Lecture 1:
Introduction & DSP

Michael Mandel mim@sci.brooklyn.cuny.edu

CUNY Graduate Center, Computer Science Program
http://mr-pc.org/t/csc83060
With much content from Dan Ellis' EE 6820 course

Outline

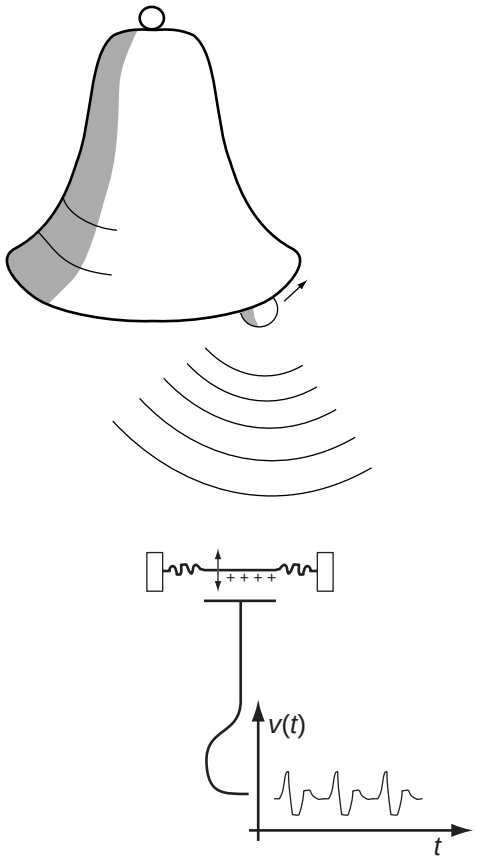
Acknowledgments: This lecture uses content from

- Dan Ellis' EE 4810 course (Lectures 1,2,3)
  `https://www.ee.columbia.edu/~dpwe/e4810`
- Dan Ellis' EE 6820 course (Lecture 1)
  `https://www.ee.columbia.edu/~dpwe/e6820/`

Outline

Sound and information

Sound is air pressure variation



Mechanical vibration

↓

Pressure waves in air

↓

Motion of sensor

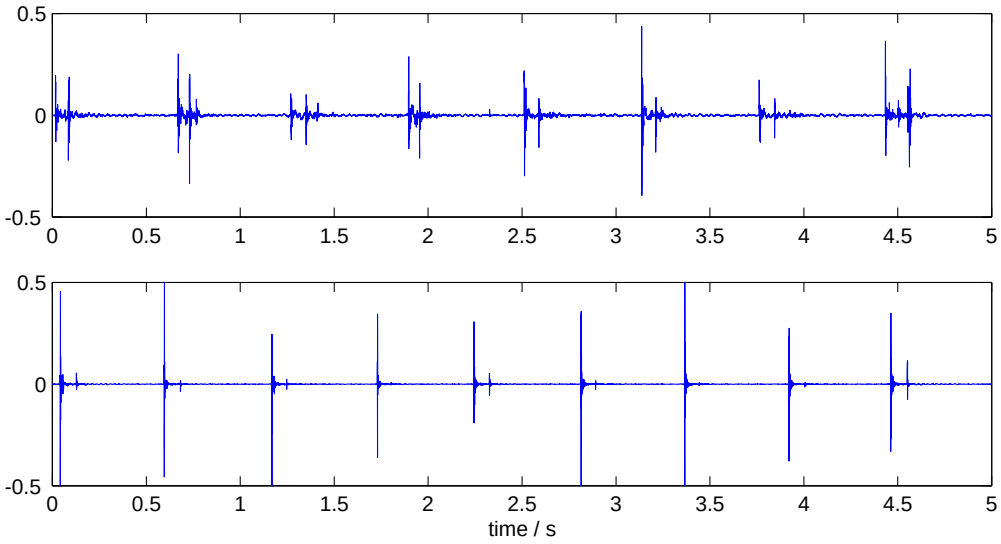
↓

Time-varying voltage

Transducers convert air pressure $\leftrightarrow$ voltage

What use is sound?

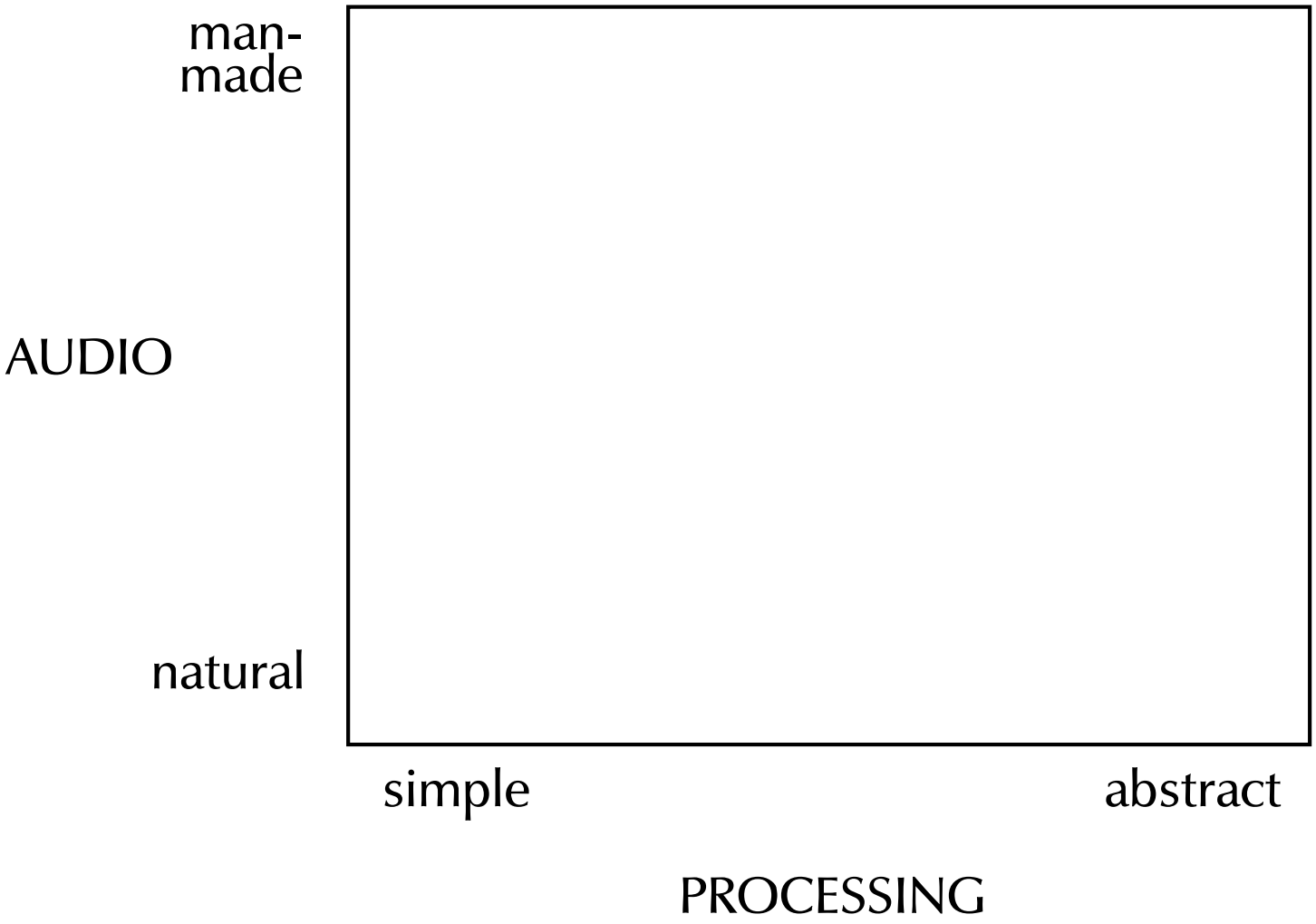
Footsteps examples:



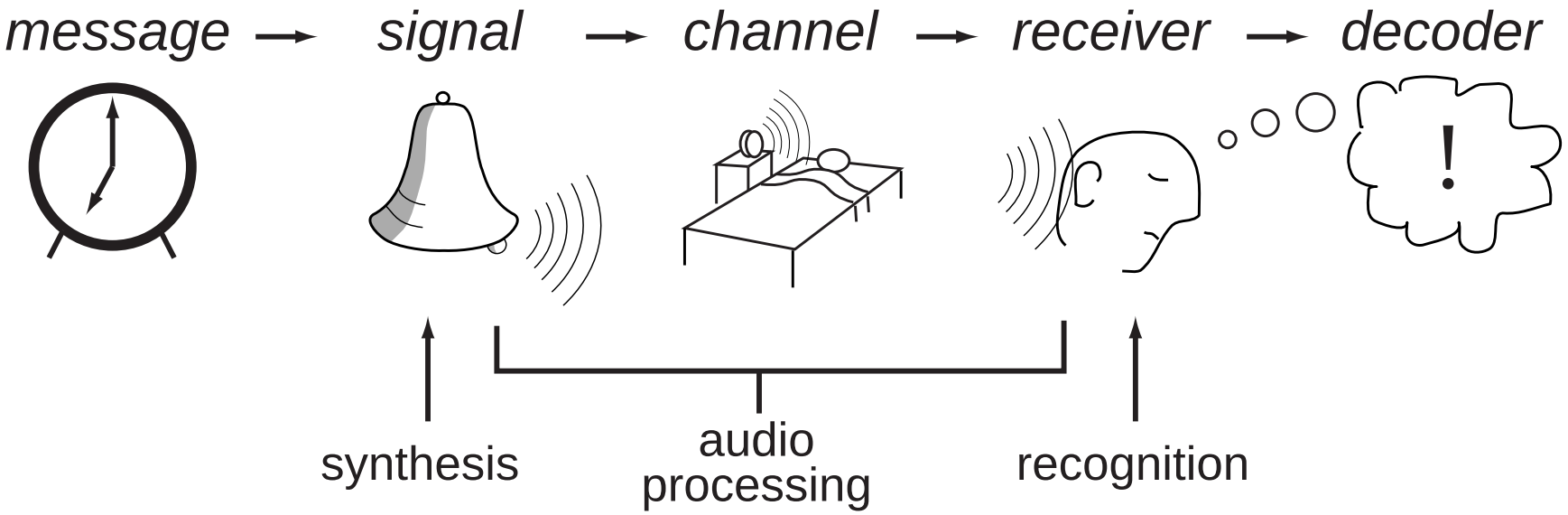
Hearing confers an evolutionary advantage

- useful information, complements vision
- . . . at a distance, in the dark, around corners
- listeners are highly adapted to 'natural sounds' (incl. speech)

The scope of audio processing



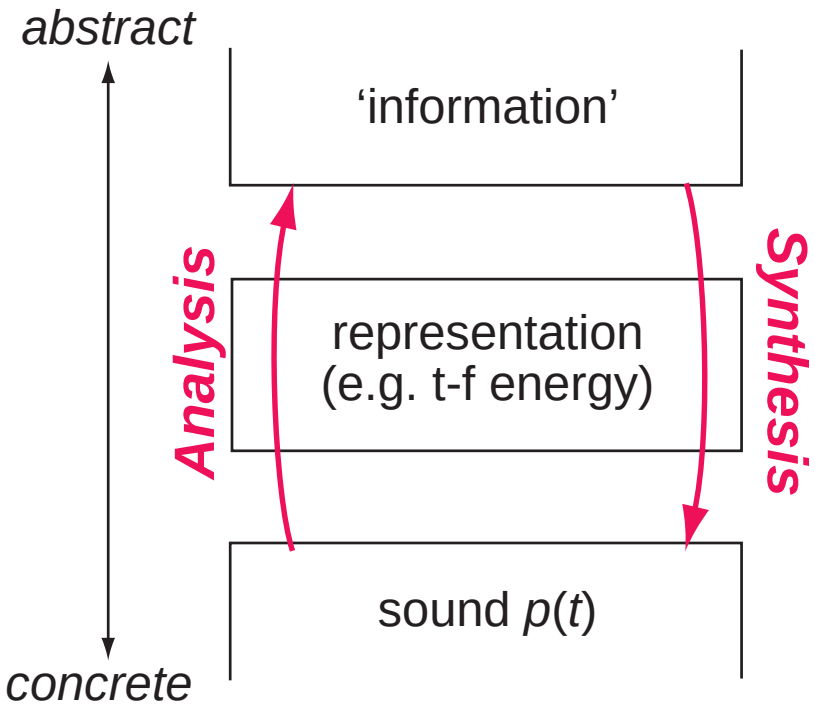
The acoustic communication chain



- Sound is an information bearer
- Received sound reflects source(s) plus effect of environment (channel)

Levels of abstraction

Much processing concerns shifting between levels of abstraction



Different representations serve different tasks

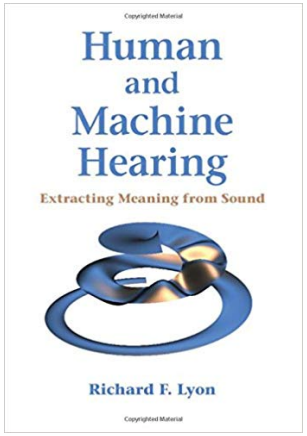
- separating aspects, making things explicit, ...

Outline

Course structure

- Goals
  - survey topics in sound analysis & processing
  - develop an intuition for sound signals
  - learn some specific technologies
- Course structure
  - weekly reading (20% + 10%)
  - midterm project proposal (20%)
  - final project (30% + 10%)
- Required textbook for first part of course. Available at `http://dicklyon.com/hmh/Lyon_Hearing_book_01jan2018.pdf`

Richard Lyon (2018), *Human and Machine Hearing: Extracting meaning from sound*. Cambridge University Press.



Web-based

Course website:

- `http://mr-pc.org/t/csc83060`
- for lecture notes, articles, examples, ...
- + Blackboard for homework, etc.

Assignments: every week

- Everyone reads book chapters and/or review papers
  - reviewing topic of next class
  - discuss in class
- One student presents a contribution paper
  - journal & conference papers presenting a novel contribution
  - everyone else reads it as well
  - everyone writes a two-paragraph response on blackboard: ¶1: summary and ¶2: future directions
- Write one-paragraph project status
  - ¶3: weekly status updates

Final project

- Most significant part of course (60%) of grade
- Oral proposals mid-semester;
  Presentations in final class
  + paper on week later
- Scope
  - practical (Python or Matlab recommended)
  - identify a problem; try some solutions
  - evaluation
- Topic
  - few restrictions within world of audio
  - investigate other resources
  - develop in discussion with me
- No copying

Examples of past projects

- Use of "bubble noise" to compare native Korean listeners to Korean learners – became masters thesis
- Use of siamese networks as learned distance function between clean and noisy speech – became INTERSPEECH paper
- Implementation of classic noise tracking algorithms and application to beamforming
- Implementation of LSTM-based single-channel speech enhancement

Some project ideas

- Podcast transcriber and diarizer
- Podcast episode recommender
- Analysis of long-term environmental recordings
- Heatmapping tools applied to automatic speech recognizers
- Use of neural speech synthesis in speech enhancement

Example responses: ¶1: Contribution article summary

The paper for this week is Mohri, Mehryar, Fernando Pereira, and Michael Riley. "Speech recognition with weighted finite-state transducers." Springer Handbook of Speech Processing. Springer Berlin Heidelberg, 2008. 559-584. The paper describes how to use weighted finite-state transducers(WFST) for automatic speech recognitions. The transducers can be efficiently used to represent major components of speech recognition like Hidden Markov Model(HMM), context-dependency model, pronunciation lexicon and statistical grammars. At first authors present general description of transducers , algorithms to determinize and minimize them. Then they describe how these methods are very good match for combining and optimizing different parts of the speech recognizer. A WFST can be described informally as a machine with fixed set of states, it has a start state, some final states and intermediate states. Given a input symbol they can transitions from one state to another with producing output symbol. WFST is weighted in the sense that transitions from state to state can have different weights. Generally these weights represents the probability or likelihood of different paths. It transduces a string that can be read along the path from start state to final state to output string with particular weight. In another word they represent a binary relation between input and output strings. For example pronunciation lexicons can be represented by transducers as mapping from pronunciation to word. Pronunciation lexicon (L) for all of the words can be generated by composing all word pronunciation transducers together thorough a super-initial state. Applying a Kleene closure on each of the word transducer allows for words to repeat in any order. If we add the mapping context dependent triphone to context independent phone with this, we get a model(C) that maps from context dependent phones to word strings. For efficiency we need to determinize and minimize the transducer. Converting non-deterministic transducer to a deterministic counter part is known as a detereminizer, there are algorithms like semiring applicable for this operation. Deterministic transducers are efficient to search given a particular input-output pair, thus we want to determinize the transducers for efficiency. Now the problem is all of the models may not be determinizable to start with, for example pronunciation has homophones , different words that sounds the same. Thus they are not determinizable,we need to add auxiliary phone symbols to make it determinizable. A unweighted transducer can be minimized with classical algorithms, but for weighted version there is an extra step pushing /reweighting. Finally if we combine HMM model (H) with this we get a transducer that maps from distribution to word strings.

My response: Good summary

Example responses: ¶2: Contribution article response

The paper for this week talks about transducers and their application to ASR. They first introduce acceptors and then define transducers. They also mention different algorithms for composition, determinization and minimization of transducers. According to me it is a very well written paper, the examples are specially well described and easy to reproduce. With the algorithms I found the mathematical descriptions very informative and useful as well. I wanted to know more about other available algorithms and whether they can be used for speech recognizer. For composition they described state-wise pairing with semiring method which increases the state space of the composite automation. Though that is expected as we are combining two transducers. I wanted to know a little bit more about other available algorithms and why we can or can't use them. I felt similarly about weight pushing algorithm, whether there are other algorithms available for reweighing the transducer. Or if there is any algorithm for directly minimizing weighted transducer pushing and then minimizing.

My response: Good questions, I don't know all of the answers to them. A semiring isn't an algorithm or a method, it is a mathematical object that has certain characteristics. Informally, it's the way to keep track of the weights and how to combine them. The semiring to represent probabilities is different from, but equivalent to, the semiring to represent log probabilities. Basically you need a plus operation and a times operation. For probabilities, plus is + and times is *. For log probabilities, plus is log(exp(x1)+exp(x2)) and times is +. So applying the plus operation to two probabilities is equivalent to applying the plus operation to two log probabilities. And similarly for times. Anyway, I think there are several algorithms for weight pushing, minimization, etc.

Example responses: ¶3: Project status

For the project I am working to train the deep neural network, which is the main part of the project.

This week I have written a code for simple WARP loss function.

I have also created the training data from CHiME2-GRID. The training data contains one clean chunk, noisy chunk and another negative example.

I am also working towards setting the DNN properly to train with training data.

My response: Ok, have you checked the code into github on your own branch yet? If so, send me a link to it. If not, please do.

Outline

Signals

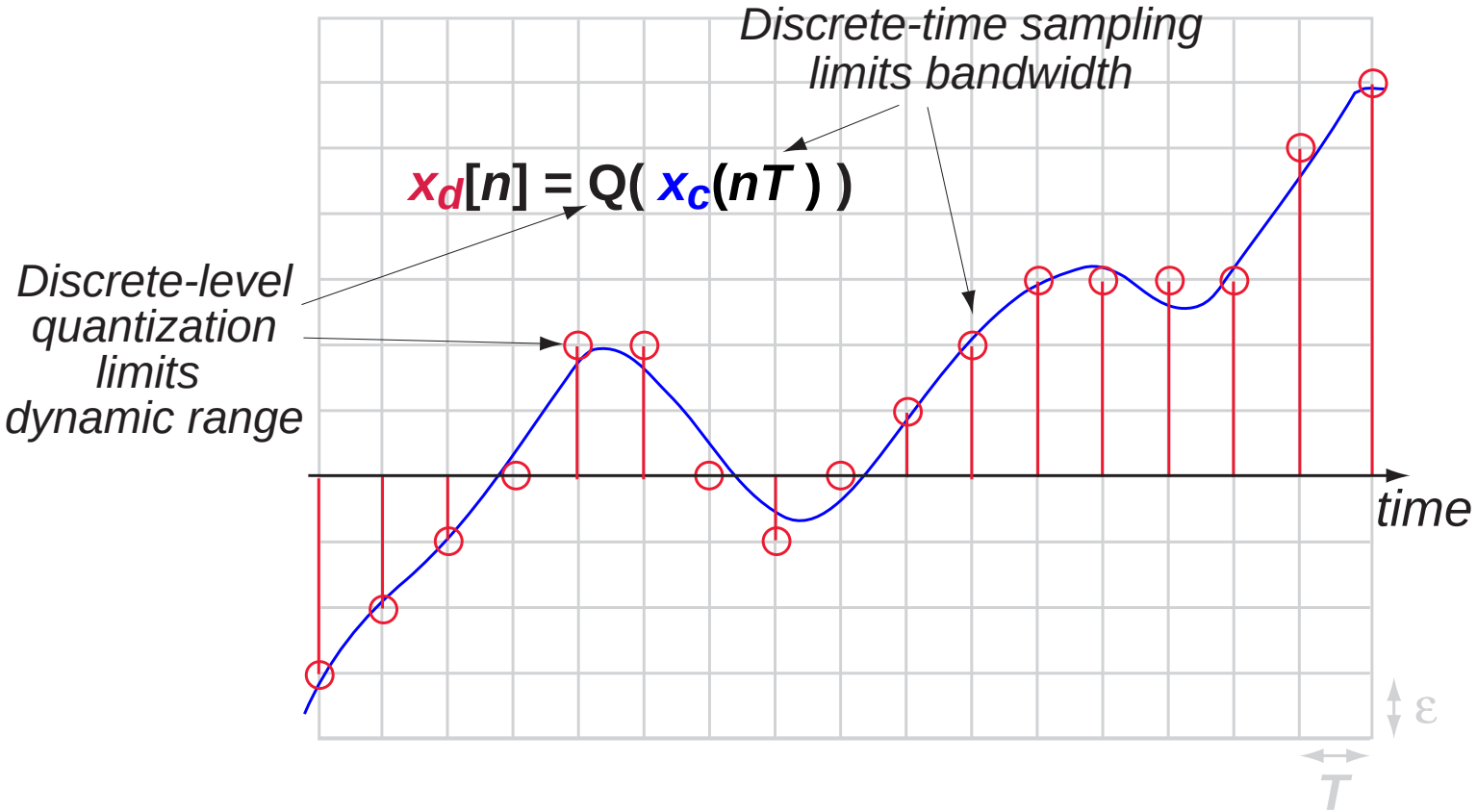
- Signals: Information-bearing function

E.g. sound: air pressure variation at a point as a function of time

- Dimensionality
  - Sound: 1-dimensional
  - Greyscale image: 2-dimensional, $i(x, y)$
  - Video: 3 x 3-D, $\{r(x, y, t), g(x, y, t), b(x, y, t)\}$

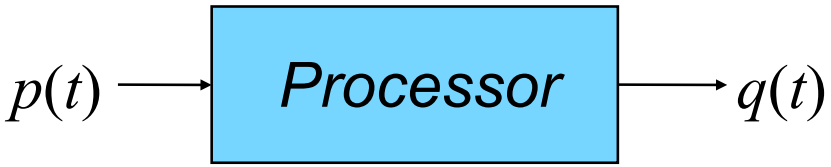
Digital signal processing

- DSP = signal processing on a computer
- Two effects: discrete-time, discrete level

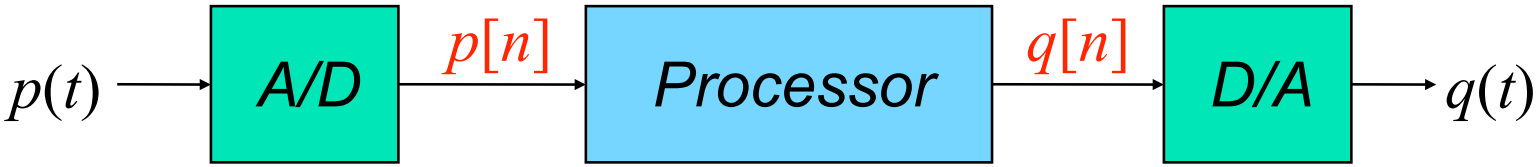


DSP vs analog signal processing

Analog signal processing



Digital signal processing system

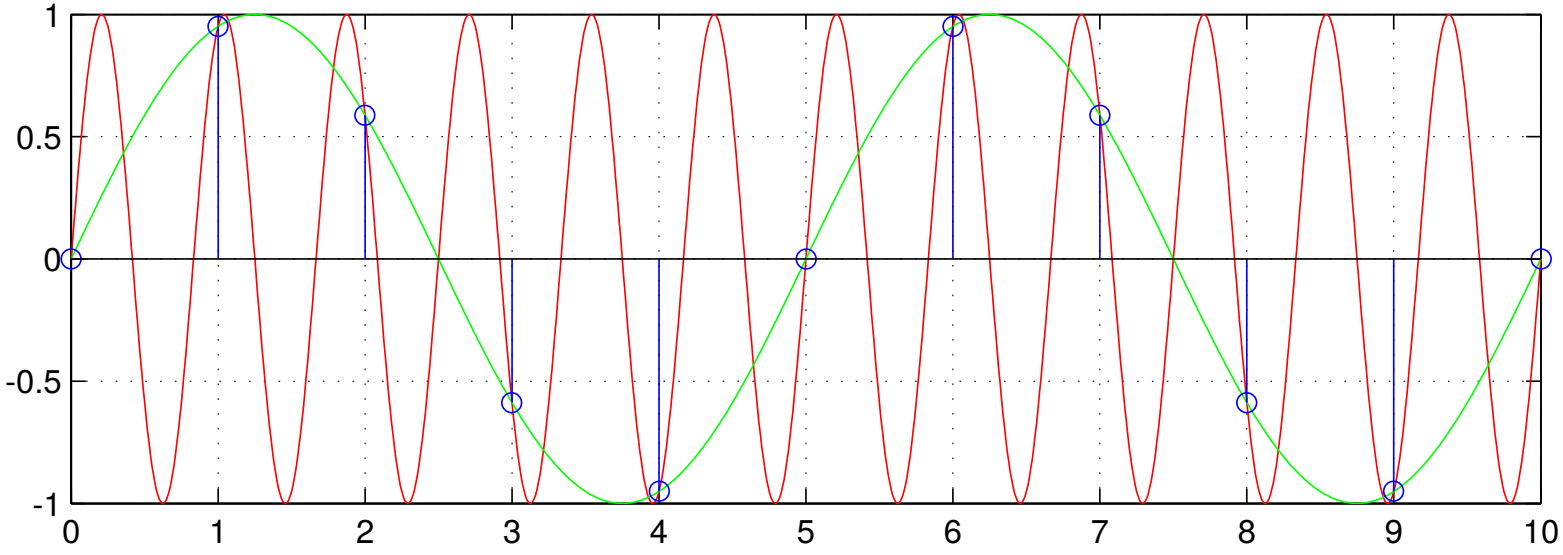


Digital vs analog

- Digital pros
  - Noise performance – quantized signal
  - Use general-purpose computer – flexible, upgradeable
  - Stability/replicability
- Digital cons
  - Limitations of analog-digital converters
  - Baseline complexity/power consumption
  - Latency

Digital limitation: sampling sinusoids

- Samples of a sinusoid could equally correspond to others



$$x_1[n] = \sin(\omega_0 n)$$
$$x_2[n] = \sin((\omega_0 + 2\pi r)n) = \sin(\omega_0 n) = x_1[n]$$

Aliasing

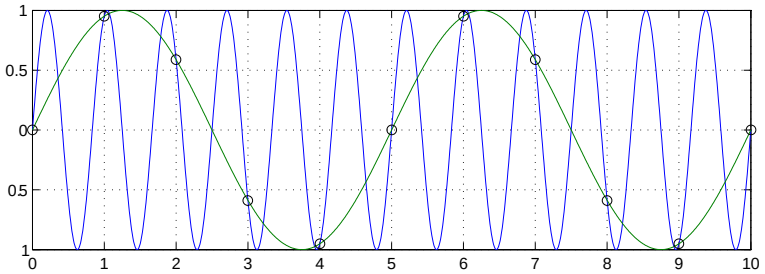
- For example, for $\cos(\omega n), \omega = 2\pi r \pm \omega_0$
  all (integer) r values appear to the same after sampling
- We say that a larger ω appears aliased to a lower frequency
- Therefore we generally consider frequencies $0 \leq \omega_0 \leq \pi$
  i.e., less than $\frac{1}{2}$ cycle per sample

Sampling and aliasing

Discrete-time signals equal the continuous time signal at discrete sampling instants:

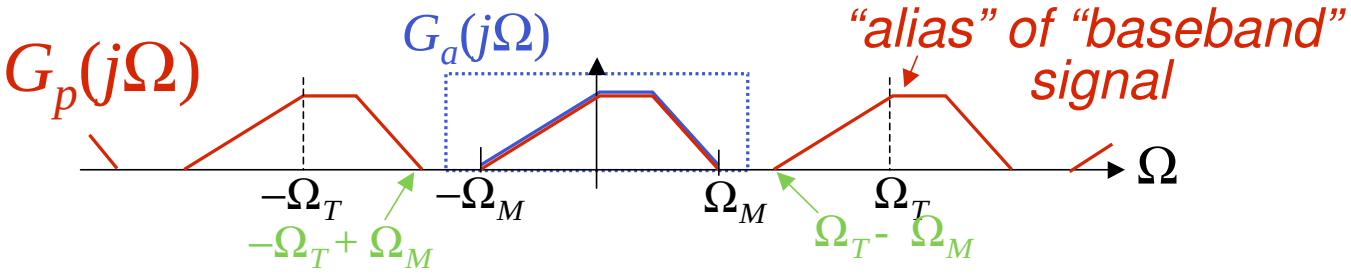
$$x_d[n] = x_c(nT)$$

Sampling cannot represent rapid fluctuations



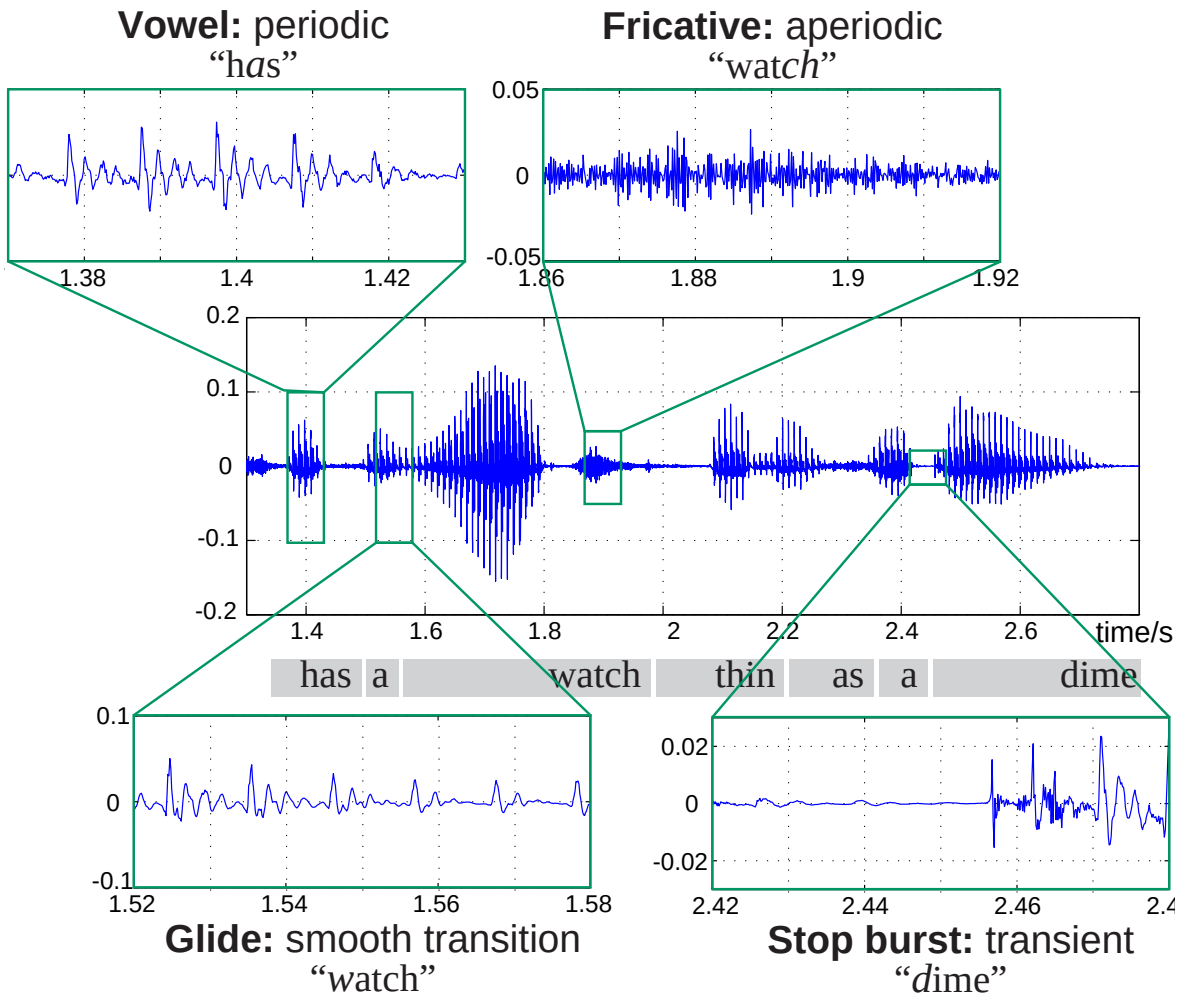
$$\sin\left(\left(\Omega_M + \frac{2\pi}{T}\right) Tn\right) = \sin(\Omega_M Tn) \quad \forall n \in \mathbb{Z}$$

Nyquist limit $(\Omega_T/2)$ from periodic spectrum:



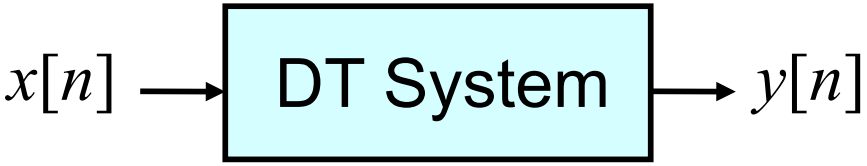
The speech signal: time domain

Speech is a sequence of different sound types



Discrete-time systems

- A system converts input to output



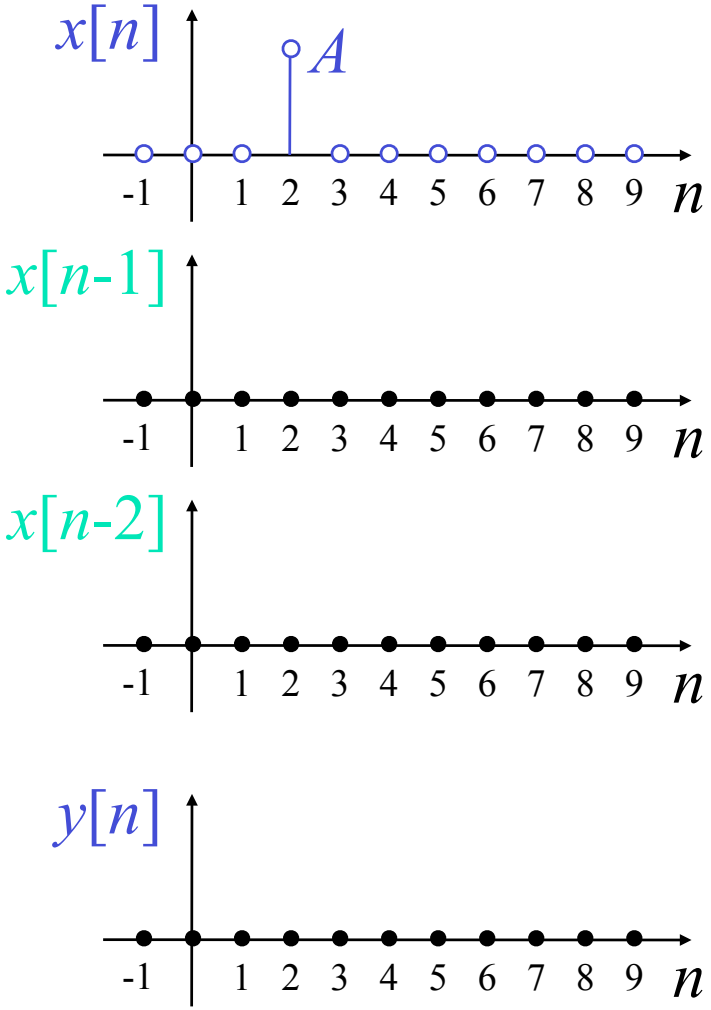
$$\{y[n]\} = f(\{x[n]\})\forall n$$

- For example, a 3-point moving average ($M = 3$)

$$y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$$

3-point moving average example

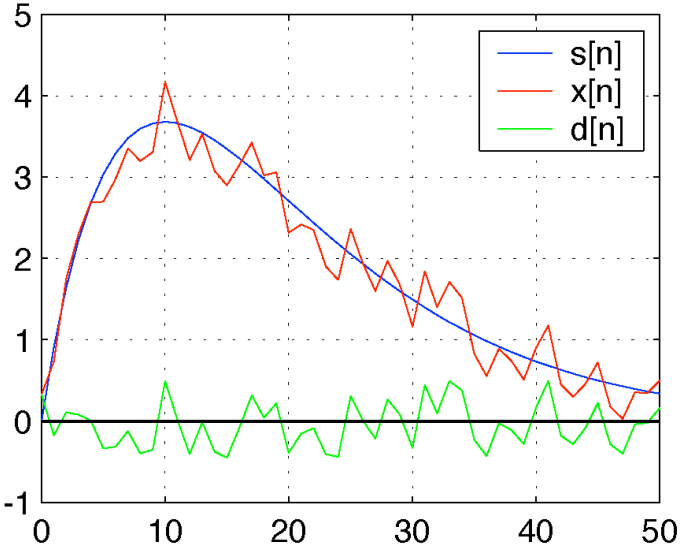
$$y[n] = \frac{1}{3}\sum_{k=0}^{2} x[n-k]$$

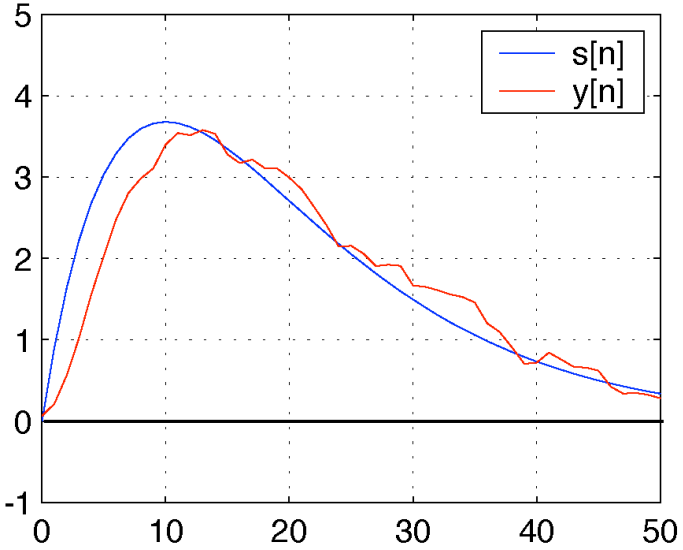


Moving average smoother

- Moving average smooths out rapid variations
  e.g., "12 month moving average"
- Example, 5-point moving average

$$x[n] = \underbrace{s[n]}_{\text{signal}} + \underbrace{d[n]}_{\text{noise}} \qquad y[n] = \frac{1}{5}\sum_{k=0}^{4} x[n-k]$$





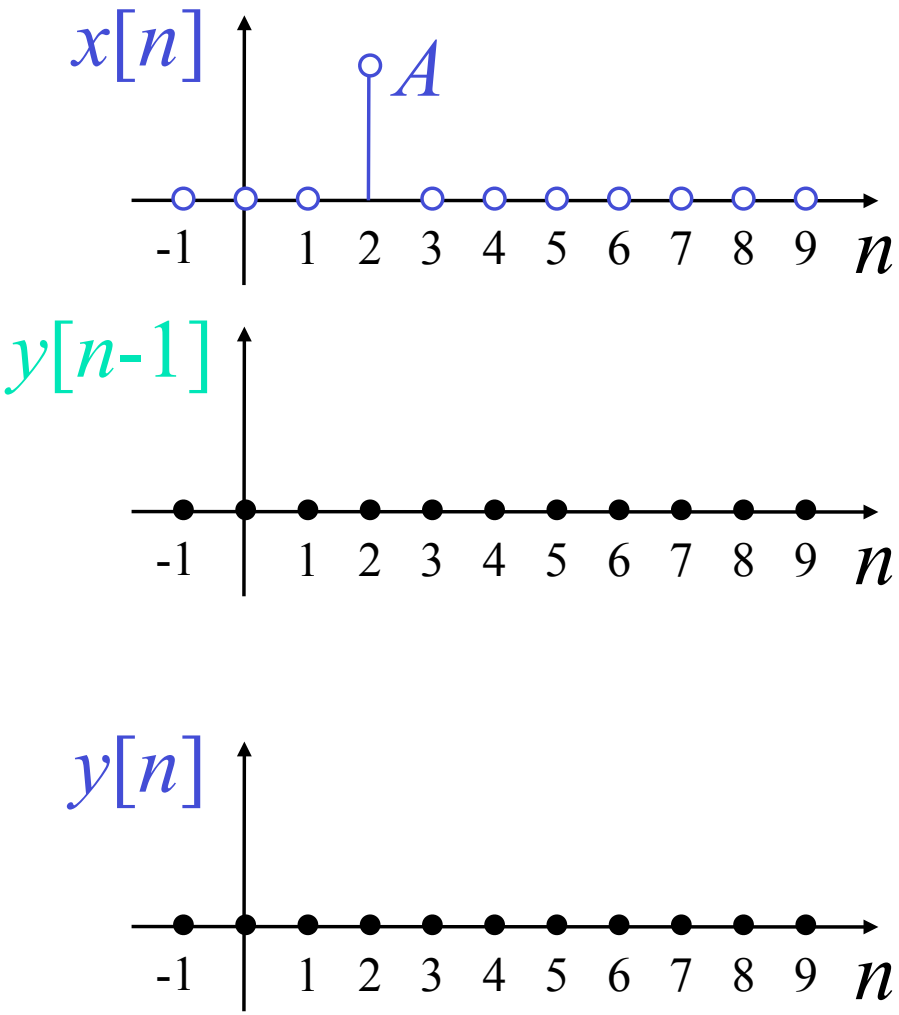
Example 2: accumulator

Accumulator: output accumulates all past inputs

$$y[n] = \sum_{\ell=-\infty}^{n} x[\ell]$$

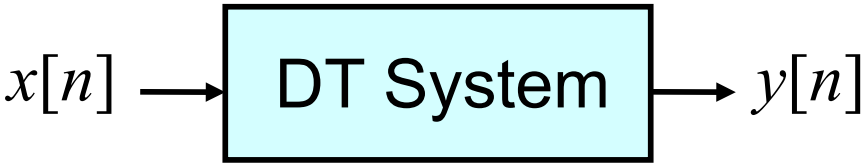
$$= x[n] + \sum_{\ell=-\infty}^{n-1} x[\ell]$$

$$= x[n] + y[n-1]$$



System property: Linearity

- Linear systems obey superposition



- If input $x_1[n] \rightarrow$ output $y_1[n]$, $x_2 \rightarrow y_2$
- Given a linear combination of inputs, a linear system produces the same linear combination of outputs

$$x[n] = \alpha x_1[n] + \beta x_2[n]$$
$$\rightarrow y[n] = \alpha y_1[n] + \beta y_2[n]$$

for all $\alpha, \beta, x1, x2$

System property: Time-invariance

- Time-invariant systems don't depend on an absolute time
- Given a time-shifted input, a time-invariant system produces a version of the output shifted by the same amount

$$\text{if } x[n] \to y[n], \text{ then } x[n-n_0] \to y[n-n_0]$$

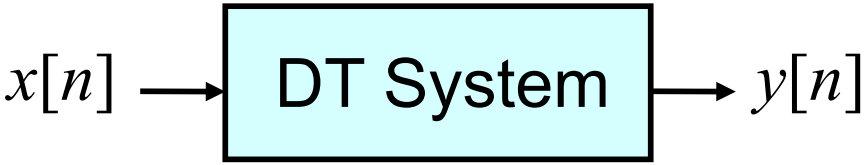
for all n_0, x

Impulse response

- An impulse is a unit sample sequence

$$\delta[n] = \begin{cases} 1, n = 0 \\ 0, n \neq 0 \end{cases}$$

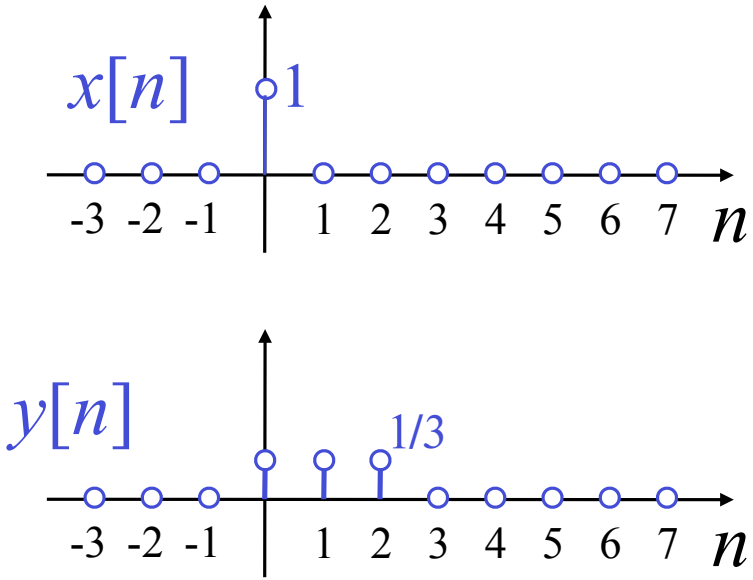
- Given a system, if $x[n] = \delta[n]$ then $y[n] \triangleq h[n]$, and $h[n]$ is the impulse response of the system



- A linear, time-invariant system is completely specified by $h[n]$

Impulse response example: Moving average

- Consider the 3-point moving average filter again
- If we put in a standard impulse $x[n] = \delta[n]$
  we get out the impulse response $y[n] = h[n]$



Convolution

- Impulse response means for an LTI system $\delta[n] \rightarrow h[n]$
- Shift invariance means $\delta[n - n_0] \rightarrow h[n - n_0]$
- Linearity means $\alpha\delta[n - k] + \beta\delta[n - l] \rightarrow \alpha h[n - k] + \beta h[n - l]$
- We can express any sequence as a sum of weighted δs

$$x[n] = x[0]\delta[n] + x[1]\delta[n - 1] + x[2]\delta[n - 2] + \ldots$$

- Therefore, we can compute the output of any system

Convolution sum

- Another way to write $x[n]$ as a sum of deltas is

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

- Therefore, for an LTI system, we get the convolution sum

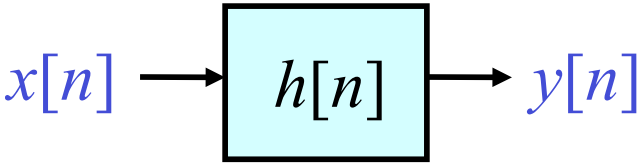
$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \triangleq x[n] * h[n]$$

- Note that this is symmetric in x and h by using $\ell = n - k$

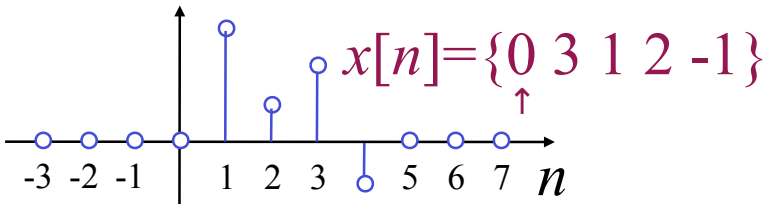
$$x[n] * h[n] = \sum_{\ell=-\infty}^{\infty} x[n-\ell]h[\ell] = h[n] * x[n]$$

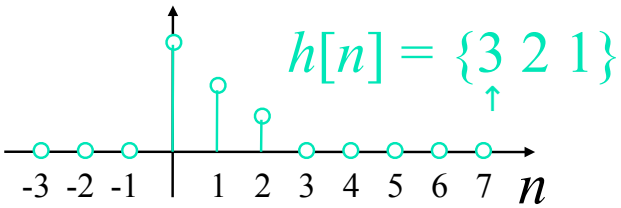
Interpreting convolution

- Passing a signal through an LTI system is equivalent to convolving it with the system's impulse response



- Let's convolve the following two signals in two different ways



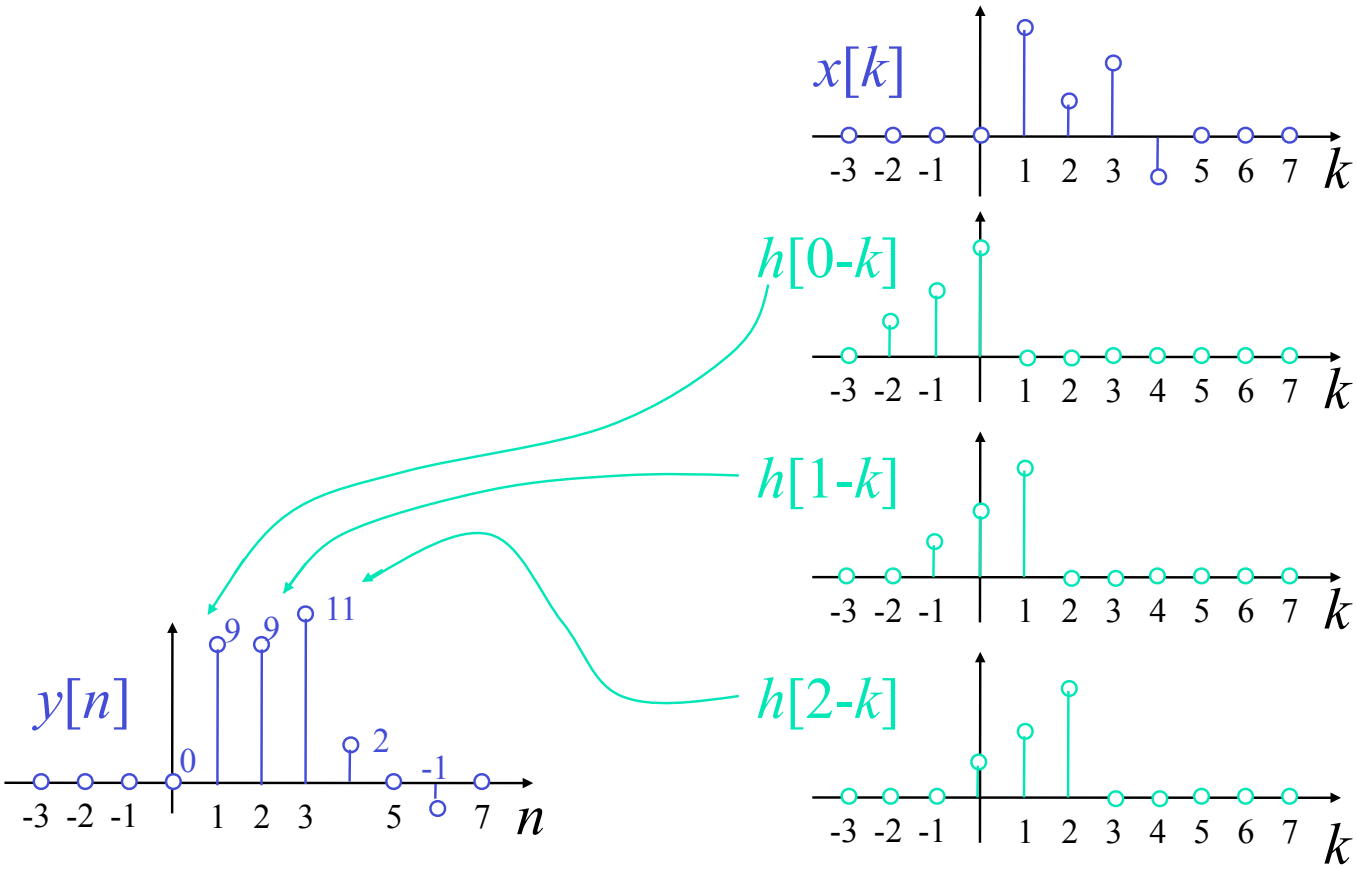


$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{\ell=-\infty}^{\infty} x[n-\ell]h[\ell]$$

Convolution interpretation 1

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

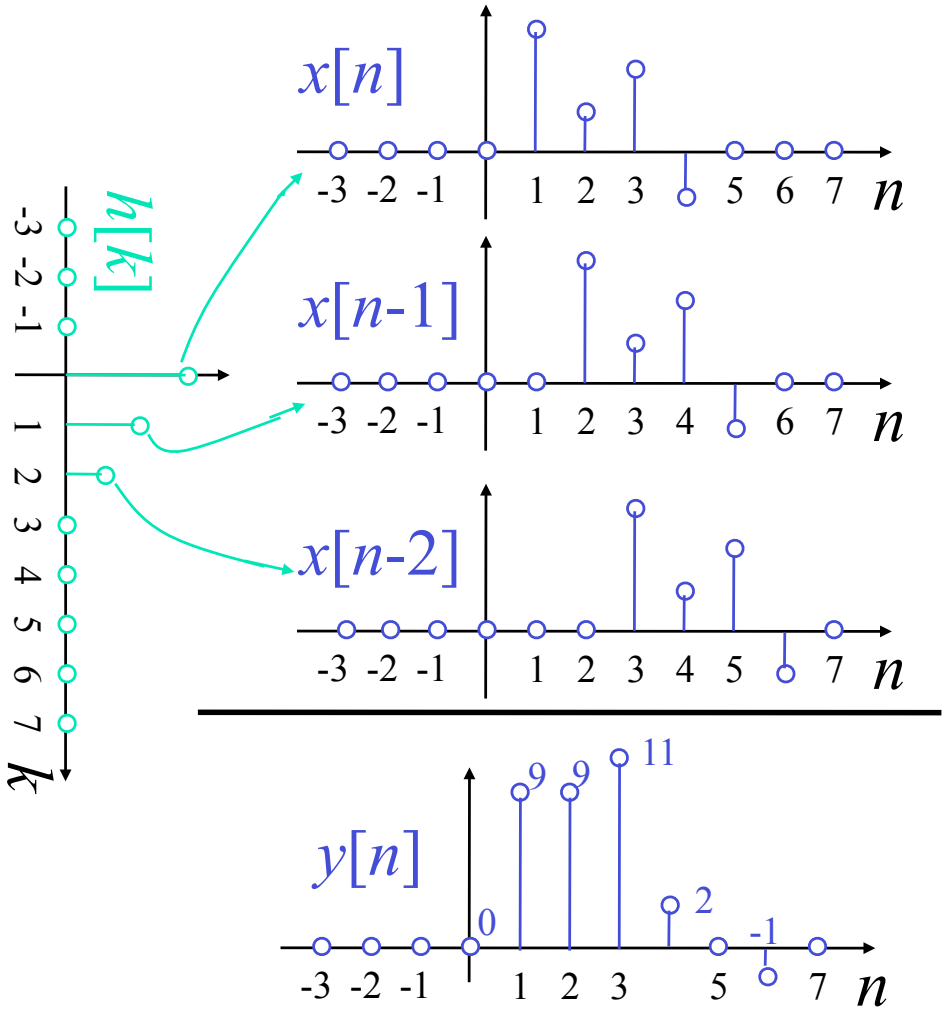
- Time-reverse h, shift by n, take inner product against fixed x



Convolution interpretation 2

$$y[n] = \sum_{\ell=-\infty}^{\infty} x[n-\ell]h[\ell]$$

- Shifted xs weighted by points in h
- Or, weighted delayed versions of h



Time-domain summary

- Computers analyze signals that are discrete in time and in level
- Sampling leads to aliasing, i.e., limited frequency bandwidth
- Linear time-invariant systems transform signals by convolving them with an impulse response

Outline

The Fourier series

- The basic observation (in continuous time) is that
  A periodic signal can be decomposed into
  a sum of weighted sinusoids at integer multiples
  of the fundamental frequency

i.e., if $x(t) = x(t+T)$ for some fundamental frequency T, then

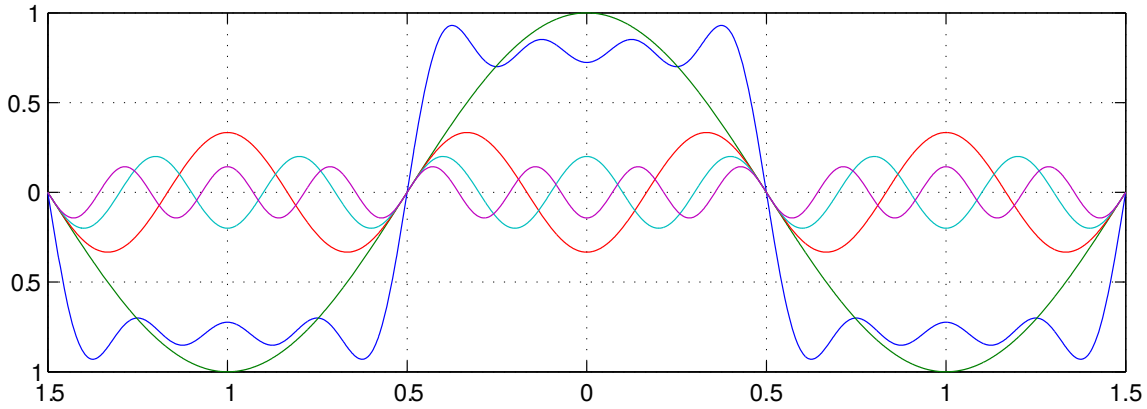
$$
\begin{aligned}
x(t) &\approx \sum_{k=0}^{M} a_k \cos\left(\frac{2\pi k}{T} t + \phi_k\right) \\
&= \sum_{k=0}^{M} a'_k \cos\left(\frac{2\pi k}{T} t\right) + b_k \sin\left(\frac{2\pi k}{T} t\right) \\
&= \sum_{k=0}^{M} \underbrace{c_k}_{\text{complex}} \exp\left(\frac{2\pi k j}{T} t\right) \quad \text{where } j = \sqrt{-1}
\end{aligned}
$$

Fourier series example: square wave

- Approximated by the following Fourier series coefficients

$$\phi_k = 0 \qquad a_k = \begin{cases} (-1)^{\frac{k-1}{2}} \frac{1}{k} & k = 1, 3, 5, \ldots \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned} i.e.\ x(t) &\approx \sum_{k=0}^{M} a_k \cos\left(\frac{2\pi k}{T} t + \phi_k\right) \\ &= \cos\left(\frac{2\pi}{T} t\right) - \frac{1}{3}\cos\left(\frac{2\pi}{T} 3t\right) + \frac{1}{5}\cos\left(\frac{2\pi}{T} 5t\right) - \ldots \end{aligned}$$



Fourier analysis

- Given a signal, how can we find Fourier coefficients?
- Let's use the a'_k and b_k representation, so

$$x(t) = \sum_{k=0}^{M} a'_k \cos\left(\frac{2\pi k}{T} t\right) + b_k \sin\left(\frac{2\pi k}{T} t\right)$$

- Take the inner product with complex sinusoids

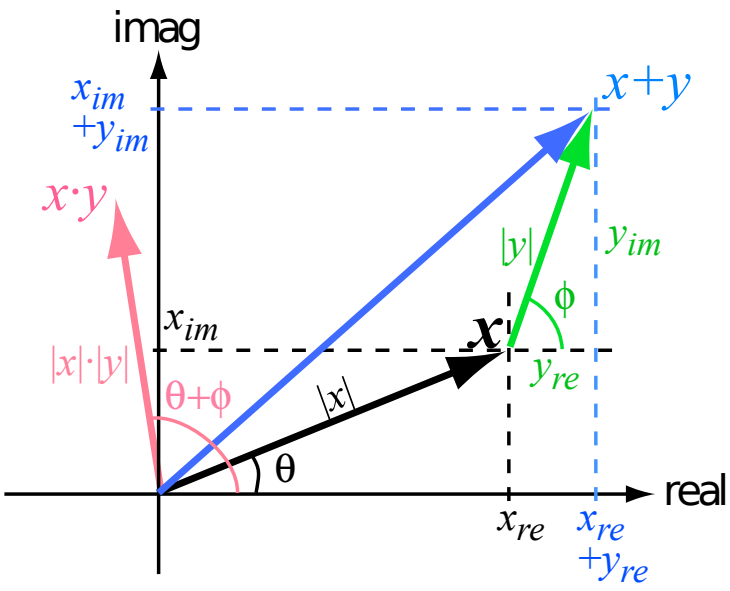
$$a'_k = \frac{1}{T} \int_{-T/2}^{T/2} x(t) \cos\left(\frac{2\pi k}{T} t\right) dt$$
$$b_k = \frac{1}{T} \int_{-T/2}^{T/2} x(t) \sin\left(\frac{2\pi k}{T} t\right) dt$$

- Works because these sinusoids are orthogonal to each other i.e., each sinusoid picks itself out of the signal

Complex numbers

- Complex numbers are for us a mathematical convenience that lead to simple expressions for things like the Fourier transform
- Add a second "imaginary" dimension ($j \triangleq \sqrt{-1}$) to all values
- Two equivalent representations:
  - Rectangular form: $x = x_r + jx_i$
    with magnitude $|x| = \sqrt{x_r^2 + x_i^2}$
    and phase $\theta = \tan^{-1}(x_i/x_r)$
  - Polar form: $x = |x|e^{j\theta} = |x|\cos\theta + j|x|\sin(\theta)$
    Euler's formula: $e^{j\theta} = \exp(j\theta) = \cos\theta + j\sin\theta$

Operations on complex numbers



- When adding, real and imaginary parts add

$$(a + jb) + (c + jd) = (a + c) + j(b + d)$$

- When multiplying, magnitudes multiply and phases add

$$re^{j\theta} \cdot se^{j\phi} = rse^{j(\theta+\phi)}$$

Complex conjugation

- Conjugation operation negates the imaginary part

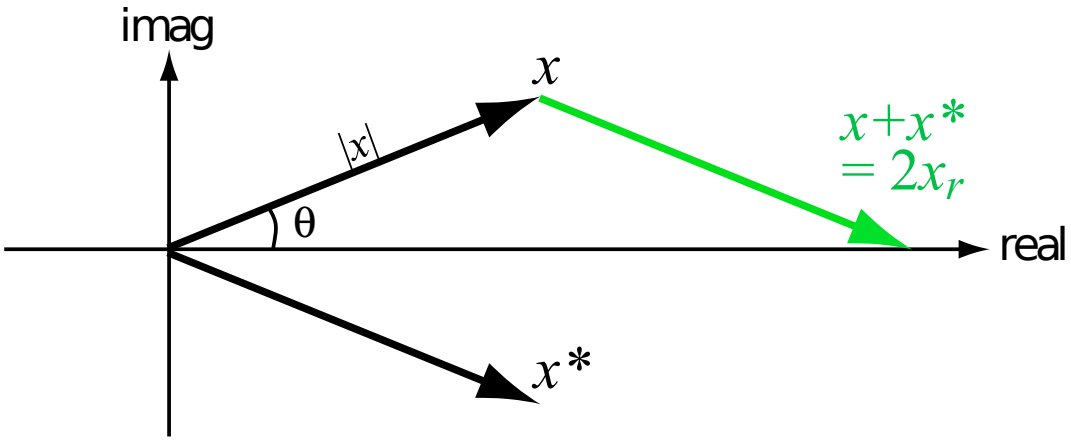
$$x^* = x_r - jx_i$$

- also negates the imaginary phase

$$x^* = |x|e^{j(-\theta)}$$

- Useful in extracting real quantities from complex expressions

$$x + x^* = x_r + jx_i + x_r - jx_i = 2x_r$$
$$x \cdot x^* = |x|e^{j\theta}|x|e^{-j\theta} = |x|^2$$



Fourier analysis using complex exponentials

- Inner product with complex sinusoids

$$a'_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\cos\left(\frac{2\pi k}{T}t\right)dt$$

$$b_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\sin\left(\frac{2\pi k}{T}t\right)dt$$

- Can be written more compactly with complex exponentials

$$c_k = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\exp\left(-j\frac{2\pi k}{T}t\right)dt$$

- Then synthesis of original signal

$$x(t) = \sum_{k=0}^{M} c_k \exp\left(j\frac{2\pi k}{T}t\right)$$

Fourier transform

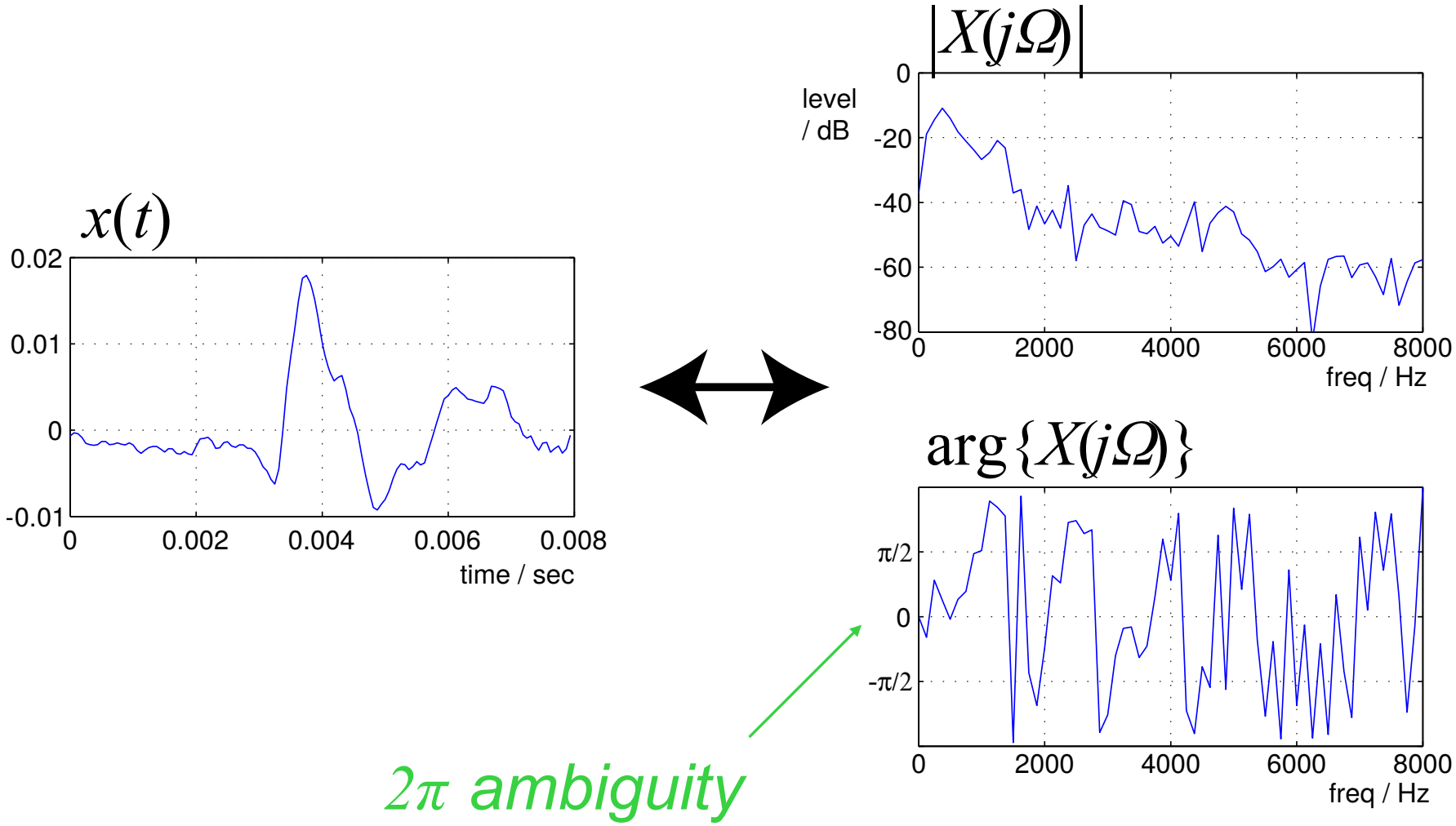
- The Fourier series (applicable to periodic signals) extends naturally to the Fourier Transform for continuous time signals

$$\text{Fourier transform} \qquad X(j\Omega) = \int_{-\infty}^{\infty} x(t)\exp(-j\Omega t)dt$$

$$\text{Inverse Fourier transform} \qquad x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\Omega)\exp(j\Omega t)d\Omega$$

- The discrete index k changes to a continuous frequency Ω

The Fourier transform is a mapping between two continuous functions



Fourier transforms

- Different Fourier analyses possible depending on signals
- continuous vs discrete, infinite vs finite (and periodic)

	Time	Frequency
Fourier series	Cont. periodic $\tilde{x}(t)$	Discrete infinite c_k
Fourier transform (FT)	Cont. infinite $x(t)$	Cont. infinite $X(\Omega)$
Discrete-time FT	Discrete infinite $x[n]$	Cont. periodic $X(e^{j\omega})$
Discrete FT	Discrete periodic $\tilde{x}[n]$	Discrete periodic $X[k]$

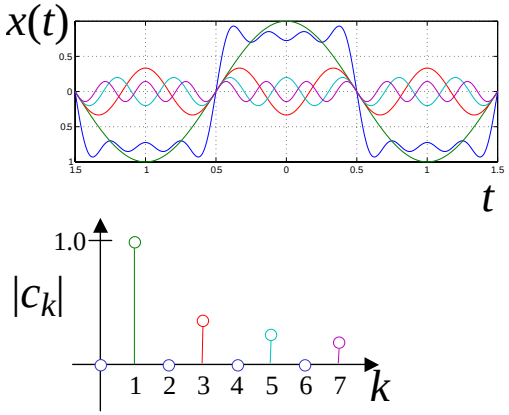
Fourier transforms

Fourier Series (periodic continuous x)

$$\Omega_0 = \frac{2\pi}{T}$$

$$x(t) = \sum_k c_k e^{jk\Omega_0 t}$$

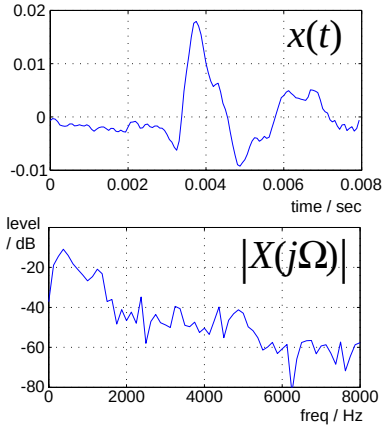
$$c_k = \frac{1}{2\pi T} \int_{-T/2}^{T/2} x(t) e^{-jk\Omega_0 t} dt$$



Fourier Transform (aperiodic continuous x)

$$x(t) = \frac{1}{2\pi} \int X(j\Omega) e^{j\Omega t} d\Omega$$

$$X(j\Omega) = \int x(t) e^{-j\Omega t} dt$$

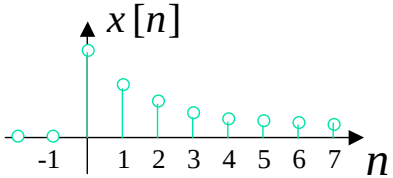


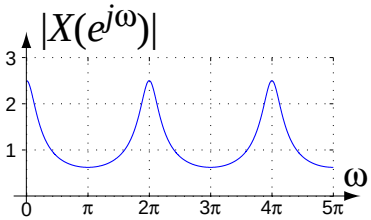
Discrete-time Fourier

DT Fourier Transform (aperiodic sampled x)

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega$$

$$X(e^{j\omega}) = \sum x[n]e^{-j\omega n}$$



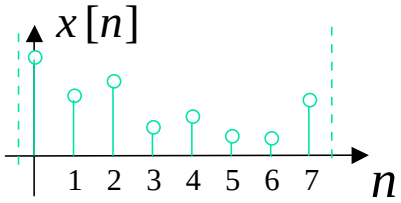


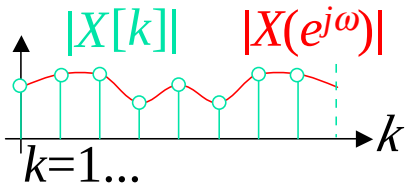
Discrete Fourier Transform (N-point x)

- the only one we will use on computers

$$x[n] = \sum_k X[k]e^{j\frac{2\pi kn}{N}}$$

$$X[k] = \sum_n x[n]e^{-j\frac{2\pi kn}{N}}$$





Fourier transform and convolution

- The discrete-time Fourier transform is defined as

$$X(e^{j\omega}) \triangleq \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

- If $x[n] = g[n] * h[n]$ is the result of a convolution
- then in the frequency domain it becomes multiplication

$$X(e^{j\omega}) = G(e^{j\omega}) \cdot H(e^{j\omega})$$

Fourier transform and convolution proof

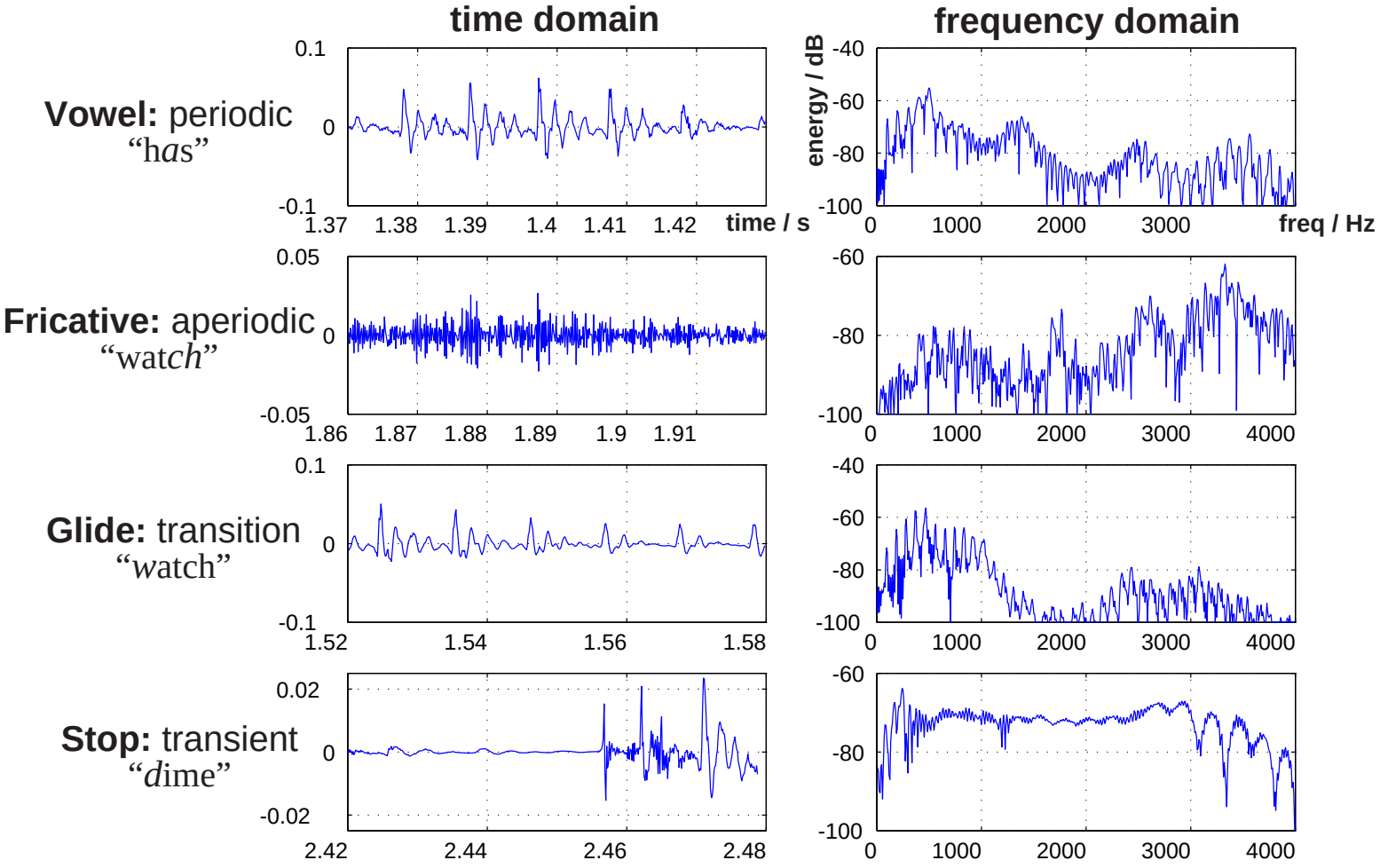
- The discrete-time Fourier transform is defined as

$$X(e^{j\omega}) \triangleq \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

- If $x[n] = g[n] * h[n]$ then

$$\begin{aligned}
X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} (g[n] * h[n])e^{-j\omega n} \\
&= \sum_{n=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} g[k]h[n-k] \right) e^{-j\omega n} \\
&= \sum_{k=-\infty}^{\infty} g[k]e^{-j\omega k} \sum_{n=-\infty}^{\infty} h[n-k]e^{-j\omega(n-k)} \\
&= G(e^{j\omega}) \cdot H(e^{j\omega})
\end{aligned}$$

Speech sounds in the Fourier domain



$$\text{dB} = 20\log_{10}(\text{amplitude}) = 10\log_{10}(\text{power})$$

Voiced spectrum has pitch + formants

Complex exponentials are eigen-functions of LTI systems

- Why do we use complex numbers in signal processing?
- B/c complex exponentials are eigen-functions of LTI systems

$$
\begin{aligned}
x[n] &= \alpha_0 \exp(j\omega n) \\
y[n] &= \sum_{\ell=-\infty}^{\infty} x[n-\ell]h[\ell] \\
&= \sum_{\ell=-\infty}^{\infty} \alpha_0 \exp(j\omega(n-\ell))h[\ell] \\
&= \alpha_0 \exp(j\omega n) \sum_{\ell=-\infty}^{\infty} \exp(-j\omega\ell)h[\ell] \\
&= \alpha_1 \exp(j\omega n)
\end{aligned}
$$

- Output same frequency as input, but different (complex) gain

Frequency-domain summary

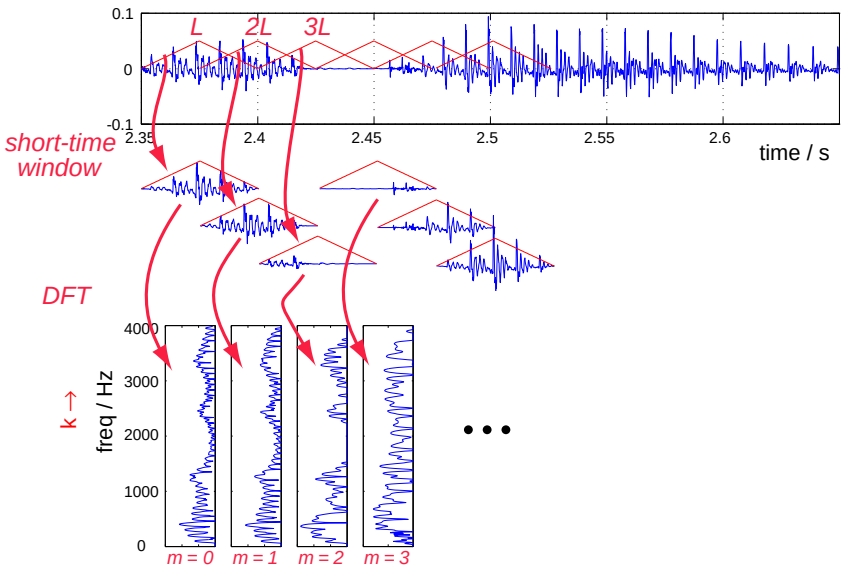
- Any signal can be approximated as sum of weighted sinusoids
- Weights of the sinusoids computed using Fourier analysis
- Convolution in the time domain is equivalent to multiplication in the frequency domain
- Complex exponentials are eigen-functions of LTI systems

Outline

Short-time Fourier Transform

Want to localize energy in time and frequency

- break sound into short-time pieces
- calculate DFT of each one

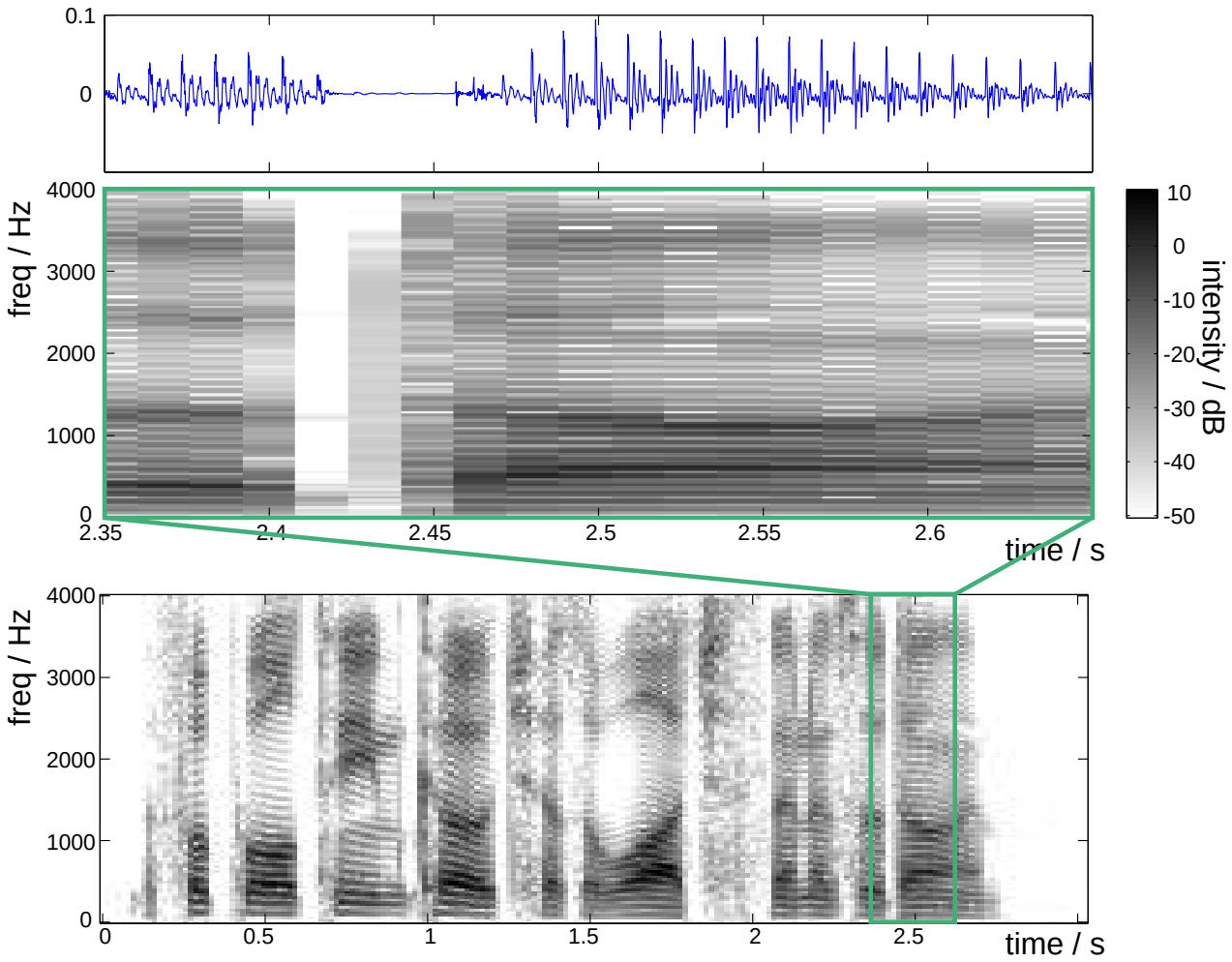


Mathematically,

$$X[k,m] = \sum_{n=0}^{N-1} x[n]\, w[n-mL] \exp\left(-j\frac{2\pi k(n-mL)}{N}\right)$$

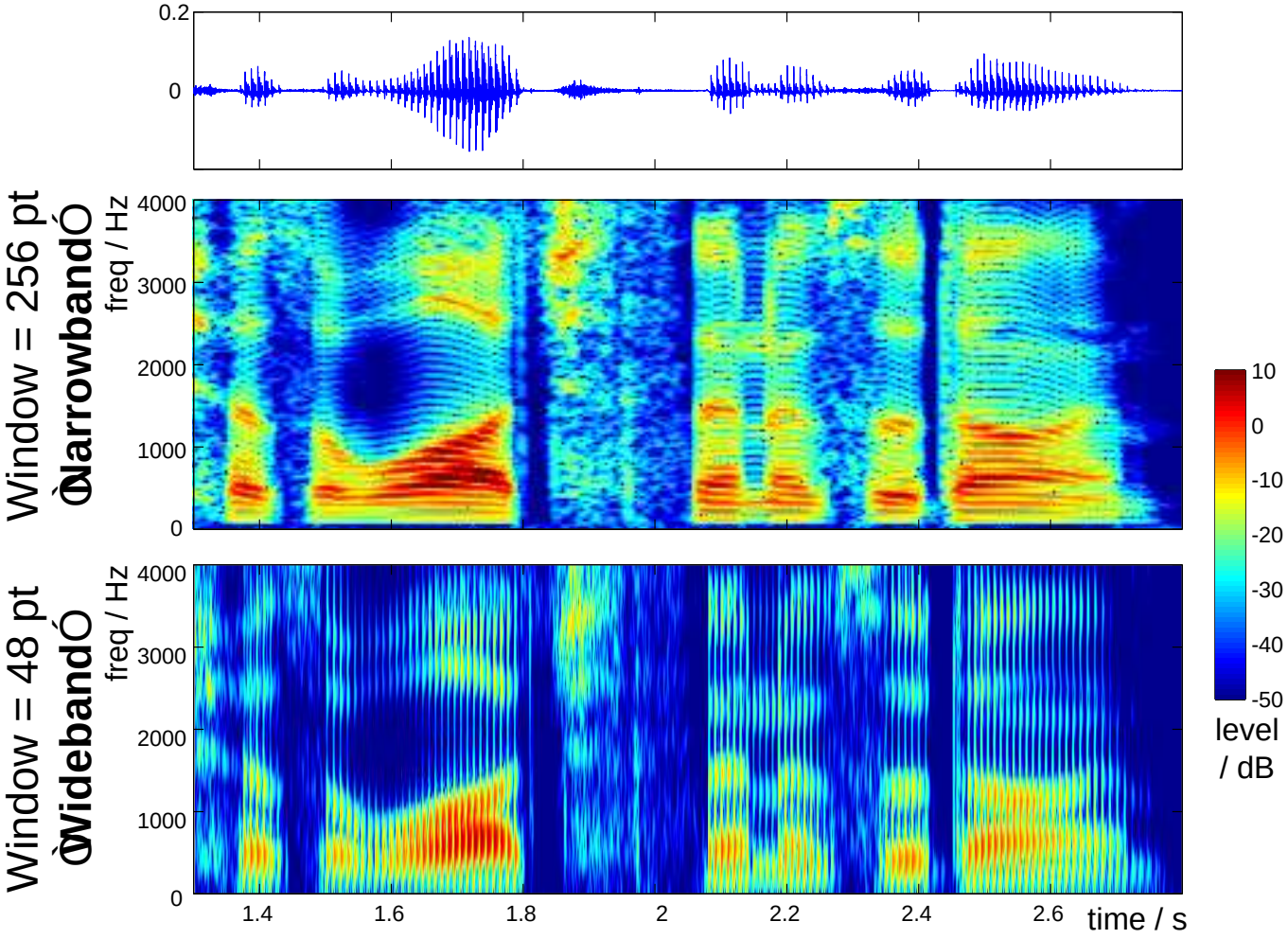
The Spectrogram

Plot log-amplitude STFT, $20 \log_{10} |X[k, m]|$, as a gray-scale image



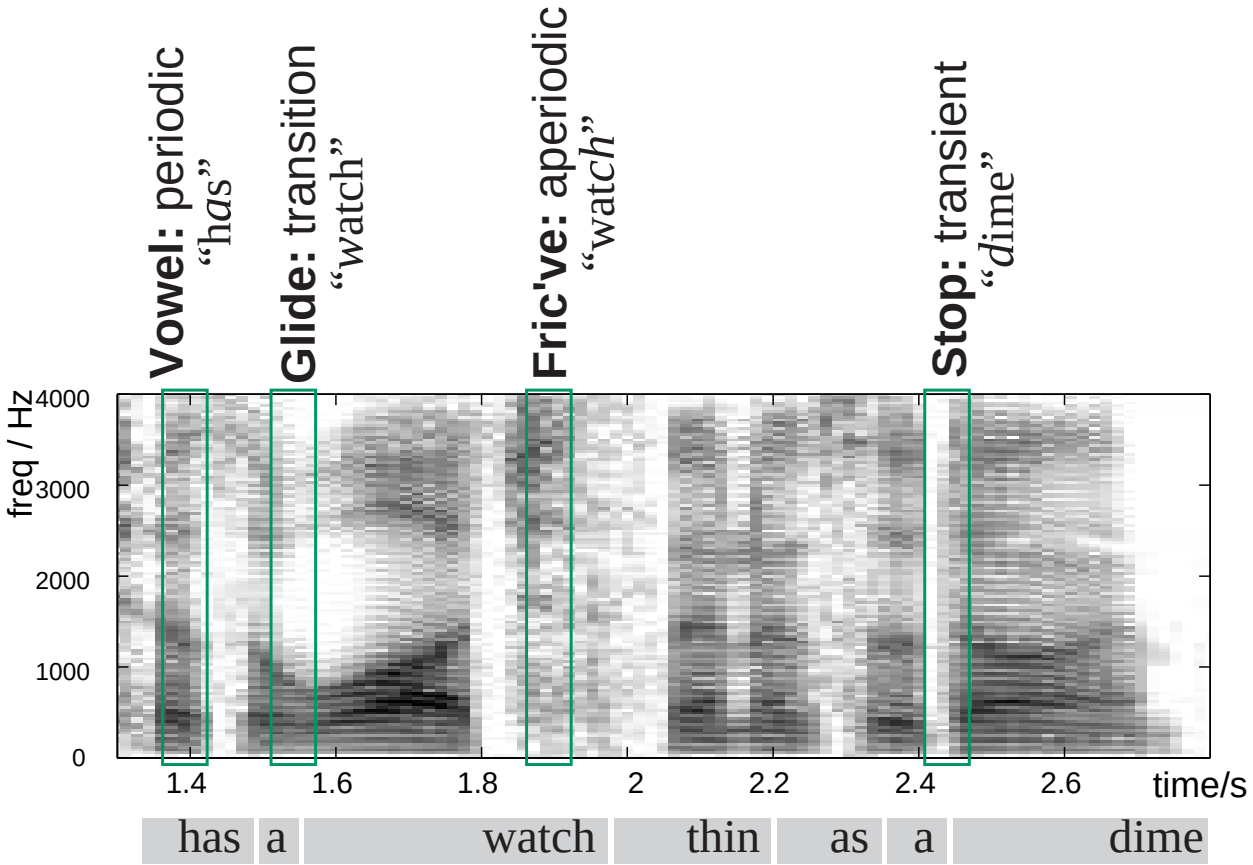
Time-frequency tradeoff

Longer window $w[n]$ gains frequency resolution at cost of time resolution



Speech sounds on the Spectrogram

Most popular speech visualization



Wideband (short window) better than narrowband (long window) to see formants

A spectrogram is equivalent to a bank of filters[1]

- The STFT can be interpreted as a bank of filters in two ways

$$X[k,m] = \sum_{n=0}^{N-1} x[n] \underbrace{w[n-mL] \exp\left(-j\frac{2\pi k(n-mL)}{N}\right)}_{\text{Bandpass filter}}$$

$$X[k,m] = \sum_{n=0}^{N-1} \underbrace{w[n-mL]}_{\text{low-pass filter}} \underbrace{x[n] \exp\left(-j\frac{2\pi k(n-mL)}{N}\right)}_{\text{Modulated signal}}$$

- k (really $2\pi k/N$) is the center frequency of each filter
- Only look at the output of the filter every L samples

[1] See `https://www.dsprelated.com/freebooks/sasp/Filter_Bank_Summation_FBS.html`

Summary

- Information in sound
  - lots of it, multiple levels of abstraction
- Course overview
  - survey of audio processing topics
  - readings, presentations, project
- DSP review
  - digital signals, time domain
  - Fourier domain, STFT