Digital Signal Processing

Lecture 11 – LTI systems

상명대학교
컴퓨터과학과
강상욱 교수

상명대학교 SANGMYUNG UNIVERSITY

Signals and systems I

- In the context of signal processing, a system is
  - An abstract representation of anything that takes a signal as input and produces a signal as output.
  - Example : an electronic amplifier is a system that takes an electrical signal as input and produces a louder signal as output.
- A linear time-invariant system
  - Linearity : if an input x_1 produces output y_1 and another input x_2 produces y_2, then $ax_1 + bx_2 = ay_1 + by_2$.
  - Time invariance : the effect of the system doesn't vary over time, or depend on the state of the system.
- Many physical systems have these properties, at least approximately.
  - Example : circuits, mechanical systems and so on.
- LTI systems are described by linear differential equations, and the solutions of those equations are complex sinusoids.

$$\frac{dy}{dx} + p(x)y = g(x), \quad y = ?$$

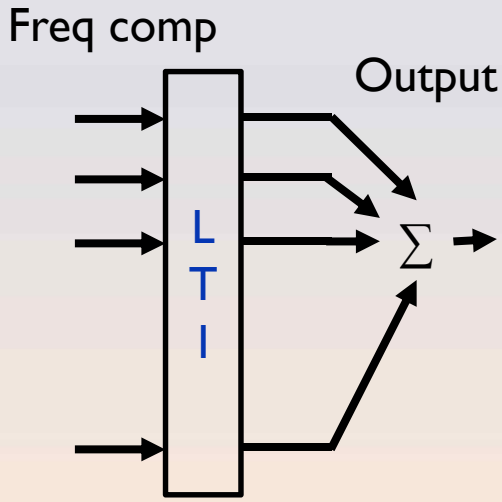
Signals and systems 2

- This result provides an algorithm for computing the effect of an LTI system on an input signal.
  - Express the input signal as the sum of complex sinusoid components.
  - For each input component, compute the corresponding output component.
  - Add up the output components.

This process is called : spectral decomposition
We decompose the input signal into its spectral components.

In order to apply this process to an LTI system,
we have to characterize the system by finding its effect on each component of the input signal.

How to do it?
Just give an impulse to the LTI system as an input signal.



Impulse response

See the spectral analysis of an impulse signal

```
impulse = np.zeros(8)
impulse[0] = 1
impulse_spectrum = np.fft.fft(impulse)
```

```
[ 1.  0.  0.  0.  0.  0.  0.  0.]
```

```
[ 1.+0.j  1.+0.j  1.+0.j  1.+0.j  1.+0.j  1.+0.j  1.+0.j  1.+0.j]
```

An impulse is the sum of components with equal magnitudes at all frequencies.

When you test a system by inputting an impulse, you are testing the response of the system at all frequencies.

You can test all components at the same time? Why?

Windows and filters

- Two-element moving average
  - A system that takes a signal as an input and produces a slightly smoother signal as an output.

```
window_array = np.array([0.5, 0.5, 0, 0, 0, 0, 0, 0,])
window = thinkdsp.Wave(window_array, framerate=8)
```

```
filtr = window.make_spectrum(full=True)
```

Low pass filter with approximate shape of Gaussian curve.

Imagine that we didn't know the window or the corresponding filter, and we wanted to characterize this system.

We need impulse response!!

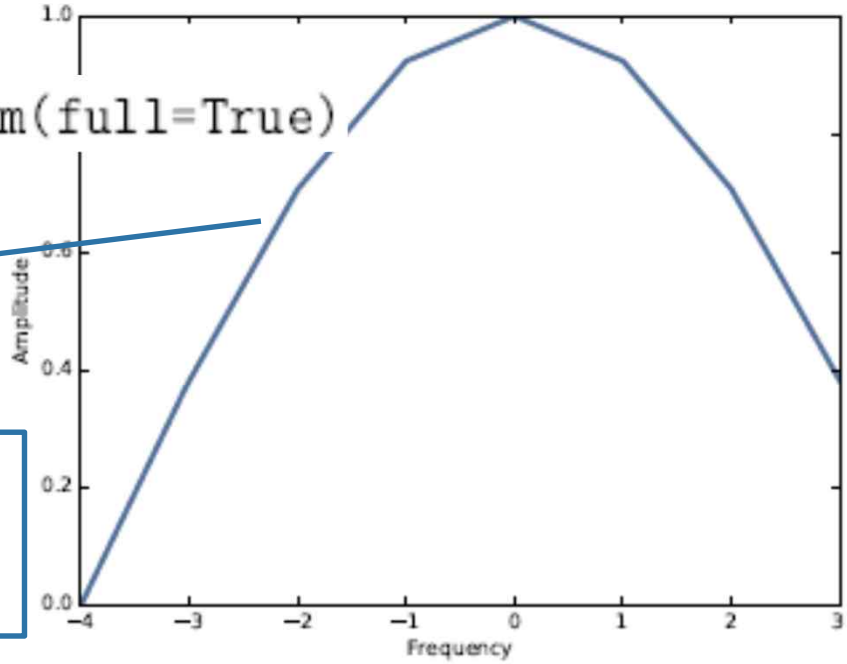


Figure 10.1: DFT of a 2-element moving average window.

Calculating the impulse response

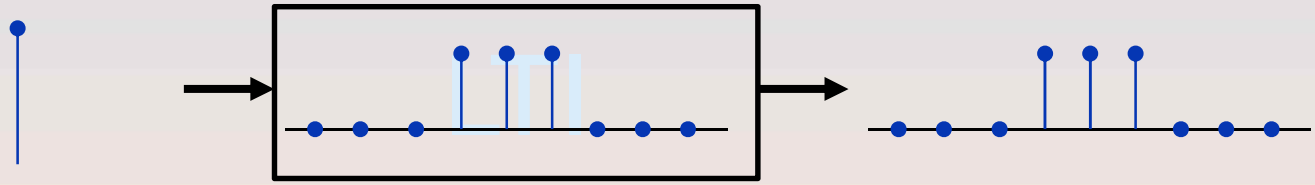
product = filtr
filtered = window

```
product = impulse_spectrum * filtr
filtered = product.make_wave()
```

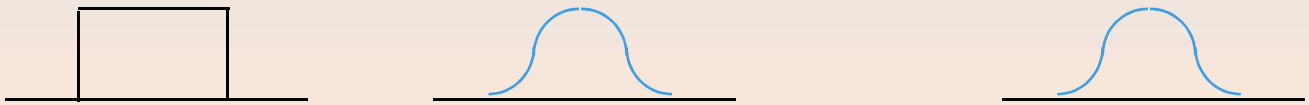
We can compute the impulse response by multiplying the spectrums of the impulse and the filter, and then converting the result from a spectrum to a wave.
Recall the convolution theorem.

Because the spectrum of an impulse is all ones, the DFT of the impulse response is identical to that of the filter that characterizes the system.

Time domain



Frequency domain



Acoustic response

- To characterize the acoustic response of a room or open space, a simple way to generate an impulse is to pop a balloon or fire a gun
  - Input signal : an approximate impulse
  - The approximate impulse response : the sound you hear.

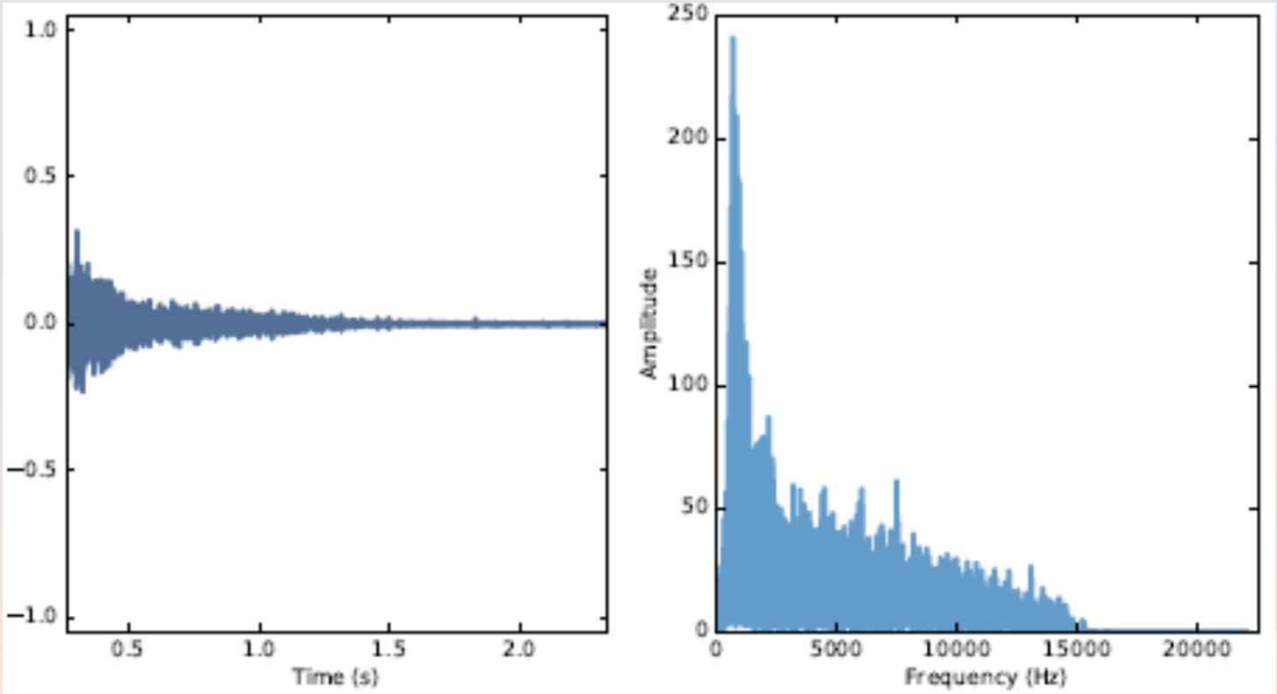


Figure 10.2: Waveform of a gunshot.

Acoustic response example 1

- An example :
  - To characterize the room where the gun was fired, a recording of a gunshot is used.
  - Then the impulse response is used to simulate the effect of that room on a violin recording.

```
response = thinkdsp.read_wave('180961__kleeb__gunshots.wav')
response = response.segment(start=0.26, duration=5.0)
response.normalize()
response.plot()
```

This is the gunshot. (See Figure10.2(left))
start=0.26 : to remove the silence before the gunshot.

```
transfer = response.make_spectrum()
transfer.plot()
```

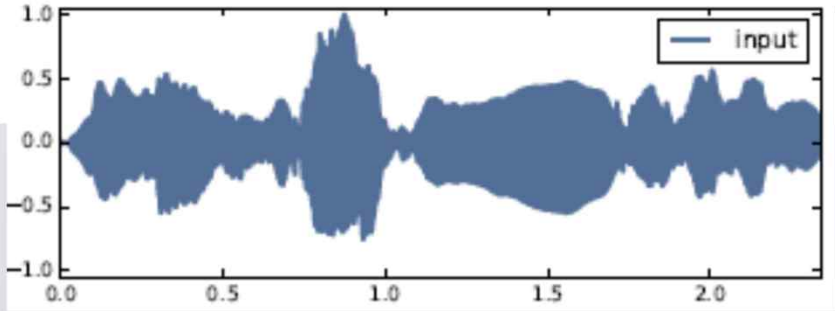
This is the DFT of the gunshot.
(See Figure10.2(right))

The spectrum encodes the response of the room. (Also called "transfer function")
For each freq. the spectrum contains a complex number.
real part : amplitude multiplier, imaginary part : phase shift

Acoustic response example 2

The violin recording is sampled at the same framerate as the gunshot at 44100Hz. And the wave is trimmed to the same length as the gunshot.

```
violin = thinkdsp.read_wave('92002__jcveliz__violin-origional.wav')
violin.truncate(len(response))
violin.normalize()
```



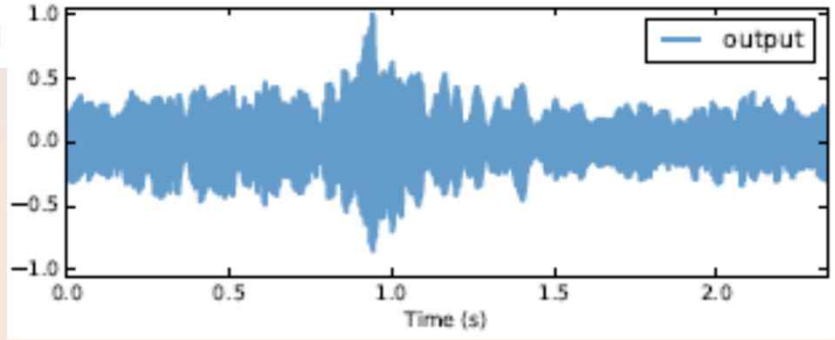
Notice that the original violin sound is recorded in a room…

The DFT of the violin wave.

```
spectrum = violin.make_spectrum()
```

'output' simulates the sound you may hear in the room with the transfer function 'transfer'

```
output = (spectrum * transfer).make_wave()
output.normalize()
output.plot()
```



After hearing the output, can you imagine what the room is like?

Systems and convolution I

- By the convolution theorem
  - Multiplication in the freq. domain corresponds to convolution in the time domain.
  - In this example, the output of the system is the convolution of the input and the system response.

Why?

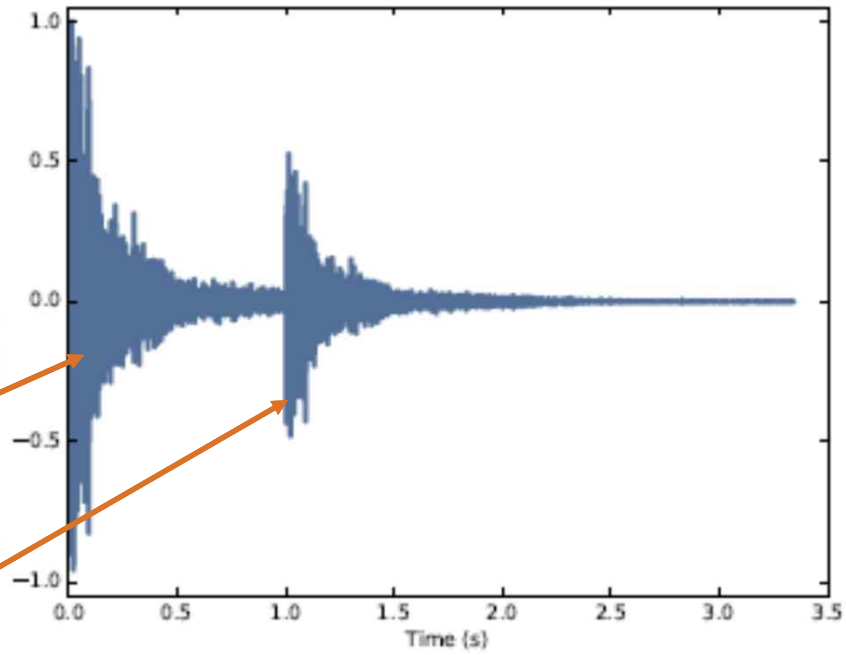
- You can think of the samples in the input wave as a sequence of impulses with varying amplitude.
- Each impulse in the input yields a copy of the impulse response, shifted in time (because the system is time-invariant) and scaled by the amplitude of the input.
- The output is the sum of the shifted, scaled copies of the impulse response. The copies add up because the system is linear.

Systems and convolution 2

- Suppose that instead of firing one gun, we fire two.
  - A big one with amp. 1 at $t = 0$, and a smaller one with amp. 0.5 at $t = 1$
  - The response of the system can be computed as follows.

```
def shifted_scaled(wave, shift, factor):
    res = wave.copy()
    res.shift(shift)
    res.scale(factor)
    return res
```

'shift' is a time shift in seconds. 'factor' is a multiplicative factor




```
shift = 1
factor = 0.5
gun2 = response + shifted_scaled(response, shift, factor)
```

Systems and convolution 2

100 guns fired at a rate of 441 shots per second

```
dt = 1 / 441
total = 0
for k in range(100):
    total += shifted_scaled(response, k*dt, 1.0)
```

It sounds like a periodic signal at 441Hz. (Car horn in a garage)
→ any wave is considered as a series of samples, where each sample is an impulse with a different amplitude.

```
signal = thinkdsp.SawtoothSignal(freq=441)
wave = signal.make_wave(duration=0.1,
                        framerate=response.framerate)
```

Adding up the impulse responses of impulses that make up the sawtooth.

```
total = 0
for t, y in zip(wave.ts, wave.ys):
    total += shifted_scaled(response, t, y)
```

Systems and convolution 3

100 guns fired at a rate of 441 shots per second

f : the input signal
g : impulse response
h : the sum of the shifted, scaled copies of g

$$
\begin{array}{lccccccc}
f[0] & [& g[0] & g[1] & g[2] & \dots & &] \\
f[1] & [& & g[0] & g[1] & g[2] & \dots &] \\
f[2] & [& & & g[0] & g[1] & g[2] & \dots \;] \\
\hline
 & [& & & h[2] & & &]
\end{array}
$$

$$h[2] = f[0]g[2] + f[1]g[1] + f[2]g[0]$$

1. The input is a sequence of impulses, so the output is the sum of scaled, shifted copies of the impulse response; that sum is the convolution of the input and the impulse response.

2. The DFT of the impulse response is a transfer function that encodes the effect of the system on each frequency component as a magnitude and phase offset. The DFT of the input encodes the magnitude and phase offset of the frequency components it contains. Multiplying the DFT of the input by the transfer function yields the DFT of the output.