Computational Methods (PHYS 2030)

Instructors: Prof. Christopher Bergevin (cberge@yorku.ca)

Schedule: Lecture: MWF 11:30-12:30 (CLH M)

Website: http://www.yorku.ca/cberge/2030W2018.html

York University
Winter 2018
Lecture 20

Fourier analysis

- Deep history throughout mathematics, physics, engineering, biology, ...

- Backbone of modern signal processing and linear systems theory

- Lays at foundation of many modern methodologies in medical imaging (e.g., MRI, CT scans)

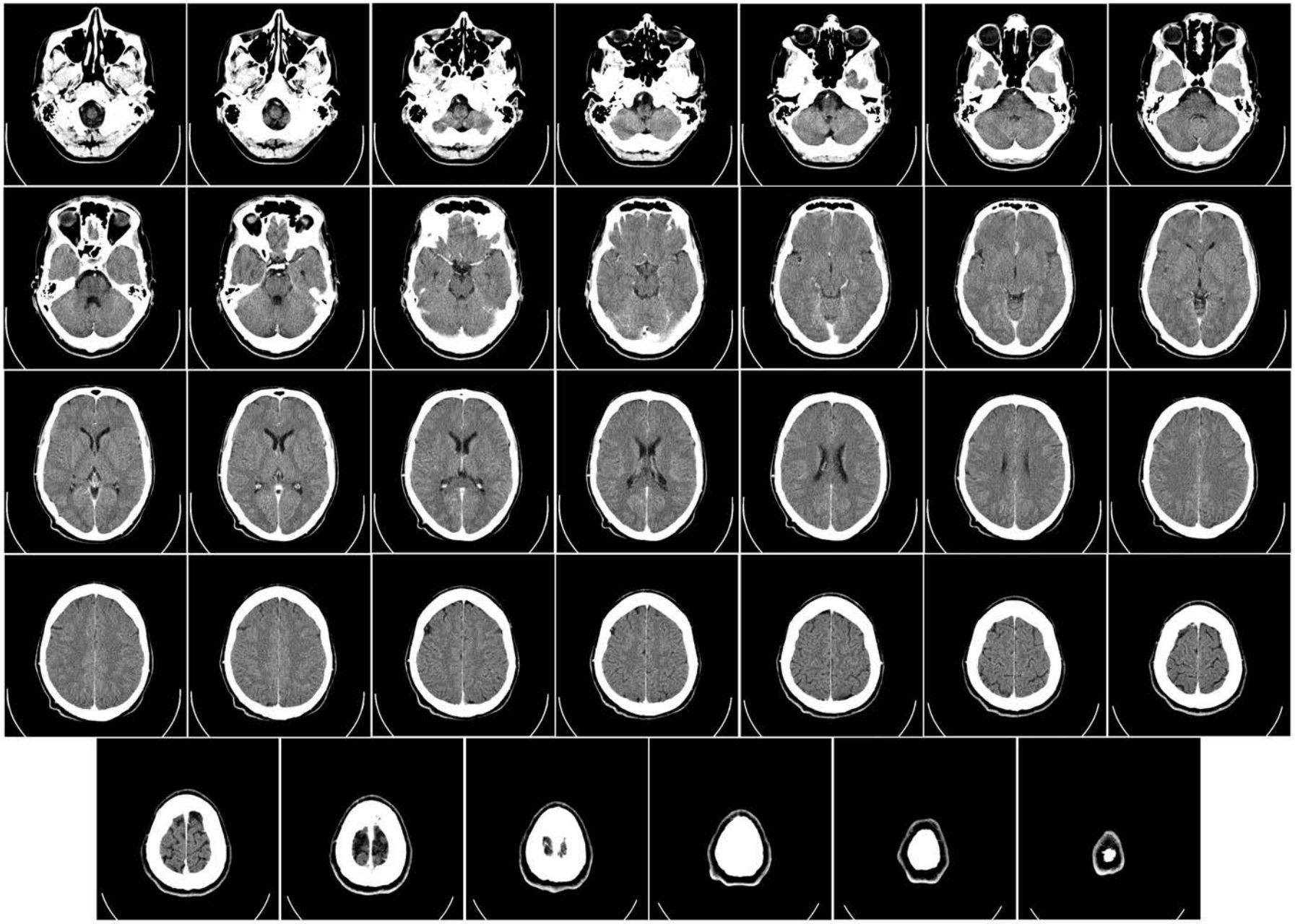
- Builds off the basic idea of a *Taylor series* (which posits we can describe a function as an infinite series of polynomials)

Joseph Fourier (1768-1830)

Basic idea: Represent 'signal' as a sum of sinusoids

Note: We focus on 1-D here for clarity, but these ideas generalize to higher dimensions (e.g., 2-D for images)

Motivation: Medical Imaging (e.g., NMR/MRI)



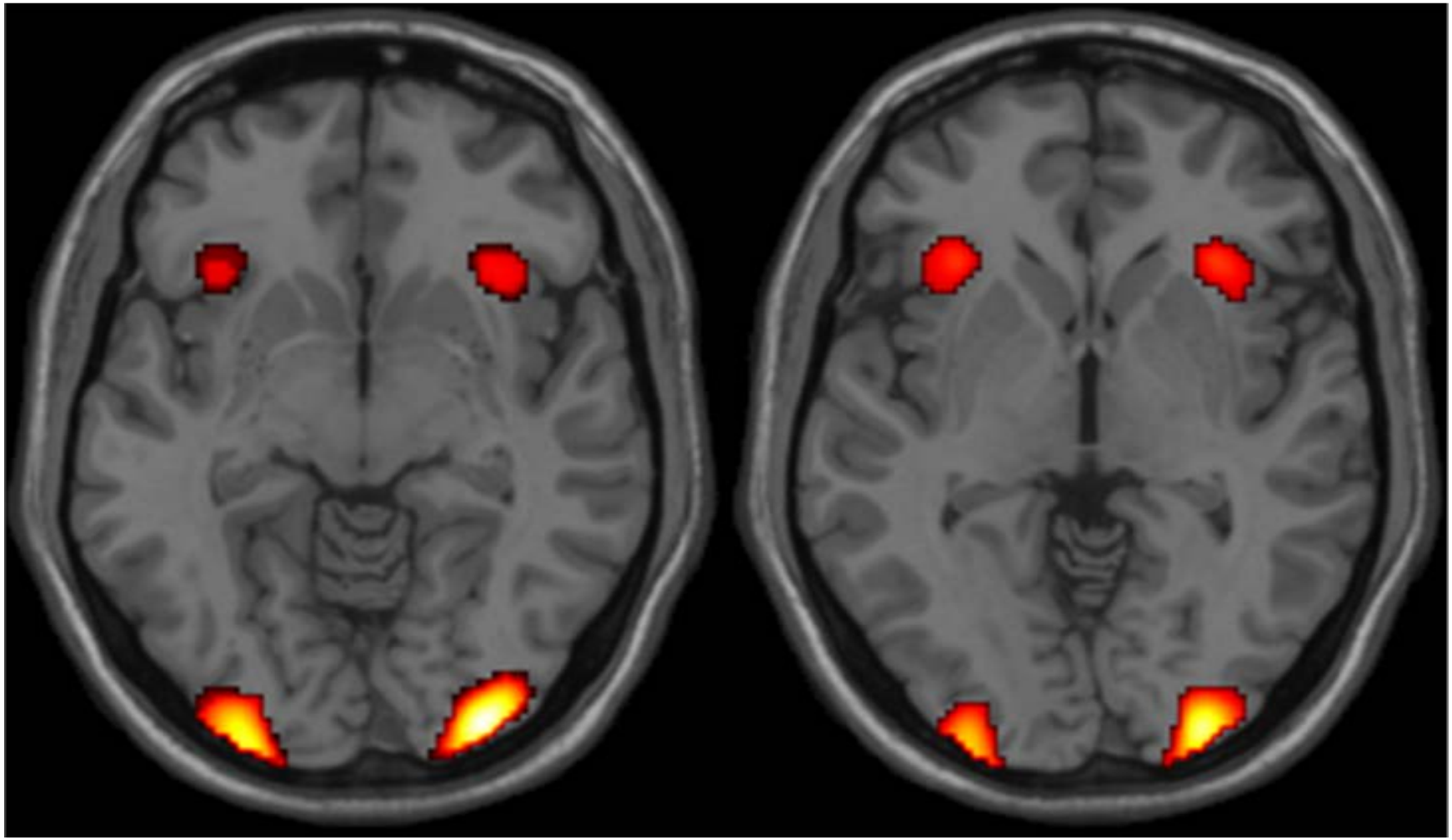
Wikipedia

Motivation: Medical Imaging (e.g., NMR/MRI)

→ A key foundation for imaging is a Fourier transform ("k-space")

Wikipedia

Motivation: Medical Imaging (e.g., NMR/MRI)

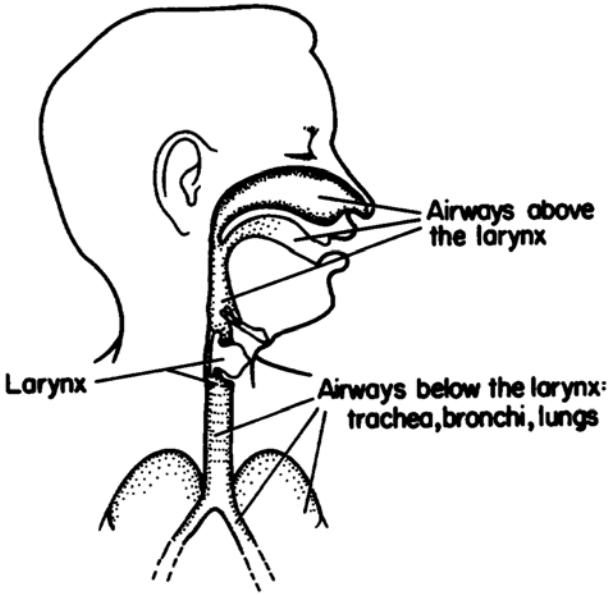


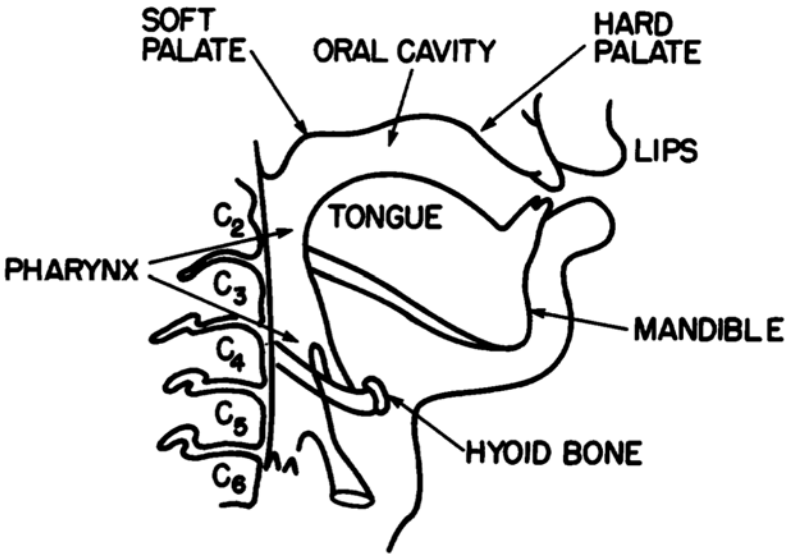
Motivation: Medical Imaging (e.g., NMR/MRI)

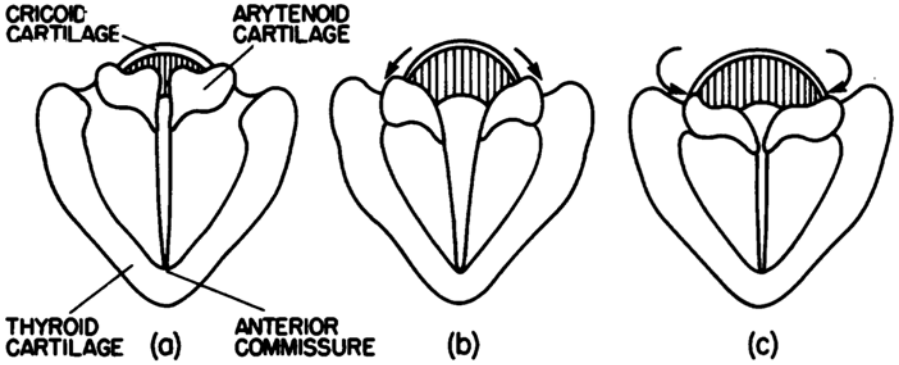
http://brainimaging.waisman.wisc.edu/facilities

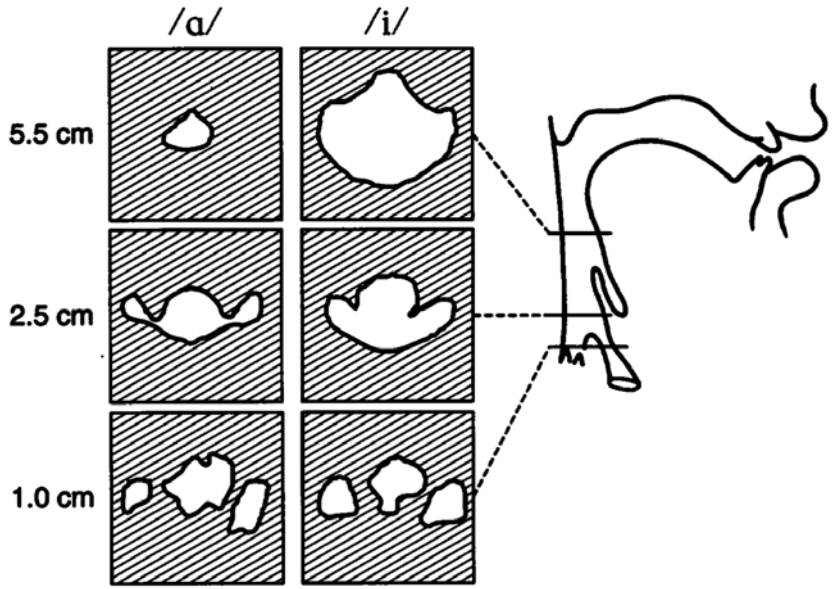

Getty

Motivation: Speech





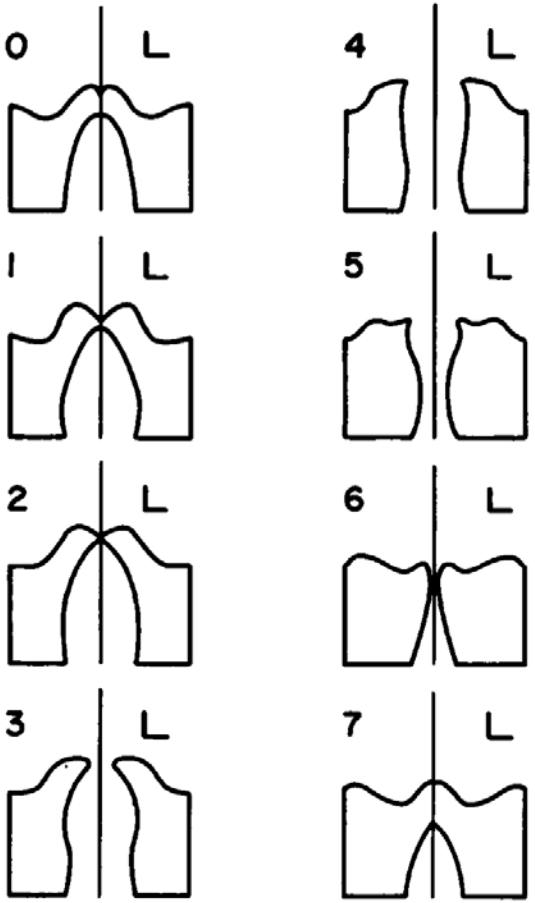


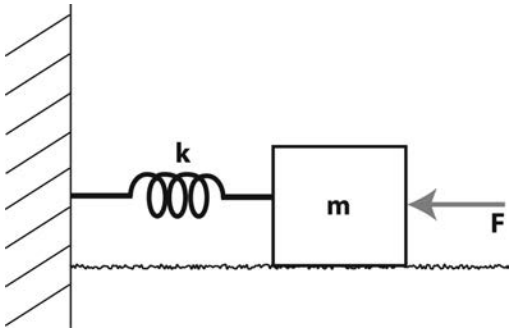


Stevens (2000)

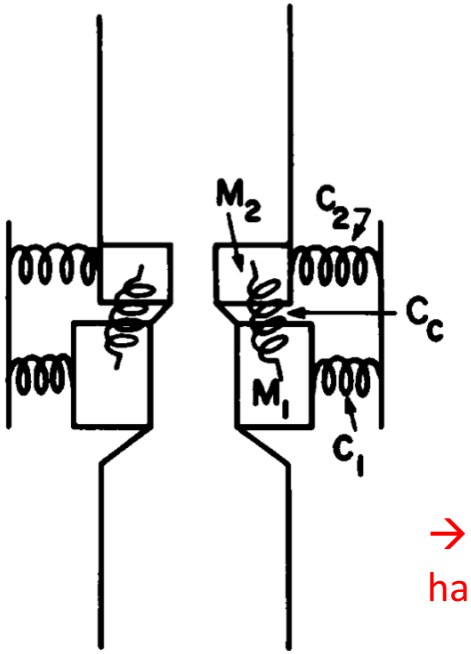
Motivation: Speech

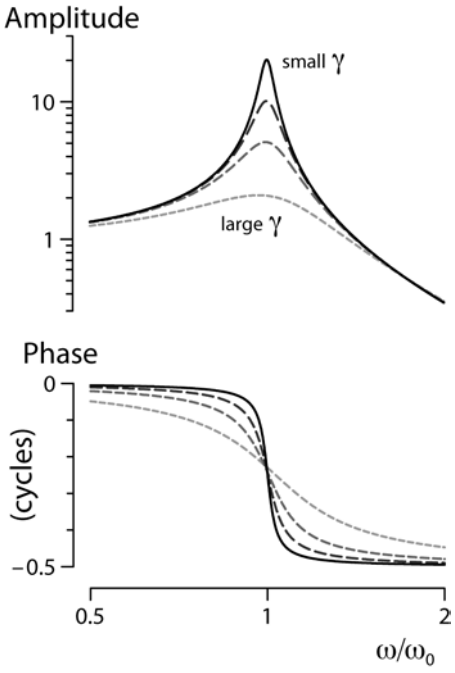
Snapshots of vibrating vocal folds





Simple two-mass model for the vocal folds



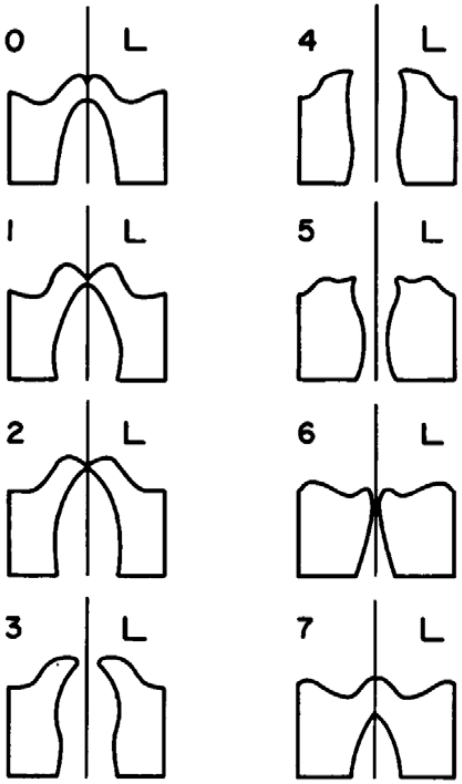


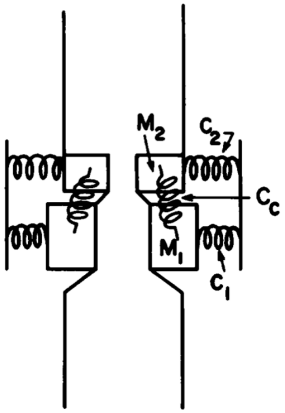
→ Vocal folds behaves like a harmonic oscillator (HO)!

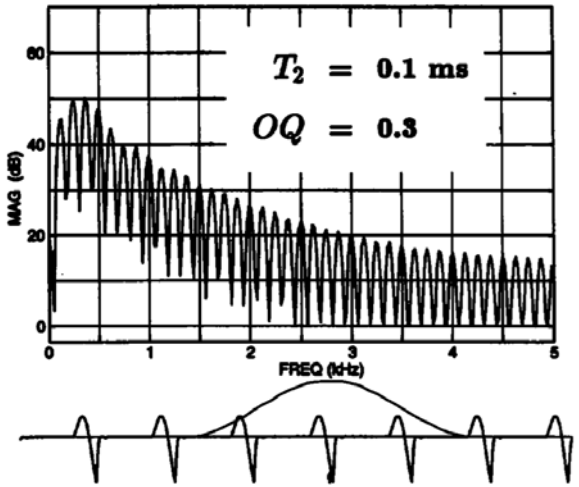
Reminder: We can describe HO in terms of 'spectral' response

Stevens (2000)

Motivation: Speech

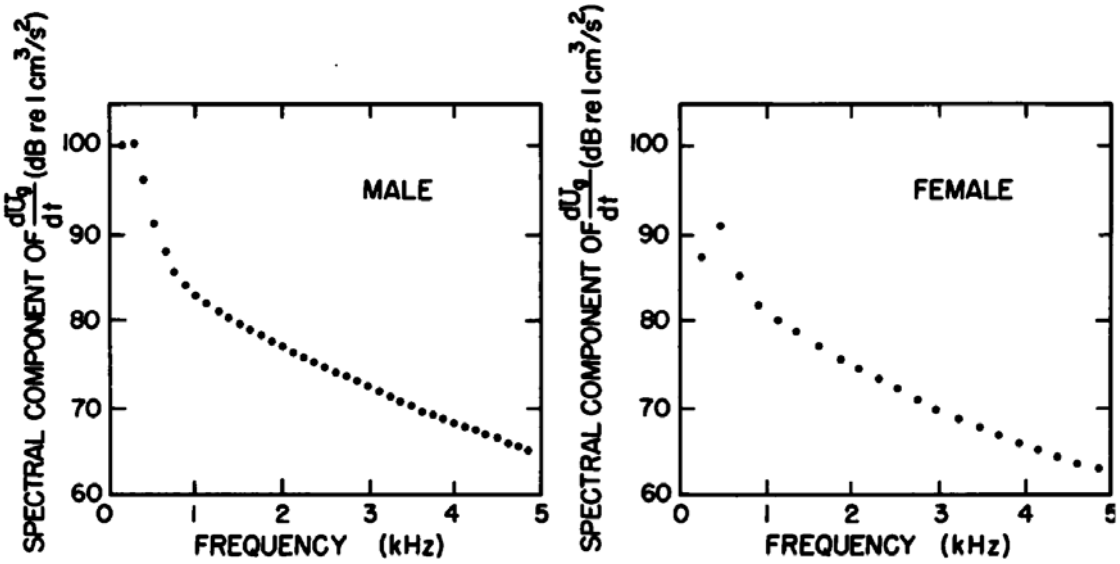






Key idea:
Spectrum
→ x-axis is frequency [Hz]

→ Vibrating vocal folds give off 'buzzy' sound due to harmonics



→ Males have lower 'fundamental' (due to more massive vocal folds)

Stevens (2000)

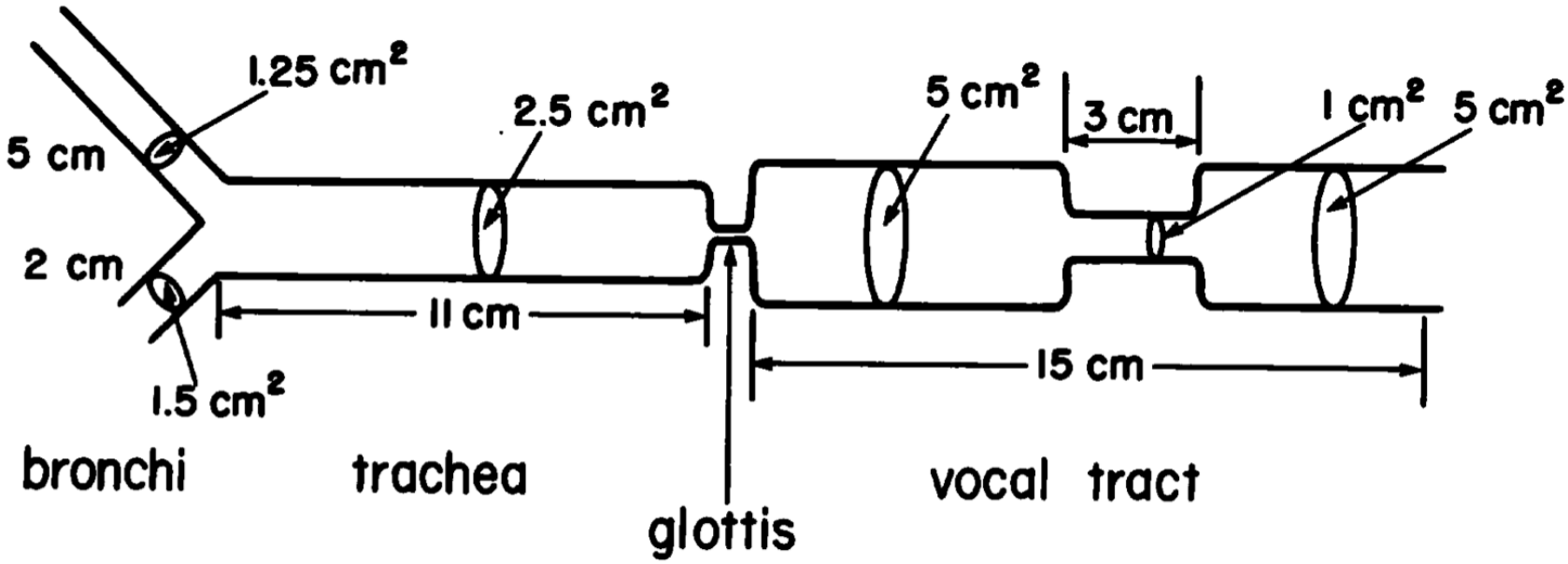
Motivation: Speech

Pressure source (lungs)

Vibration source (vocal folds)

Filter (vocal tract)

Output (mouth)



→ Complex acoustic process is boiled down to a relatively simple/tractable framework of 'sources' and tubes!

Stevens (2000)

Motivation: Speech

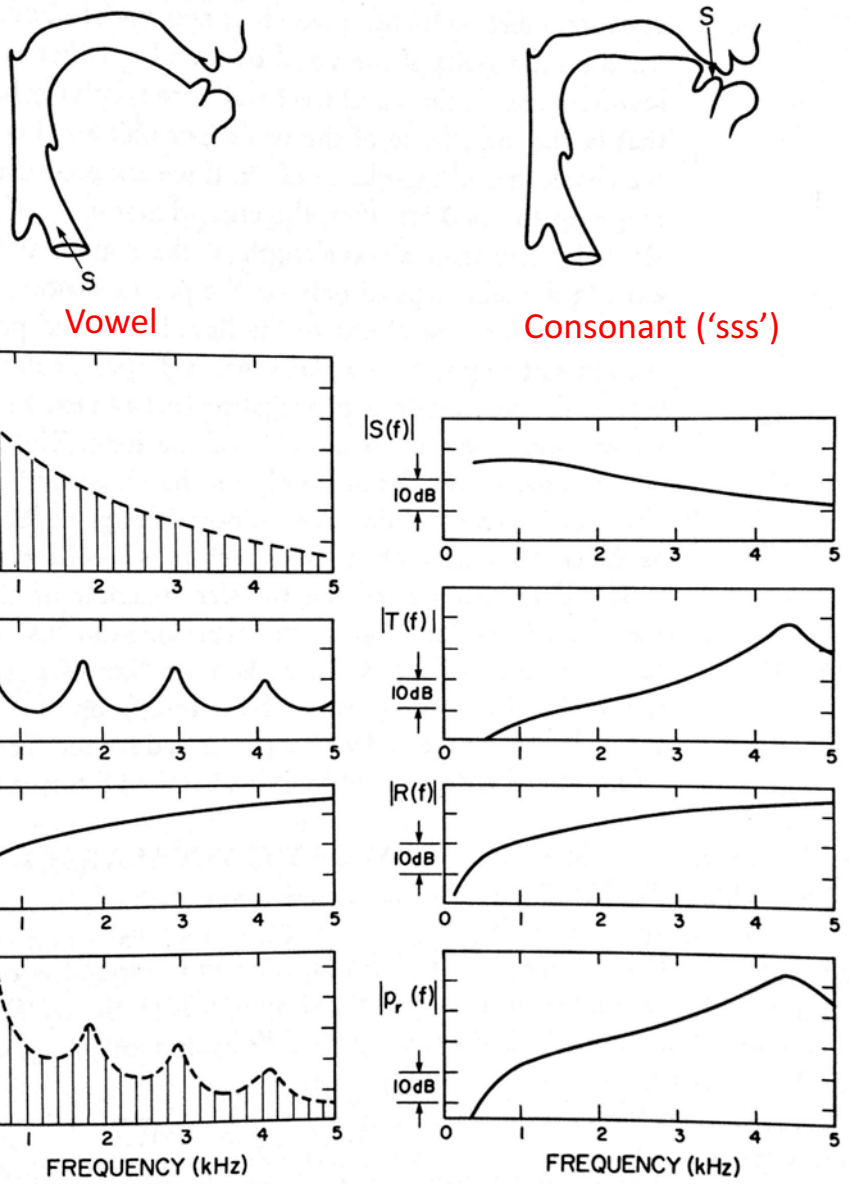


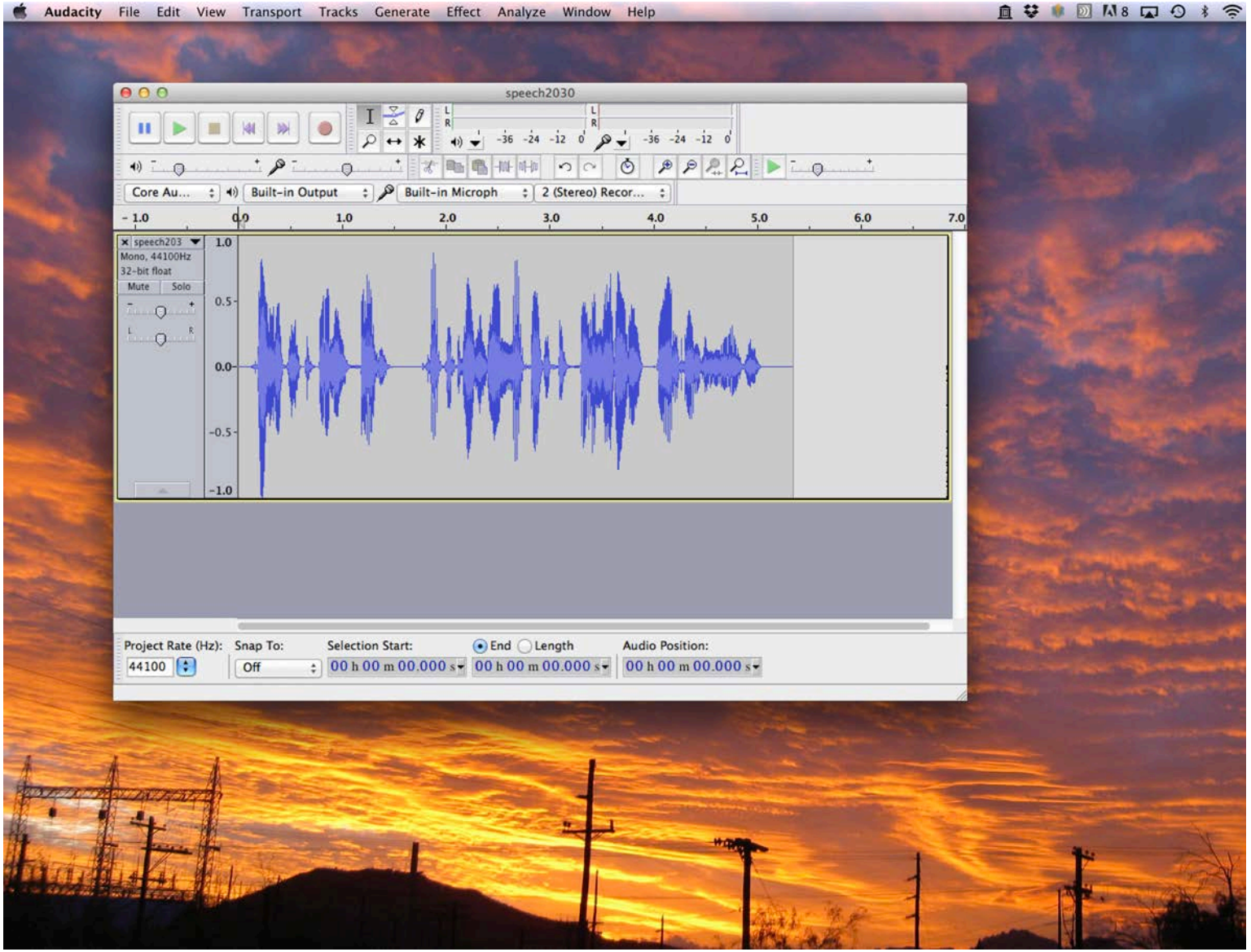
Figure 3.1 Sketches indicating components of the output spectrum $|p_r(f)|$ for a vowel and a fricative consonant. The output spectrum is the product of a source spectrum $S(f)$, a transfer function $T(f)$, and a radiation characteristic $R(f)$. The source spectra are similar to those derived in figures 2.10 and 2.33 in chapter 2. For the periodic source, $S(f)$ represents the amplitudes of spectral components; for the noise source, $S(f)$ is amplitude in a specified bandwidth. See text.

- Vibrating vocal folds make 'broadband' sound
- Vocal tract shapes that sound
- Resulting 'shape' emphasizes features which we then pick up with our ear (e.g., formants of vowels)

→ But what does it mean for everything to be a function of frequency?

Stevens (2000)

Aside: Recording sound



Note: Many freeware programs exist for recording sound (Matlab lets you do it on PCs)

Aside: Recording sound

- Several basic ingredients:
  - sound source
  - microphone
  - A/D converter (e.g., laptop, Arduino)
  - software (e.g., Matlab, C, LabView, ...)

- Think about physically what each 'step' does (e.g., microphone transduces by either inductive or capacitive changes, thereby creating an electric current)

- Sound (i.e., pressure fluctuations) are thereby converted to voltage signals

- For a 'mono' signal, this is a 1-D system (i.e., voltage is a function of time)

- A continuous signal when digitized becomes discrete (i.e., 'sampled')

Aside: Discrete vs Continuous

Start w/ some familiar ideas:

- Quantum vs. Classical mechanics
- Wave-particle duality
- Statistical mechanics

1. We ‘live’ in a world that is simultaneously discrete and continuous

2. Such a reality flavors how we ‘measure’ anything

Aside: Sampling

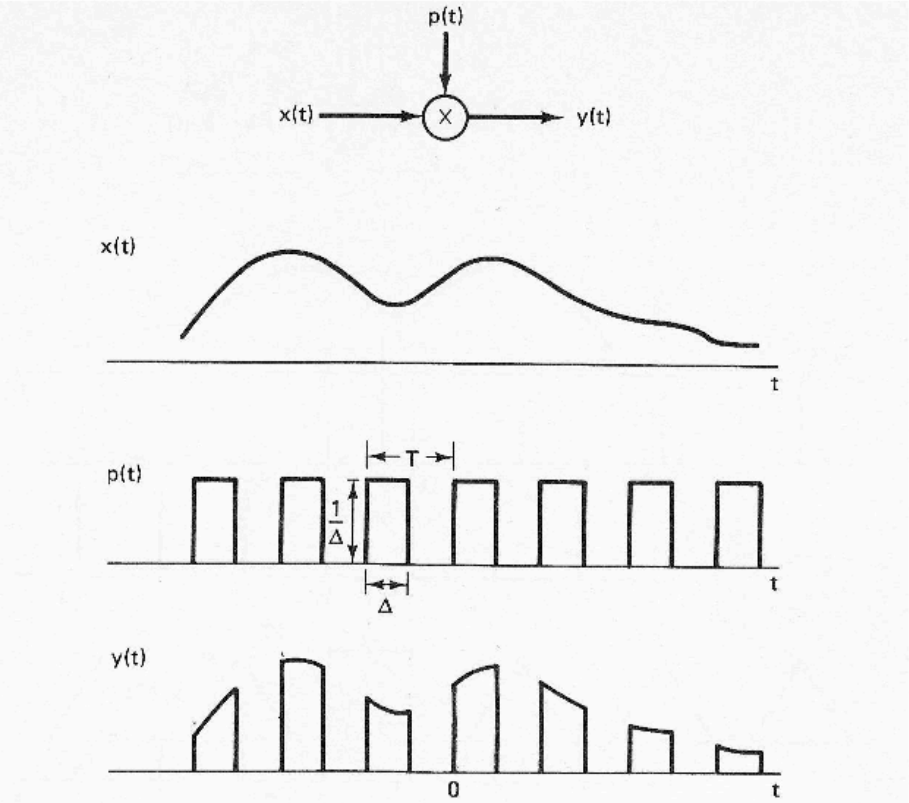


Figure 8.2 Pulse amplitude modulation. As $\Delta \longrightarrow 0$, $p(t)$ approaches an impulse train.

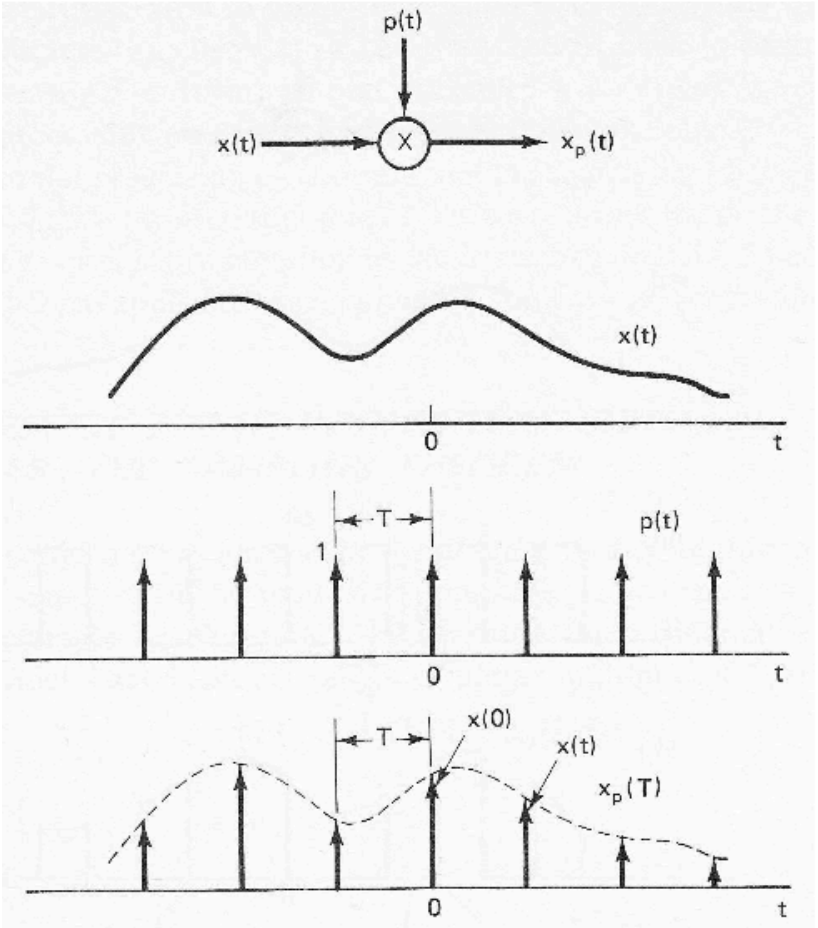


Figure 8.3 Pulse amplitude modulation with an impulse train.

'Snapshots' (i.e., sampling)

Very short snapshots' → Delta functions

Oppenheim & Willsky

Aside: Sampling

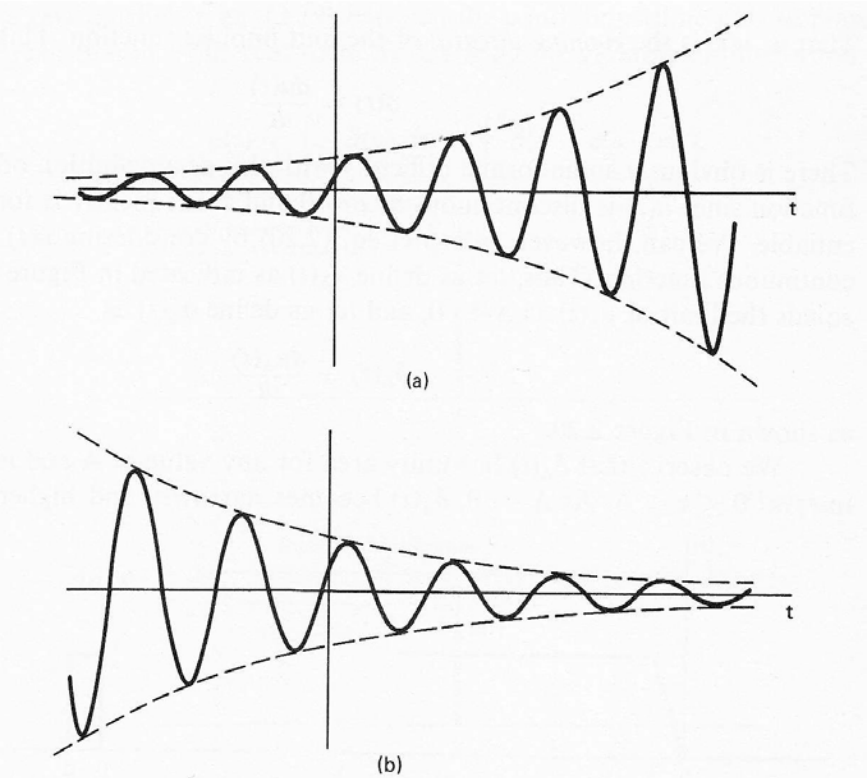


Figure 2.17 (a) Growing sinusoidal signal $x(t) = Ce^{rt}\cos(\omega_0 t + \theta)$, $r > 0$; (b) decaying sinusoid $x(t) = Ce^{rt}\cos(\omega_0 t + \theta)$, $r < 0$.

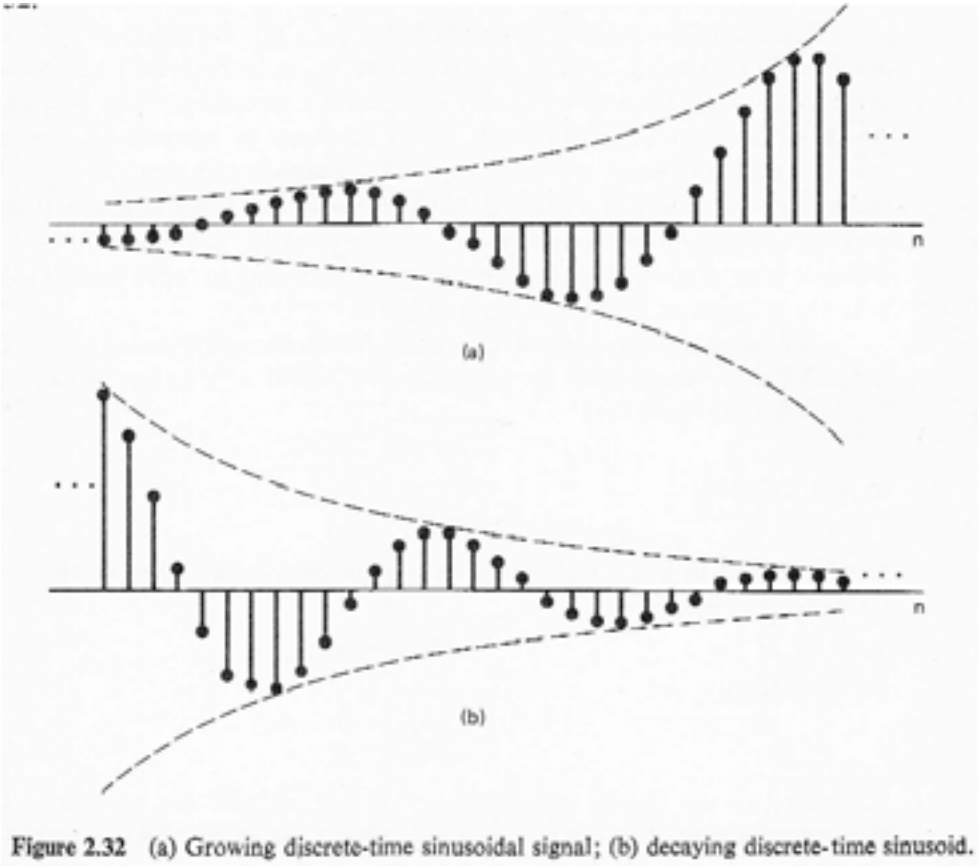


Figure 2.32 (a) Growing discrete-time sinusoidal signal; (b) decaying discrete-time sinusoid.

Continuous signal (analog)

Discretized signal (digital)

→ Typically when we measure a signal, we measure a discretized version of such (which is both similar and different!)

Oppenheim & Willsky

Aside: Sampling

- Note that there is some sort of timing associated with our sampling

- That is, there is a *sampling rate* associated with converting from analog to digital

ex. compact discs use a sample rate (SR) of 44.1 kHz

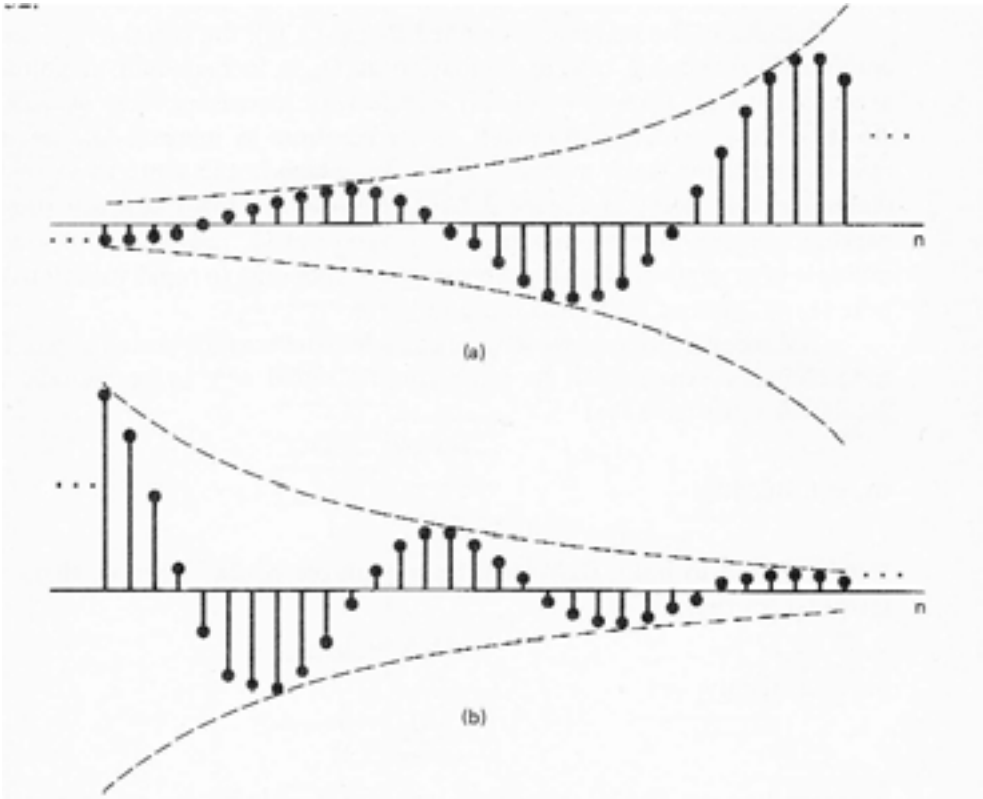


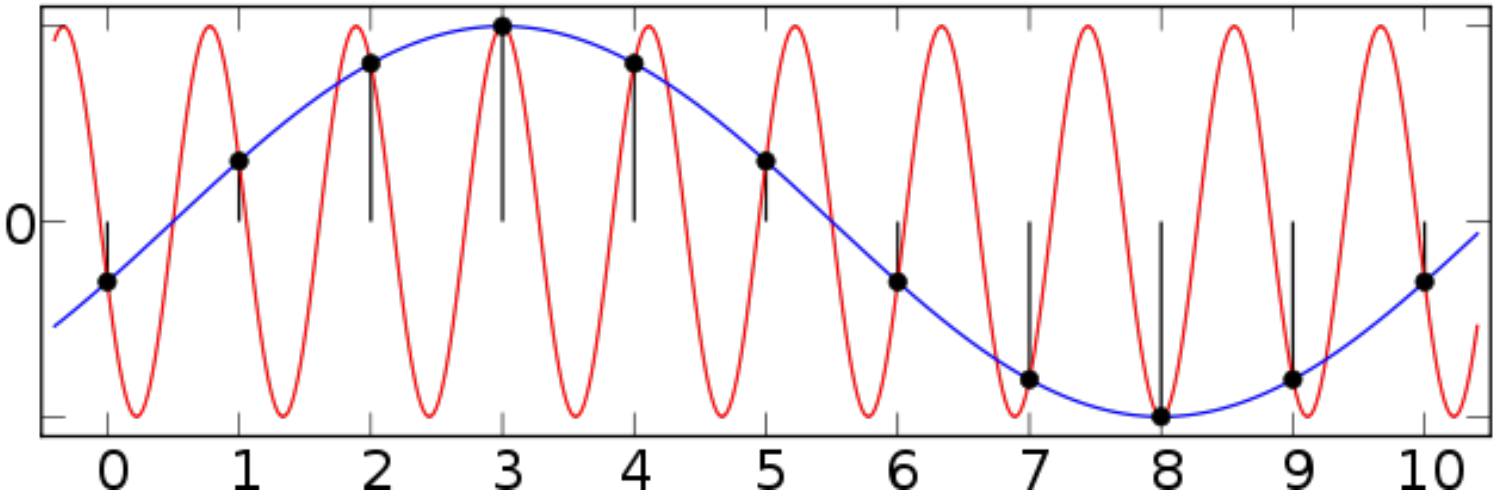
Figure 2.32 (a) Growing discrete-time sinusoidal signal; (b) decaying discrete-time sinusoid.

- The faster we sample, the more information we capture (to a point)

Oppenheim & Willsky

Aside: Aliasing

- Be careful that your sample rate is not too low...

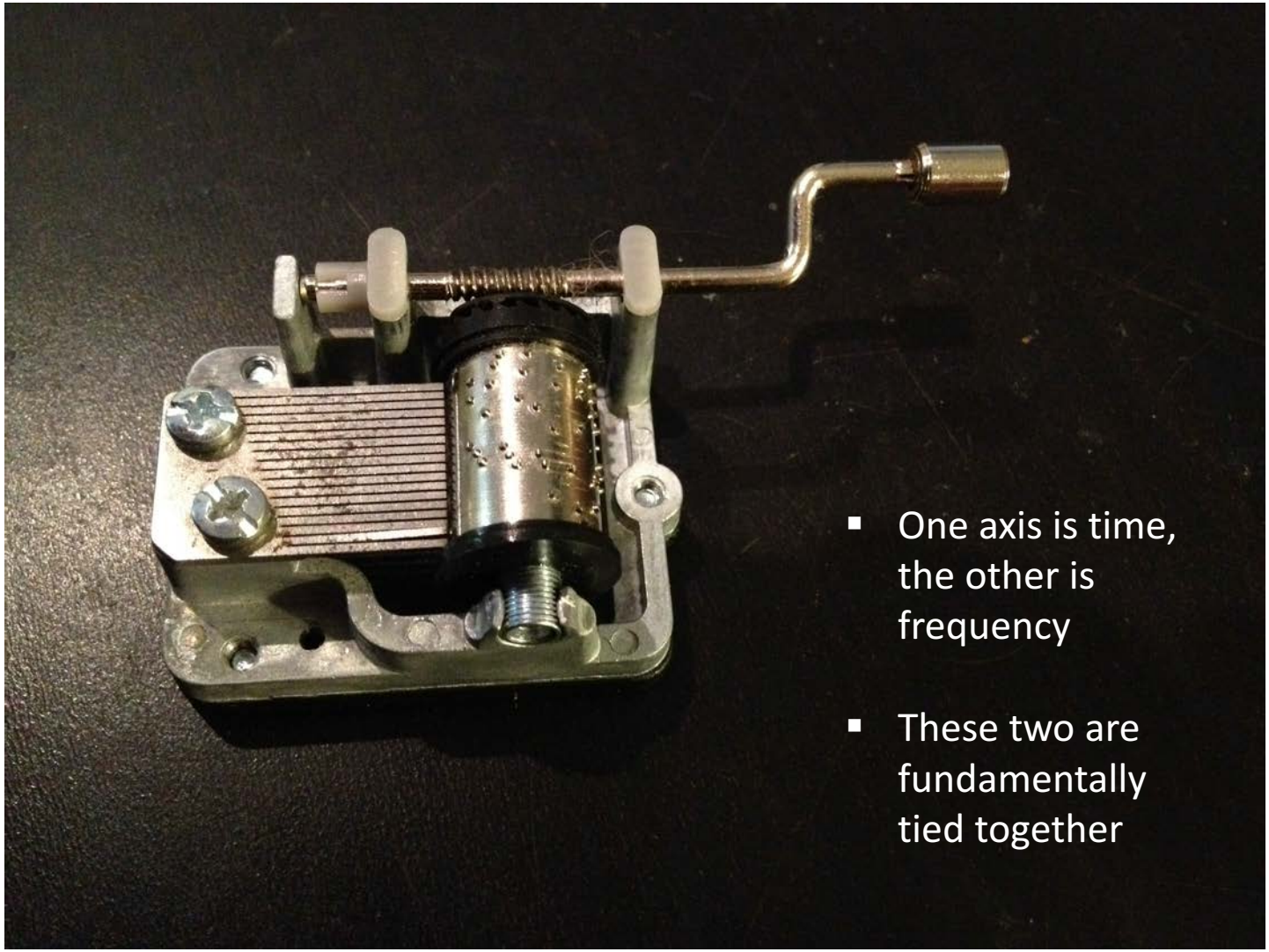


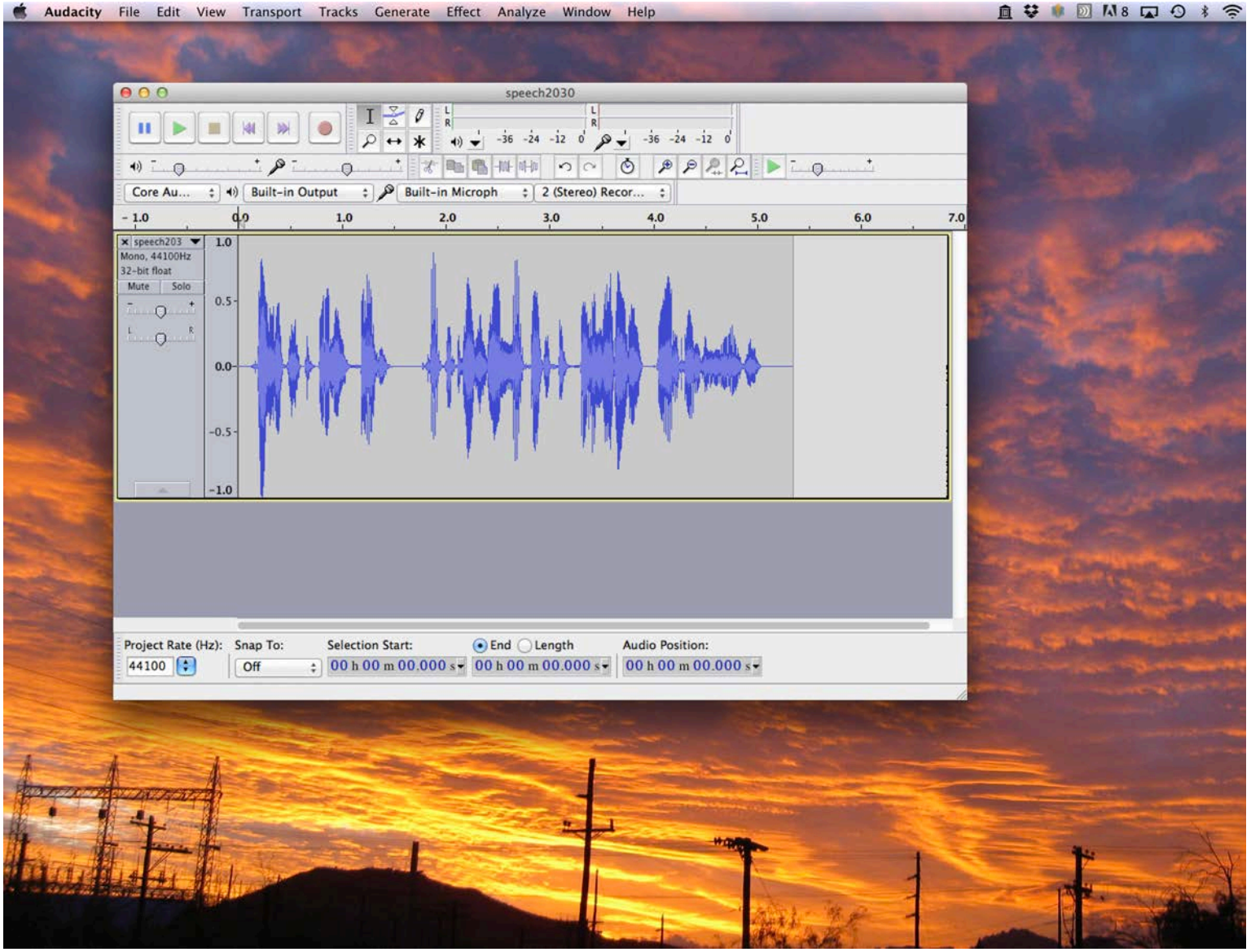
→ “Aliasing”

wikipedia (aliasing)

Key idea: Fourier transform

- Allows one to go from a time domain description (e.g., our recorded mic signal) to a spectral description (i.e., what frequency components make up that signal)

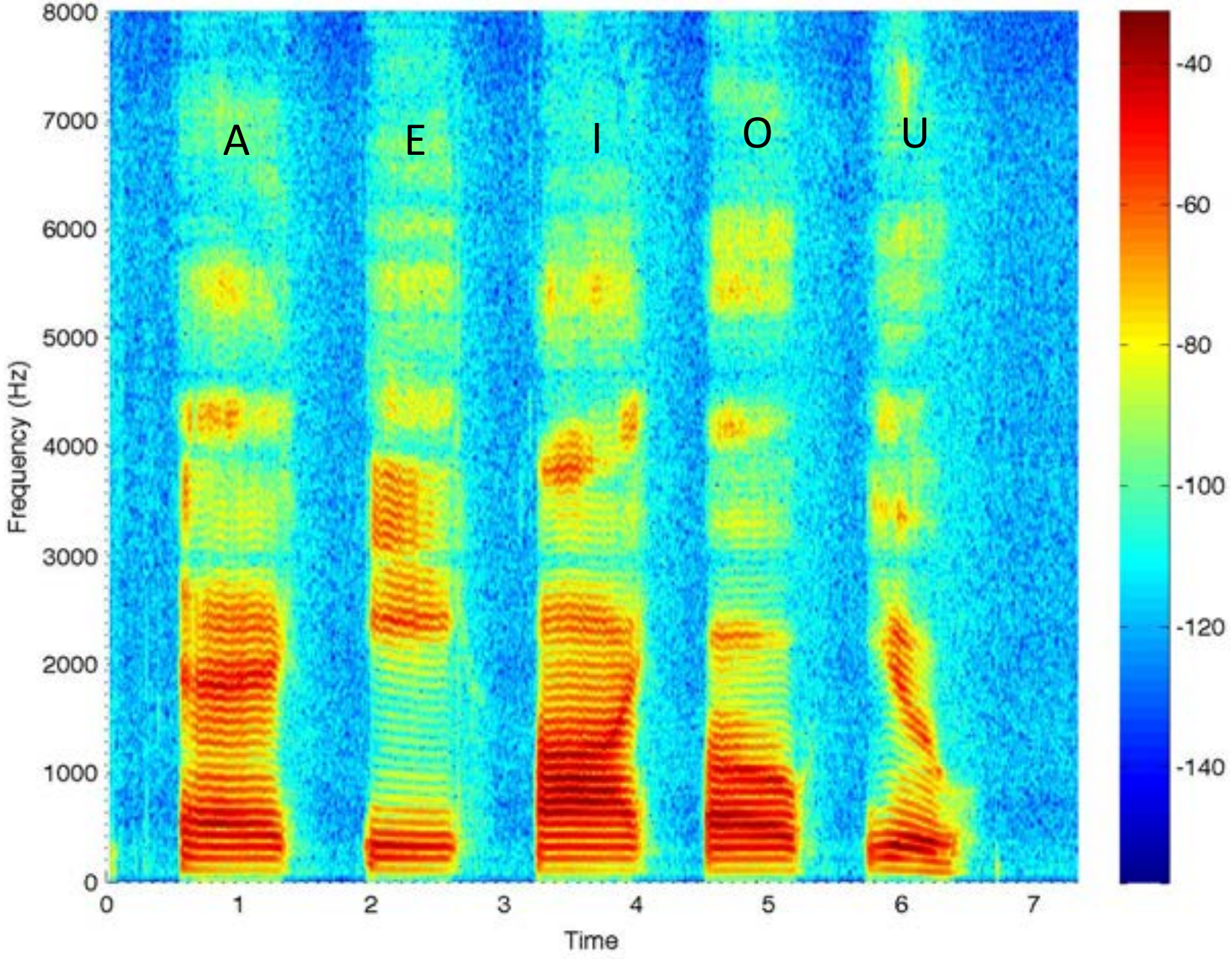




→ We are going to perform a specific type of spectral analysis called the 'Short Time Fourier Transform' (STFT) to make what is called a *spectrogram*

```
function y = makeSpectrogram(file)
% ### EXspectrogram.m ###          10.27.14
% Reads in wav file created via separate program (e.g., Audacity) and makes a spectrogram
% NOTE: make sure sample rate specified here matches that used when recording the data!

% -------
P.SR= 44100;  % SR data collected at [Hz]
P.windowL= 2048;   % length of window segment for FFT {2048}
P.overlap= 0.8;    % fractional overlap between window, from 0 to 1 {0.8}
P.maxF= 8000;   % max. freq. for spectrogram [Hz] {8000}
fileN= './spectrogram2030.jpg';    % filename to save image to
% -------
pts= round(P.windowL*P.overlap);      % convert fractional overlap to # of points
disp(sprintf('Assumed sample rate = %g kHz', P.SR/1000));
A= wavread(file);
spectrogram(A,blackman(P.windowL),pts,P.windowL,P.SR,'yaxis');  % create spectrogram and plot (via
built-in function)
axis([0 size(A,1)/P.SR 0 P.maxF])
colorbar

% -------
% save picture to file as a jpg w/ a user-specified resolution
REZ= '-r180';     % resolution for exporting colormaps to jpg
print('-djpeg',REZ,[fileN]);


% NOTE: to play back the audio, type:
% > sound(A,SR)
% where SR is the appropriate sample rate (e.g., fiddle with if you want to
% change the pitch)

% NOTE: To save an array (A) to .wav file, type:
% > wavwrite(A,SR,16,filename);
```

→ Requires pre-recorded sound file


A
E
I
O
U
8000
7000
6000
5000
4000
3000
2000
1000
0
Frequency (Hz)
0
1
2
3
4
5
6
7
Time
-40
-60
-80
-100
-120
-140

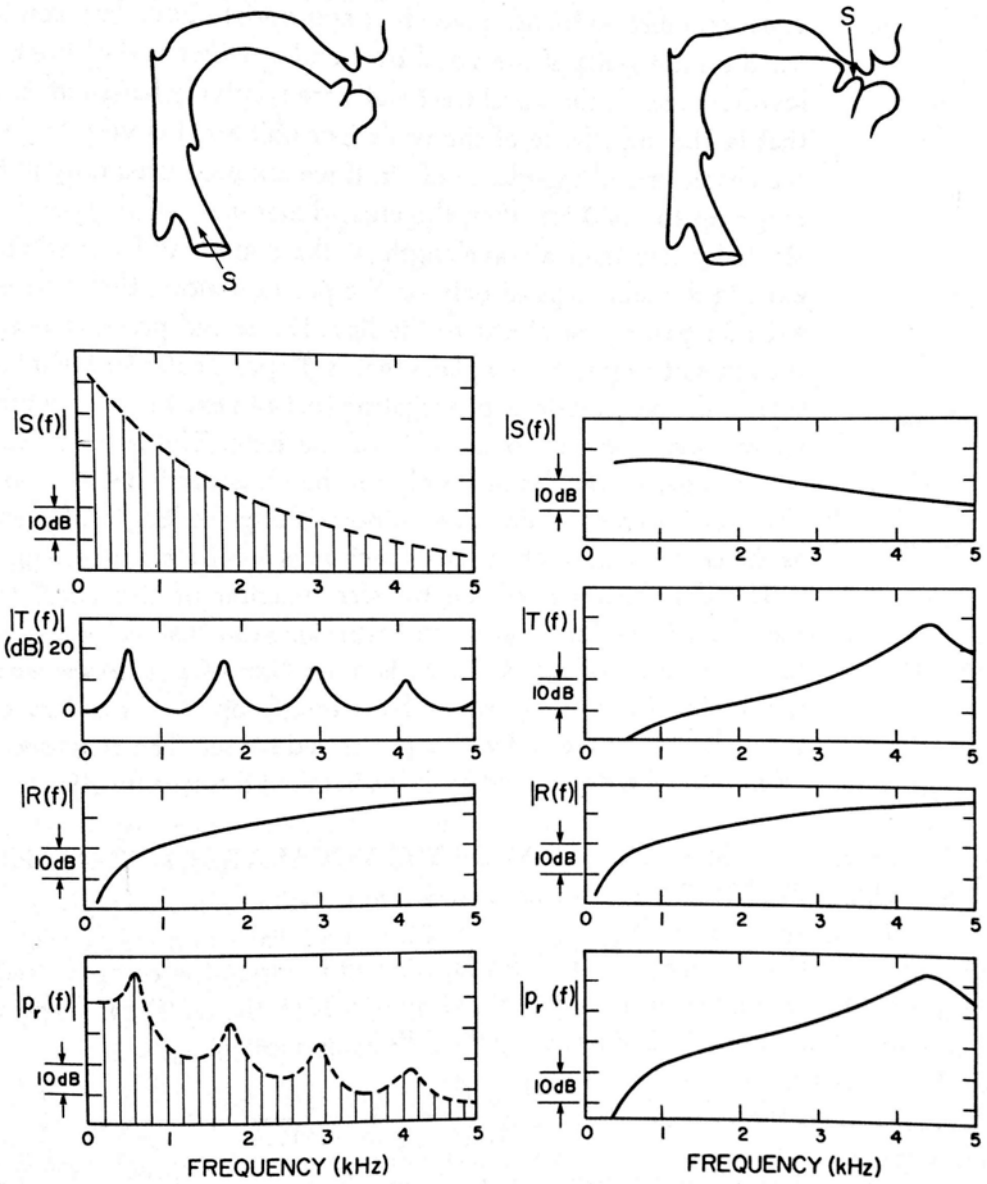
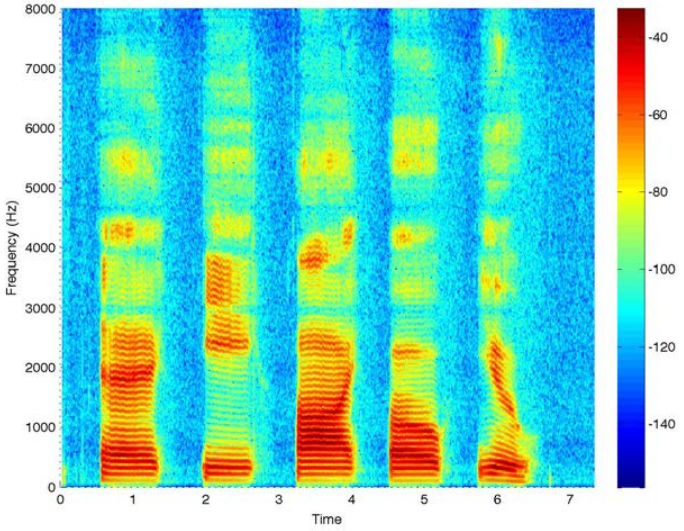
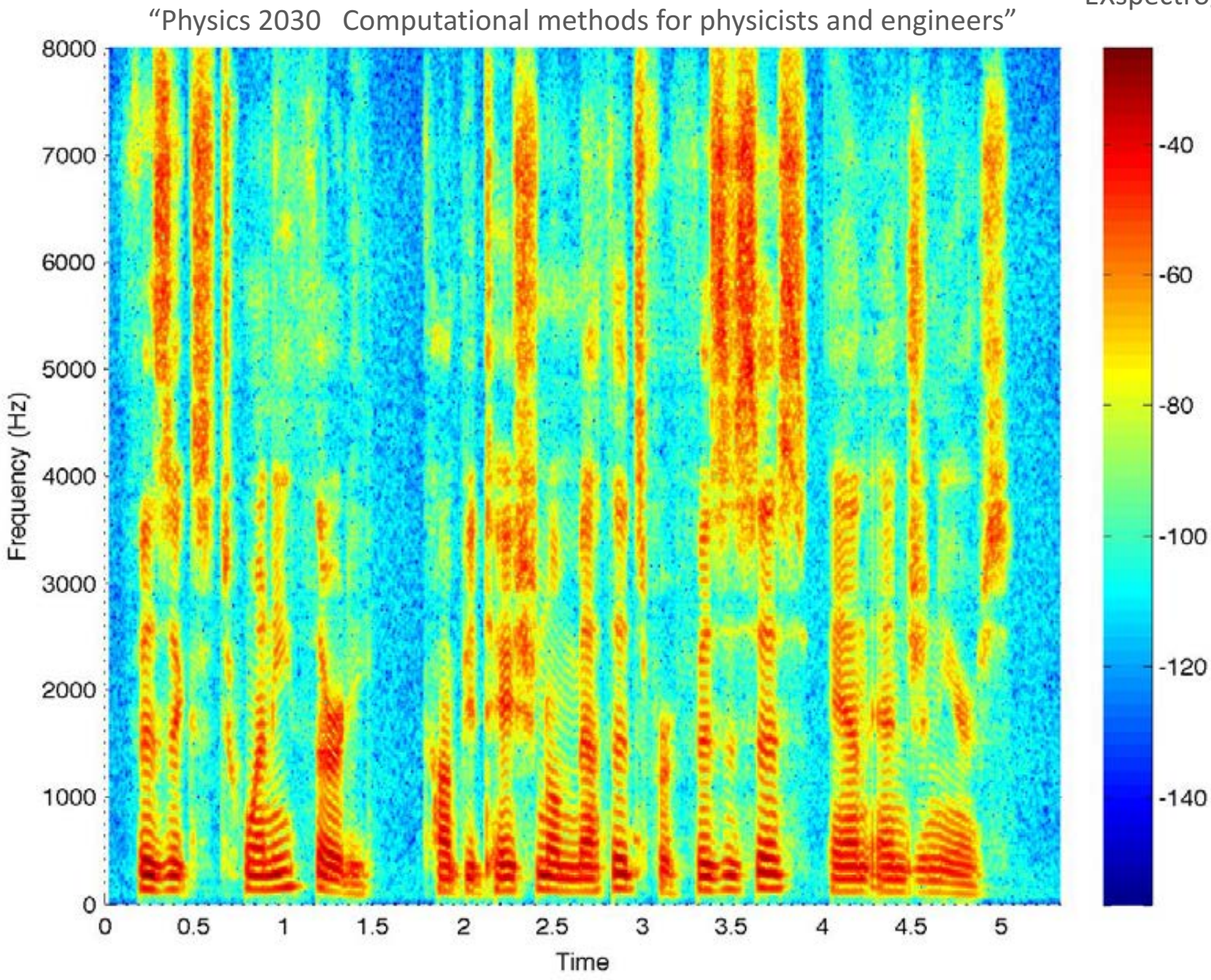


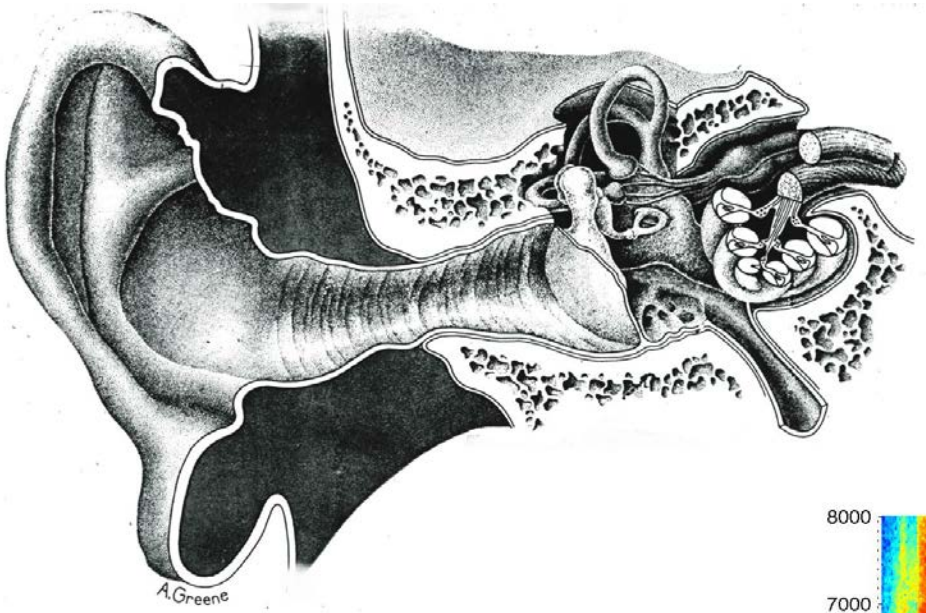
Figure 3.1 Sketches indicating components of the output spectrum $|p_r(f)|$ for a vowel and a fricative consonant. The output spectrum is the product of a source spectrum $S(f)$, a transfer function $T(f)$, and a radiation characteristic $R(f)$. The source spectra are similar to those derived in figures 2.10 and 2.33 in chapter 2. For the periodic source, $S(f)$ represents the amplitudes of spectral components; for the noise source, $S(f)$ is amplitude in a specified bandwidth. See text.

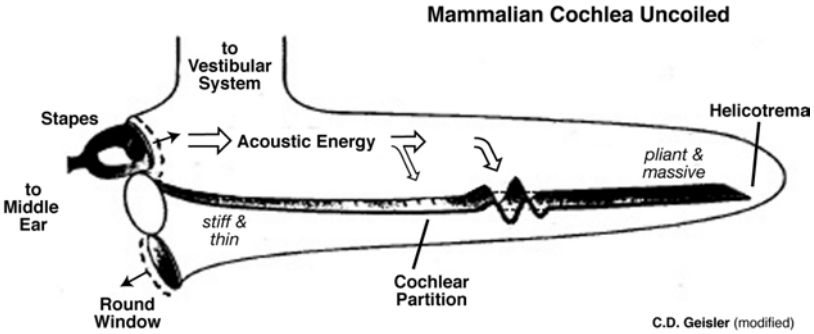


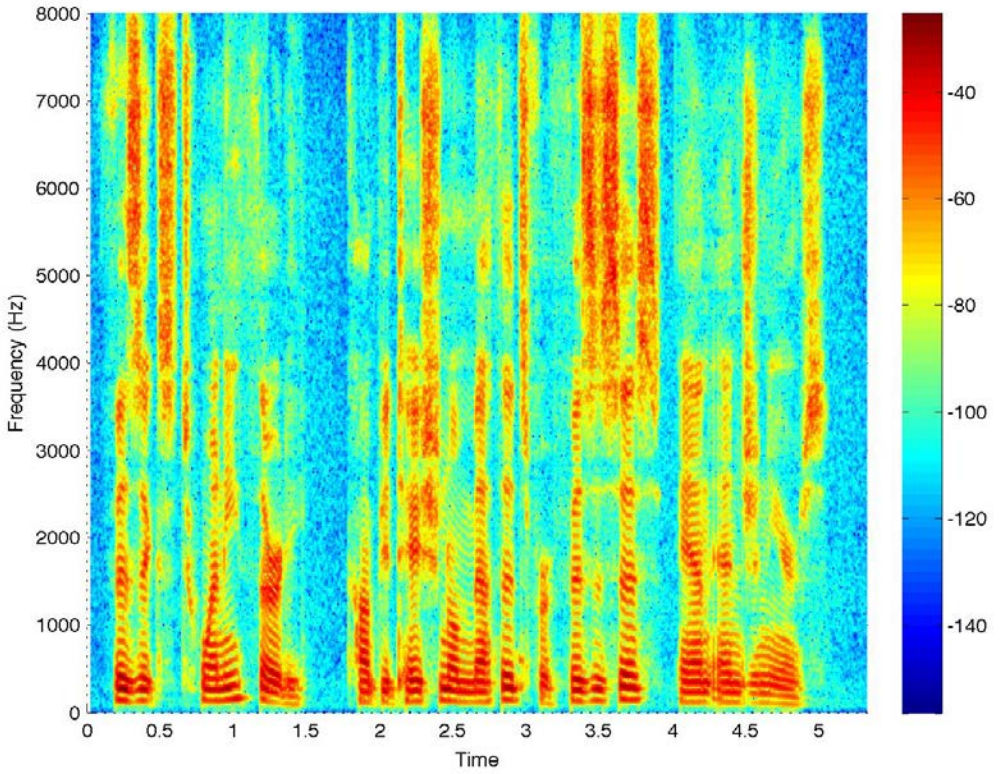
Stevens (2000)

EXspectrogram.m









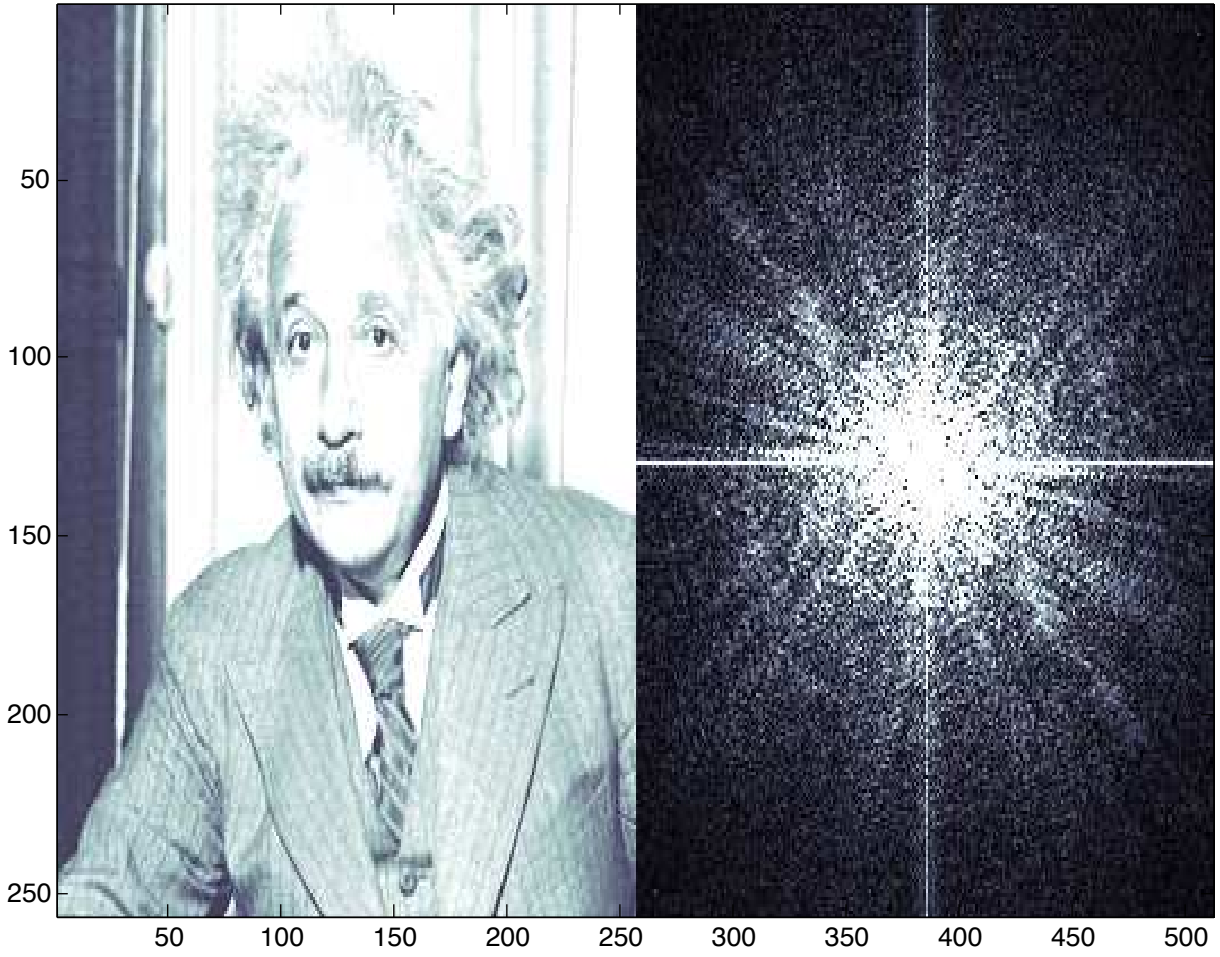
→ The ear's main job to to do this sort of spectral decomposition!

```
% ### EXfourier2D.m ###     10.27.14
% Shows a dynamic reconstruction of an image stemming from adding
% successively higher order terms from a 2D Fourier transform
% [source: Richard Murray, York CVR]
clear;
% ------------
fileI= 'Einstein.tif';    % image to analyze
N = 4;  % set number of radial sections per step
rstep= 1/2;     % set radial step size going outward (larger means smaller steps)
delay= 0.1; % animation delay [s]
Mmovie= 0;  % make a movie version (avi) (0-no,1-yes)
% ------------
im = double(imread(fileI)); % load image
imft = fftshift(fft2(im));  % compute 2D FFT
% +++
dim = size(im,1);
ftmax = max(abs(imft(:)));  % determine largest amplitude (for scaling)
[r,t] = matrt([ dim dim ]); % make distance map (via external function)
f = find(t<0);
t(f) = t(f)+pi;
t = t/pi;
rmap = floor(power(r,rstep)/.6);
dmap = rmap+t;
F = ceil(N*max(dmap(:)));
% +++
imC = uint8(127*[ im/255 500*abs(imft)/ftmax ]); % write initial frame (whole image and FFT)
figure(1); clf; colormap(bone)
image(imC); drawnow;
% +++
% make movie as well
if Mmovie==1
    obj = VideoWriter('fourier_synth_einstein.avi');  % open movie file
    open(obj); writeVideo(obj,imC);
end
% +++
% step through frames
figure(2); clf; colormap(bone);
for f=1:F
    mask = (dmap<(f/N));    % create low-pass filter mask
    imftmask = imft.*mask;  % filter image by applying mask in freq. domain
    immask = real(ifft2(ifftshift(imftmask)))/255;  % inverse Fourier transform
    imC = uint8(127*[ max(min(immask,1),0) 500*abs(imftmask)/ftmax ]);  % update image
    image(imC); drawnow;    pause(delay);
    if (Mmovie==1), writeVideo(obj,imC); end
end
```

As an example, Fourier transforms are not limited to 1-D

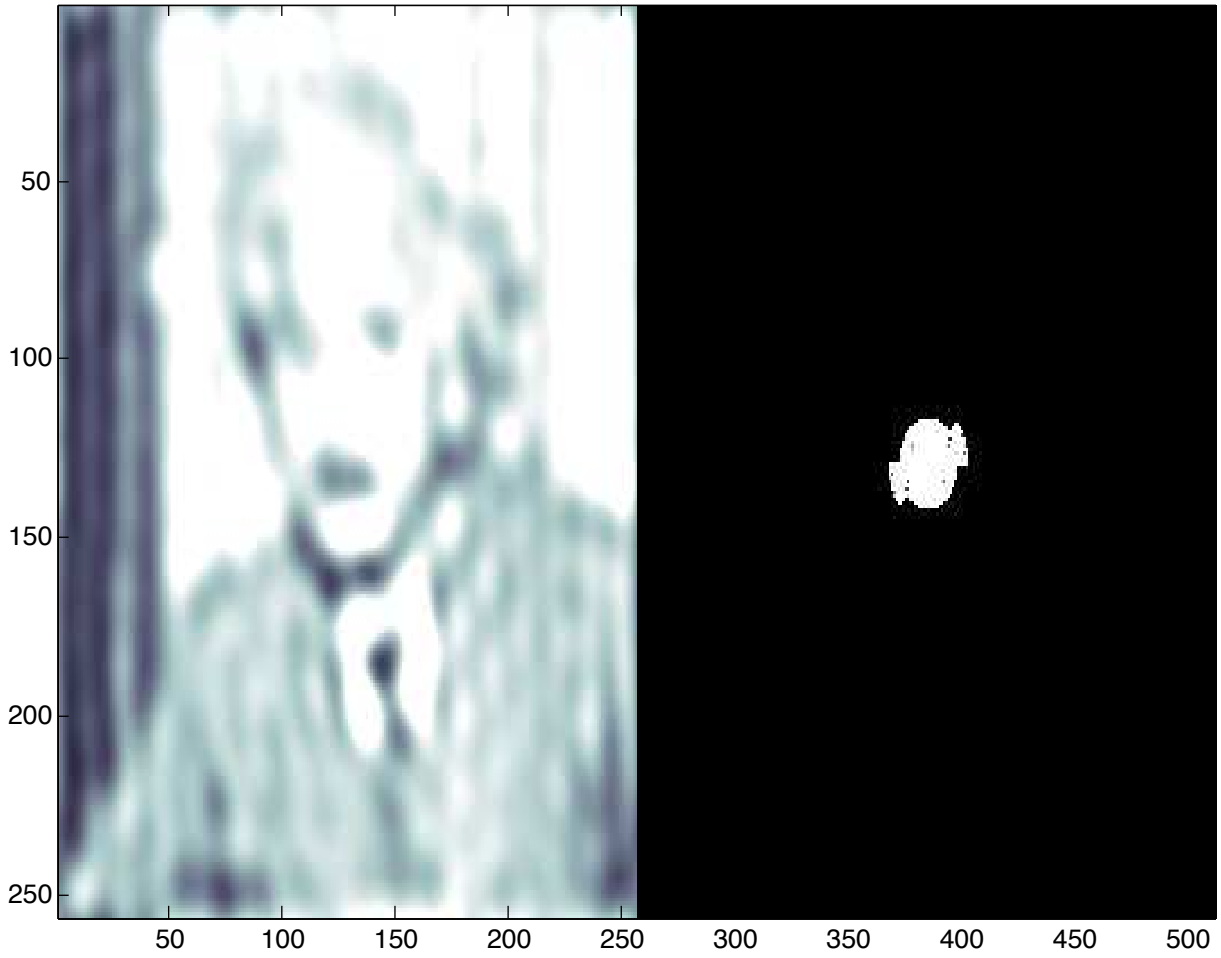
→ Requires several ‘sub’ functions

‘Spatial domain’ ‘Frequency domain’



Note: Only ½ of the information is shown on the right (amplitude only; phase not shown)

EXfourier2D.m



→ ‘Low-pass filtered’ version of the image

Fourier series

Intuitive connection back to Taylor series:

$$y(x_1 + \Delta x) \approx y(x_1) + \sum_{n=1}^{N} \frac{1}{n!} \left. \frac{d^n y}{dx^n} \right|_{x_1} (\Delta x)^n. \qquad \text{(D.2)}$$

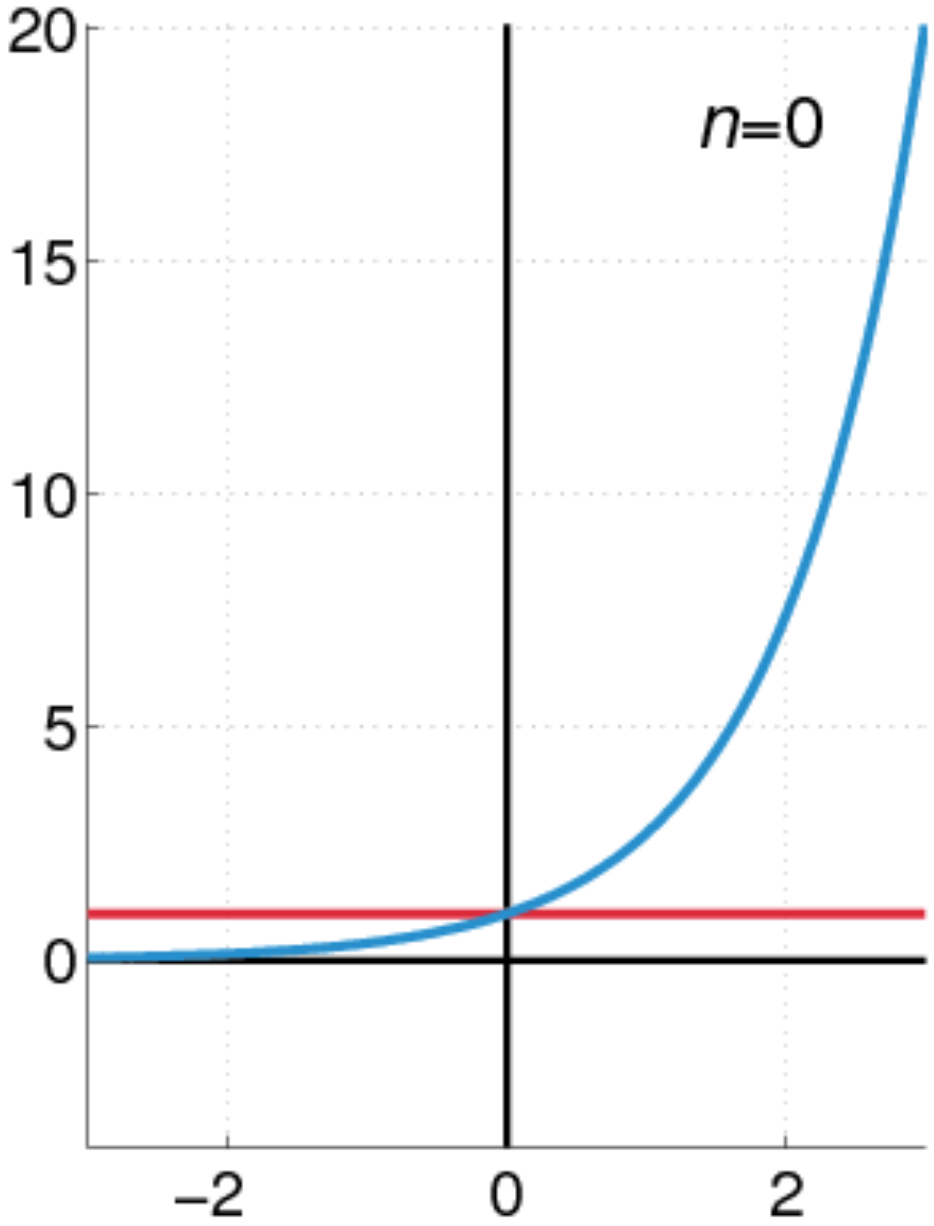
$$f(x) = f(x_o) + f'(x_o)(x - x_o) + \frac{f''(x_o)}{2!}(x - x_o)^2 + \cdots + \frac{f^{(n)}(x_o)}{n!}(x - x_o)^n + \cdots$$

$$= \sum_{n=0}^{\infty} \frac{f^{(n)}(x_o)}{n!}(x - x_o)^n$$

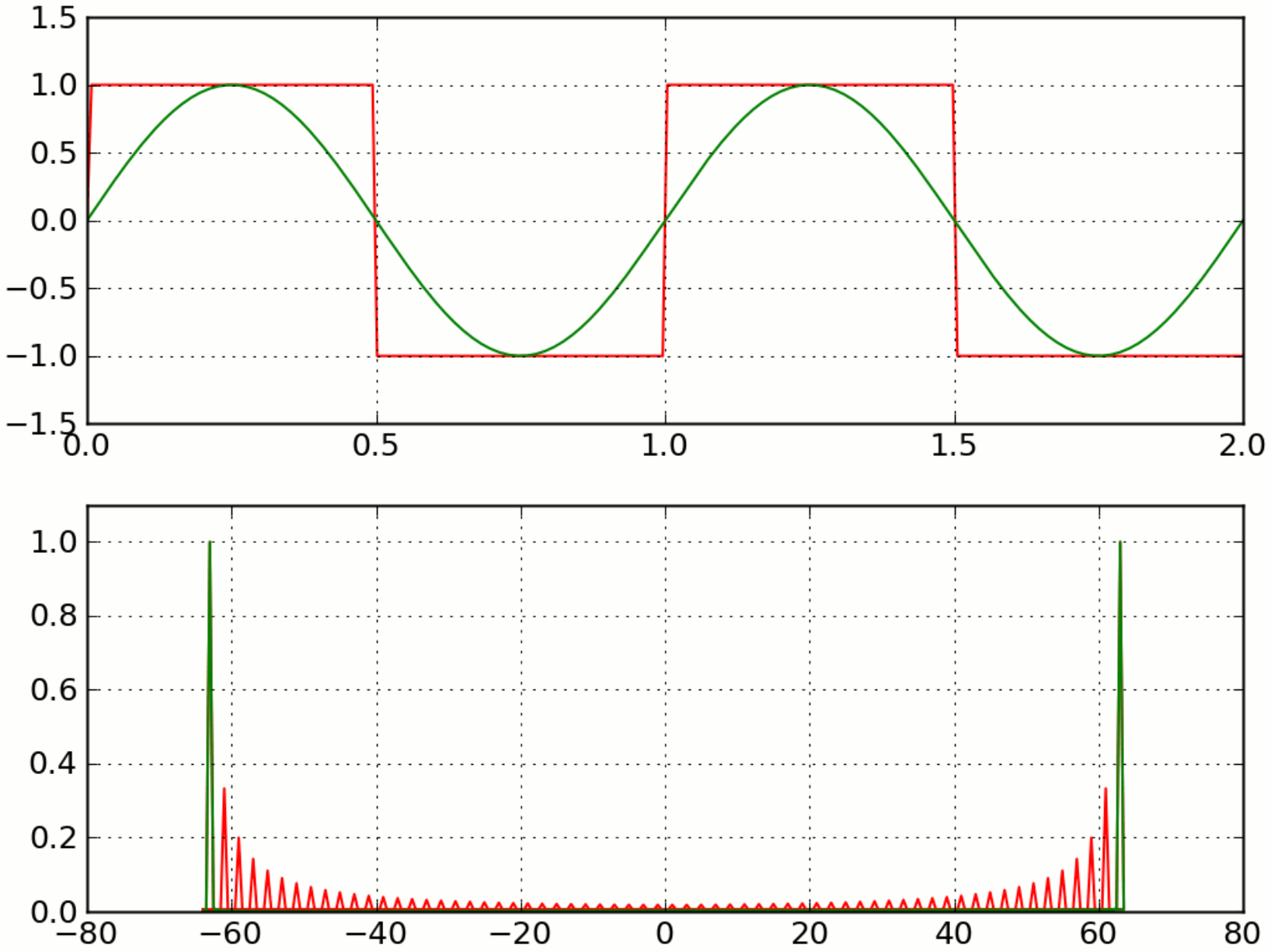
Taylor series → Expand as a (infinite) sum of polynomials

Different Idea: Fourier series → Expand as a (infinite) sum of sinusoids

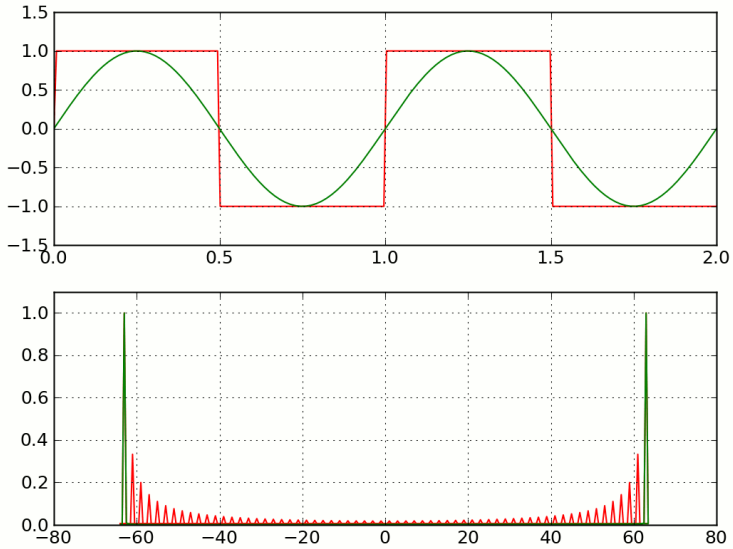
"The exponential function e^x (in blue), and the sum of the first n+1 terms of its Taylor series at 0 (in red)."



wikipedia (Taylor series)

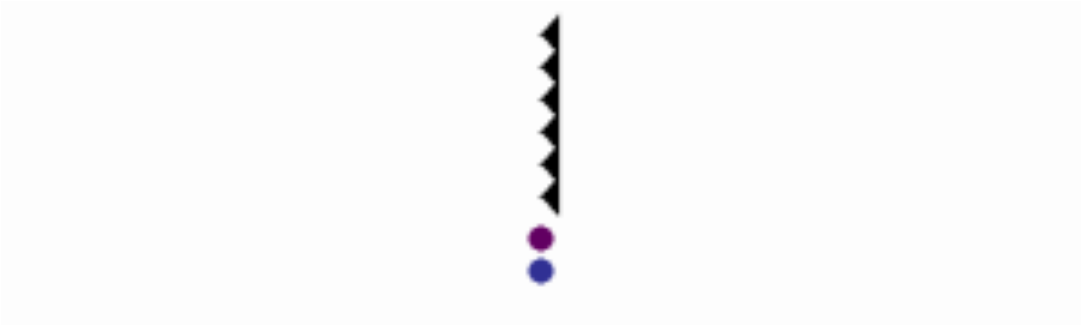


wikipedia (square wave)



“Animation of the additive synthesis of a square wave with an increasing number of harmonics.”

“The six arrows represent the first six terms of the Fourier series of a square wave. The two circles at the bottom represent the exact square wave (blue) and its Fourier-series approximation (purple).”



wikipedia (square wave)

Fourier series

$$f(t) = a_o + a_1 \sin(\omega t) + b_1 \cos(\omega t) + \\ + a_2 \sin(2\omega t) + b_2 \cos(2\omega t) + \\ + a_3 \sin(3\omega t) + b_3 \cos(3\omega t) + \cdots$$

$$= A_0 + A_1 \sin(\omega t + \phi_1) \\ + A_2 \sin(2\omega t + \phi_2) \\ + A_3 \sin(3\omega t + \phi_3) + \cdots$$

$$= \sum_{n=0}^{\infty} A_n \sin(n\omega t + \phi_n)$$

$$= \sum_{n=0}^{\infty} B_n e^{in\omega t} \qquad \text{where } B_n \in \mathbb{C},\ i = \sqrt{-1}$$

Complex #s are much more compact and easier to deal with

Summary

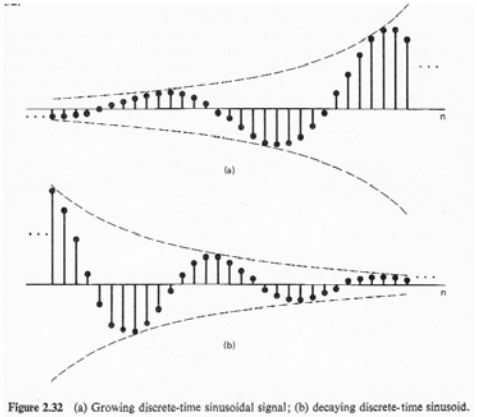


Figure 2.32 (a) Growing discrete-time sinusoidal signal; (b) decaying discrete-time sinusoid.