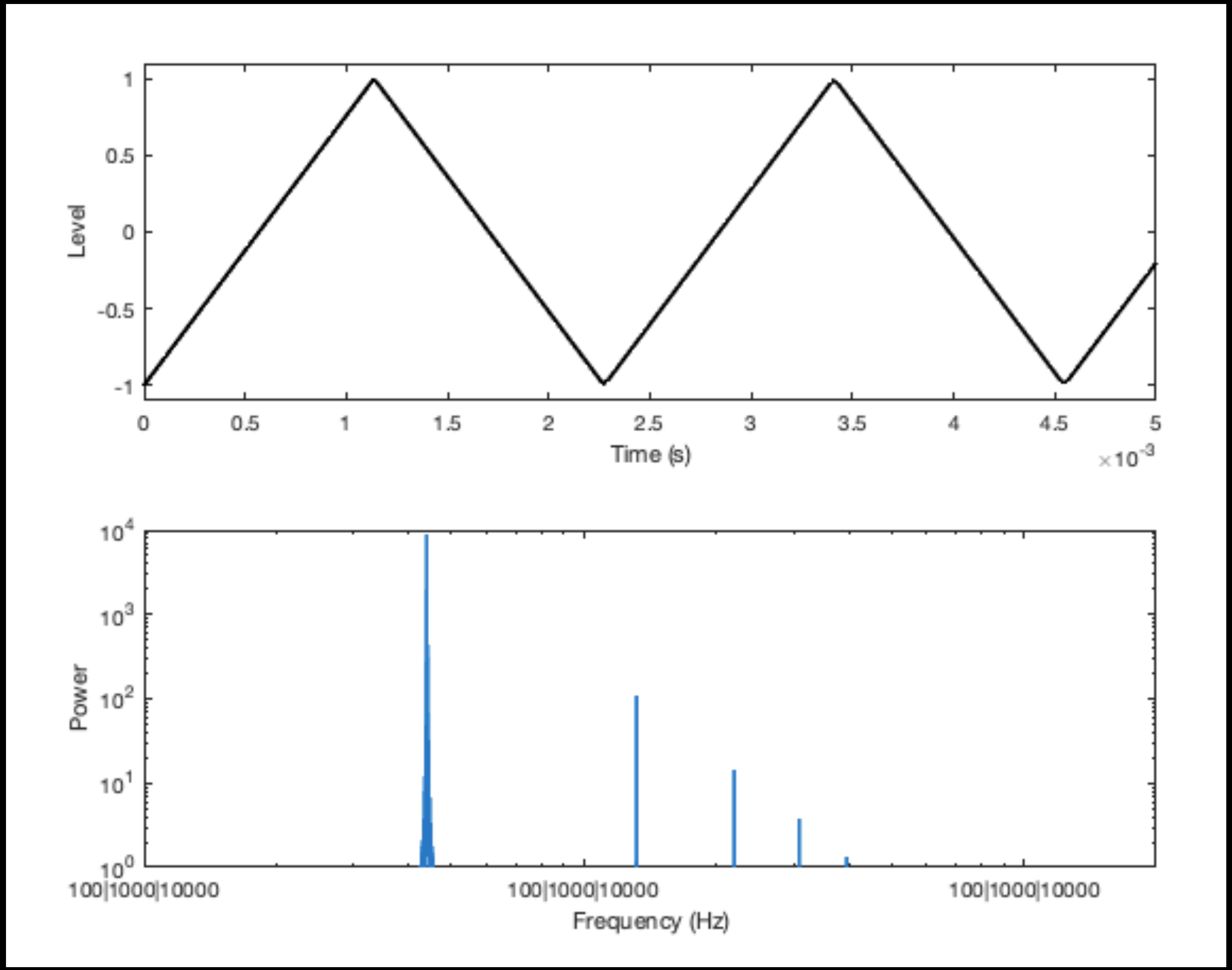
SINE WAVE

- No overtones or harmonics
- SinOsc.ar(freq: frequency, phase: 0, mul: 1.0, add: 0)
- freq = the frequency in Hz
- phase = the phase of the wave form at start
- mul = multiplication of the waveform, i.e., the sound level
- add = the offset of the waveform



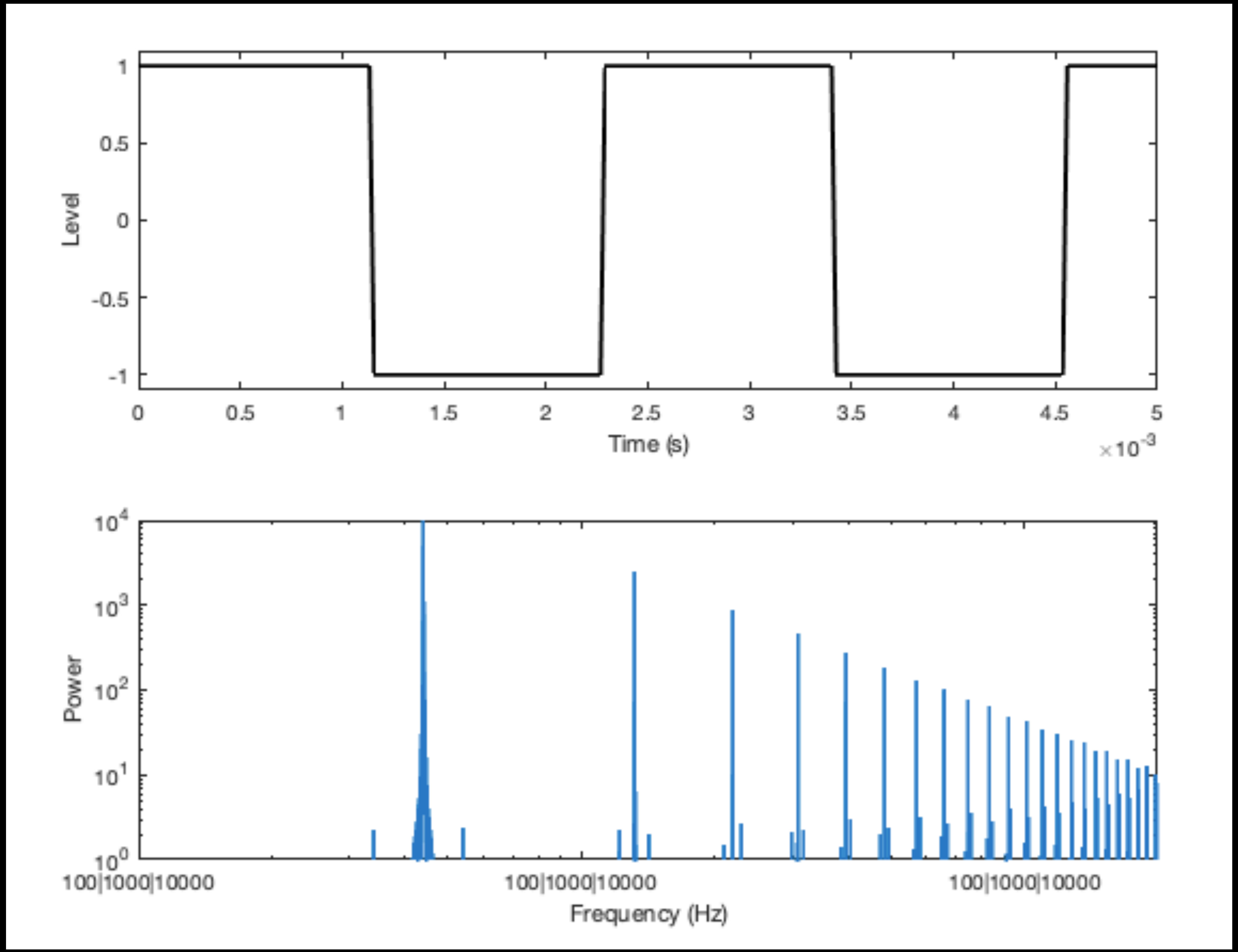
TRIANGLE WAVE

- Odd harmonics with quite steep rolloff
- LFTri.ar(freq: frequency, iphase: 0, mul: 0.3, add: 0);
- freq = the frequency in Hz
- iphase = the phase of the wave form at start
- mul = multiplication of the waveform, i.e., the sound level
- add = the offset of the waveform



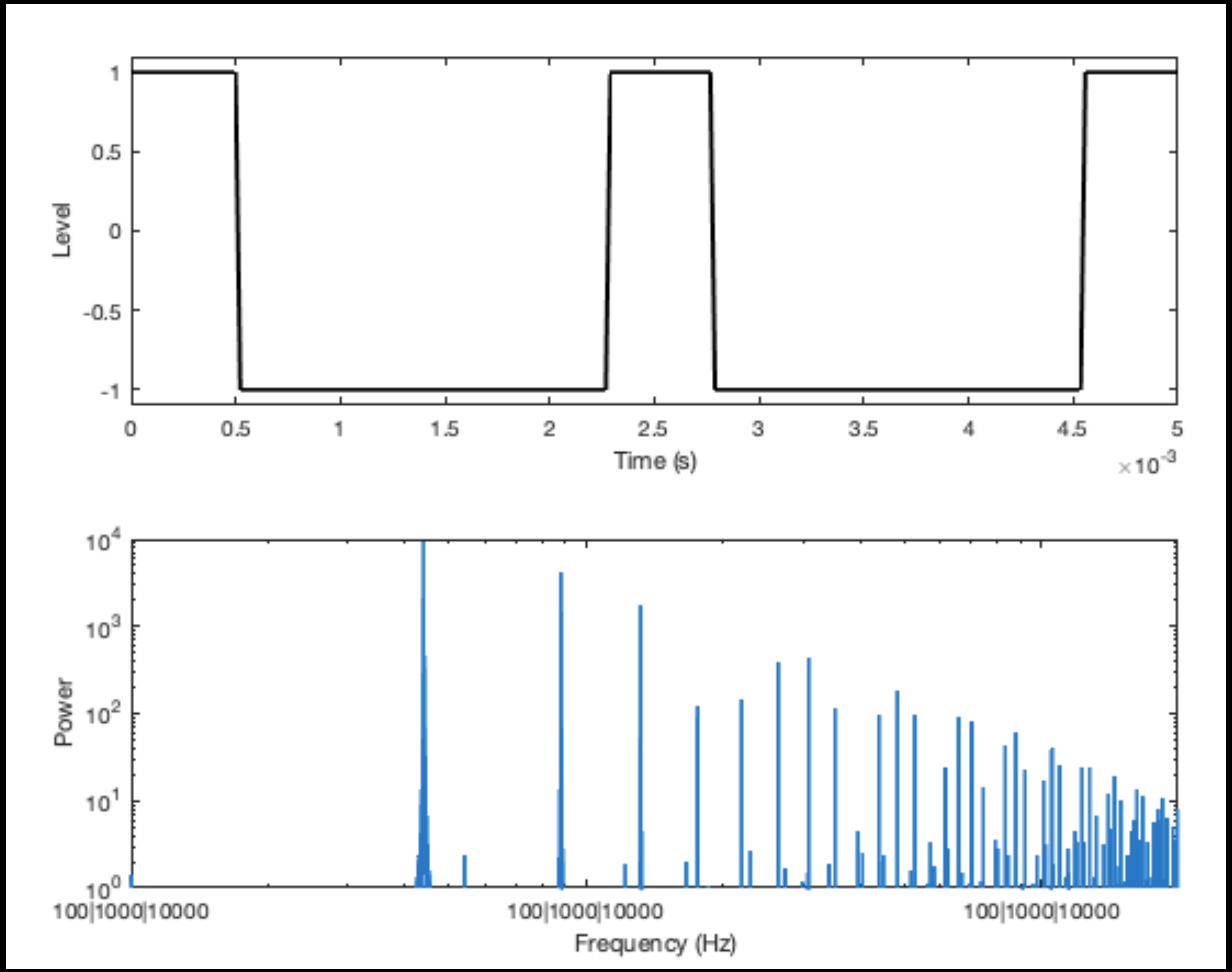
SQUARE WAVE

- Odd harmonics with less steep rolloff
- LFPulse.ar(freq: frequency, iphase: 0, width: 0.5, mul: 0.3);
- freq = the frequency in Hz
- iphase = the phase of the wave form at start
- width = the pulse width
- mul = multiplication of the waveform, i.e., the sound level
- add = the offset of the waveform



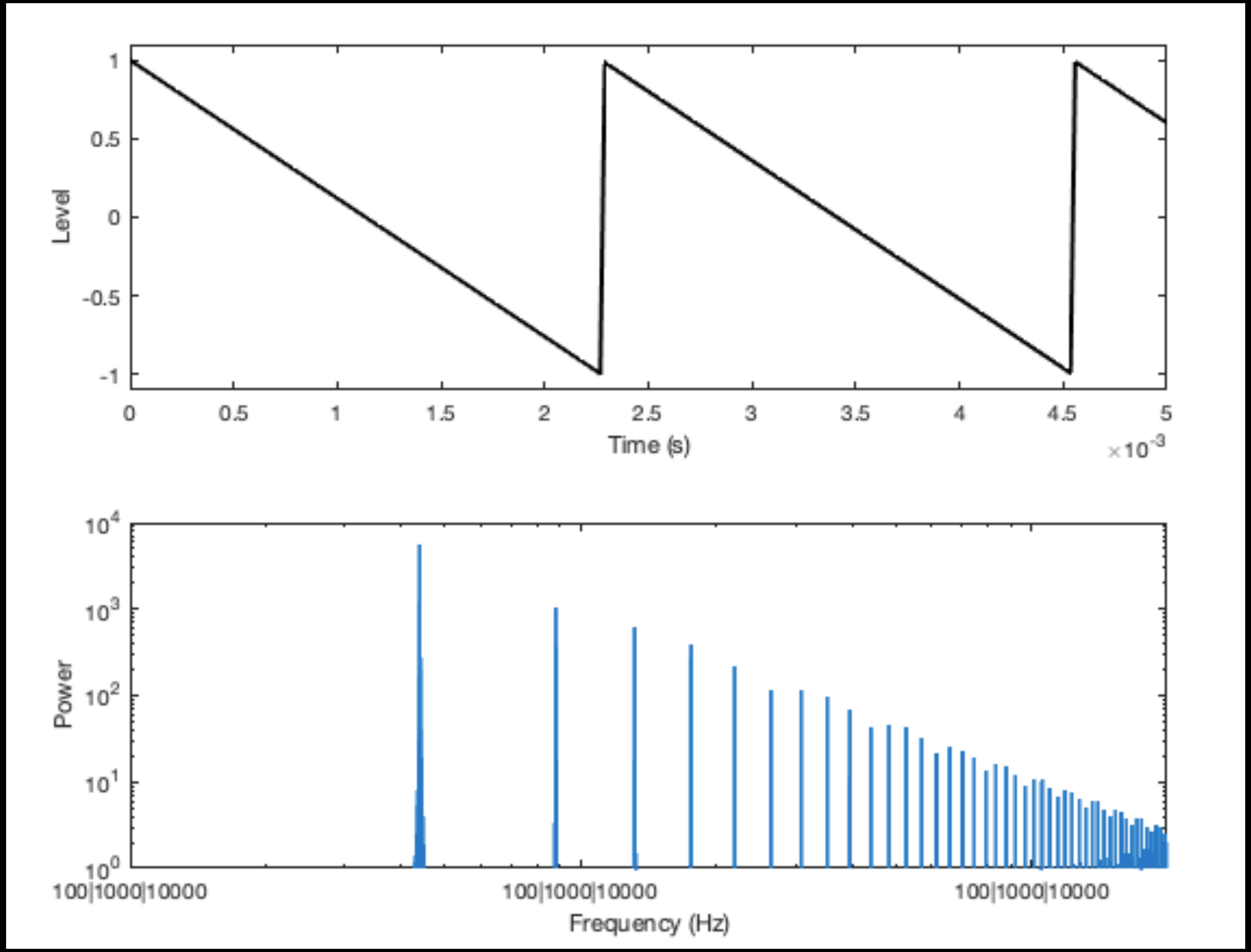
PULSE WAVE

- Changes to the pulse width add plenty of harmonics
- LFPulse.ar(freq: frequency, iphase: 0, width: 0.2, mul: 0.3);
- width = the pulse width, 20% in this example
- 20% and 80% sounds the same to the human ear



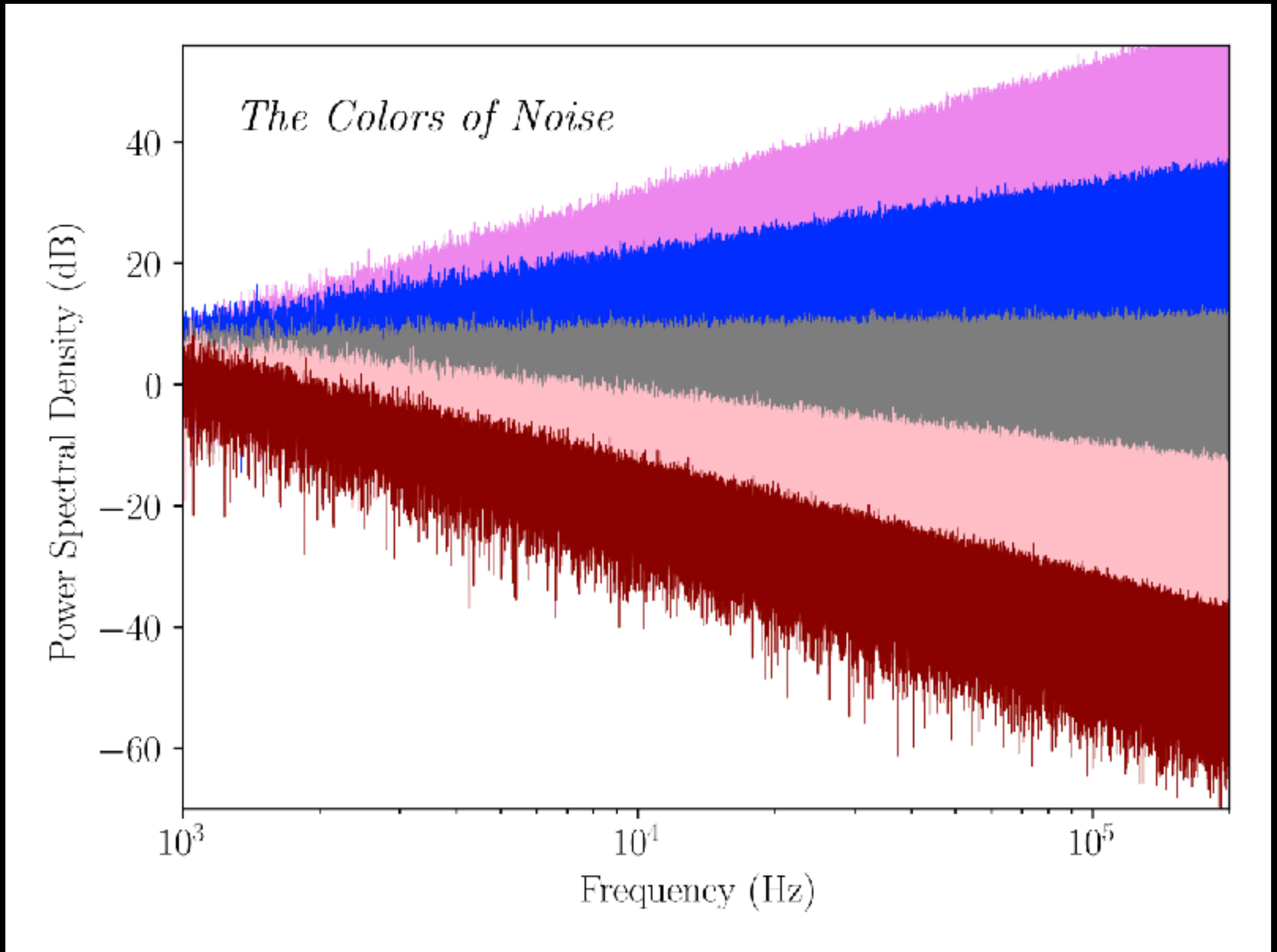
SAWTOOTH WAVE

- Odd and even harmonics harmonics
- LFSaw.ar(freq: frequency, iphase: 0, mul: 0.3, add: 0);
- freq = the frequency in Hz
- iphase = the phase of the wave form at start
- mul = multiplication of the waveform, i.e., the sound level
- add = the offset of the waveform
- Rising or falling sounds the same to the human ear



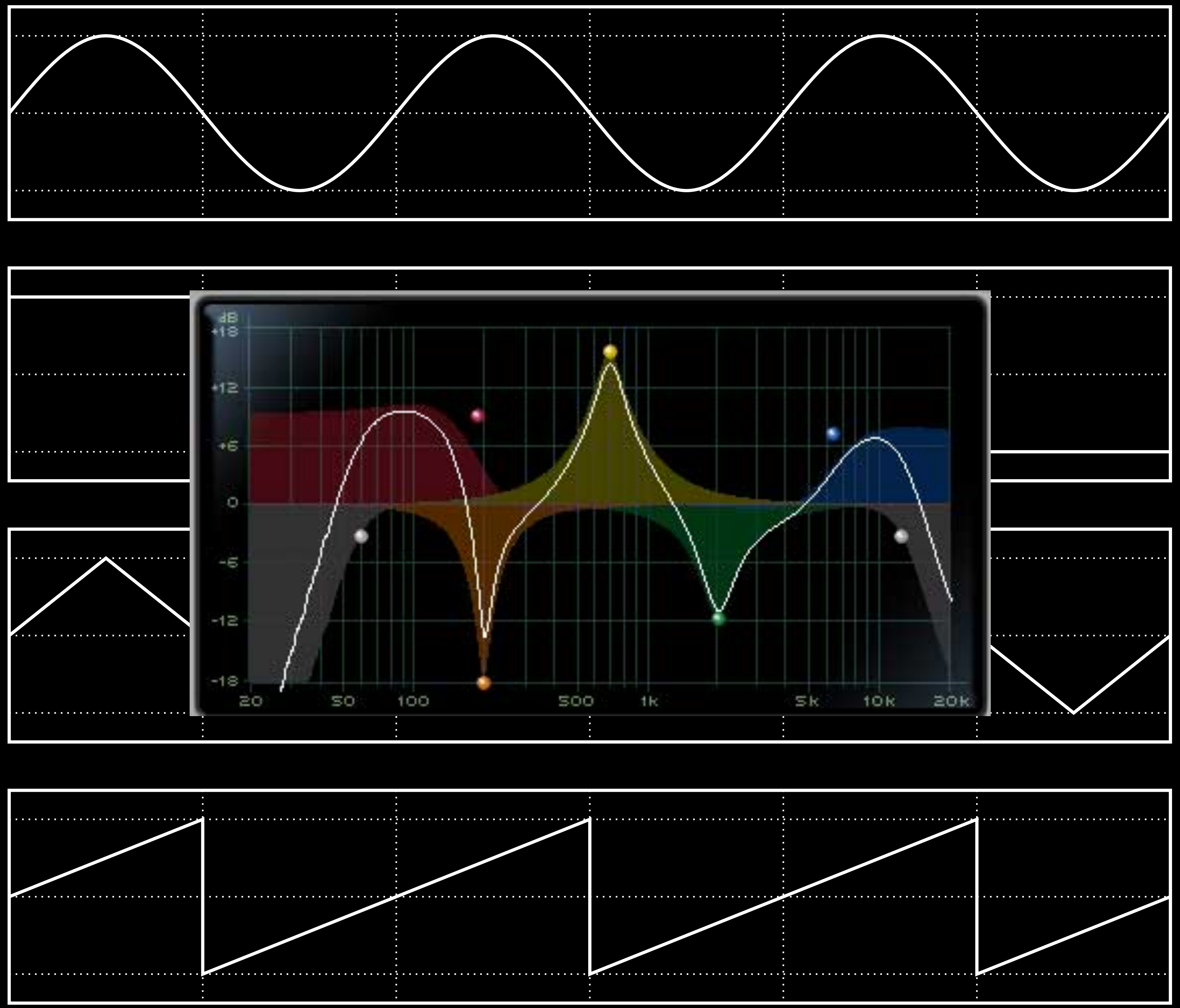
NOISE

- Random frequencies
- Different colors of the noise contains different frequency ranges, but
- https://en.wikipedia.org/wiki/Colors_of_noise



ADJUSTING THE HARMONICS

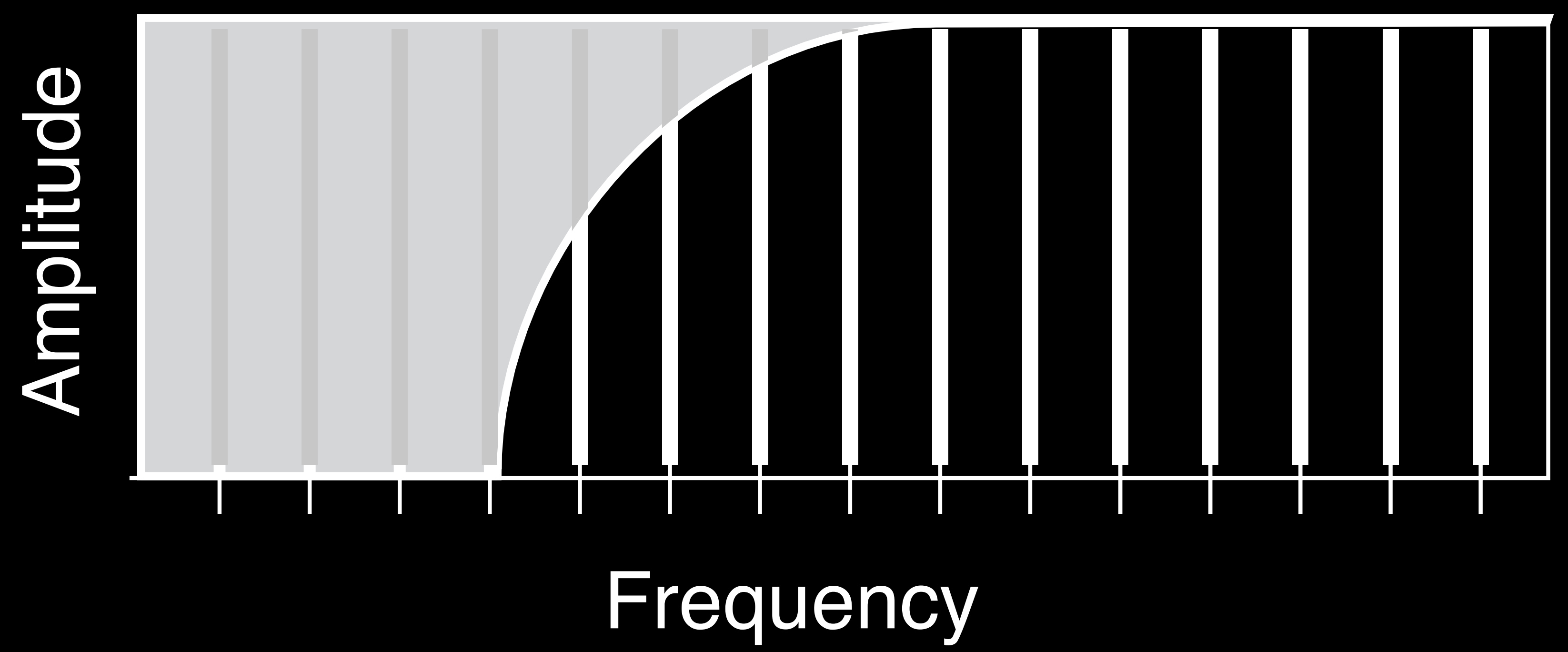
- Filters
- Highpass filter
- Bandpass filter
- Band reject filter
- Lowpass filter
- Shelving filter



HIGHPASS FILTER

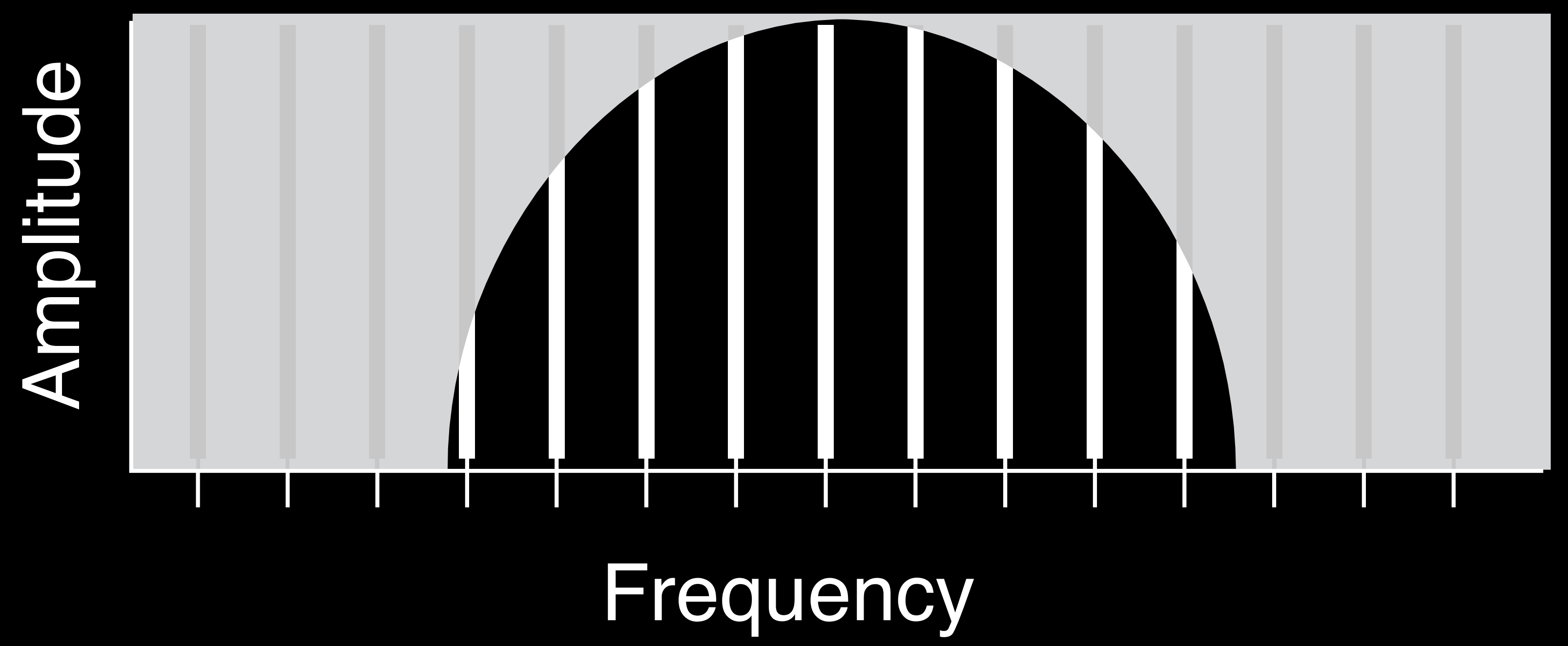
- Attenuates low frequencies
- HPF.ar(in: sound, freq: frequency, mul: 1, add: 0)
- in = the sound input
- freq = the cutoff frequency of the filter
- mul = multiplication of the filter output
- add = the offset of the sound

- In SuperCollider do not use 0Hz as cutoff frequency



BANDPASS FILTER

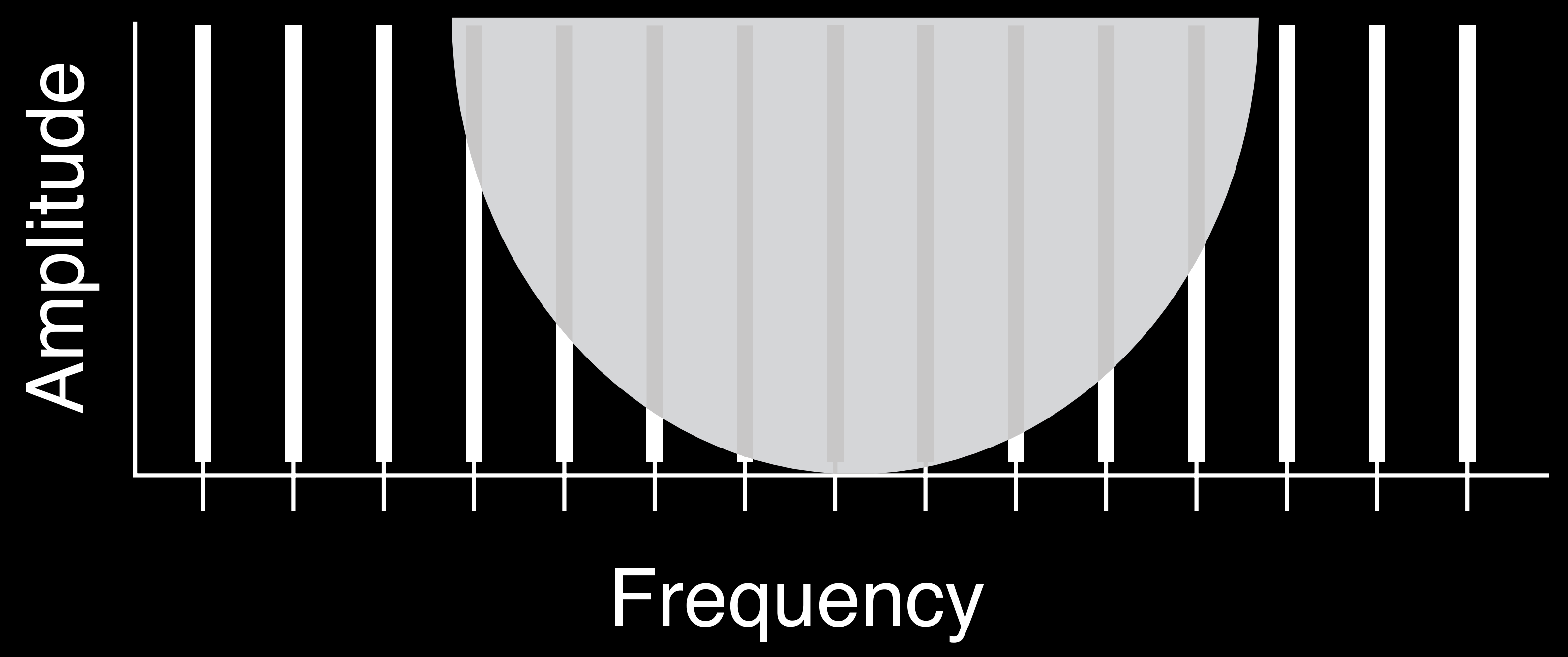
- Attenuates low and high frequencies
- BPF.ar(in: sound, freq: frequency, rq: 1, mul: 1, add: 0)
- in = the sound input
- freq = the cutoff frequency of the filter
- rq = the q (or resonance) of the filter
- mul = multiplication of the filter output
- add = the offset of the sound



- In SuperCollider do not use 0Hz as cutoff frequency

BAND REJECT FILTER

- Attenuates mid frequencies
- BRF.ar(in: sound, freq: frequency, rq: 1, mul: 1, add: 0)
- in = the sound input
- freq = the cutoff frequency of the filter
- rq = the q (or resonance) of the filter
- mul = multiplication of the filter output
- add = the offset of the sound

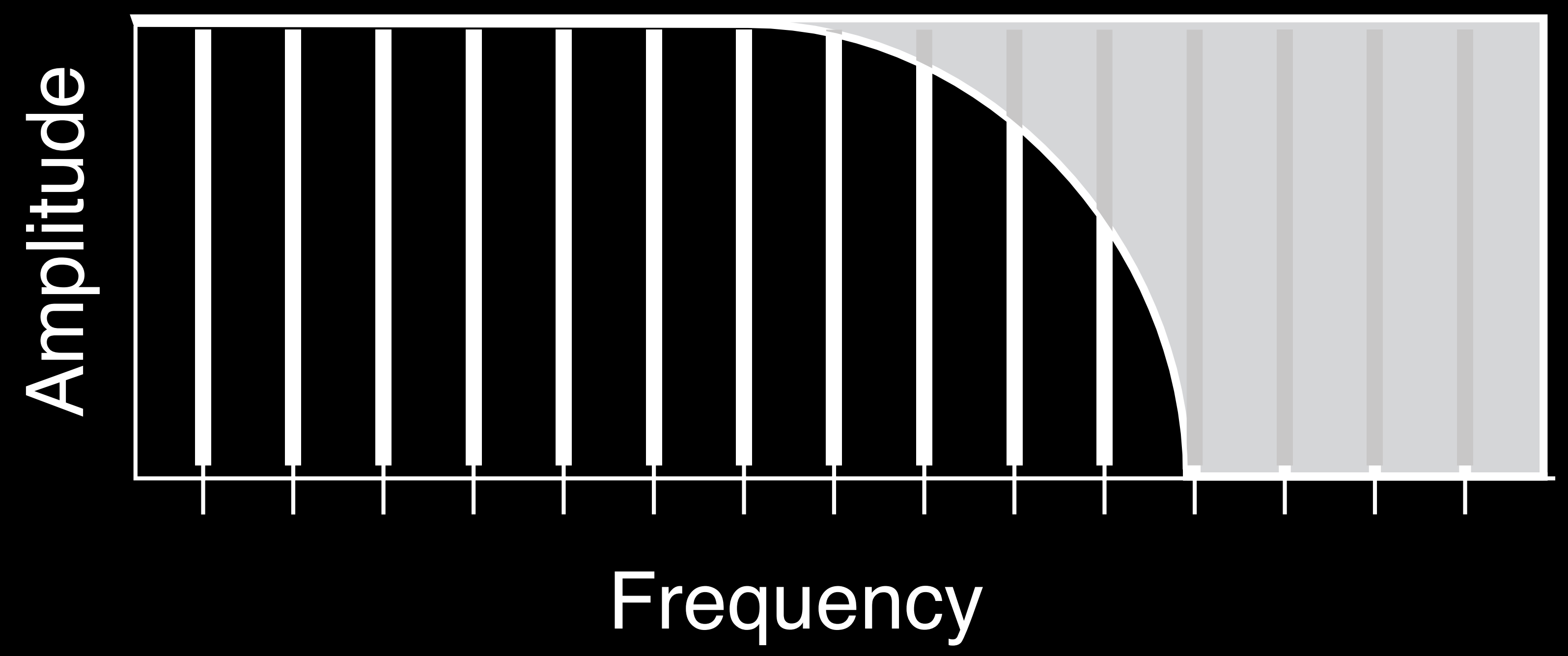


- In SuperCollider do not use 0Hz as cutoff frequency

LOWPASS FILTER

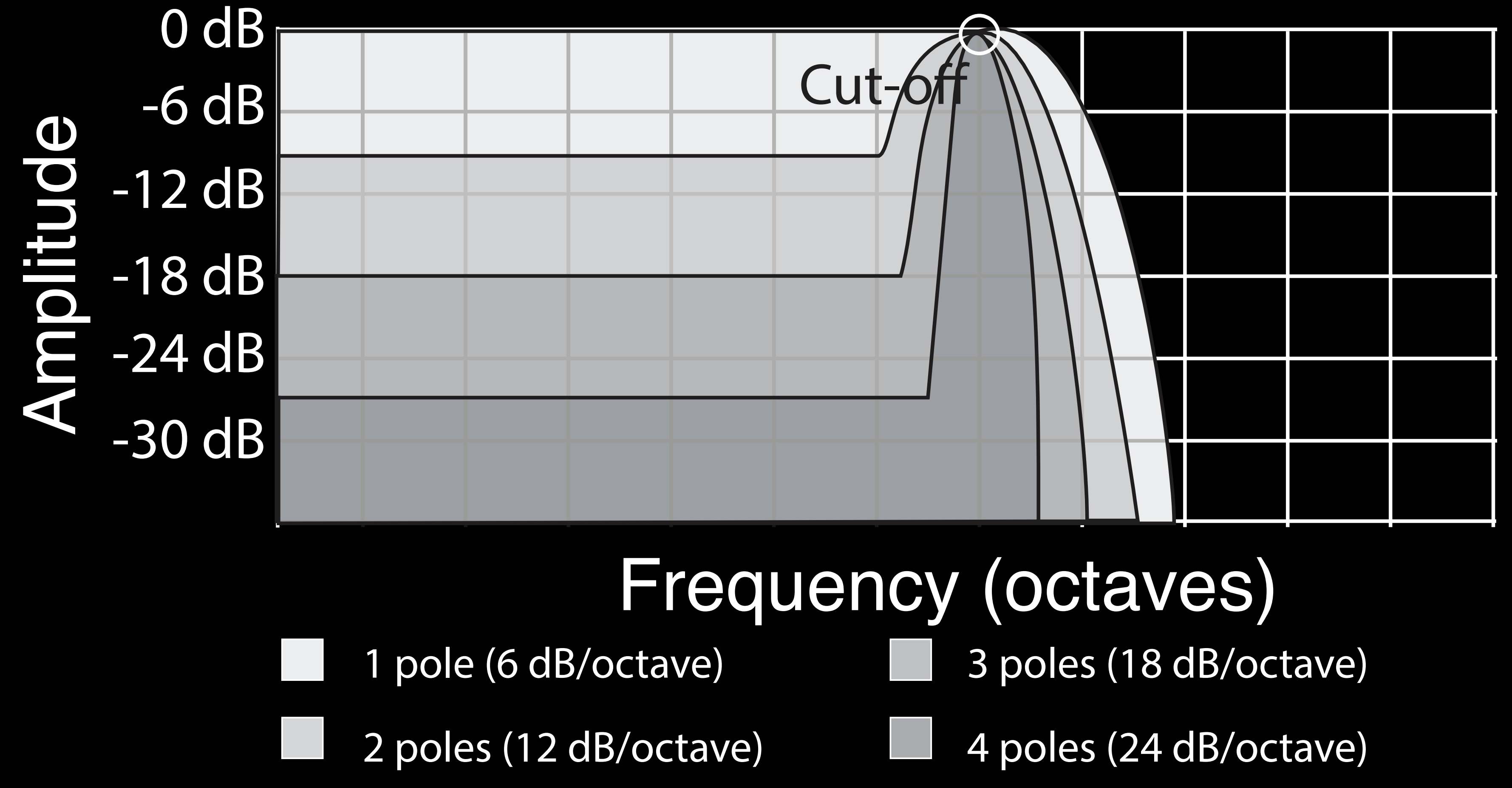
- Attenuates high frequencies
- LPF.ar(in: sound, freq: frequency, mul: 1, add: 0)
- in = the sound input
- freq = the cutoff frequency of the filter
- mul = multiplication of the filter output
- add = the offset of the sound

- In SuperCollider do not use 0Hz as cutoff frequency



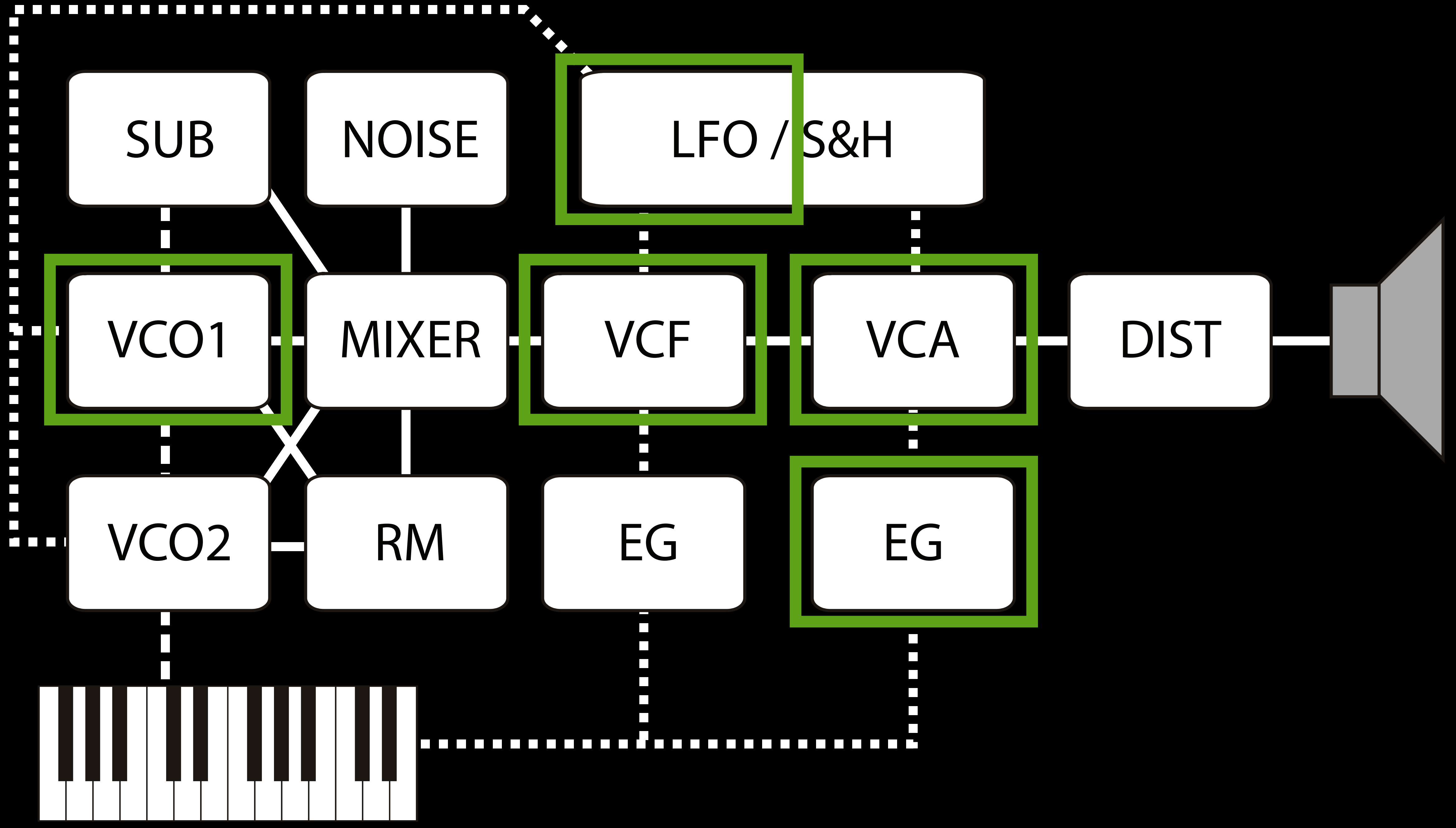
RESONANCE IN FILTERS

- Adds a feedback loop creating an emphasis at the cutoff frequency
- BPF and BRF have this by default, but
- RHPF.ar(in: sound, freq: frequency, rq: 1, mul: 1, add: 0)
- RLPF.ar(in: sound, freq: frequency, rq: 1, mul: 1, add: 0)
- In SuperCollider do not use 0 as rq value



COMBINING PARTS

- var frequencySweep = SinOsc.kr(0.25).range(150, 5000);
- Create a low frequency sine wave oscillator
- Set the oscillation range to be between 150 and 5000
- Put this in a variable (frequencySweep)
- Use this as the cutoff frequency in the lowpass filter



LET'S CONTINUE CODING

- Workshop examples and extras
  https://www.itn.liu.se/~nikro27/am2023_ws/