AUDIO MOSTLY 2023 WORKSHOP ON SUPERCOLLIDER

ADDITIVE SYNTHESIS

NIKLAS RÖNNBERG

LINKÖPING UNIVERSITY
niklas.ronnberg@liu.se

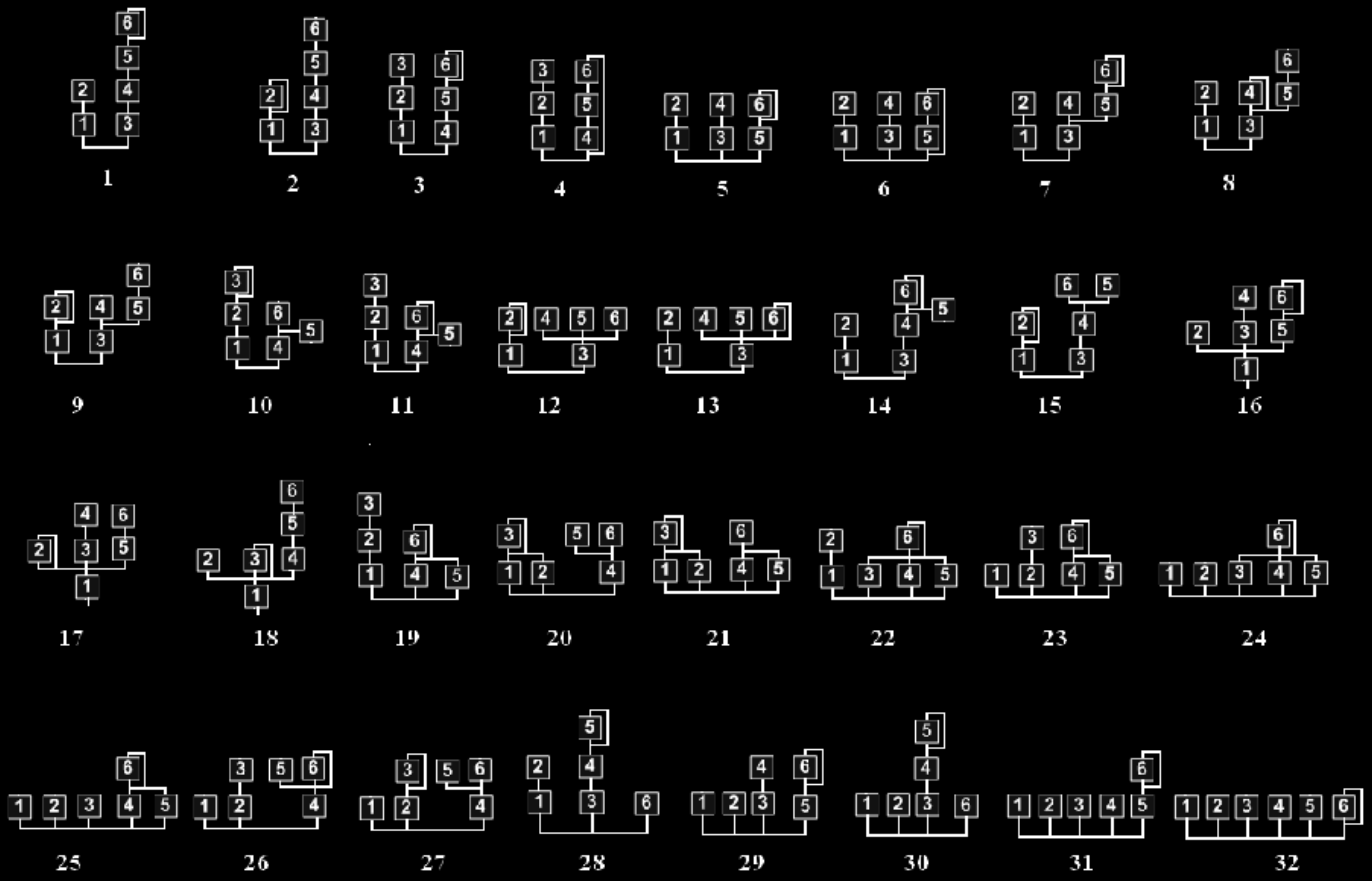
ADDITIVE SYNTHESIS

- Additive sound synthesis creates complex wave forms by adding simple wave forms.

- Each frequency component (partial) can have it’s own envelope.

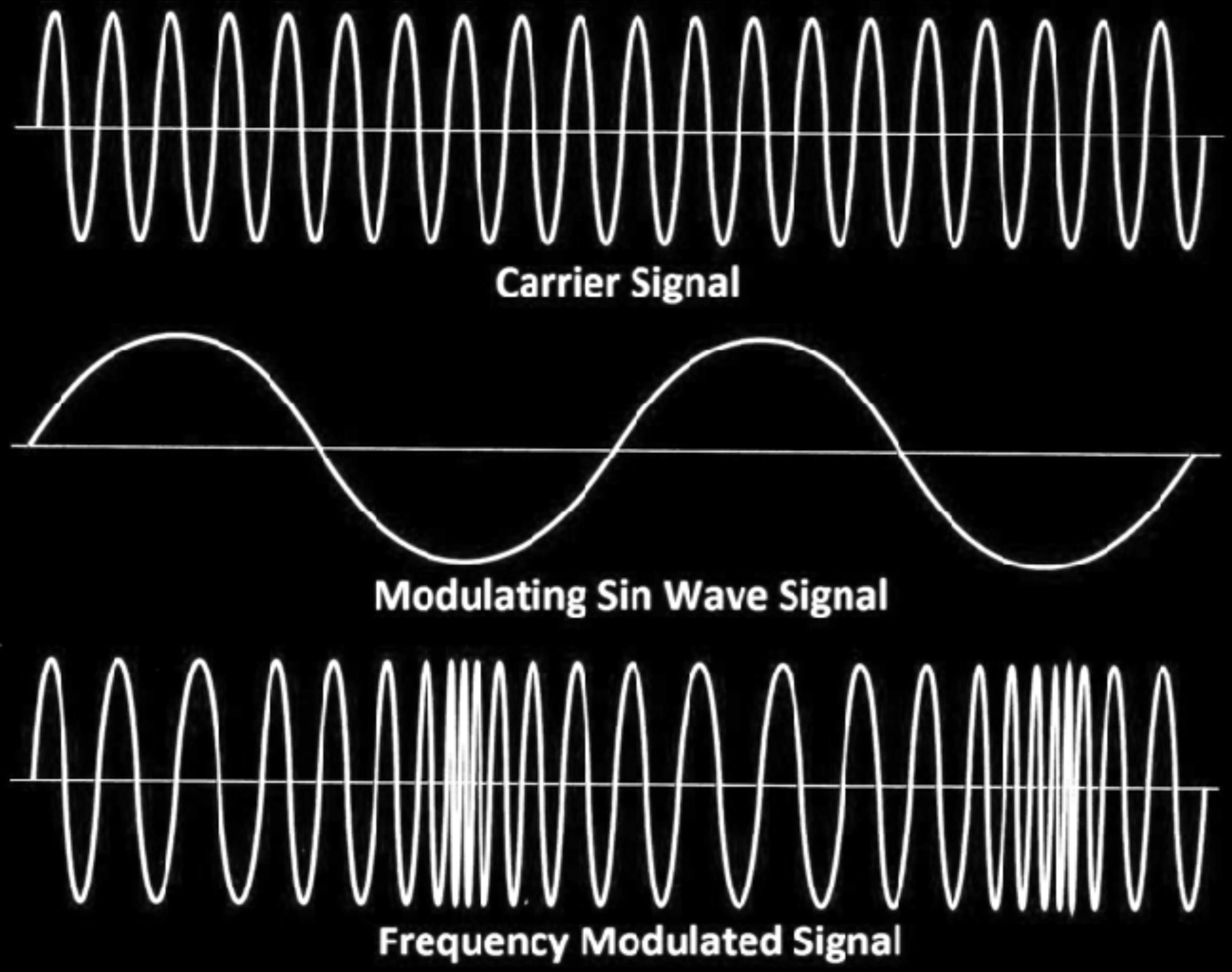
- This creates a simple way to create changing sounds with independent control of sound changes and harmonics.

- FM-synthesis creates both harmonic and disharmonious sounds very well.



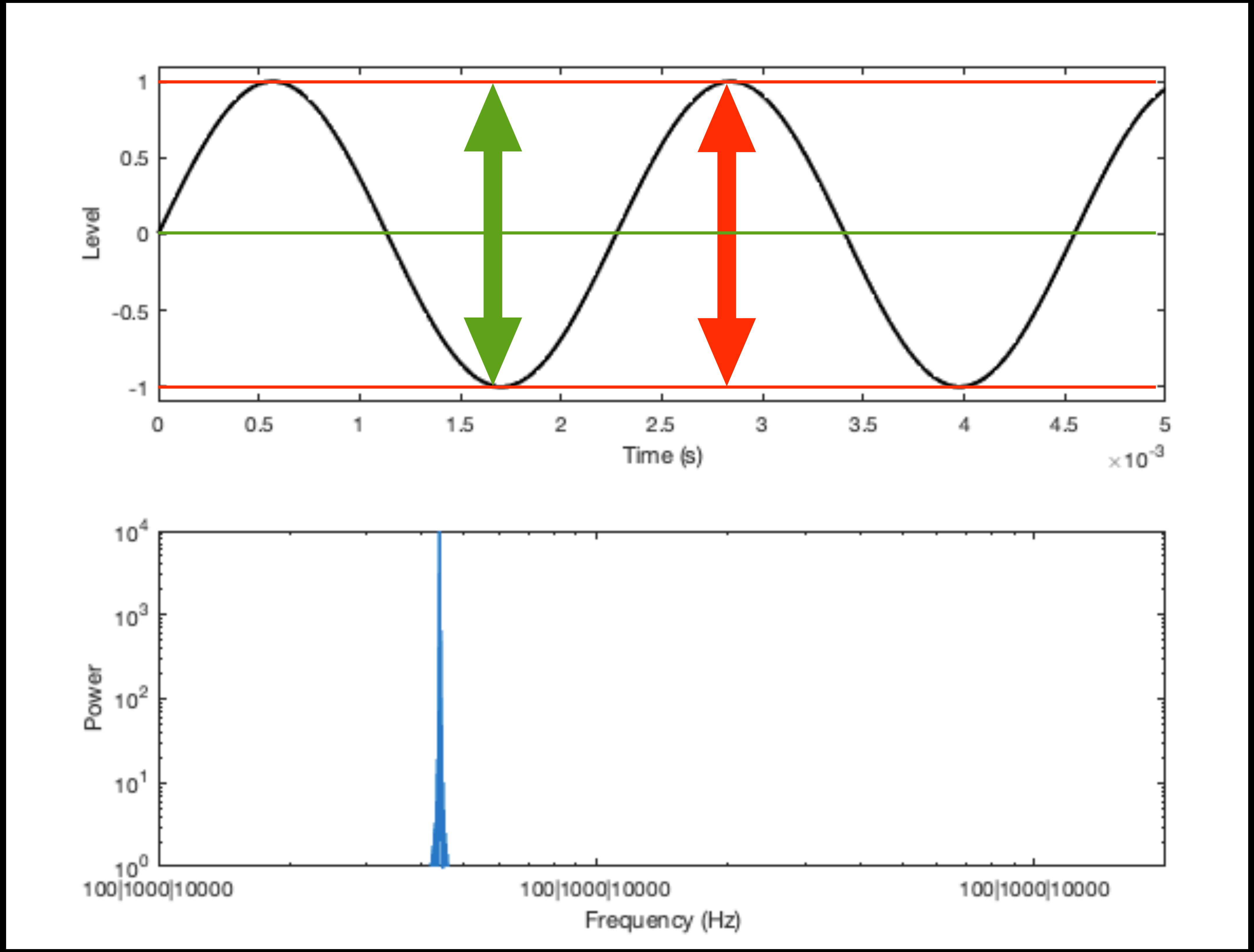
BASIC WAVEFORM

- Sine wave
- Operator/Carrier/Modulator
- Combined in different algorithms



SINE WAVE - OPERATOR

- 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



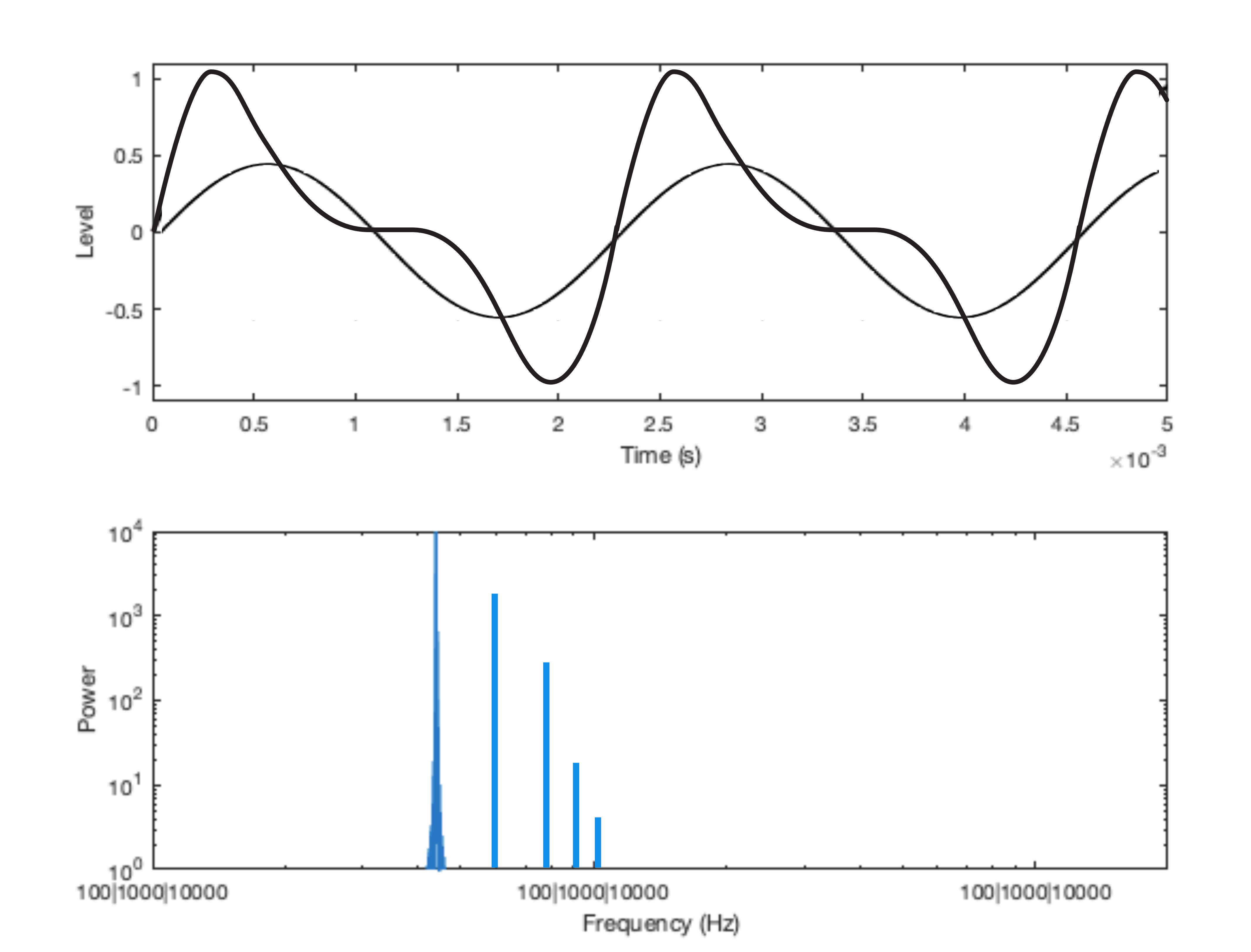
FREQUENCY MODULATION

- Frequency mixing

- op1 = SinOsc.ar(freq: frequency).range(0, 1);

- op2 = SinOsc.ar(freq: frequency * op1);

- range - is setting the range of the modulator to be positive



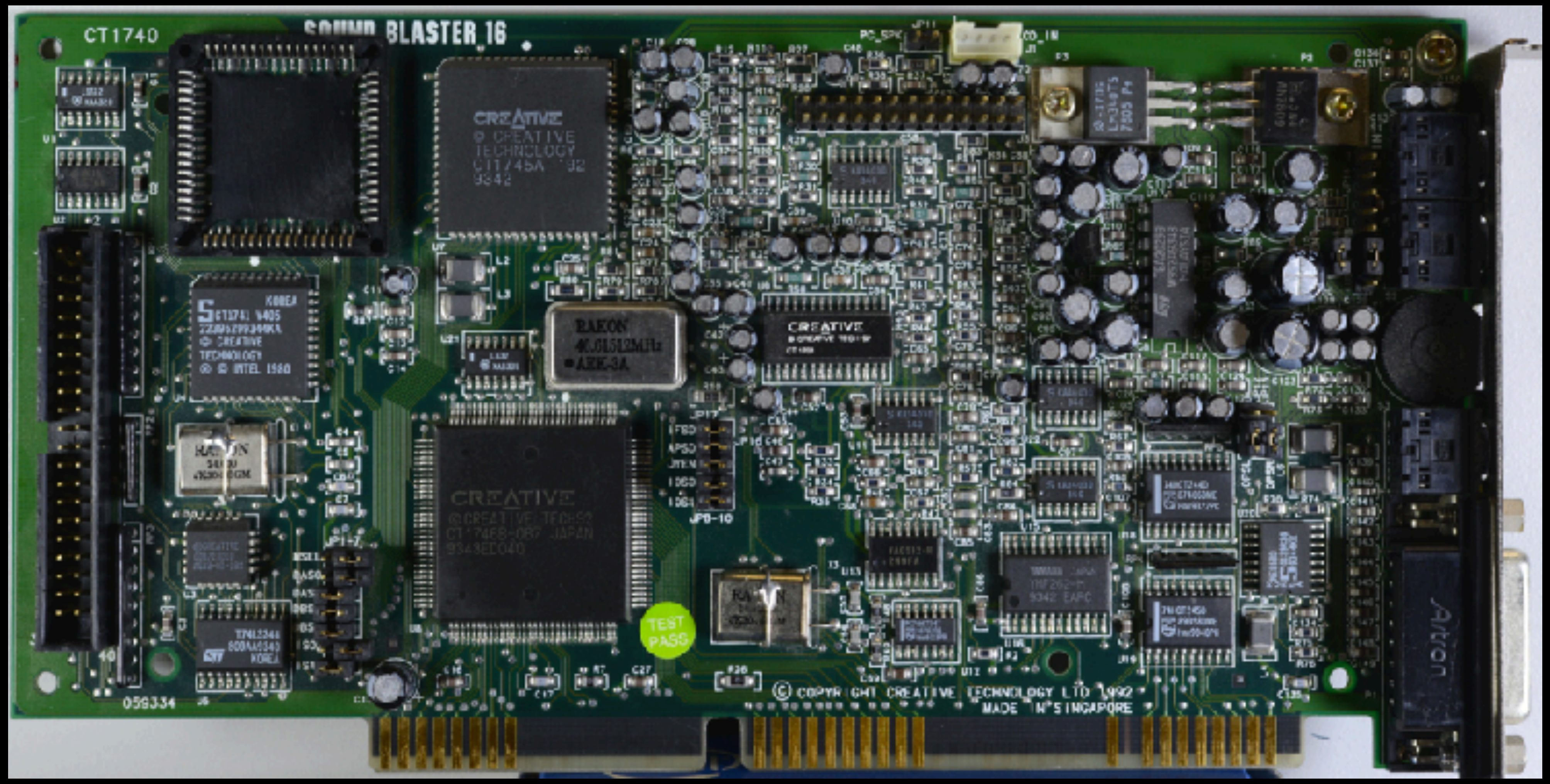
FREQUENCY MODULATION

- The amplitude of the modulating waveform sets the modulation depth.

- A greater modulation depth creates more complex wave forms.

- op1 = SinOsc.ar(freq: frequency).range(0, 1) * 5;

- op2 = SinOsc.ar(freq: frequency * op1);



FREQUENCY MODULATION

- For harmonic sounds, the modulation signal needs to be harmonically connected to the original signal (the carrier).

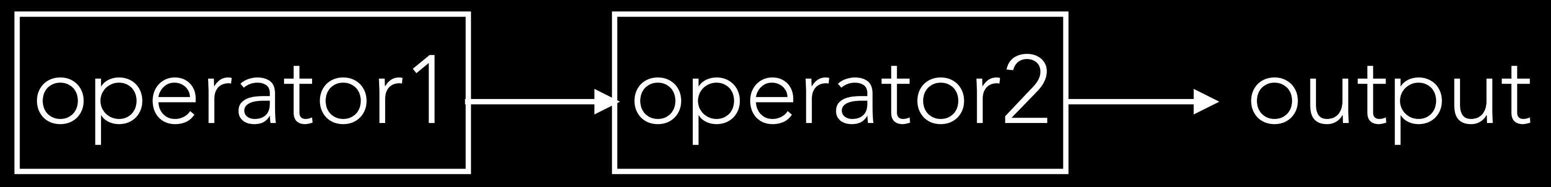
- If the modulators are not harmonic, i.e., integer multiples of the carrier frequency, the sounds gets dissonant, non harmonious, bell like, metallic and percussive sounds.

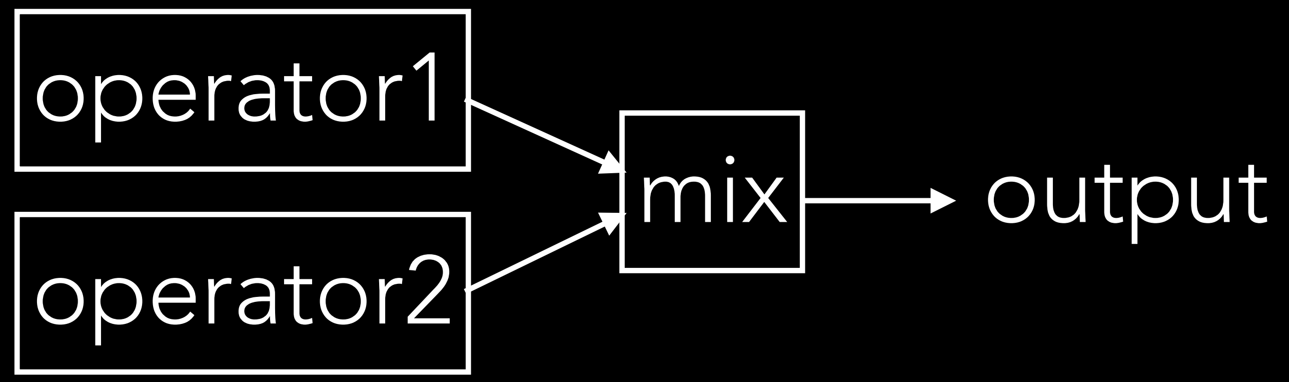
- op1 = SinOsc.ar(freq: frequency * 3.3).range(0, 1) * 5;

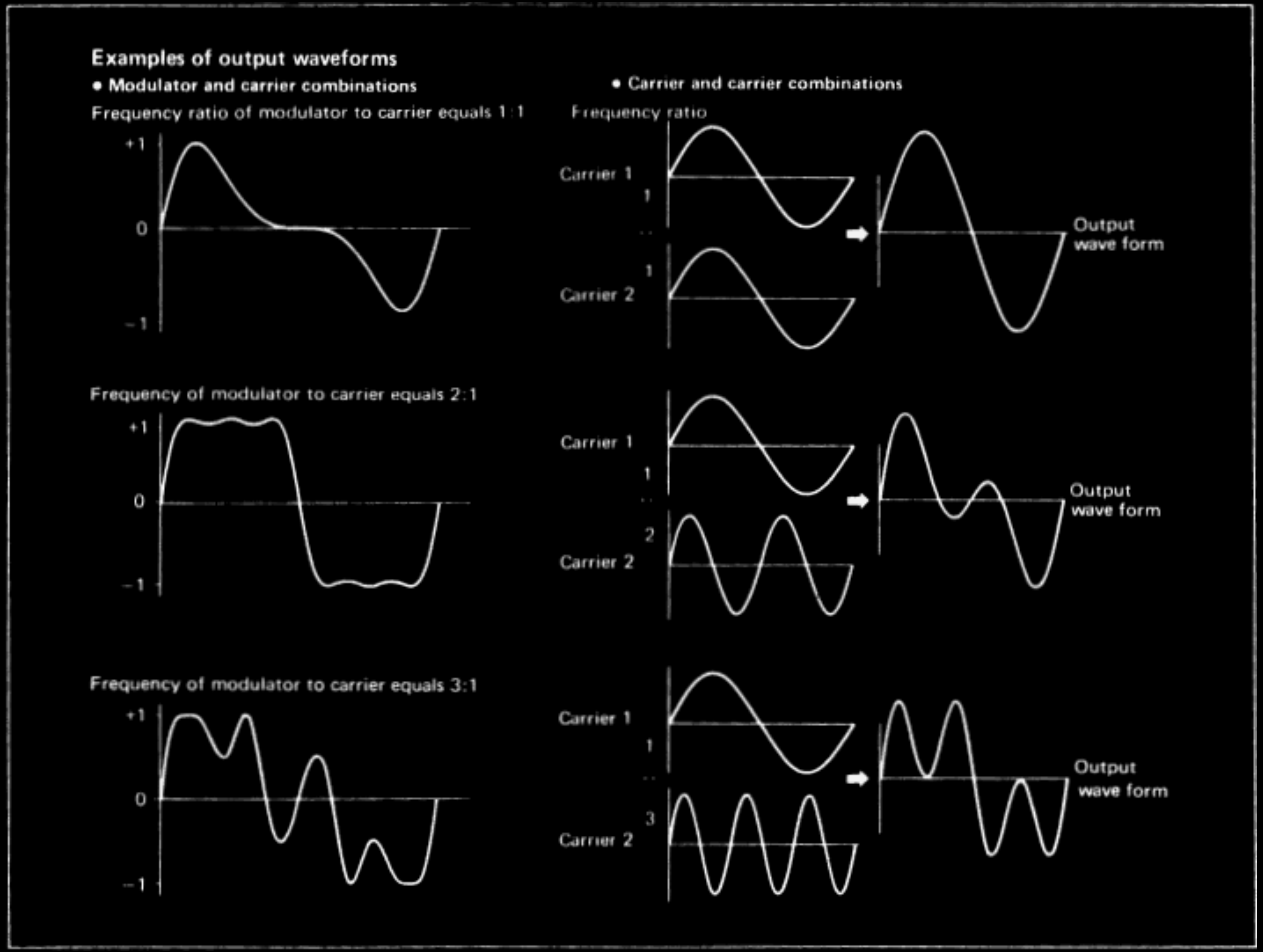
- op2 = SinOsc.ar(freq: frequency * op1);

ALGORITHMS

- Two main approaches for algorithms
  - FM synthesis (frequency mix)
  - Additive (waveform mix)







Examples of output waveforms

- **Modulator and carrier combinations**

Frequency ratio of modulator to carrier equals 1:1

Frequency of modulator to carrier equals 2:1

Frequency of modulator to carrier equals 3:1

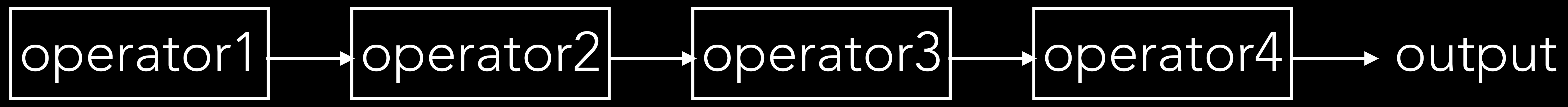
- **Carrier and carrier combinations**

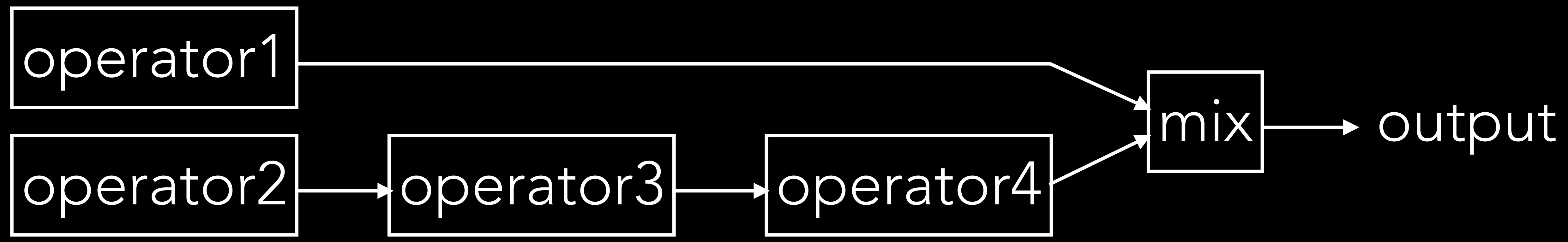
Frequency ratio

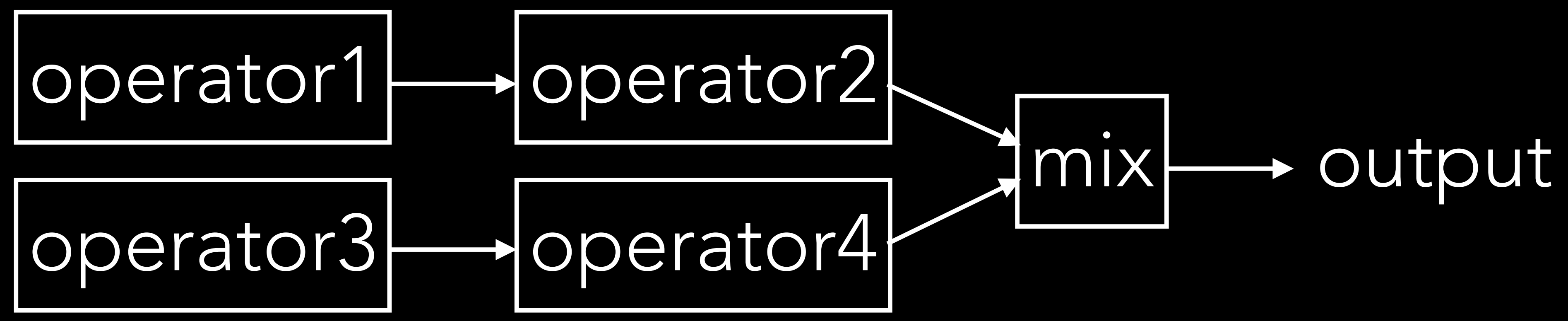
ALGORITHMS

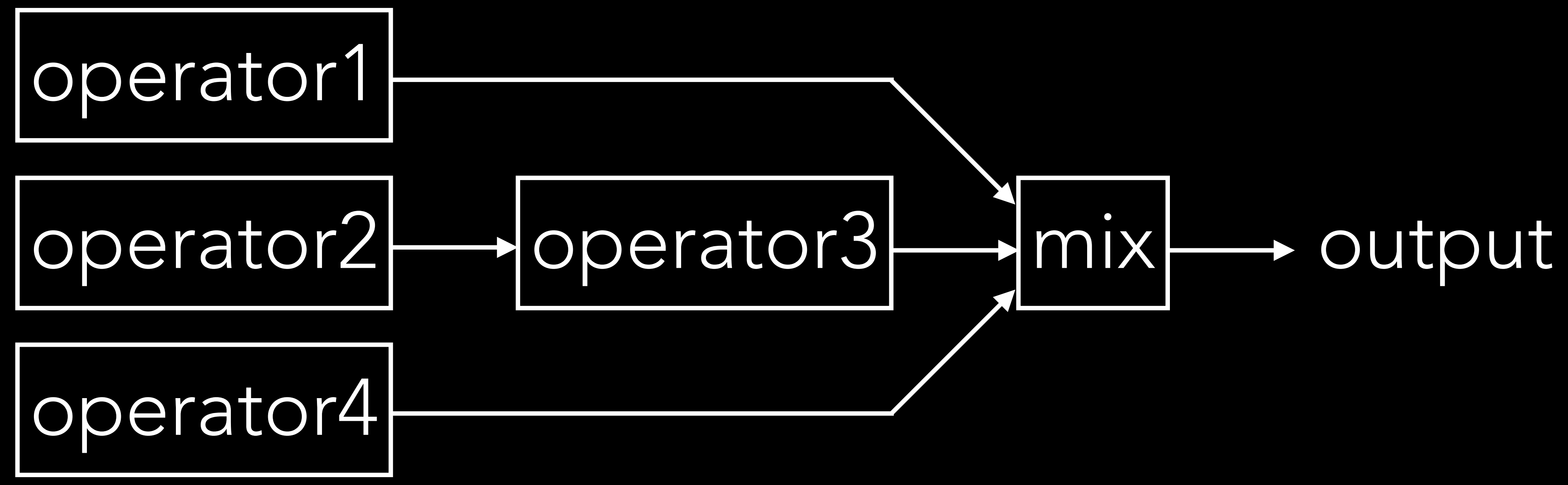
- FM synthesis (frequency mix)

- Additive (waveform mix)



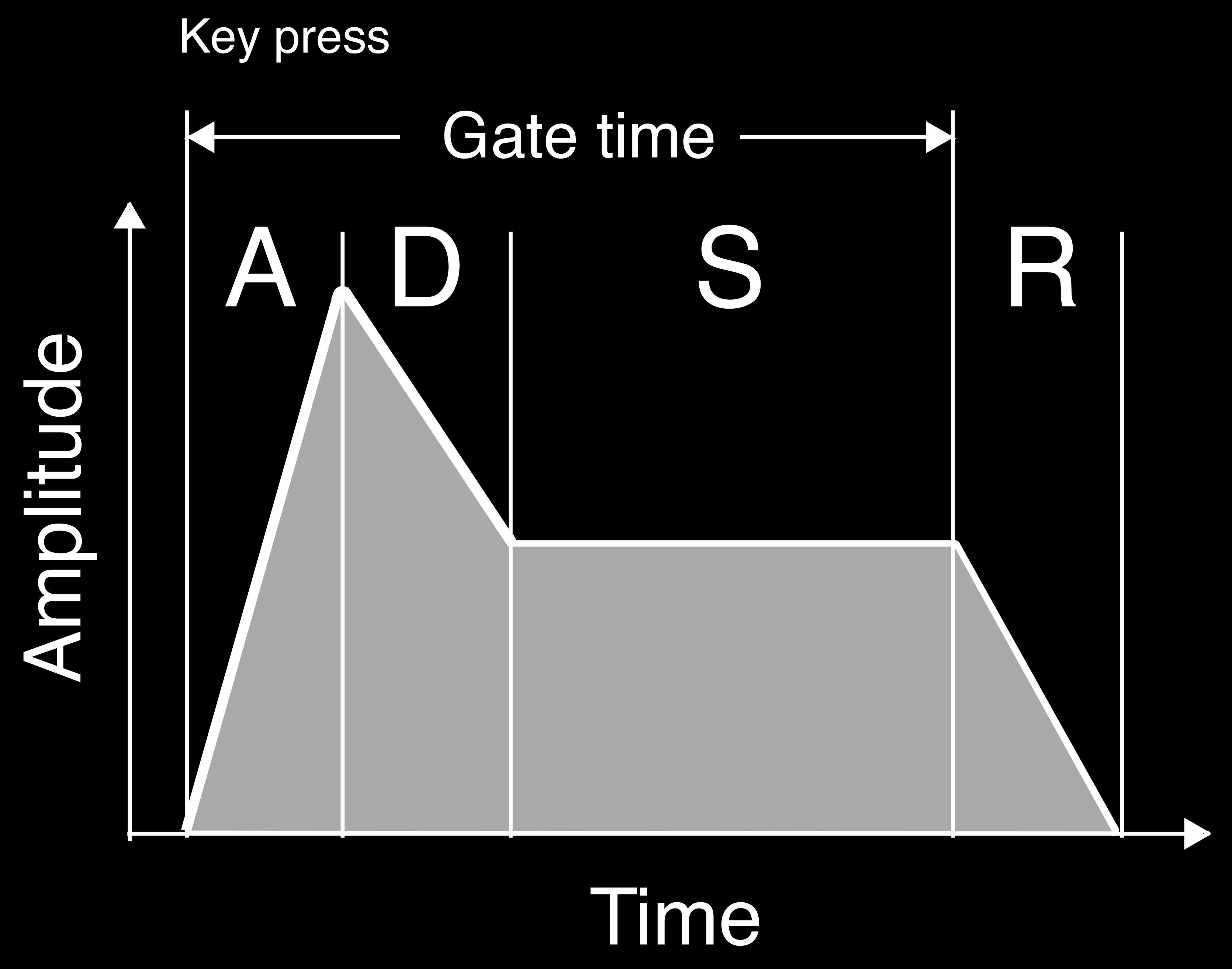






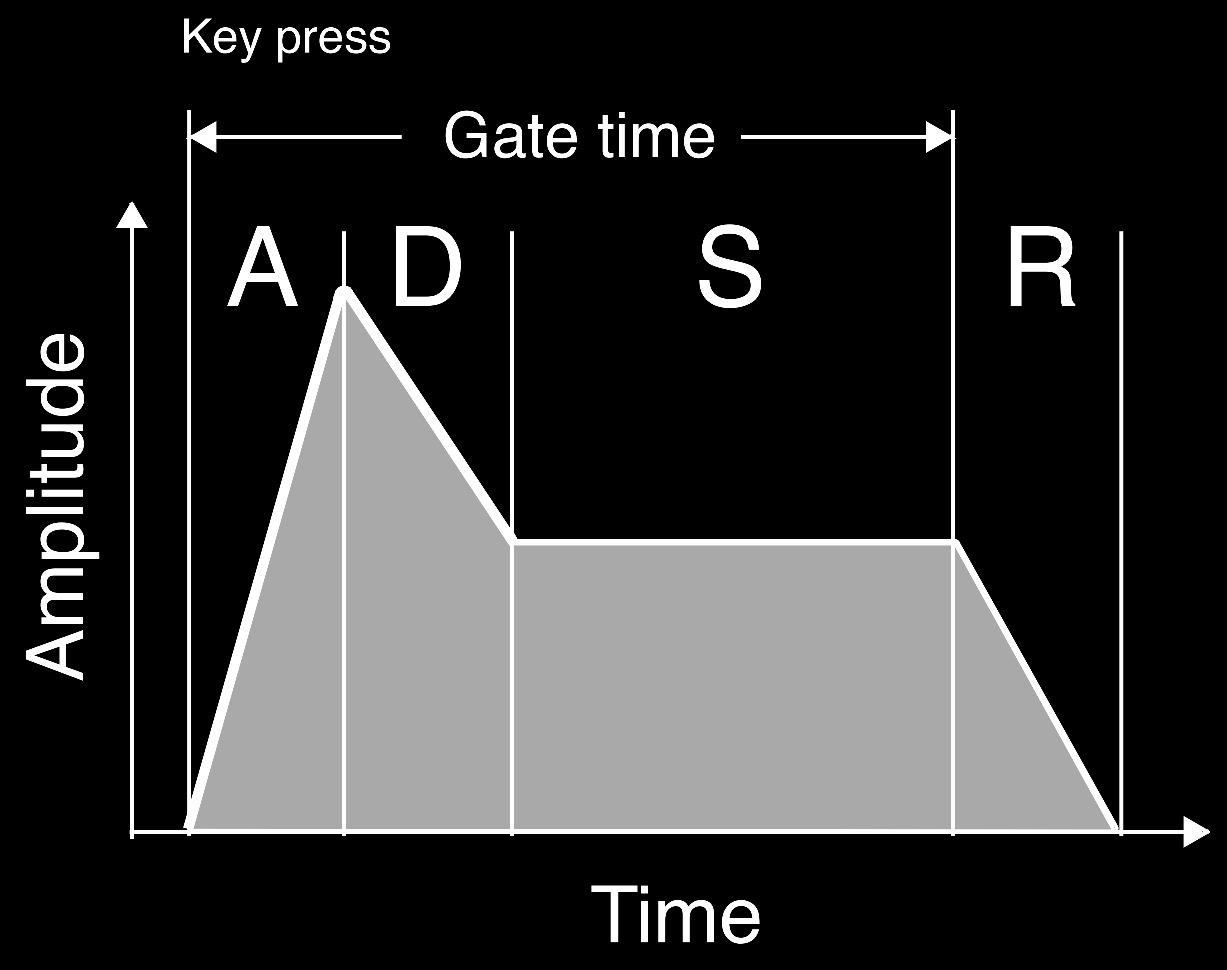
ENVELOPE GENERATORS

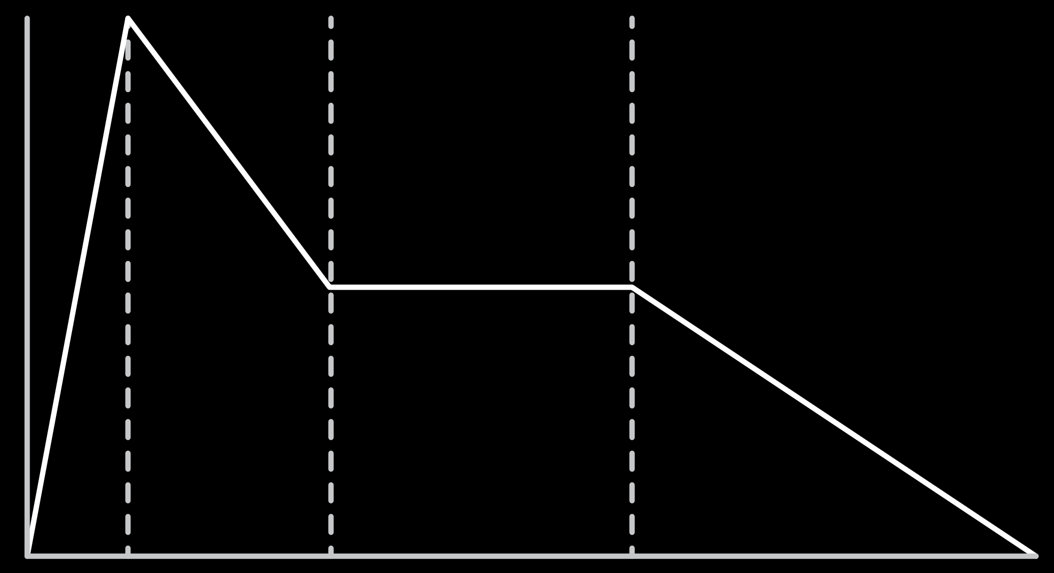
- Adjusts the contour of the amplitude (or any signal)
- EnvGen.kr(envelope, gate, levelScale, levelBias, timeScale, doneAction)
- envelope = different types of contour
- gate = the turn-on-signal
- levelScale = scaling of the envelope
- levelBias = the offset of the envelope
- timeScale = scaling of the timing in the envelope
- doneAction = what should happen after the envelope is done



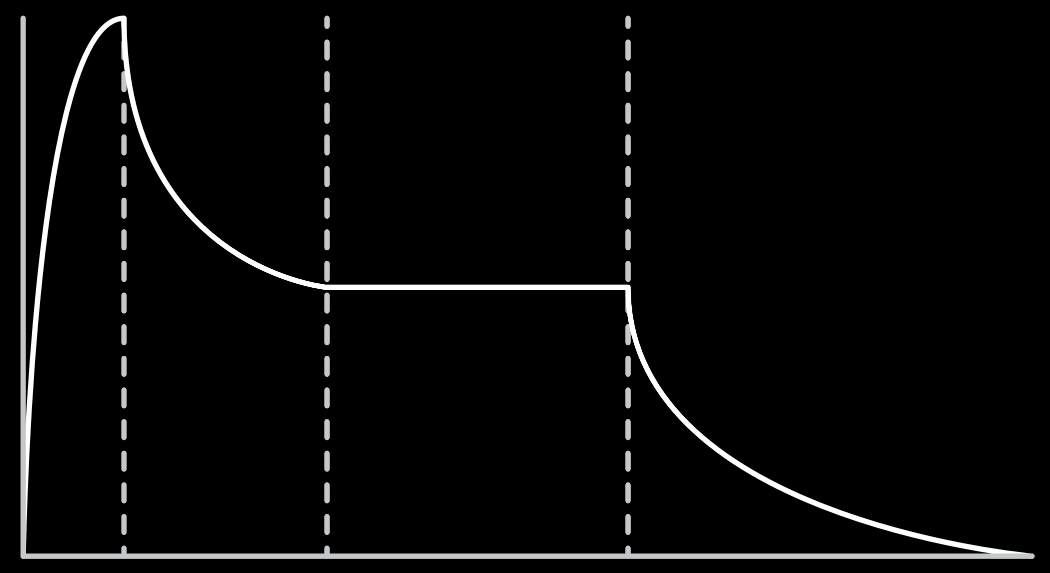
ENVELOPE GENERATORS

- ADSR (Attack, Decay, Sustain, Release)
- Env.adsr(attackTime, decayTime, sustainLevel, releaseTime, peakLevel, curve, bias)
- The time settings are in relation to the timeScale
- peakLevel = the max level of the envelope
- curve = the curvature of the envelope
- bias = the offset of the envelope

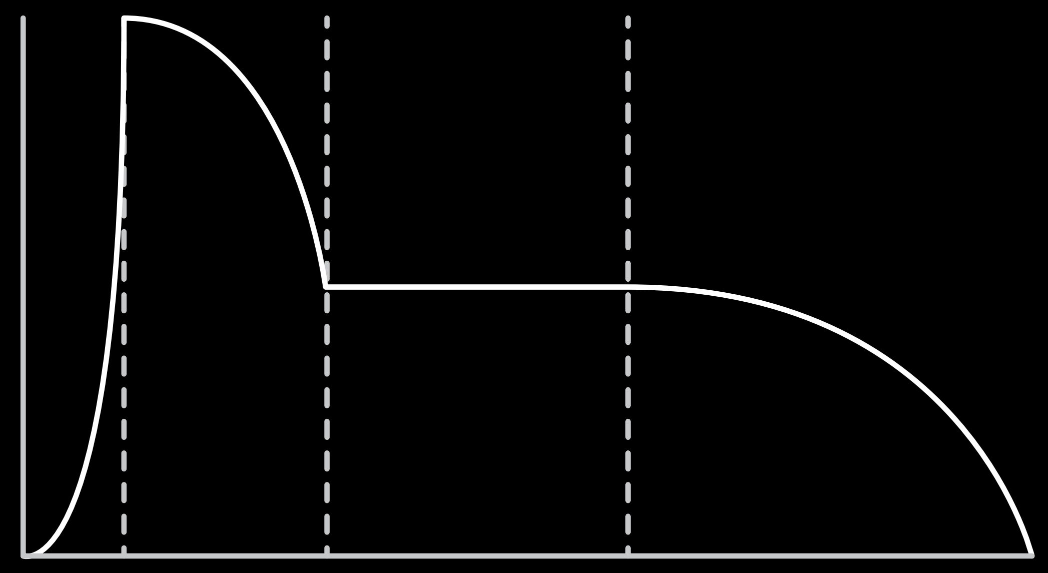




curve: 0



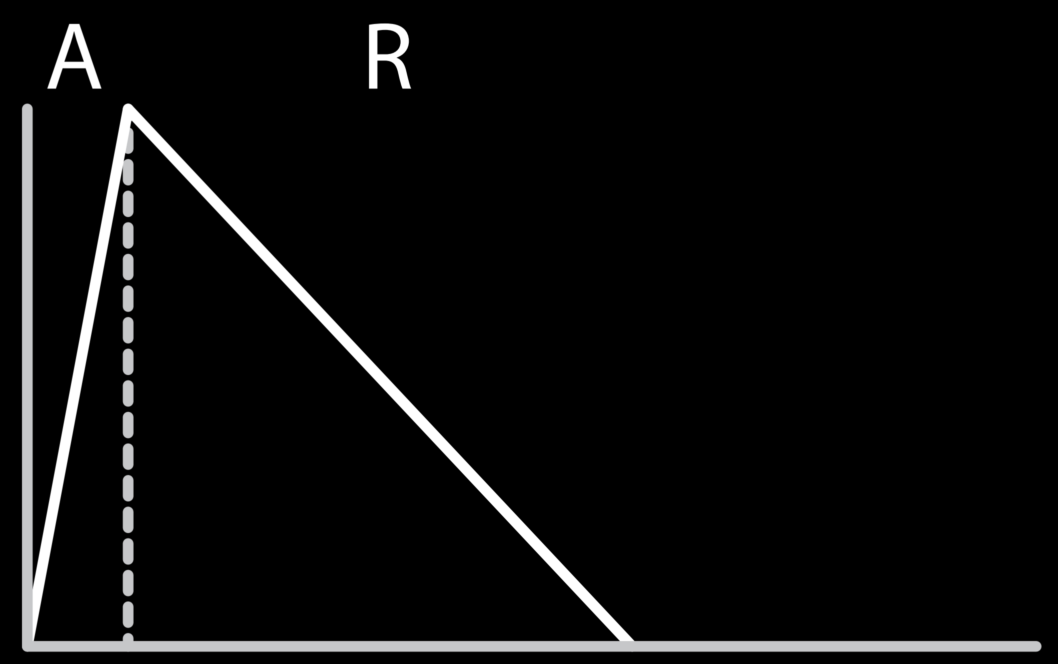
curve: -4

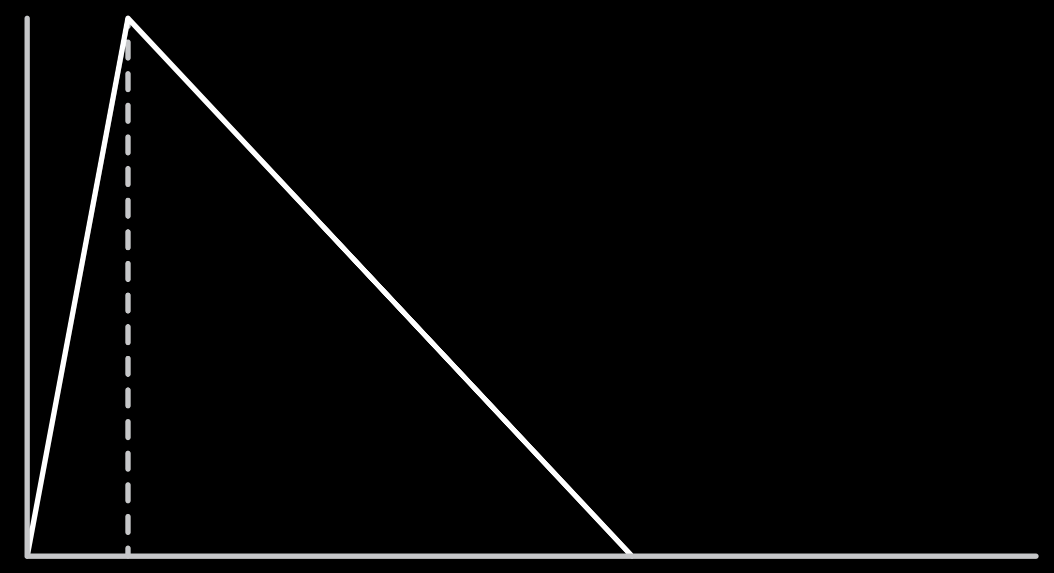


curve: 4

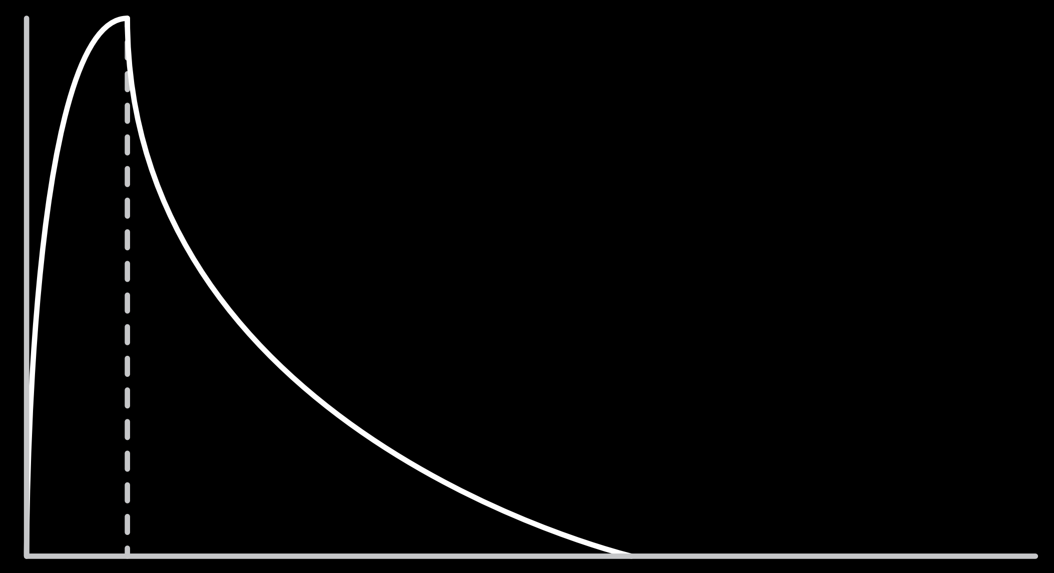
ENVELOPE GENERATORS

- AR (Attack, Release)
- Env.perc(attackTime, releaseTime, level, curve)
- The time settings are in relation to the timeScale
- level = the max level of the envelope
- curve = the curvature of the envelope
- .plot = plots the envelope shape

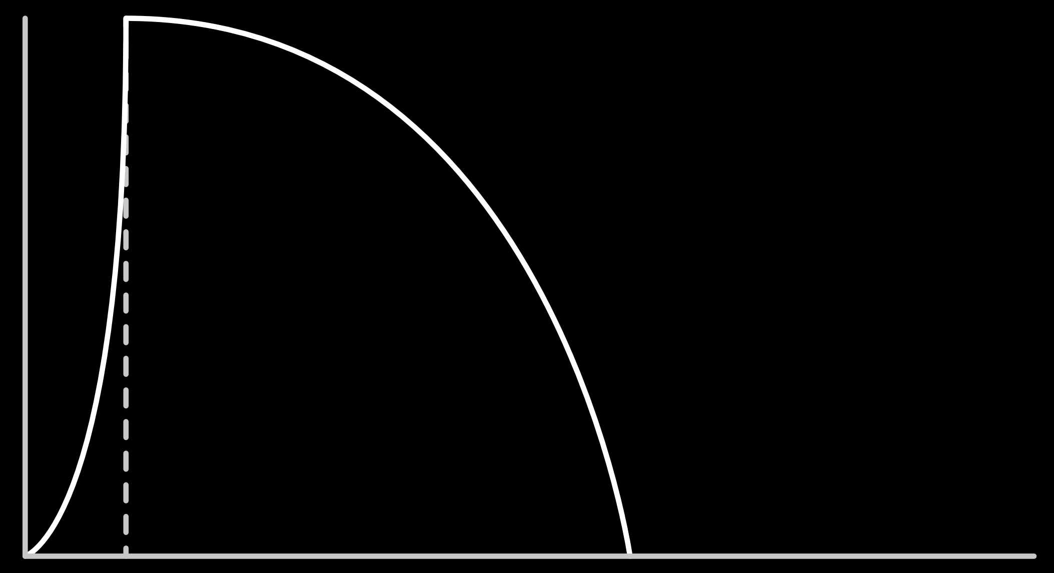




curve: 0



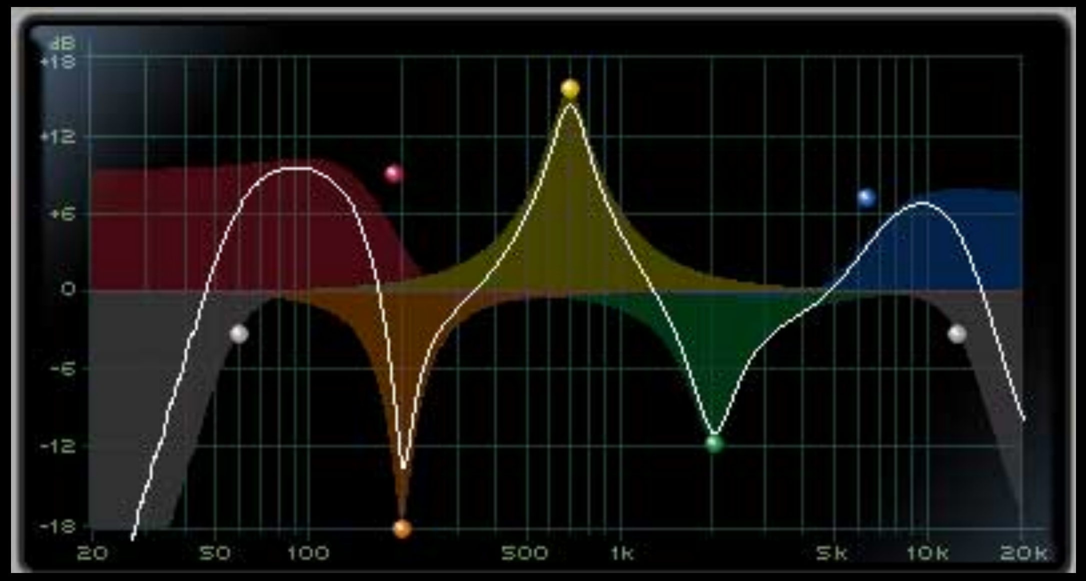
curve: -4

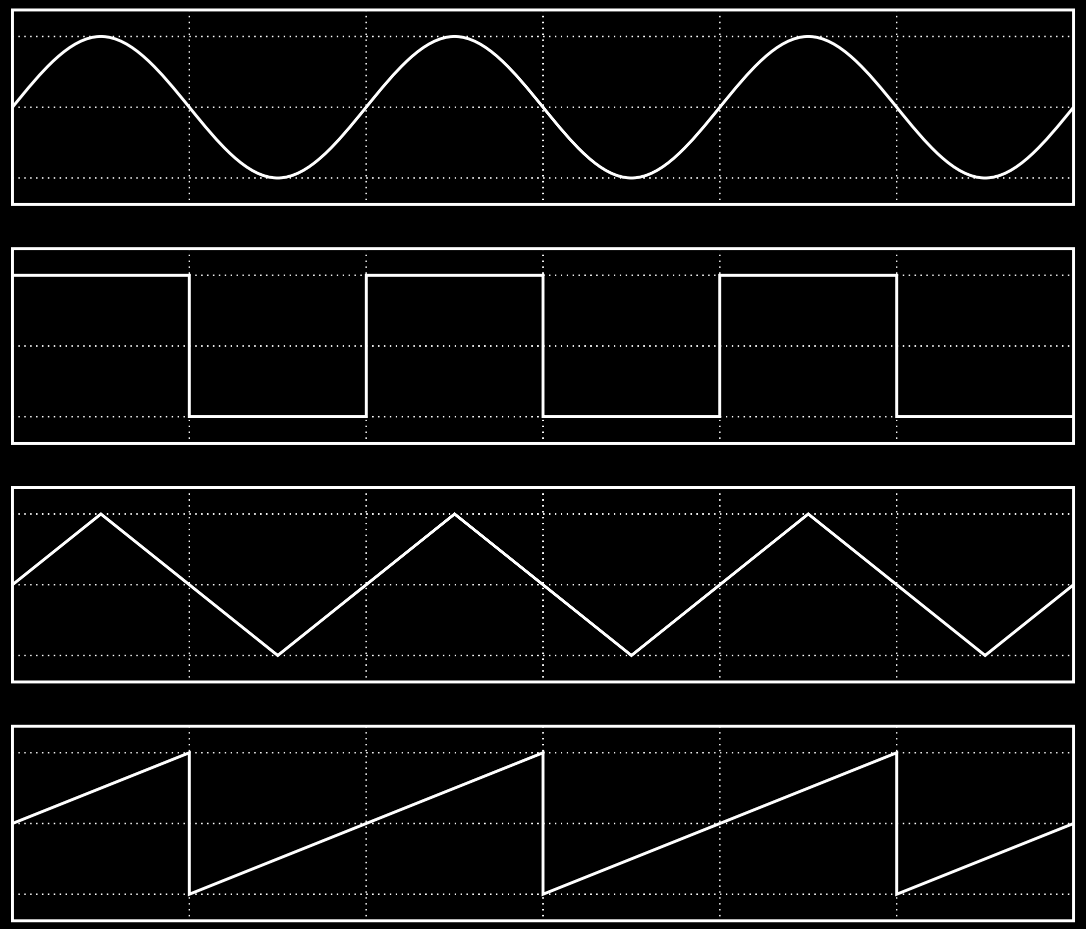


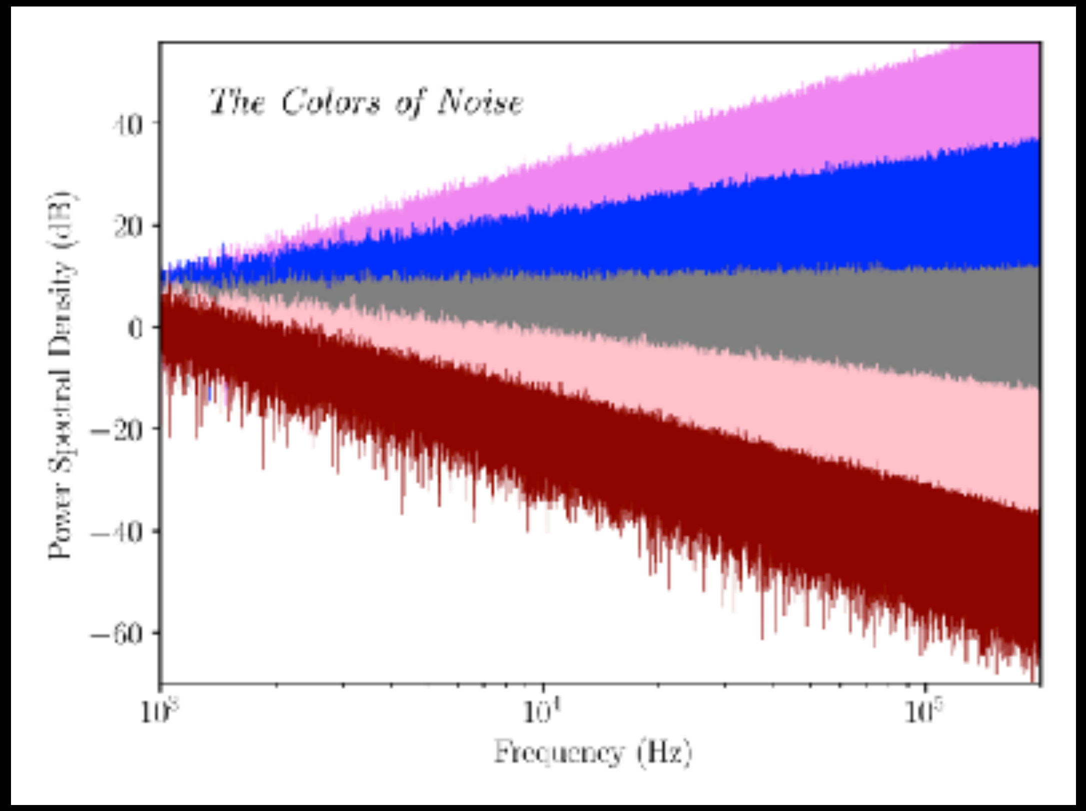
curve: 4

GOING FURTHER

- Filters (HPF, BPF, BRF, LPF, Shelving)
- Other waveforms (LFTri.ar, LFPulse.ar, LFSaw.ar, …)
- Noise (WhiteNoise.ar, PinkNoise.ar, BrownNoise.ar)







LET'S CONTINUE CODING

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