Perception & Multimedia Computing

Week 15 – Music Perception Pt. 2; Intro. To Signals & Systems

Michael Zbyszyński
Lecturer, Department of Computing
Goldsmiths University of London

Melody, Harmony, Tempo, & Rhythm

Melody

Definition and structural use of melody varies among cultures (e.g., rock, classical, gamelan, …)

Basic definition: Sequence of pitched events (notes) perceived as a single entity, unfolding in time

Harmony

Perceptual quality when pitched sounds (notes) occur simultaneously

Relative intervals determine harmony/chord, not *absolute* notes/pitches

e.g., C-E-G equivalent to D-F#-A
C-Eb-G equiv. to D-F-A

Music theory: Certain chords tend to follow other chords; certain notes tend to move to other notes when chord changes

Cultural exposure leads to learned expectations

We learn through repeated exposure that certain melodic patterns are likely to resolve in certain ways, and that certain harmonies follow others.

Composers and songwriters work within learned system of expectations, surprising us or satisfying us through careful deviation from & adherence to “the rules.”

Individual compositions also develop and then break their own systems of expectations.

Schematic expectation

Expectation learned from cultural context of music

How do you expect a scale to resolve?

How do you feel when it resolves?

satisfaction ("prediction response")

How do you feel when it doesn't?

attention ("tension response")

Attention

Certain musical phenomena demand attention: unexpected phenomena, increases in volume, approaching sounds, ...

Arousal: physiological response to attention (increasing heart rate, perspiration); stress!

Can be pleasurable or stressful

Veridical expectation

Expectation learned by exposure to a piece of music

Satisfaction also arises when we know what note is coming next in *this particular song*

But schematic expectations are strong, and unusual phenomena in a familiar piece of music still “surprise” us

Consequences

Your unique listening history shapes the way you hear.

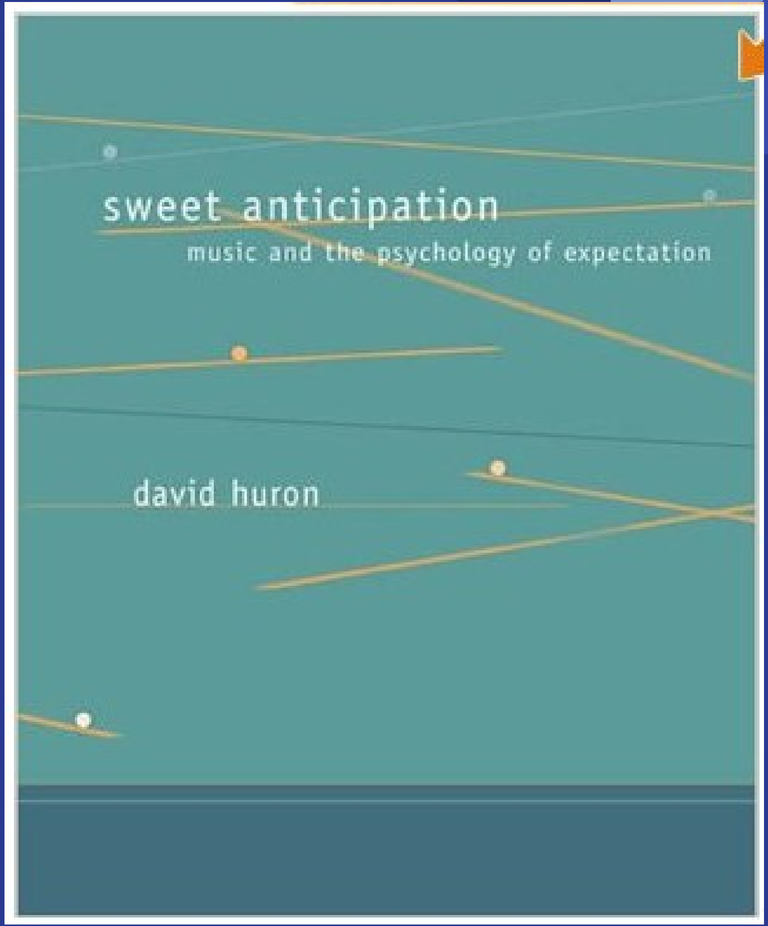
Composers & musicians manipulate expectations to capture your attention, surprise you, satisfy you, …

Consequences

Your unique listening history shapes the way you hear.

Composers & musicians manipulate expectations to capture your attention, surprise you, satisfy you, …

Recommended reading!



Also:
http://www.music-cog.ohio-state.edu/Music829D/Notes/Expectation.html#Reality versus Appearances

Tempo

Natural “pulse” speed of music

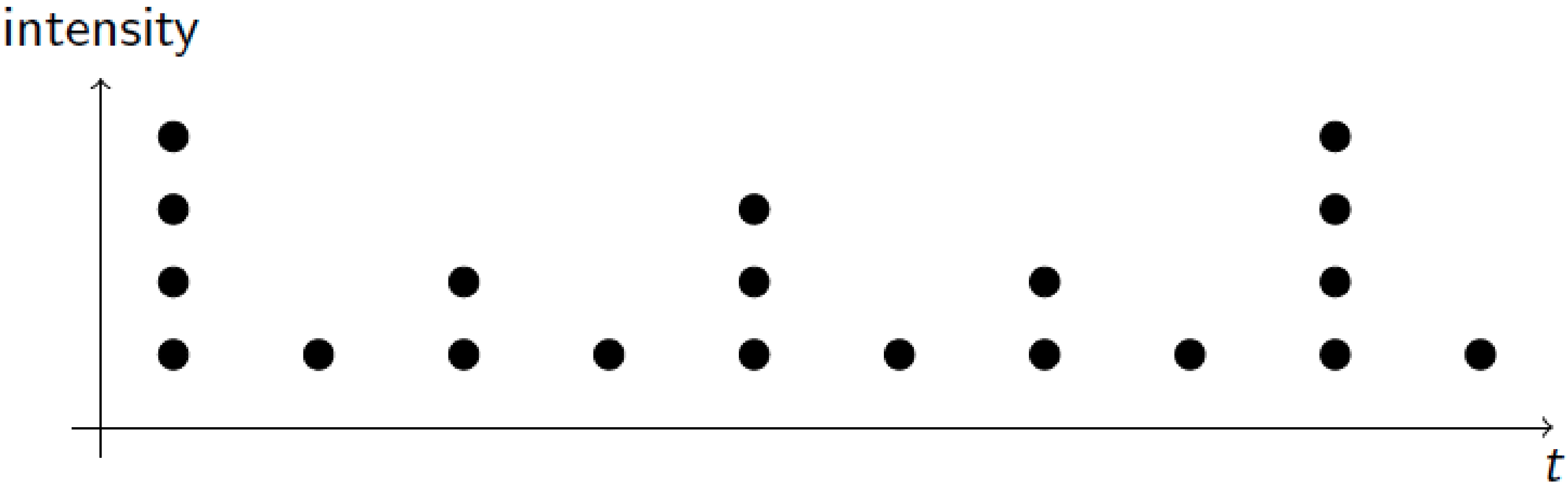
Often ambiguous (half/double speed)

Humans’ preferred beat interval: around 0.2-0.8s

Metrical structure

Hierarchy of temporal groups:

Beats – Bars – Four-bar patterns – larger groups (e.g., 12- or 16-bar segments)



Rhythm

In a nutshell, which elements of the hierarchy to emphasize?

Music Summary:

- When do you hear sound as consonant/dissonant? Pitched/unpitched? Single/multiple sound sources?
- What are timbre, pitch, melody, harmony, tempo, rhythm?
- How do amplitude, frequency, and phase impact what we hear?
- What is a spectrum? Why is spectrum useful in reasoning about sound perception?
- Can you look at a spectrum and reason about what you will hear?
- Can you reason about how your expectations for melody, rhythm, harmony impact your experience of music?

Signals & Systems

What is a Signal?

- Any quantity varying over space or time
- Maths: *function* Physics: *field*

What is a Signal?

- Any quantity varying over space or time
- Maths: *function* Physics: *field*
  - *Electrical current at a point in a circuit*
  - *Count of students in weekly lectures*
  - *Temperature at all points in room*

For now, one-dimensional signals:

- Continuous time signals
  - Current at a point in a circuit
  - Temperature at a location
  - Sound pressure level

For now, one-dimensional signals:

- Discrete time signals
    - Current at a point in a circuit, measured once per second
    - Temperature at a location, measured hourly
    - Sound pressure level, measured 44,100 times per second

1-dimensional, discrete-time signals in Python

```
x = [1,2,3]
t = np.arange(0, 1, 1/44100)
s = sin(2*pi*100*t)
```

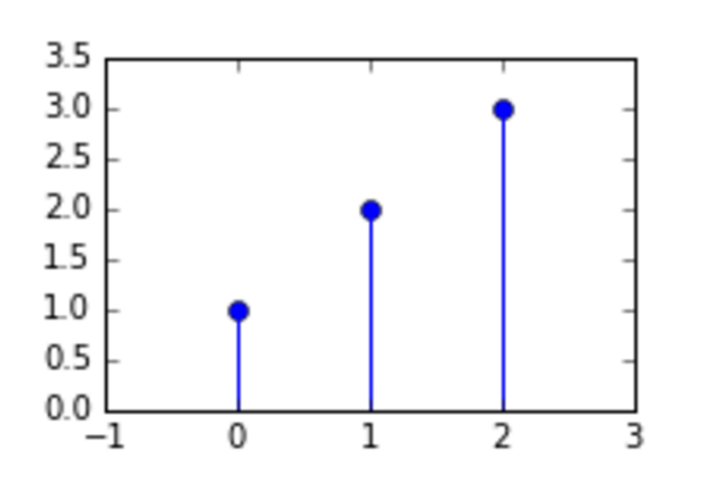
These are just arrays!

We interpret each element as happening at a certain time

e.g., s[10] happens 10 samples after "time 0"

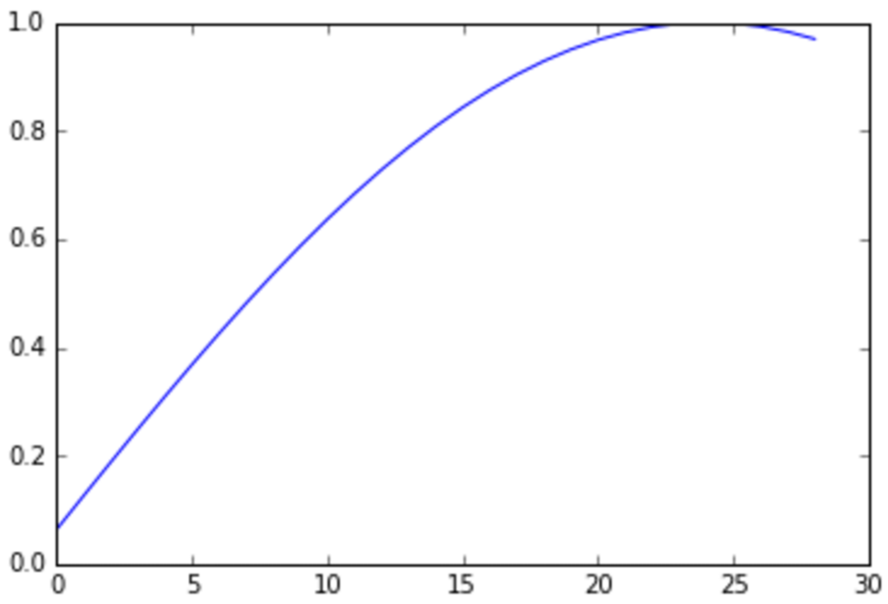
```
x = [1,2,3]
print x
fig, axes = plt.subplots(figsize=(3,2))
axes.stem(x)
ylim([0, 3.5])
plt.xticks(np.arange(-1, 4, 1.0))
```

```
[1, 2, 3]
```



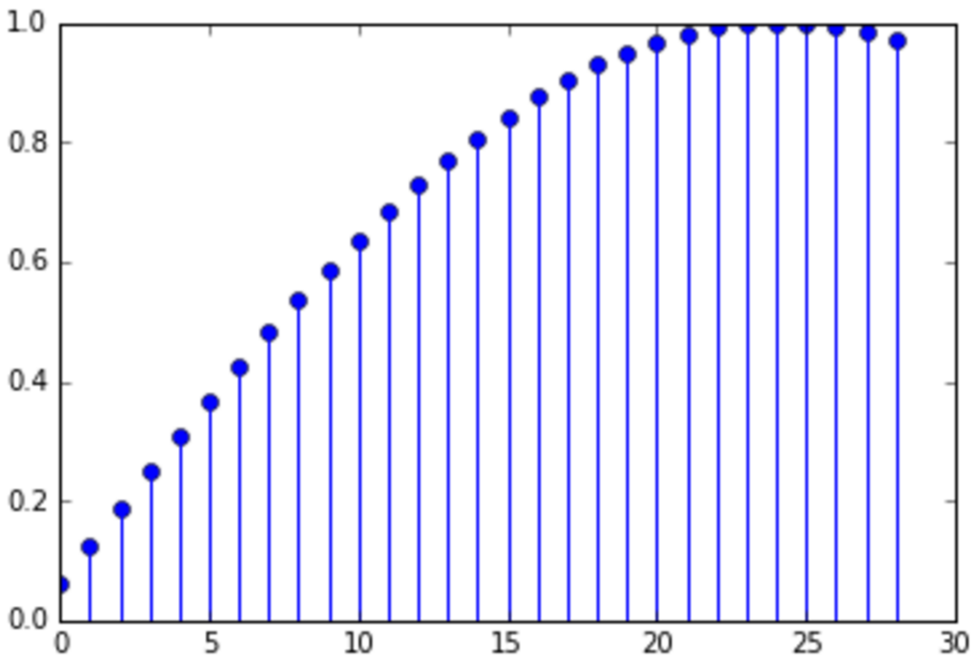
```
t = np.arange(0, 1, 1/44100)
s = sin(2*pi*440*t)
plot(s[1:30])
```

```
[<matplotlib.lines.Line2D at 0x10ff02210>]
```



```
t = np.arange(0, 1, 1/44100)
s = sin(2*pi*440*t)
stem(s[1:30])
```

<Container object of 3 artists>



Systems

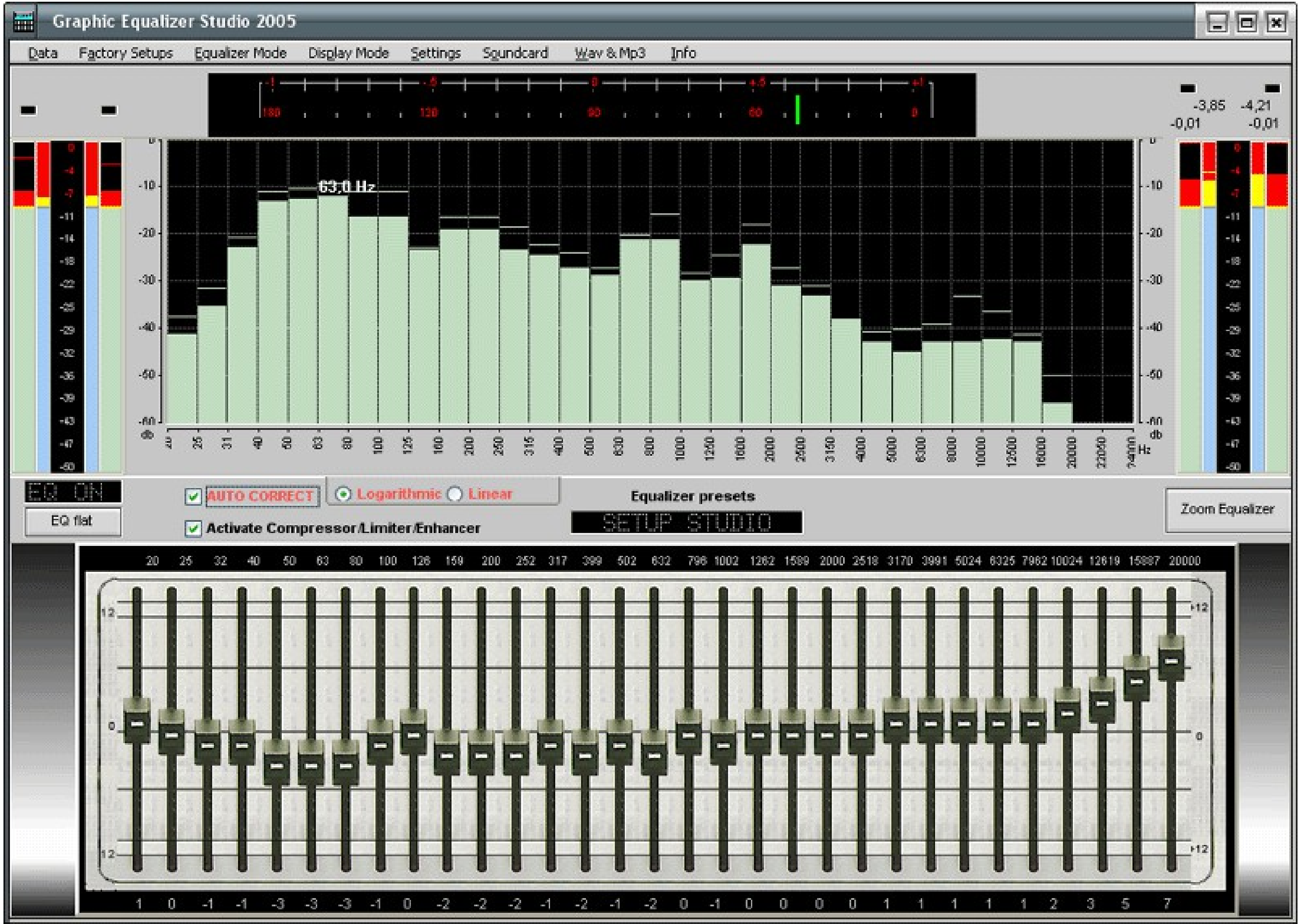
- Systems “construct” new signals from existing ones
- Examples
  - Computer monitor (in: electrical signal, out: light)
  - Violin (in: bow movement; out: sound)
  - Car suspension (in: bumps; out: smooth ride)

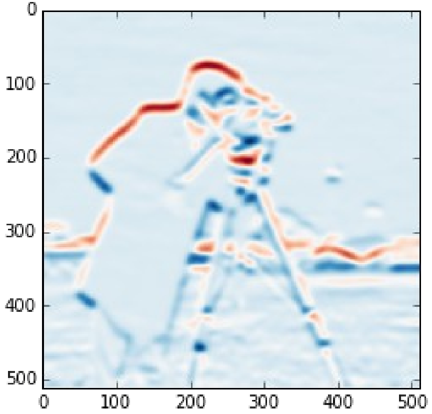
Systems

Compute output signal (response) from input

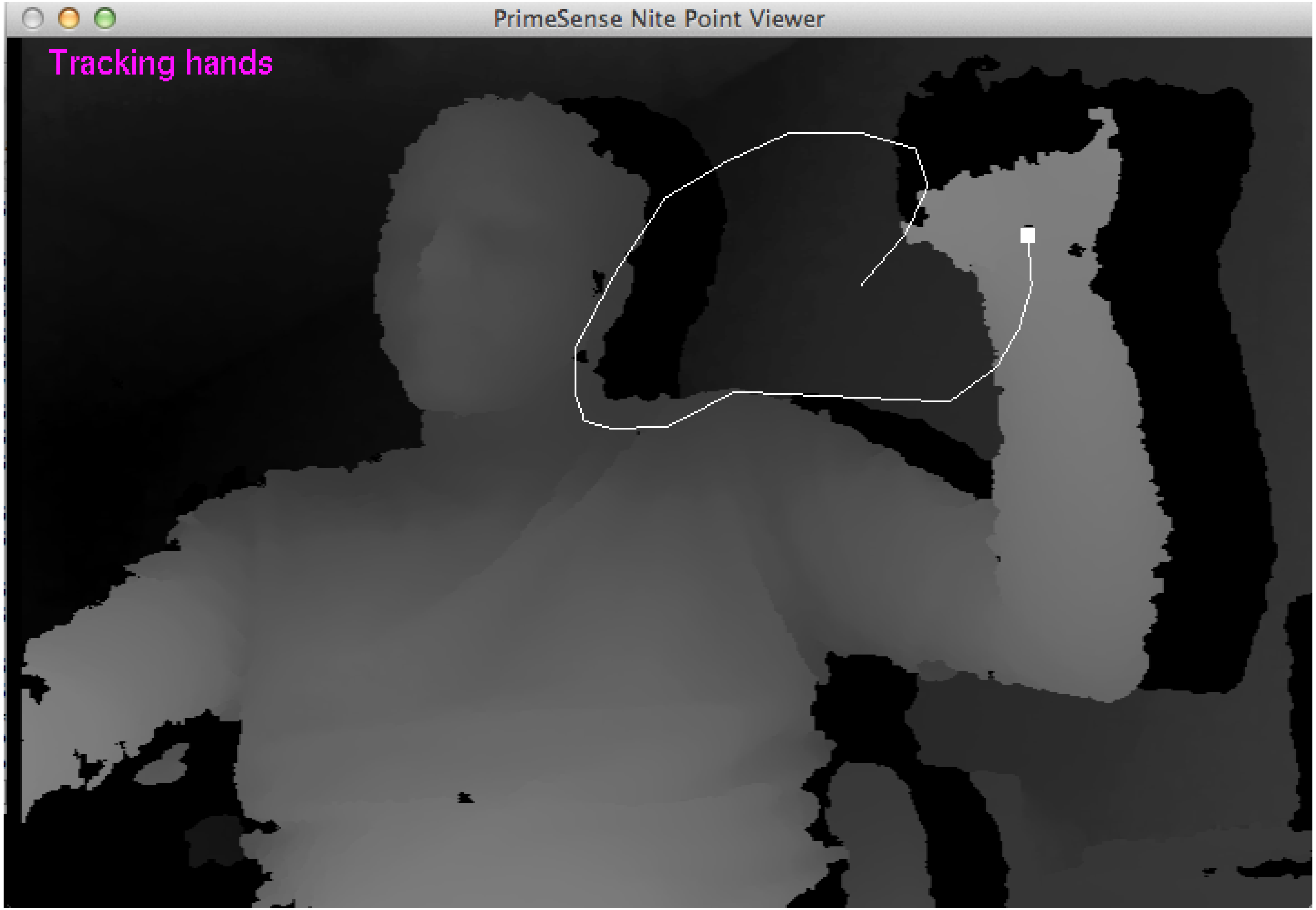
[1,2,3,4,5] → System → [2,4,6,8,10]

Examples of systems used in multimedia


Graphic Equalizer Studio 2005
Data
Factory Setups
Equalizer Mode
Display Mode
Settings
Soundcard
Wav & Mp3
Info
-3,85
-4,21
-0,01
-0,01
63,0 Hz
db
Hz
EQ ON
EQ flat
AUTO CORRECT
Logarithmic
Linear
Activate Compressor/Limiter/Enhancer
Equalizer presets
SETUP STUDIO
Zoom Equalizer
20 25 32 40 50 63 80 100 126 159 200 252 317 399 502 632 796 1002 1262 1589 2000 2518 3170 3991 5024 6325 7962 10024 12619 15887 20000
1 0 -1 -1 -3 -3 -3 -1 0 -2 -2 -2 -1 -2 -1 -2 0 -1 0 0 0 0 1 1 1 1 1 2 3 5 7



0
100
200
300
400
500
0
100
200
300
400
500

设计悬空
4KONG.COM
Topaz DeNoise 注册版效果图
处理前
处理后


PrimeSense Nite Point Viewer
Tracking hands

The magical system-building genie

“I wish I had a system (e.g., an audio or image filter) that could change the spectrum of my signal in *exactly this way…*”

e.g., “Remove frequencies at 60Hz and leave everything else untouched’

or “Boost all frequencies in 1000Hz to3000Hz range”

or “Remove the high-frequency noise in this image”

Done!!!

(If you meet just a few conditions, then follow my instructions)

“I have a system here that I like. I wish I could re-create this system in my own software…”

e.g., “I like how music sounds when played in this giant cathedral. I wish I could make all my music sound like it’s recorded here.”

or

“I really like a ‘blur’ effect in this photo editing software. I wish I could make this effect in my own software.”

Done!!!

(If you meet just a few conditions, then follow my instructions)

The magical system-analysis genie

“I have a system here (e.g. I can apply it to an example, or look at its code). I wish I could reason about what kind of effect it would have on any new signal, without having to try it out!”

Done!!!

(If you meet just a few conditions, then follow my instructions)

What are the conditions?

Systems have to be

1. Linear
2. Time-invariant

(Or for images, shift-invariant)

Notation
+
Linearity & Time-Invariance

Signal notation

These are all equivalent:

- $x[n] = [3, 5, 12]$

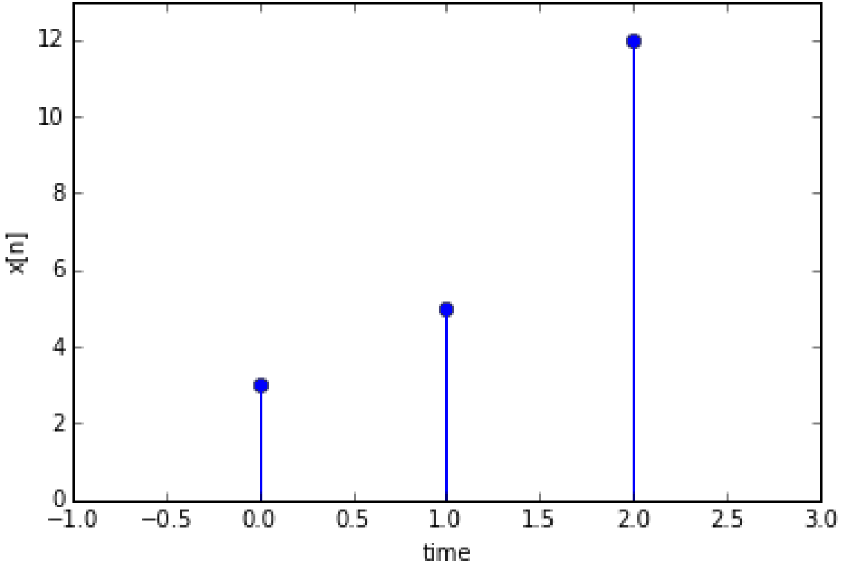
- $x[0] = 3$, $x[1] = 5$, $x[2] = 12$

- $x[n] = [3, 5, 12, 0, 0, 0, \ldots.]$

and $x[-1] = 0$, $x[-100] = 0$, etc.

Signal notation

Also equivalent: $x[n] = [3, 5, 12]$



Notation: applying a system to a signal

Given an input signal *x*, the action of a system H on that signal, producing an output signal y is denoted:

$$y = H\{x\}$$

Notation: applying a system to a signal

For a discrete-time signal x[n] :

$$y[n] = H\{x[n]\} \text{ or } y[n] = H\{x\}[n]$$

Notation: applying a system to a signal

For a discrete-time signal x[n] :

$$x[n] \longrightarrow \boxed{\text{System } H} \longrightarrow y[n] = H\{x\}[n]$$

Example

A system:

$$H\{x[n]\} = 0.5 * x[n]$$

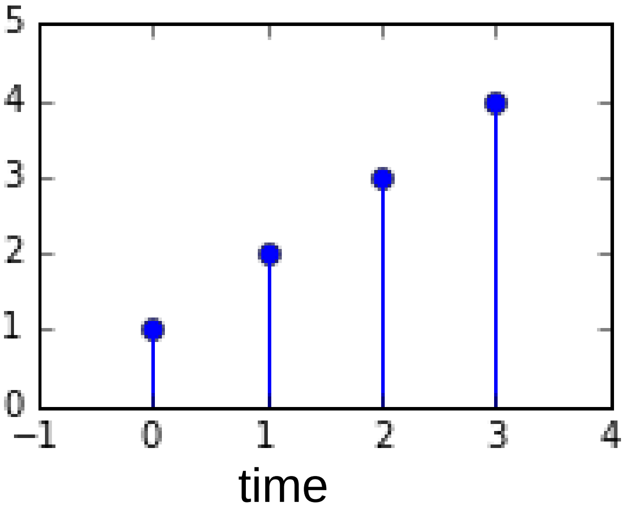
A signal:

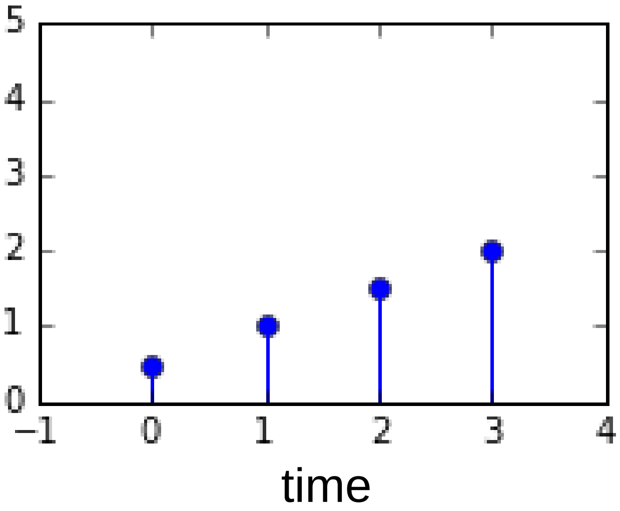
$$x[\mathrm{n}] = [1, 2, 3, 4]$$

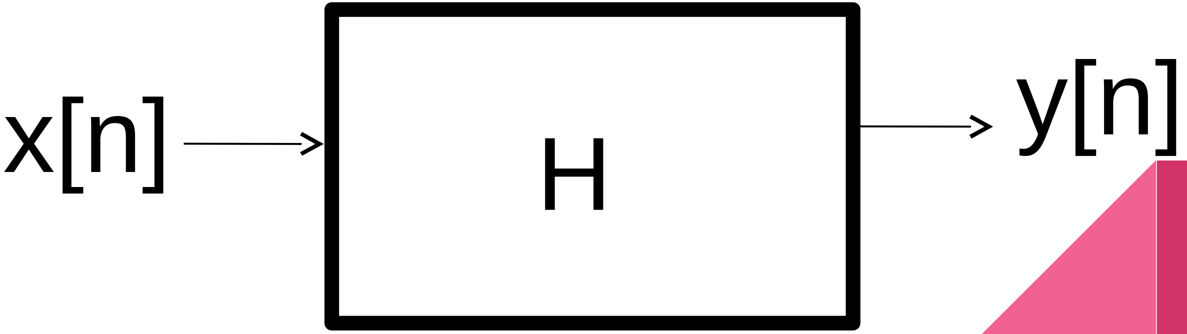
$$y[n] = H\{x[n]\} = ???$$

$$y[n] = [0.5, 1, 1.5, 2]$$

What kind of system might *H* be?


0
1
2
3
4
5
−1
0
1
2
3
4
time

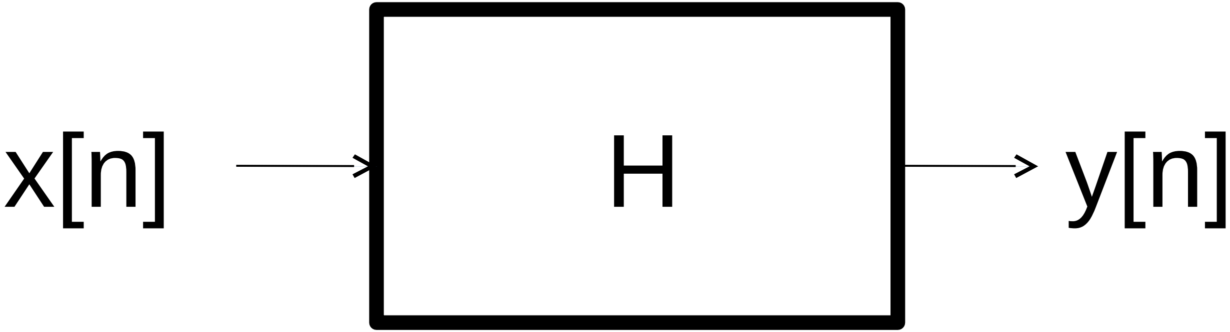

0
1
2
3
4
5
−1
0
1
2
3
4
time


x[n]
H
y[n]

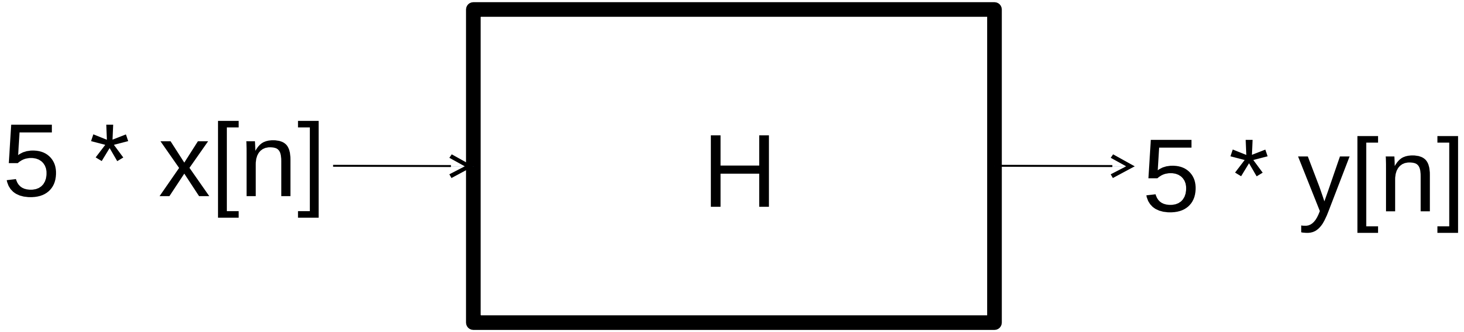
Linearity

Linear systems have the property that *scaling* and *superposition* of their input signals lead to a corresponding scaled superposition of their outputs.

Scaling



Scaling



Scaling

$$-3 * x[n] \longrightarrow \boxed{\quad H \quad} \longrightarrow -3 * y[n]$$

Example

Scaling: for any input signal x_1 where $y_1 = H\{x_1\}$,

$H\{\alpha x_1\} = \alpha y_1$ for any scalar value α

x1 = [2, 4, 6] y1 = H{x1} = [3, 18, 4]

x2 = [4, 8, 12] y2 = H{x2} = ?

x1 = 2 * x1 so y2 = 2* y1

Linearity

Linear systems have the property that *scaling* and *superposition* of their input signals lead to a corresponding scaled superposition of their outputs.

Superposition

$$x_1[n] \longrightarrow \boxed{H} \longrightarrow y_1[n]$$

$$x_2[n] \longrightarrow \boxed{H} \longrightarrow y_2[n]$$

$$x_1[n]+ x_2[n] \longrightarrow \boxed{H} \longrightarrow y_1[n]+ y_2[n]$$

Example

Scaling: for any two input signals x_1 and x_2 where $y_1 = H\{x_1\}$ and $y_2 = H\{x_2\}$, $H\{x_1 + x_2\} = y_1 + y_2$

x1 = [2, 4, 6]
x2 = [1, 5]
X3 = [5, 2, 4]
x3 = x1 + x2

so

y1 = H{x1} = [3, 18, 4]
y2 = H{x2} = [1, -3, 5]
y3 = ???
y3 = y1 + y2

y3 = [4, 13, 7]

Linearity

Linear systems have the property that *scaling* and *superposition* of their input signals lead to a corresponding scaled superposition of their outputs.

Example

x1 = [5]
x2 = [0, 1, 2]
x3 = [5, 2, 4]

y1 = H{x1} = [-5]
y2 = H{x2} = [0, 6, 3]
y3 = ?

x3 = x1 + 2*x2 so y3 = y1 + 2*y2

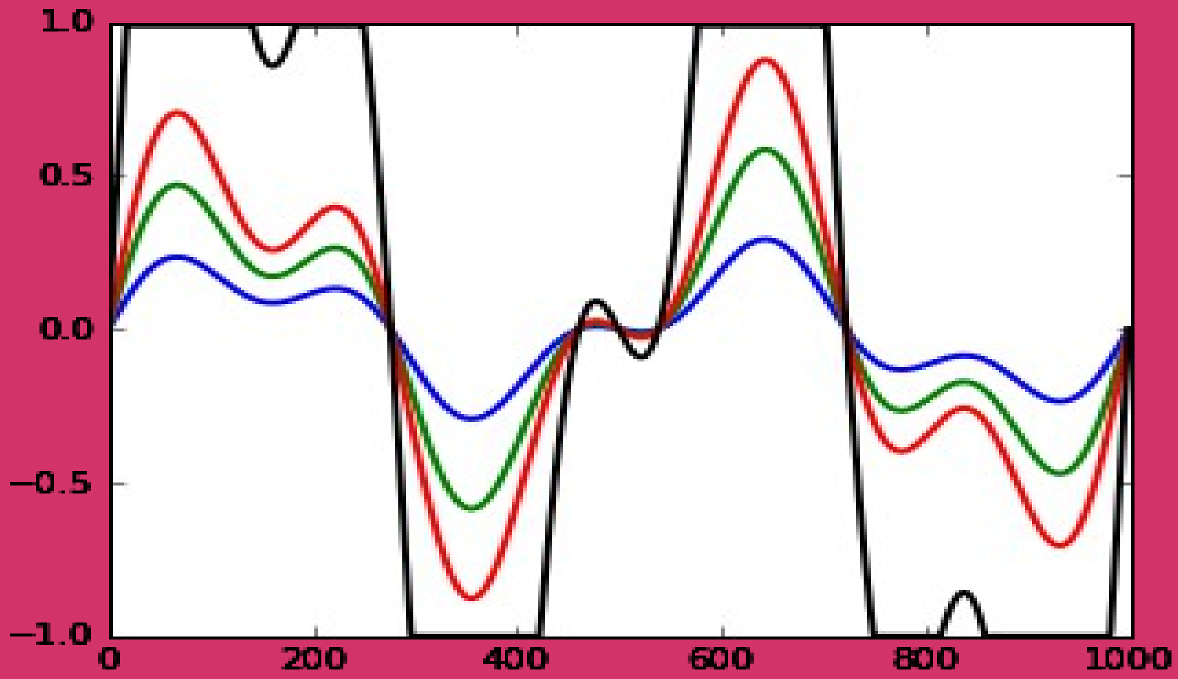
y3 = [-5, 12, 6]

Linearity

Almost all systems in the real world are linear systems for *small enough signals*.

e.g. audio signal that doesn't clip

Nonlinear: Volume clipping



Time-invariance

Time-invariant systems have the property that the output signal of the system for a given input signal does not depend explicitly on absolute time.

Time-invariance

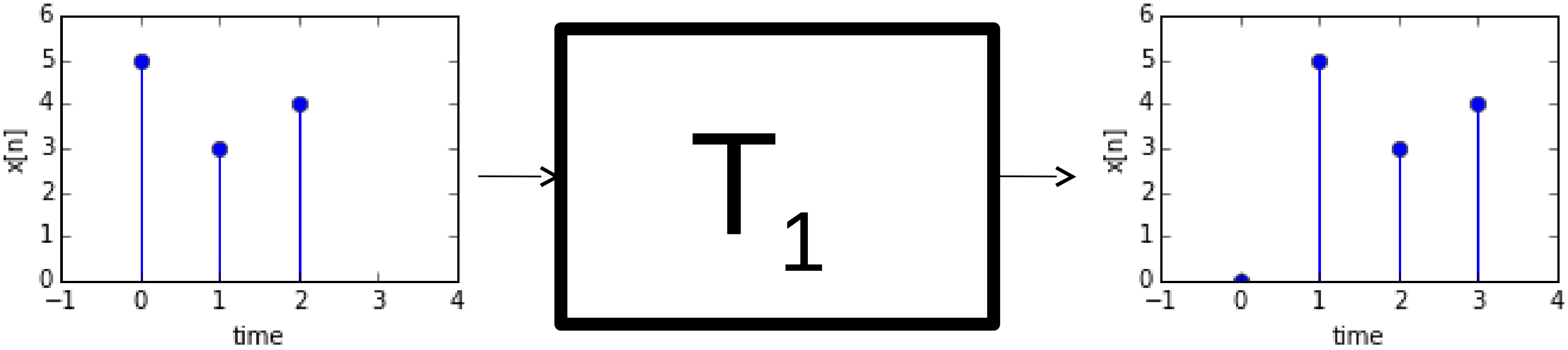
E.g., does your guitar amp affect your guitar differently on Monday than on Tuesday?

Does your image filter blur your image differently at 12:00 than at 11:00?

Probably not: These are time-invariant!

Unit Delay System

Just delays signal by 1 sample



The Unit Delay System

A special linear system T_1 whose output signal is the input signal, but delayed by one time unit.

$$y = T_1\{x\}$$

Or with discrete time explicitly represented:

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

This unit delay system is the building block of the systems we will cover in this course.

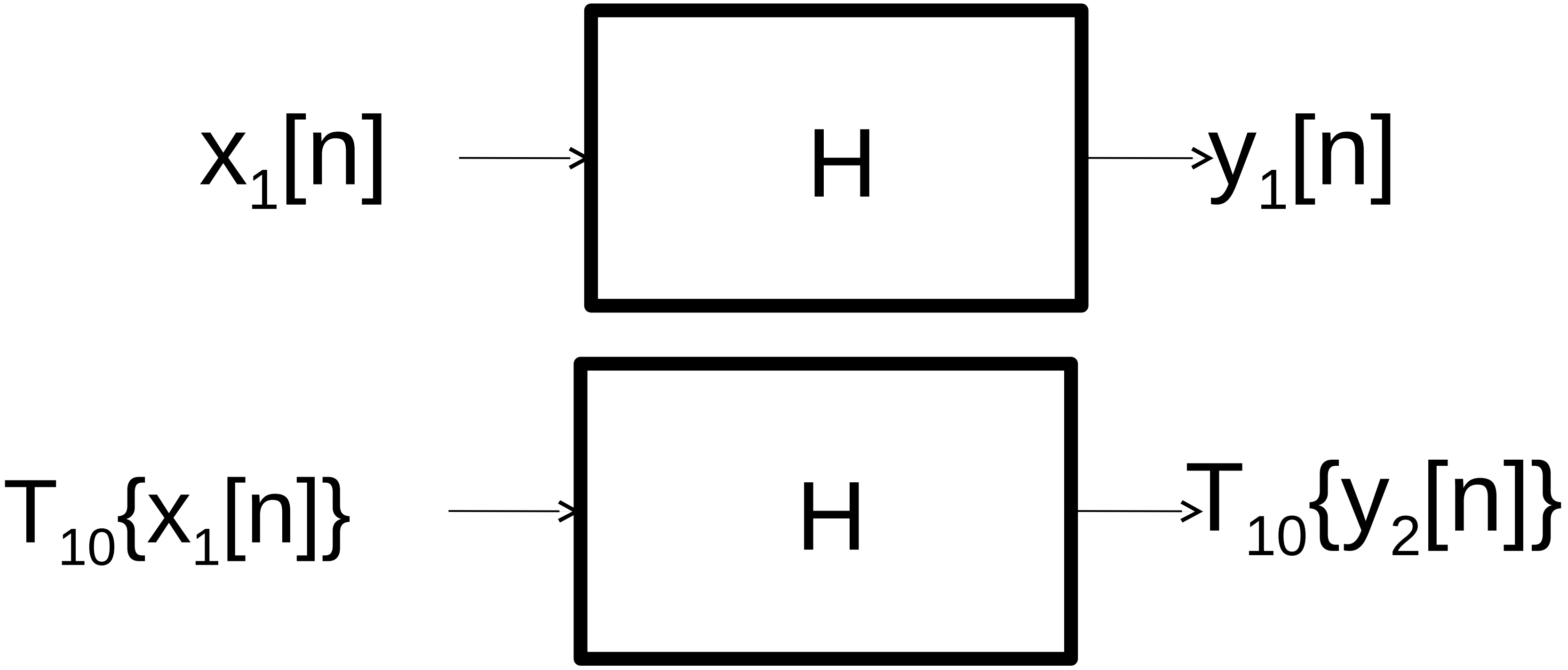
Time-invariant systems have the property that the output signal of the system for a given input signal does not depend explicitly on absolute time.
For any input signal x with y = H{x}, the system H is time-invariant if:

$$H\{T_\delta\{x\}\} = T_\delta\{y\}$$

Where T_δ is a delay system for arbitrary delay.

Passing a delayed sound through a time-invariant system gives the same result as passing the sound through the system first,
then delaying it.

Time invariance



Example

H is time-invariant

$x1 = [1, 2, 3]$

and $y1 = H\{x1\} = [3, 4, 5]$

$x2 = [0, 0, 1, 2, 3]$

$y2 = H\{x2\} = ?$

$x2 = T2\{x1\}$

so $y2 = T2\{y1\} = [0, 0, 3, 4, 5]$

Linear Time-Invariant or **LTI** Systems have both the linear property and the time-invariant property.

Why do we care?

Reason 1:

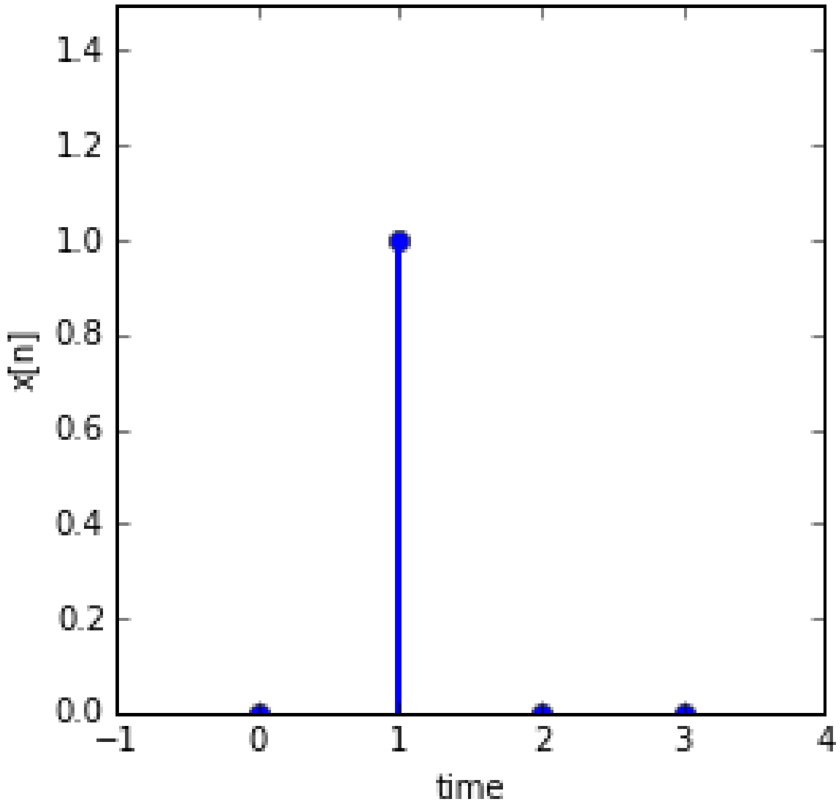
If we know the response of an LTI system for the signal below, we can compute the response to any signal whatsoever!

$$d[n] = [1, 0, 0, 0, \ldots] = [1]$$

The unit impulse signal

This special signal is called the unit impulse

d[n] = [1, 0, 0, 0, …] = [1]



The unit impulse signal

d[n]

1 at time 0, 0 everywhere else