Reminder: do Start-of-Semester forms!

TECH 350: DSP

Class III: Properties of Systems, Signal Flow Diagrams, Filters

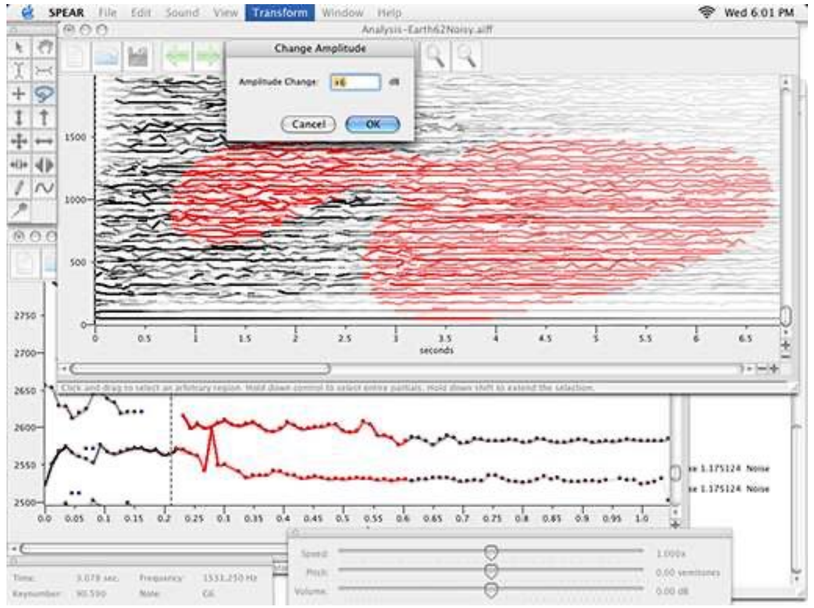
But, first…

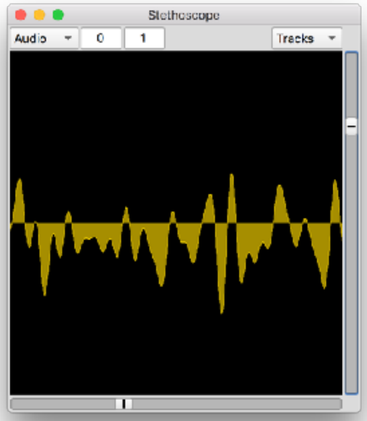
Digital Synthesis Varieties

- Additive Synthesis
- Subtractive Synthesis
- Modulation Synthesis
- Granular Synthesis
- And others (will talk about later…)

Additive Synthesis

- The creation of timbres via the summation of simple waves (often sinusoids)
- Cases in point: pipe organs, SPEAR, Max Demo.
- Considerations: harmonicity/ inharmonicity and relative amplitudes as a function of fundamental (or other parameters), envelope(s), number of oscillators and computation

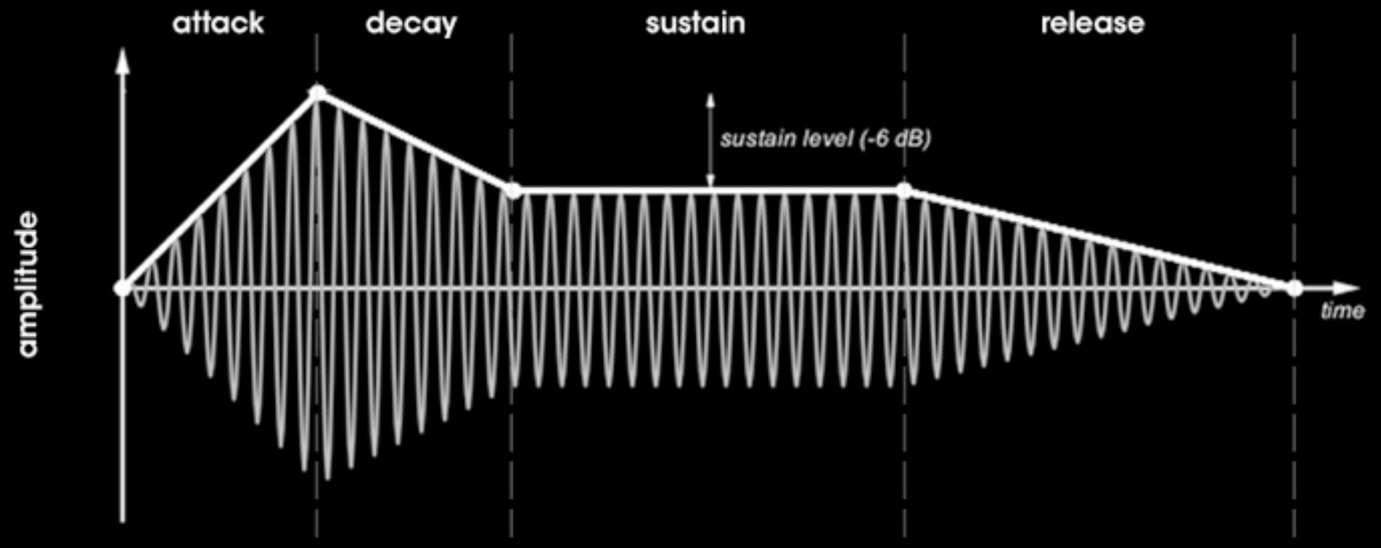




ENVELOPES

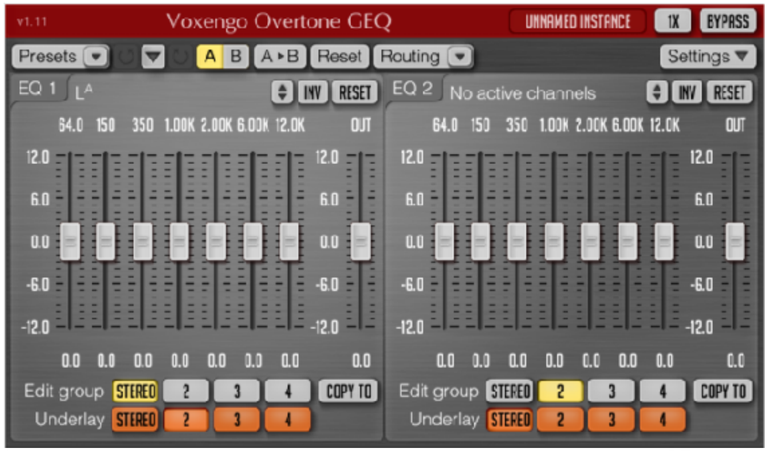
An envelope generator produces a control voltage that rises and falls once, according to a voltage command. The output rises to full on (ATTACK) and then falls over some time (DECAY) to an intermediate value (SUSTAIN) remains there before continuing to zero (RELEASE), often when the key is released.

ADSR design built by Moog at request of Ussachavesky



Subtractive Synthesis

- The creation of timbres via the attenuation by filter of an (often) harmonically rich sound
- Cases in point: many analog/ digital synths, traditional vocoding
- Considerations: source signal, filter types and properties, envelope(s)



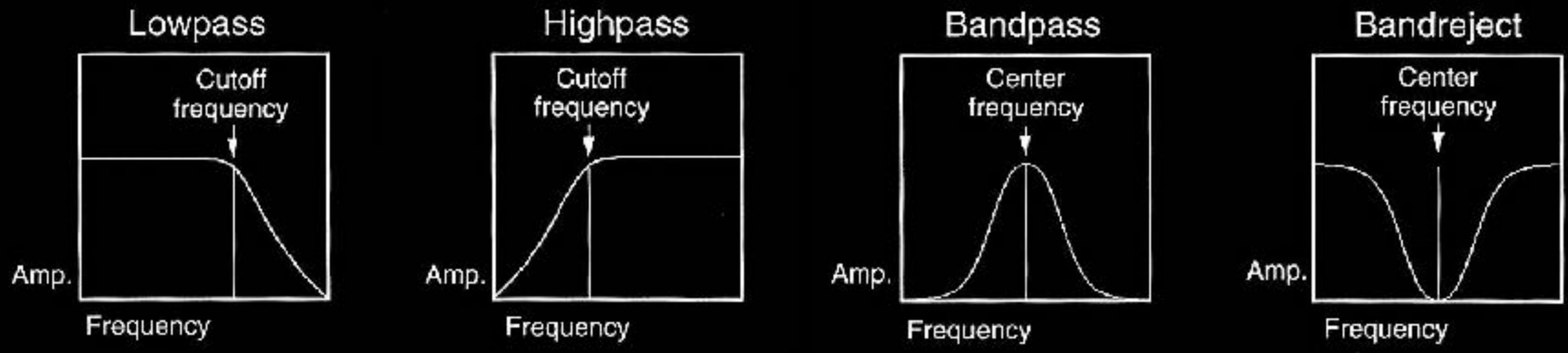
FILTERS

signal processing module, Voltage-controlled filter (VCF)

much of the timbral flexibility of a synthesizer comes from the filters

Boost or cut the amplitude of spectral components

Common varieties: low pass (LPF), high pass (HPF), band pass (BP), notch



“Q” characterizes a resonator's bandwidth relative to its center frequency. Higher the Q, narrower the filter

Modulation Synthesis

- The creation of timbres via the interaction (multiplication, etc.) of multiple (often two, to start) simple waves
- Cases in point: ring modulation, amplitude modulation, frequency modulation
- Considerations: control, CPU efficiency (wow-wee!)

Types of Modulation Synthesis

Ring Modulation = **Multiplying** two bipolar (-1 to 1) signals

Result: If sines, 2 frequencies

Amplitude Modulation = Multiplying a bipolar signal (carrier wave) with a unipolar signal (0 to 1, **amplitude**) (modulating wave)

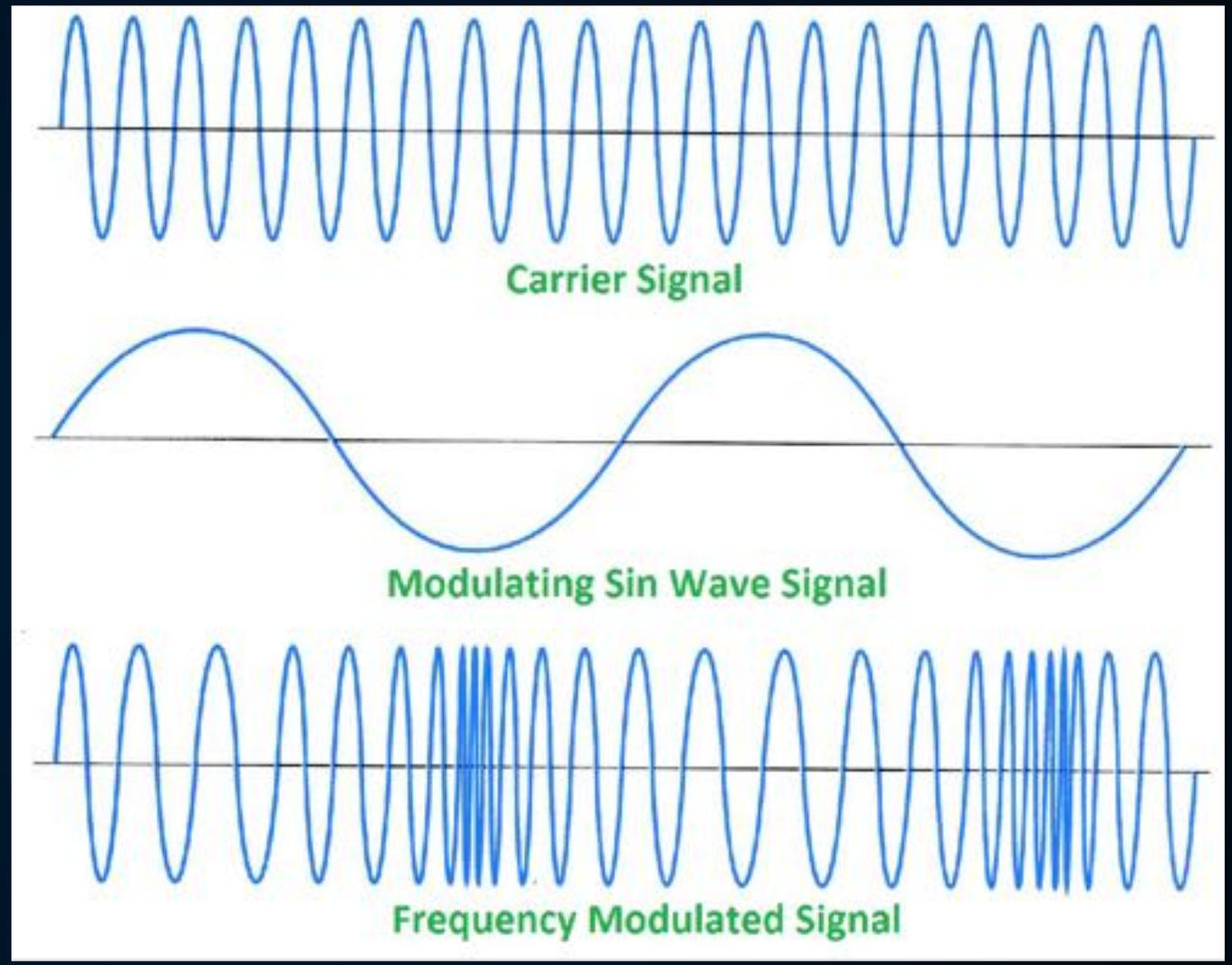
Result: If sines, 3 frequencies

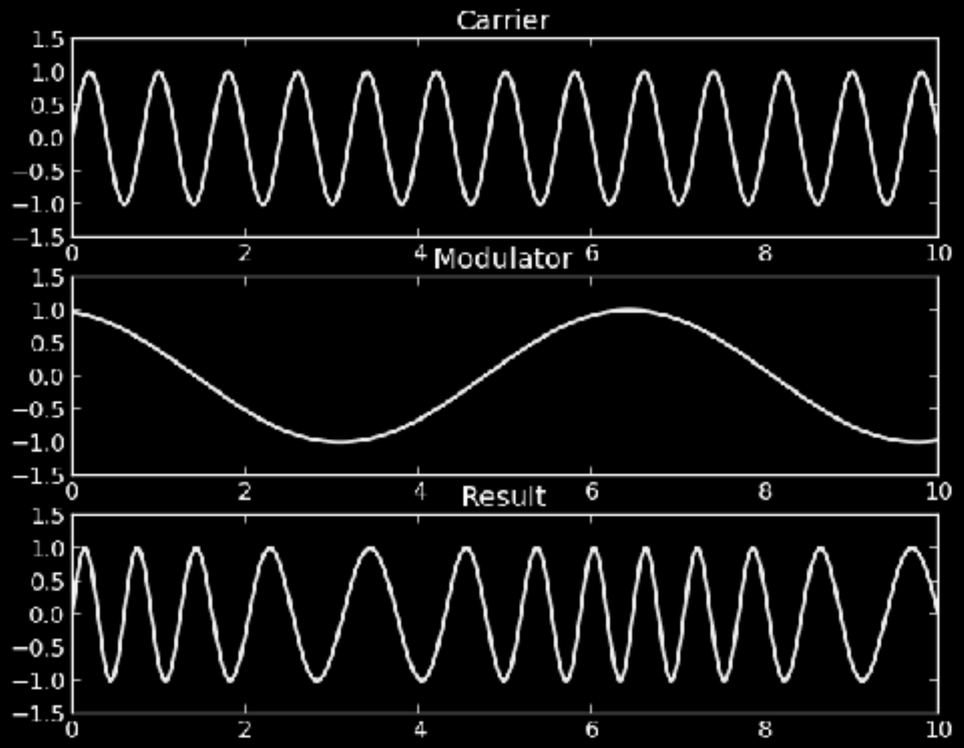
Frequency Modulation = Modulating wave changes the **frequency** of the carrier wave

Result: If sines… infinite frequencies (!)

FM Synthesis

the modulating wave or operator changes the frequency of the carrier wave.





FM SYNTHESIS

Frequency modulation first used in radio

FM synthesis developed by John Chowning in the early 1970s

efficient algorithm - little computation to generate rich sound palettes.

Yamaha DX7 (1980), one of the most popular synths of all time

DX7

FM Synthesizer based on the research of John Chowning

first commercially successful digital synthesizer

MIDI (Musical Instrument Digital Interface)

The Synthesis of Complex Audio Spectra by Means of Frequency Modulation

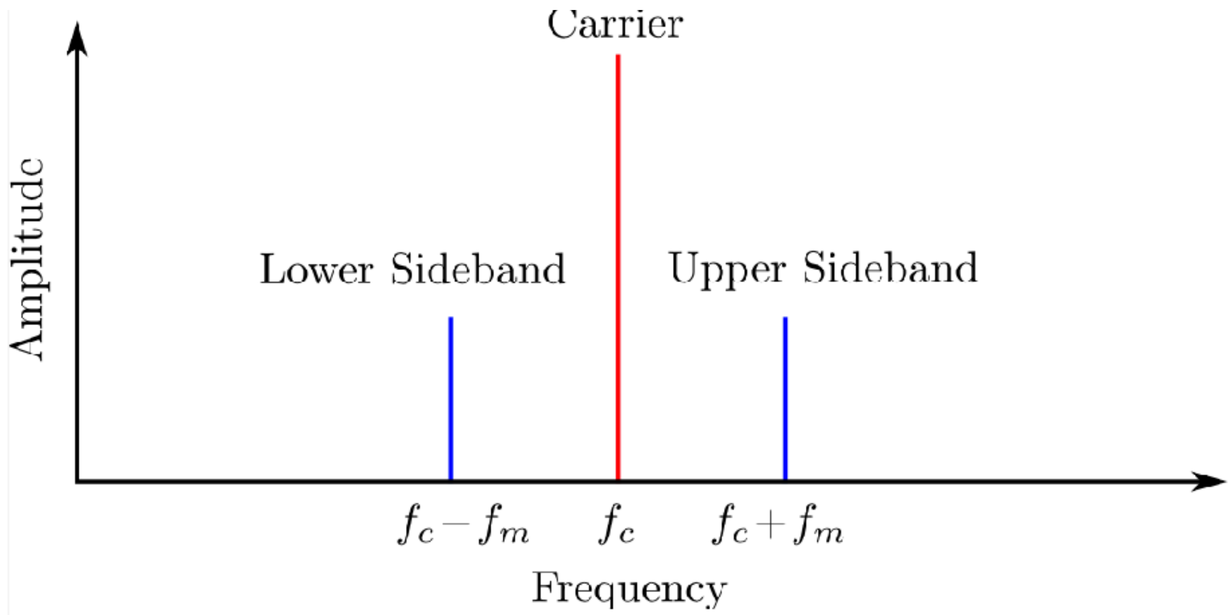
JOHN M. CHOWNING

Stanford Artificial Intelligence Laboratory, Stanford, California

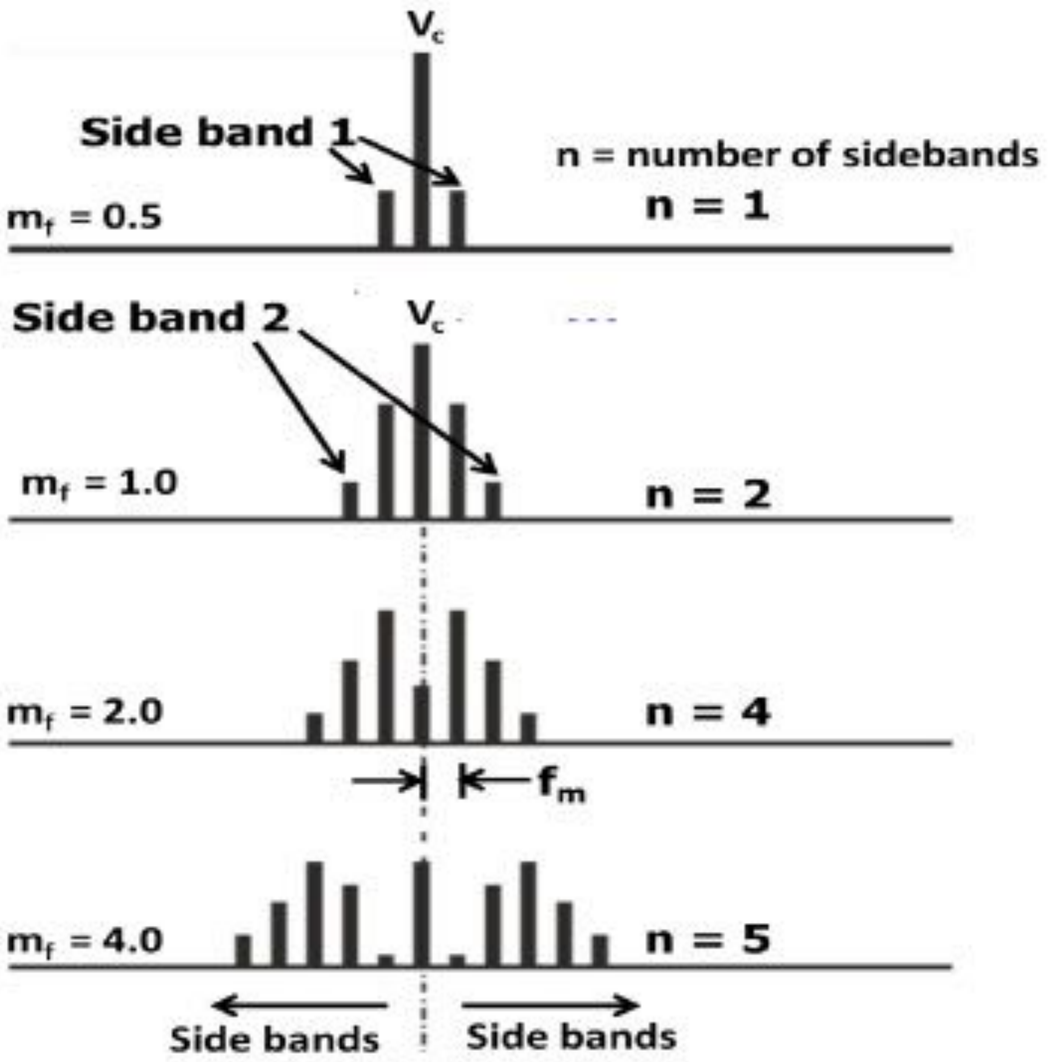
A new application of the well-known process of frequency modulation is shown to result in a surprising control of audio spectra. The technique provides a means of great simplicity to control the spectral components and their evolution in time. Such dynamic spectra are diverse in their subjective impressions and include sounds both known and unknown.

Chowning with Max Mathews and his radio baton

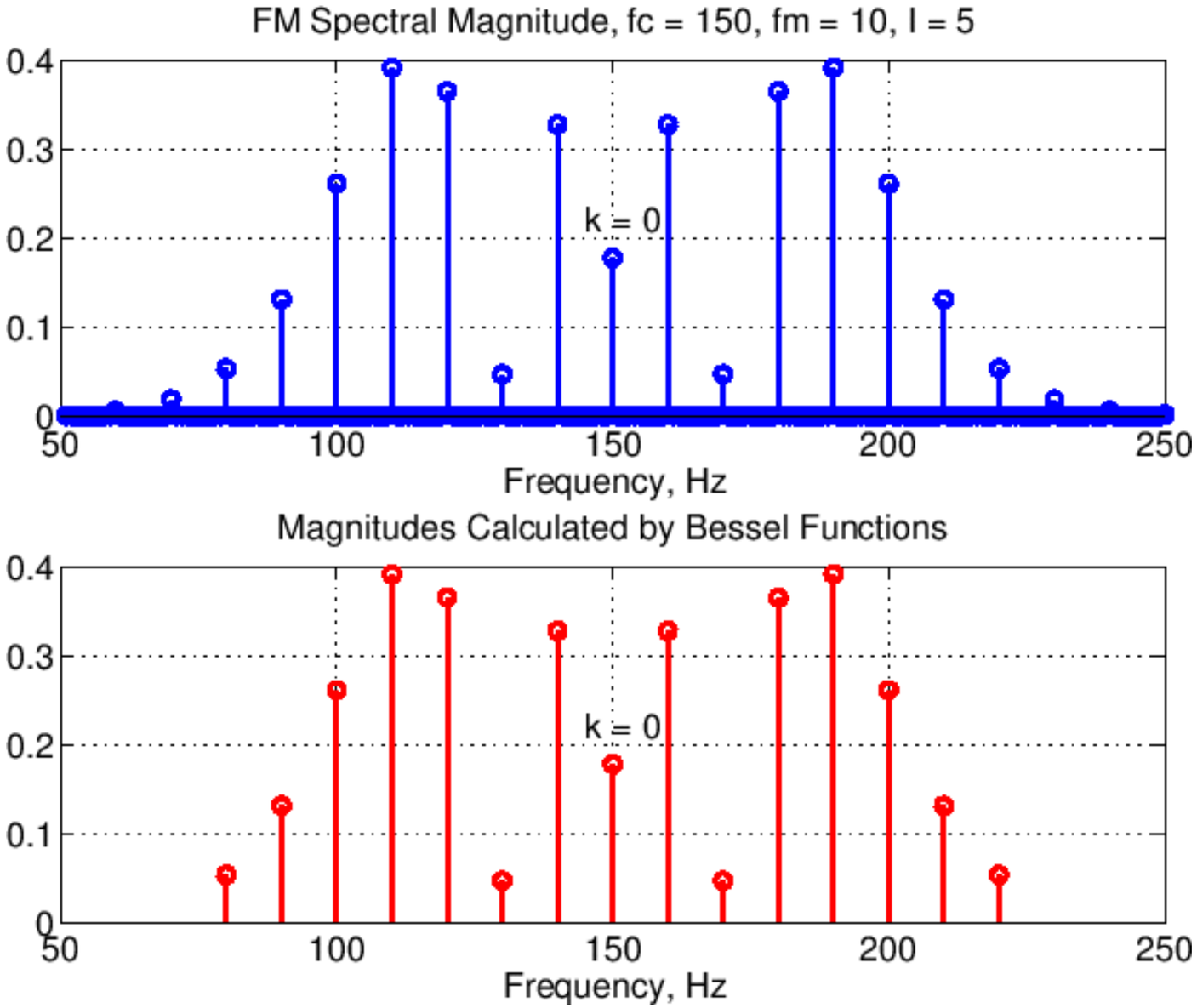
Excerpt from Stria (1977)



AM Sidebands



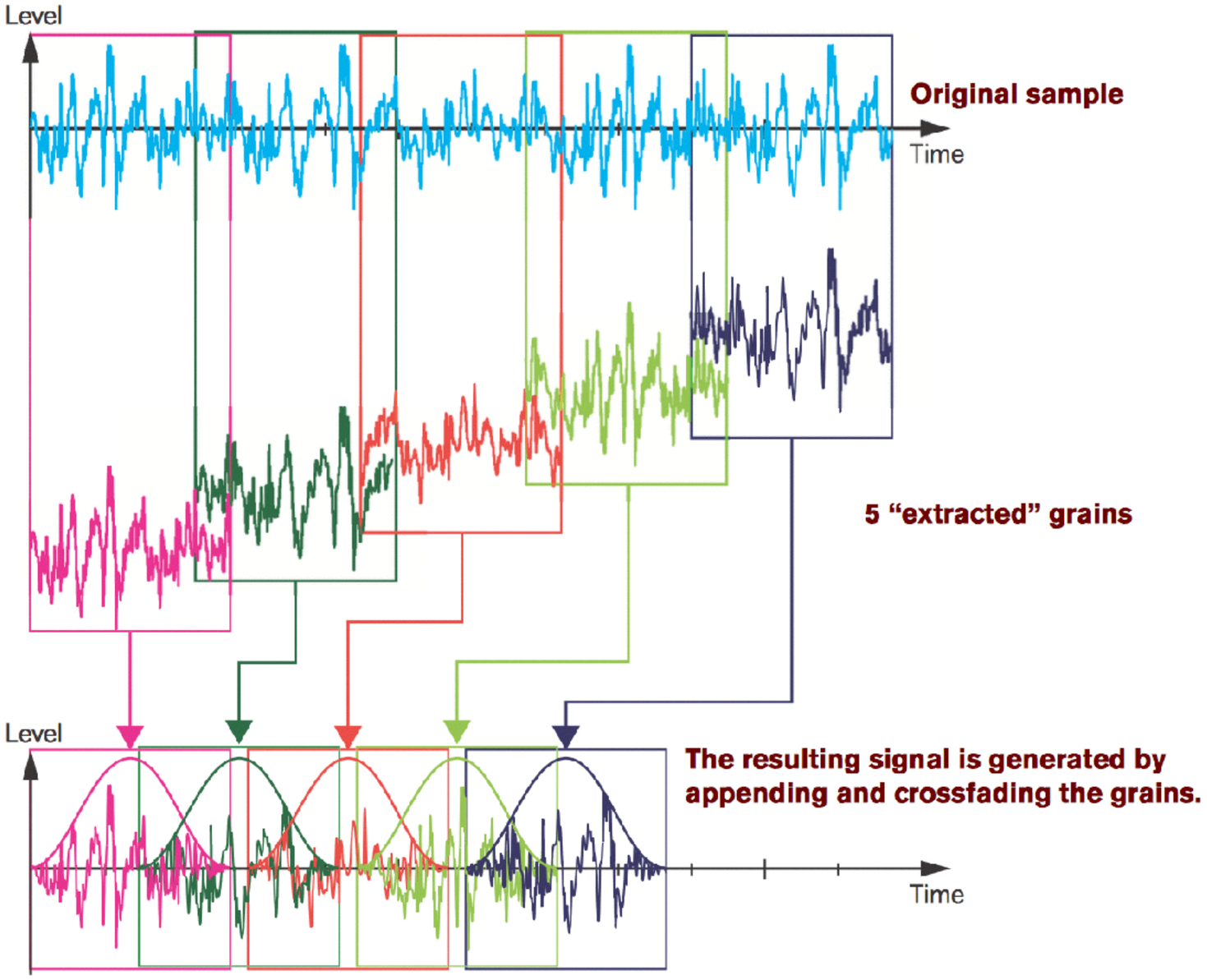
FM Sidebands

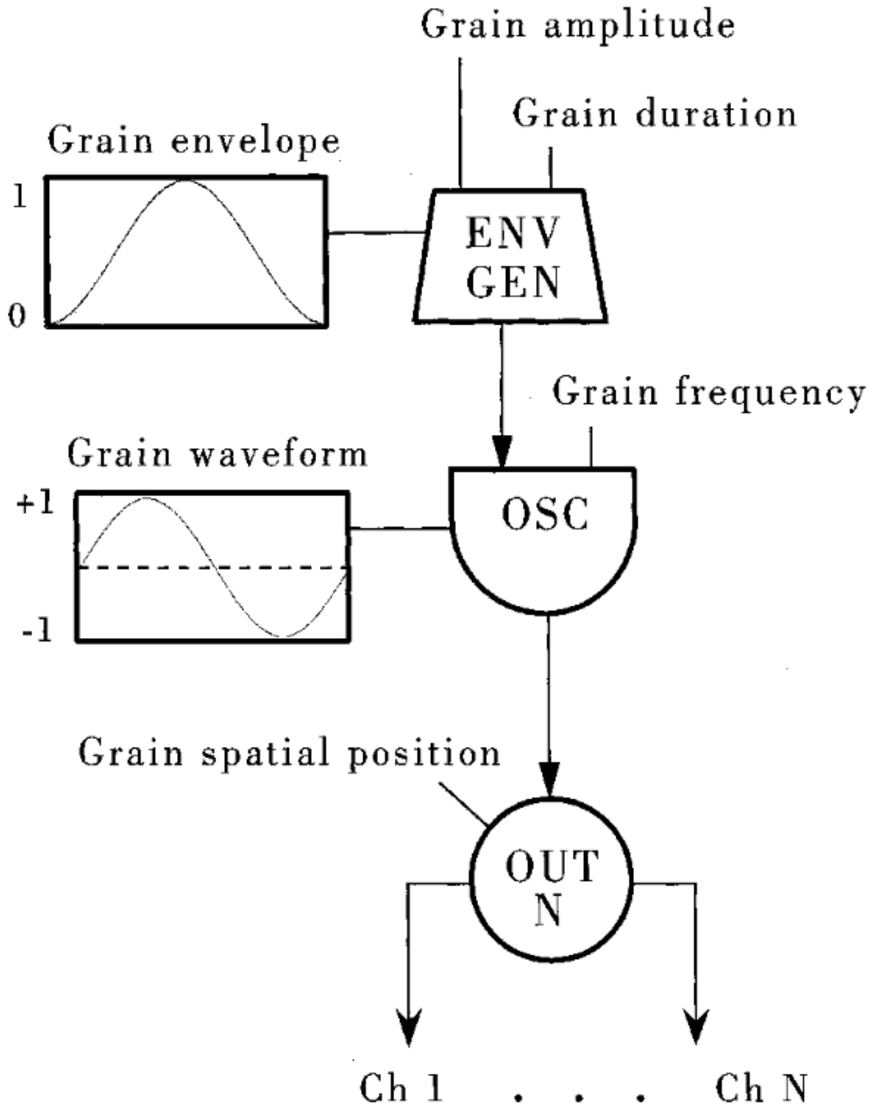


FM Sideband Magnitudes <=> Bessel Functions

Granular Synthesis

- The creation of timbres via the amalgamation of small snippets of windowed waveforms (grains)
- Cases in point: many interesting digital instruments (e.g. crusher-X)
- Considerations: source (real-world of synthetic) grain size, window, synchronicity, density (voices)


Level
Original sample
Time
5 “extracted” grains
Level
The resulting signal is generated by appending and crossfading the grains.
Time



Block diagram of a synchronous granular system

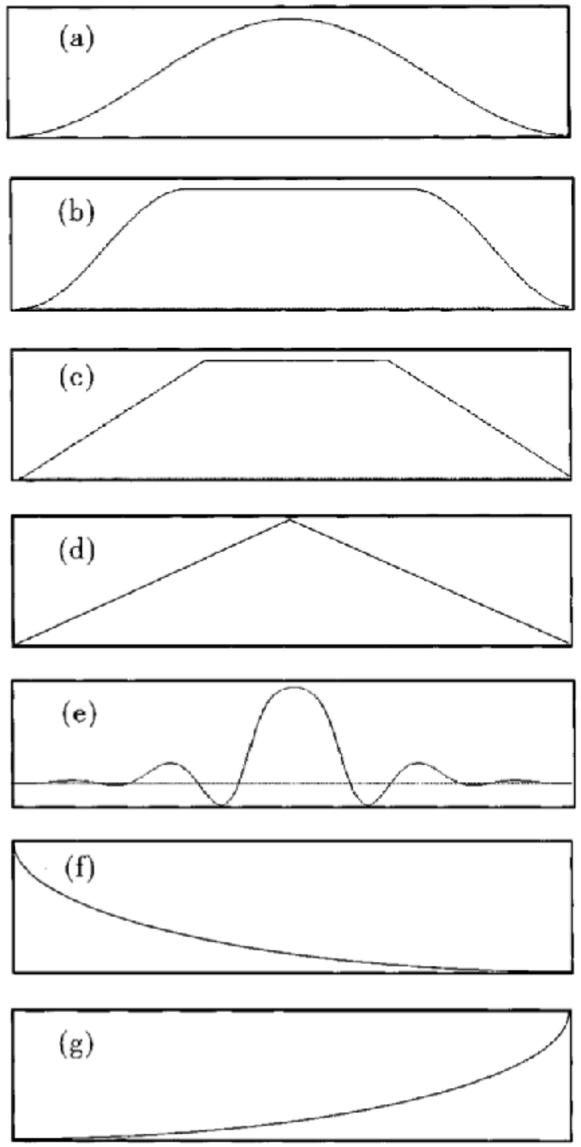


Figure 3.2 Grain envelopes. (a) Gaussian. (b) Quasi-Gaussian. (c) Three-stage line segment. (d) Triangular. (e) Sinc function. (f) Expodec. (g) Rexpodec.

Window envelopes

Types of Granular Synthesis

- Synchronous granular synthesis: grains follow one another at regular intervals
- Quasi-synchronous granular synthesis: random deviation of interval
- Asynchronous granular synthesis: scattering of grains within regions of time-frequency (a la Truax and others)

Now, back to our original programming…

DSP Systems

For our purposes, these systems take in a digital audio signal and output another, transformed digital audio signal.

For example,

a system could take in a recording of speech and output a signal with all of the high frequencies of that speech attenuated (filter)

a system could take in a synthesizer arpeggio and output a signal with all of the loudest peaks of that input reduced (compressor)

Properties of DSP Systems We'll Start With

Deterministic

easily predictable under reasonably simple circumstances

Linear

the sum of their effects is the effect of their sums / the output due to a sum of input signals equals the sum of outputs due to each signal alone

Time Invariant

same output, regardless of when input occurs

LTI Systems = Linear, Time Invariant Systems

Other Important Properties

Causal

output is a function of past and current inputs (not future inputs)

(only matters for real-time applications; offline (e.g. look-ahead), not a problem)

Invertible

can determine input if given output (e.g. deconvolution)

Stable

impulse response is non-infinite (will get to this soon);

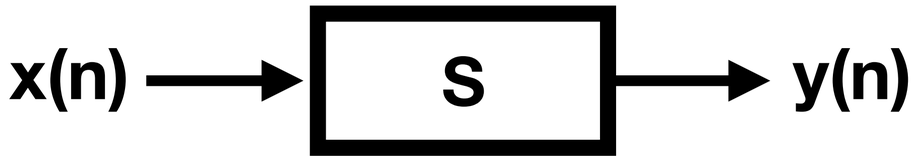
FIR = stable, IIR = maybe stable, but not guaranteed

Static

output is only a function of current input (as opposed to dynamic, which requires memory)

Representing DSP Systems

Signal Flow (or Block) Diagrams



(Difference) Equations

$$y = S(x)$$

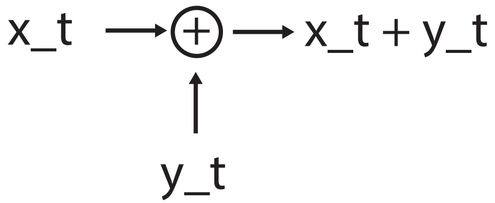
Pseudo-Code

```
audioOutput = applyS(audioInput);
#audioInput is a vector corresponding to samples in audio file
```

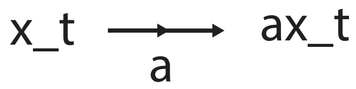
Signal Flow Diagrams

Components

addition/subtraction



multiplication
(by a constant only!)



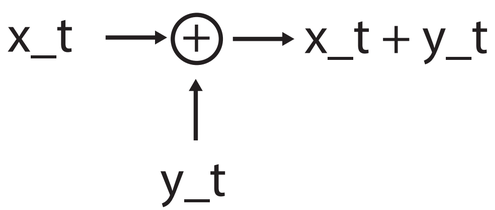
delay

advance

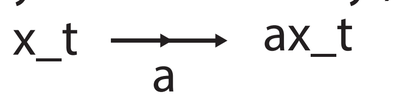
Signal Flow Diagrams

Components

addition/subtraction



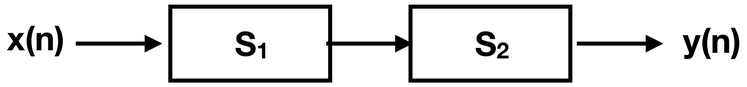
multiplication
(by a constant only!)



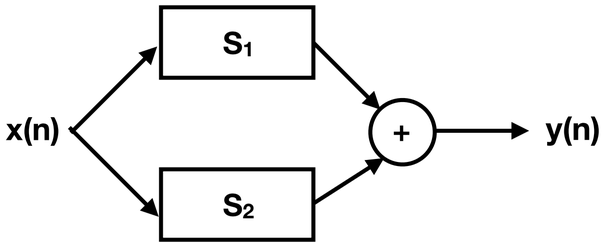
delay

advance

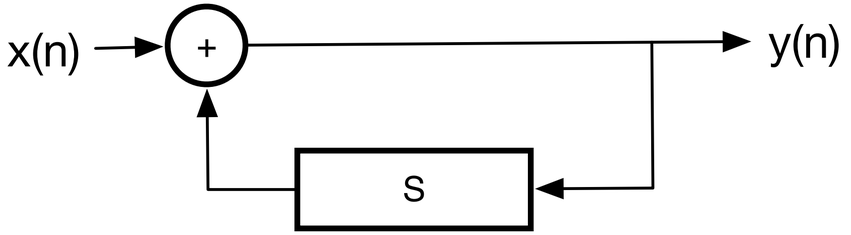
Organizational Schemes



series



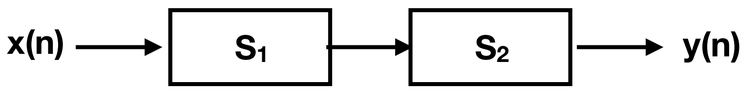
parallel



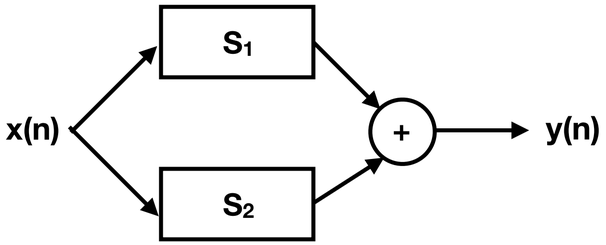
feedback

Signal Flow Diagram <-> Equation

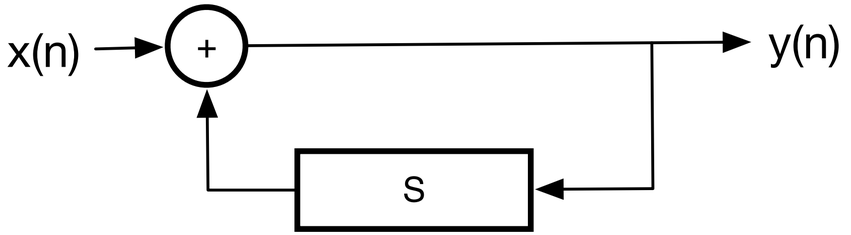
Organizational Schemes



series



parallel

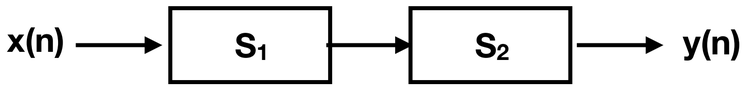


feedback

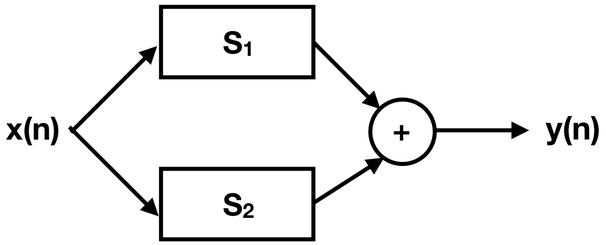
Conversions to Difference Equations (a bit Pseudo-Code-y)

Signal Flow Diagram <-> Equation

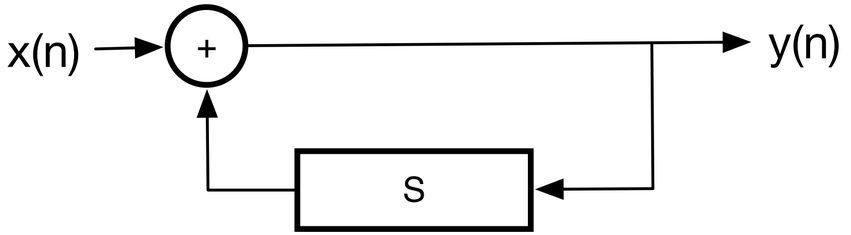
Organizational Schemes



series



parallel



feedback

Conversions to Difference Equations (a bit Pseudo-Code-y)

$y = S_2(S_1(x))$

$y = S_1(x) + S_2(x)$

$y = x + S(y)$

For Next Time:

- Install Octave (free): https://octave.org/
- Or for Mac: https://octave-app.org/
- …or Matlab (possible, but difficult, to get licenses from OCTET) on your machine (don't do this).
- If you want to get a head start, check out the Introduction to Octave at timara.net/dspresources