ELEC 421
Digital Signal and Image Processing

Siamak Najarian, Ph.D., P.Eng.,

Professor of Biomedical Engineering (retired),

Electrical and Computer Engineering Department,

University of British Columbia

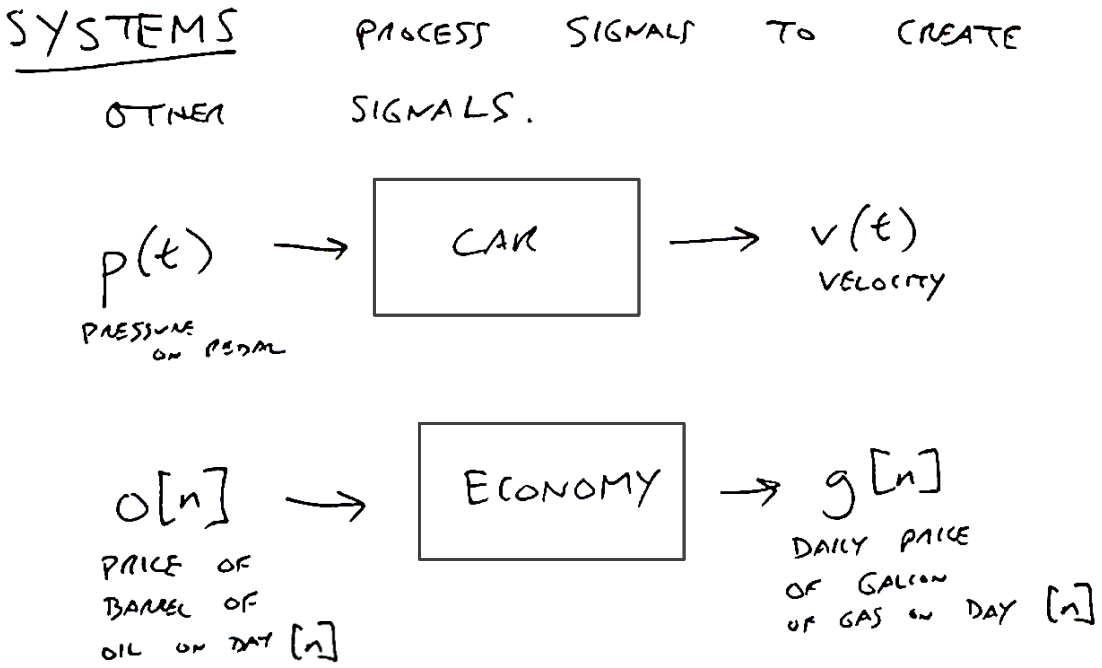
Course Roadmap for DSP

Lecture	Title
Lecture 0	Introduction to DSP and DIP
Lecture 1	Signals
Lecture 2	Linear Time-Invariant System
Lecture 3	Convolution and its Properties
Lecture 4	The Fourier Series
Lecture 5	The Fourier Transform
Lecture 6	Frequency Response
Lecture 7	Discrete-Time Fourier Transform
Lecture 8	Introduction to the z-Transform
Lecture 9	Inverse z-Transform; Poles and Zeros
Lecture 10	The Discrete Fourier Transform
Lecture 11	Radix-2 Fast Fourier Transforms
Lecture 12	The Cooley-Tukey and Good-Thomas FFTs
Lecture 13	The Sampling Theorem
Lecture 14	Continuous-Time Filtering with Digital Systems; Upsampling and Downsampling
Lecture 15	MATLAB Implementation of Filter Design

Lecture 2:
Linear Time-Invariant System

Table of Contents

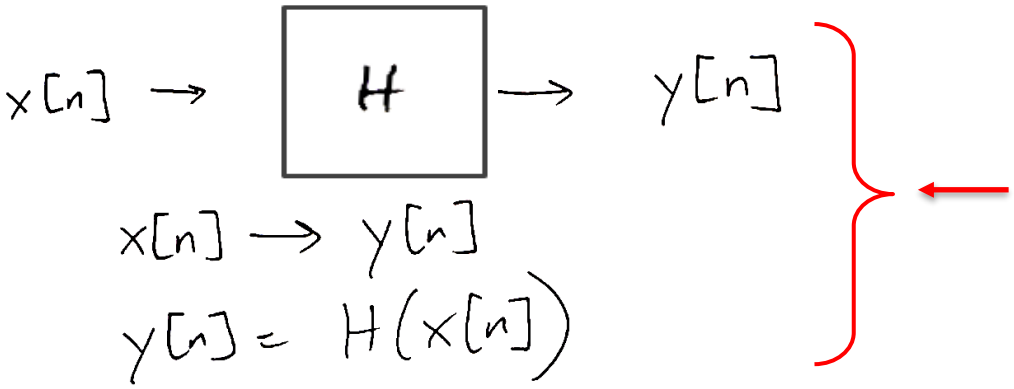
What are systems?



- **Car System:** We have got some sort of pressure, **p(t)**, on the gas pedal that goes into a complex control system, that is our car, and what comes out is the velocity of the car, **v(t)**.

- **Economy System:** The price of the gallon of gas, **g[n]**, lags behind the price of a barrel of oil, **o[n]**, by several days while it gets refined and then people add taxes.

Representing a system



- These three diagrams or representations are all different ways of saying the same thing, i.e., **representing a system**. When we are talking about a system, we are actually talking about **H**.

Example:

$$y[n] = \frac{1}{5}\sum_{k=-2}^{2} x[n-k]$$

$$= \frac{1}{5}\left(x[n+2] + x[n+1] + x[n] + x[n-1] + x[n-2]\right)$$

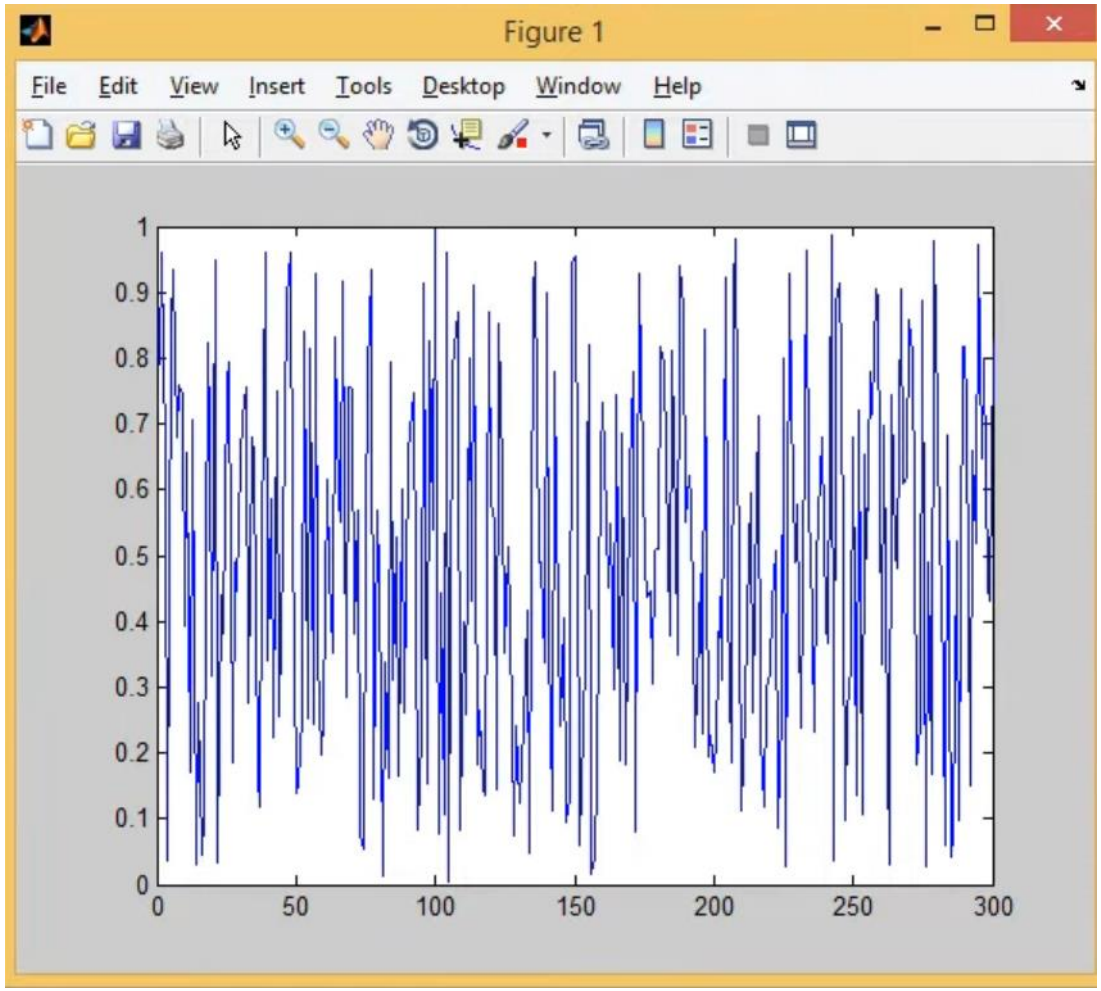
MOVING AVERAGE FILTER.

- **Example:** We may specify the **entries of y[n]** as functions of the **entries of x[n]**. We may say that **y[n]** is the sum from **k** equals **-2** to **2** multiply by a **1/5**. It is like saying we have got five entries of the signal **x** that are getting added up and averaged to produce the entry of **y[n]**. This is actually a special case of what is called a **moving average filter**.

Preview: a simple filter (with MATLAB demo)

```
>> x = rand(1, 300);
>> close all
>> plot(x)
```

MATLAB example: A *noisy* signal



Preview: a simple filter (with MATLAB demo)

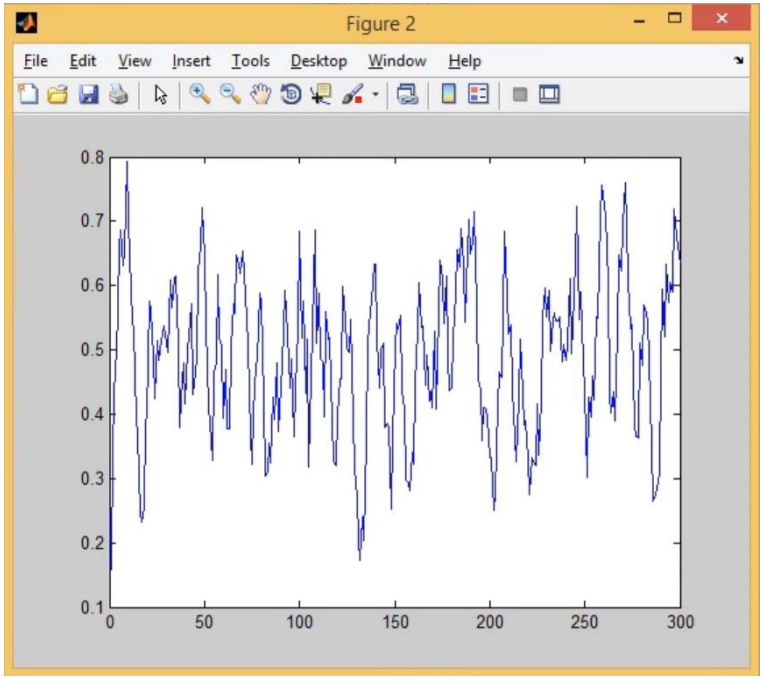
```
>> h = [1 1 1 1 1]/5

h =

    0.2000    0.2000    0.2000    0.2000    0.2000

>> y = filter(h,1,x);
>> figure
>> plot(y)
```

A *smoother* signal compared to the original.



- We can define a system, such as "**h**", which is going to act on entries of **x** and then we can make our **y** as filtering **x** by **h**.

y = filter(h, 1, x):

The line **y = filter(h, 1, x)** in MATLAB applies a filter described by the coefficients in the vector **h** to the signal **x**. Here is a breakdown of what each part does:

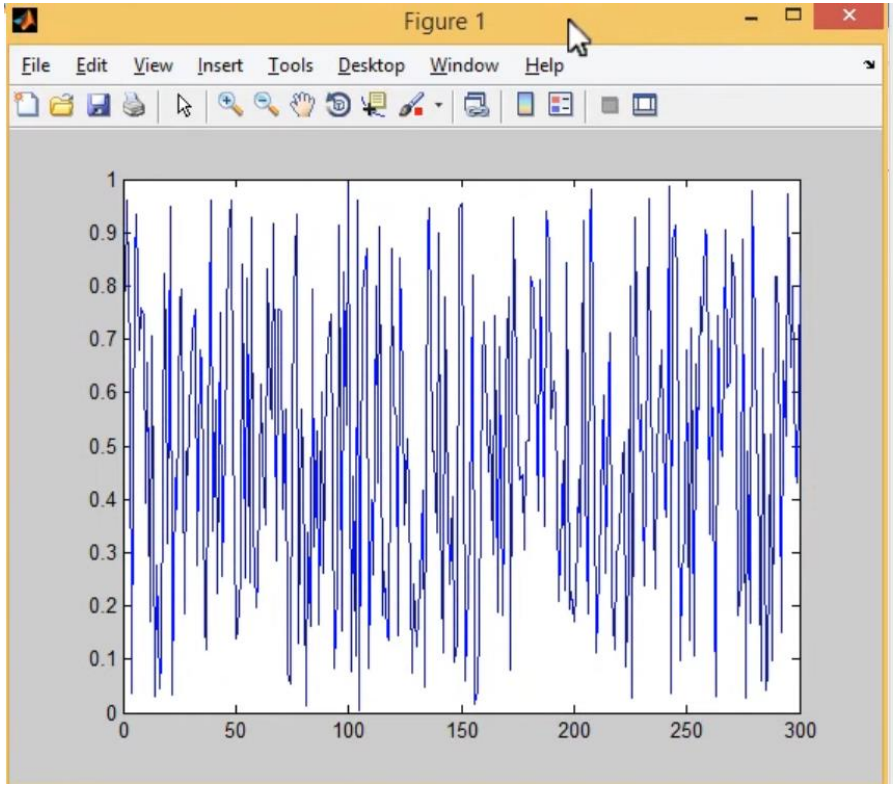
• **filter**: This is a built-in function in MATLAB used for digital filtering. It takes three arguments:

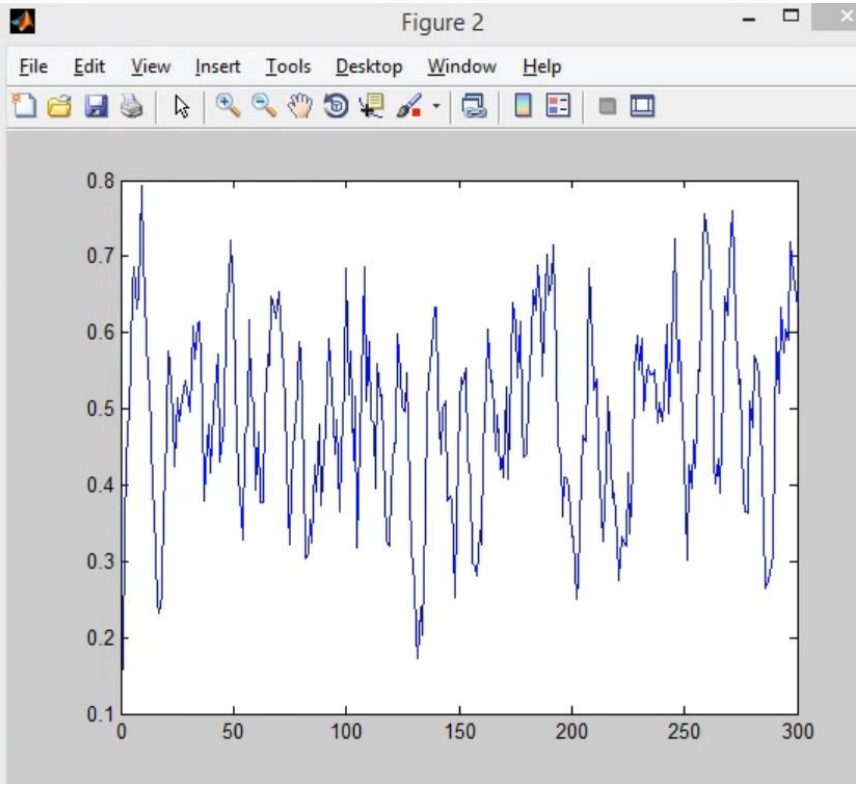
- **h**: This is a vector representing the filter coefficients (impulse response). It defines how the filter responds to an impulse signal.
- **1**: This is a dummy argument in this specific case. It is typically used to specify the denominator of the filter transfer function, but here it is set to **1**, which is equivalent to assuming a denominator of **1** (no filtering in the denominator).
- **x**: This is the input signal that we want to filter.

• **y**: This is the output of the function, which is the filtered version of the signal **x**.

Preview: a simple filter (with MATLAB demo)

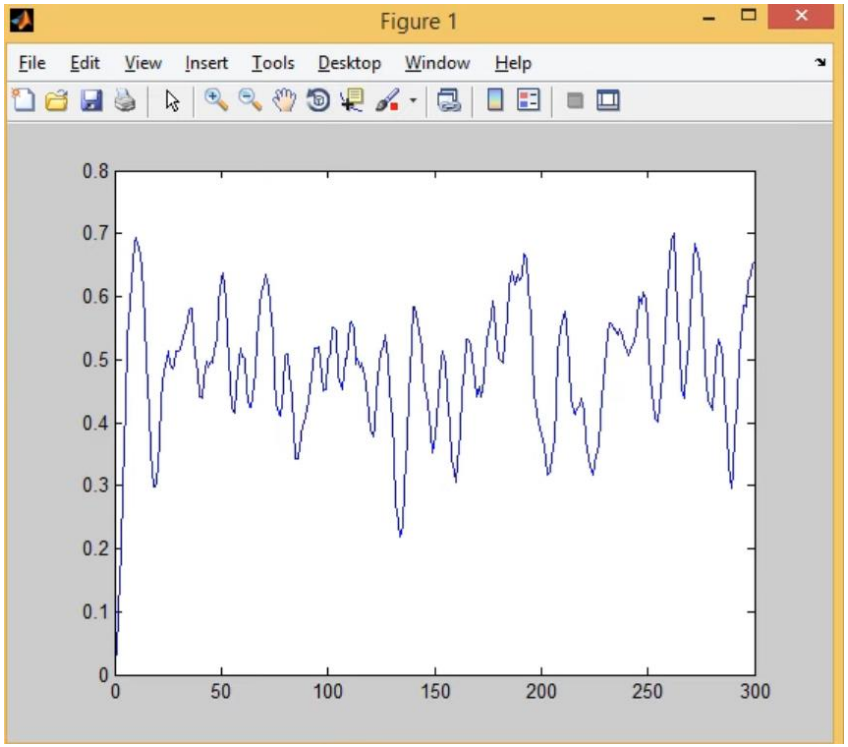
- In this process, we are taking away these spikings.





Preview: a simple filter (with MATLAB demo)

```
>> y = filter(h,1,x);
>> figure
>> plot(y)
>> y = filter(h,1,y);
>> plot(y)
```



- If we were to continue to filter that signal again (kind of ***double filter***) and then plot it, we can see that now things are getting even smoother. That is why it is called the **moving average filter**. It is like saying take the neighboring points around a specific point and average out. It will mitigate the spikiness and turn things to be smooth.
- If we kept on filtering it forever and ever, we eventually get down to a constant signal.

Relationships to differential and difference equations

$$y'(t) + a\,y(t) = b\,x(t)$$

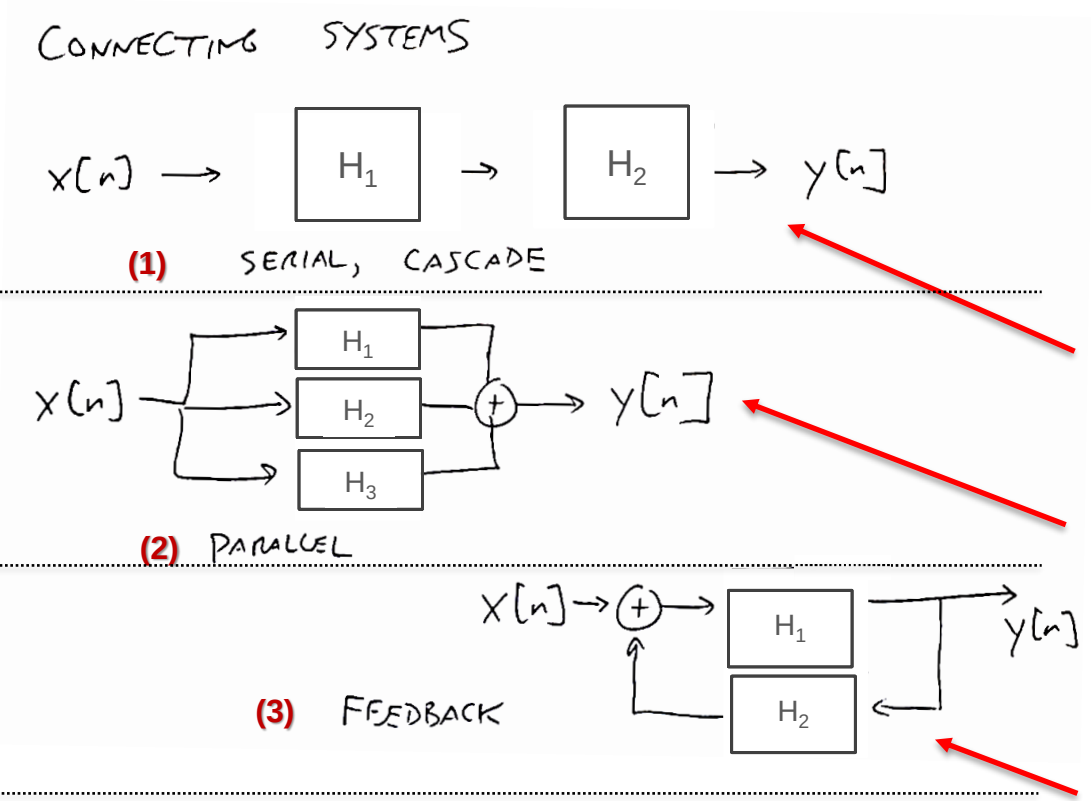
$$y[n] + a\,y[n-1] = b\,x[n]$$

DIFFERENCE EQUATIONS ARISE FROM ELECTRICAL, MECHANICAL PROBLEMS.

- In continuous-time, we see a lot of systems that are described by **differential equations**.
- These kinds of systems come up in things like mechanical properties of masses and springs and when we have forces and accelerations or in various electronic circuits.

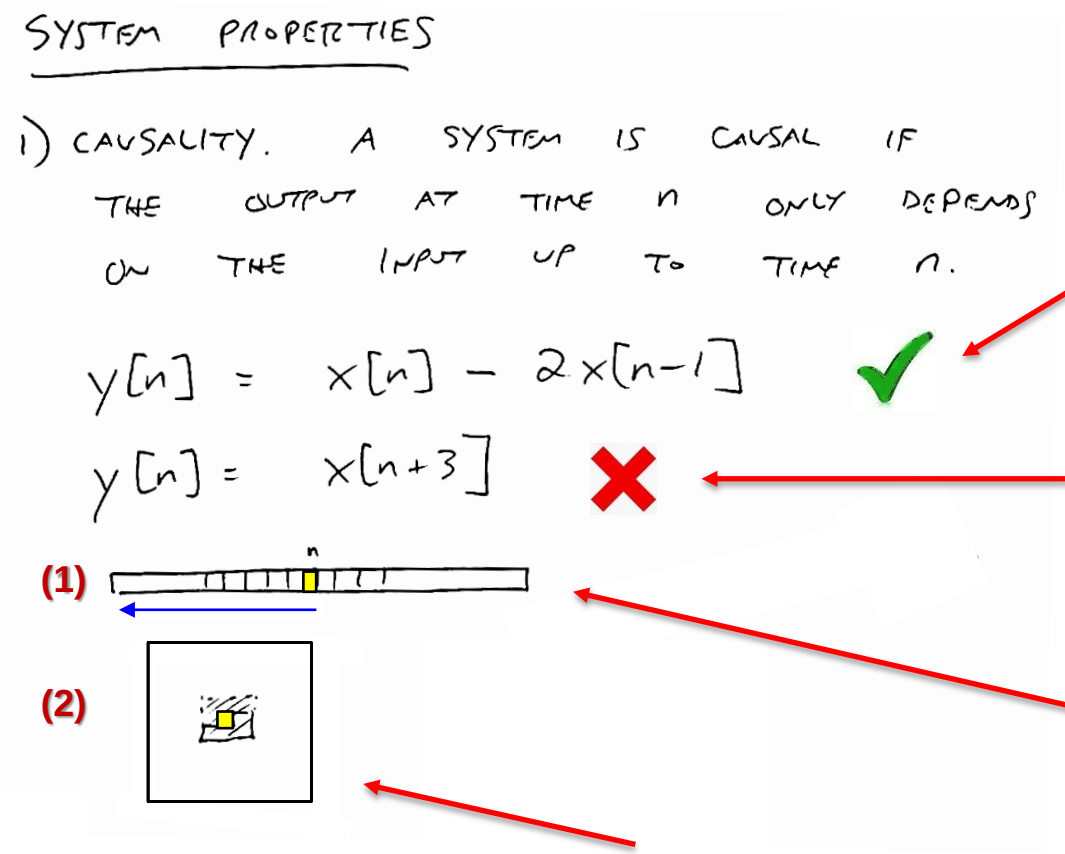
- If we were to take the continuous world and ***discretize*** it into digital, we would end up with what is called the **difference equation**.

Connecting systems together (serial, parallel, feedback)



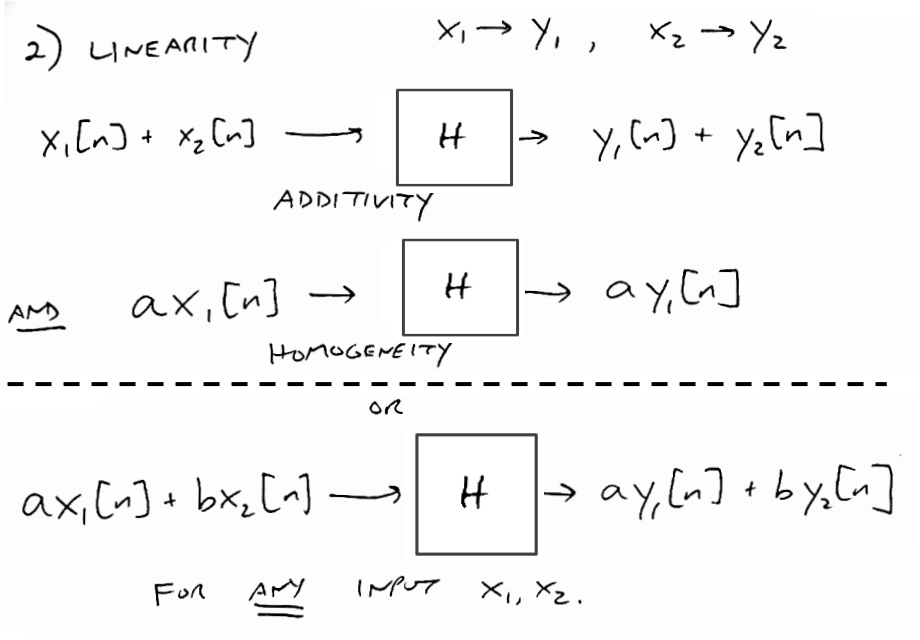
- A lot of times, we have to connect systems together (**connecting systems**). This kind of system has an analogy to circuit design.
- In circuit design, we use the idea that we may have a signal (**x[n]**) that goes through one system and then goes through another system and then we get our final output (**y[n]**), shown in **(1)**. One example is **serial** or **cascade system** that consists of some subsystems.
- Another example is the **parallel system**, shown in **(2)**. Here, we have a signal that goes through multiple systems and then we combine those multiple systems at the output to get our final signal.
- Still another example is the **feedback system**, shown in **(3)**. Here, the output is fed back (like the cruise control in a car).

System properties; Causality



- A system is **causal** if the output at time **n** only depends on the input up to time **n** (such as the example with green check mark). That is, a causal system uses the current and past inputs. A **strictly causal** system uses only past inputs, not the current input.
- In the second example, we are looking at the value of the input **3** units from now. So, it not causal (or **non-causal**). Here, some ***future time*** appear in the equation.
- Let us look at Example **(1)**, as an example in image processing. If we have a one-dimensional signal, with the units of signal shown by small boxes, and the time that we are at shown by **n** (the shaded box), then the system is causal. This is because we are only looking at point **n** and the previous times.
- In Example **(2)**, if we think about it as an image and if we want to blur the shaded pixel, usually what we do is to take the pixel that we map, and average it with its neighbors. If the system was strictly causal, we would only be allowed to think about the neighbors that were before we got to this pixel.

Linearity



- When we are testing for **Linearity**, we can combine **Additivity** with **Homogeneity** together, instead of testing these two things separately. That is, we can just test the combination of them (shown at the bottom).
- If we have something times the first signal plus something times the second signal and we put it through the system, what comes out has to be the corresponding weights on the outputs. This has to be true for any input signals. We cannot just test it for the delta function or step function. We have to prove that this is true for any possible input signaling system.

Formally proving that a system is linear

- How do we prove whether a system is linear or not?

Example: $y[n] = x[n] - 2x[n-1]$

$x_1[n] \longrightarrow x_1[n] - 2x_1[n-1] = y_1[n]$

$x_2[n] \longrightarrow x_2[n] - 2x_2[n-1] = y_2[n]$

WHAT IS THE RESPONSE TO $z[n] = x_1[n] + x_2[n]$?

$z[n] \rightarrow \boxed{H} \rightarrow z[n] - 2z[n-1]$

$= x_1[n] + x_2[n] - 2x_1[n-1] - 2x_2[n-1]$

$= (x_1[n] - 2x_1[n-1]) + (x_2[n] - 2x_2[n-1])$

$= y_1[n] + y_2[n]$ ADDITIVITY ✓

We put $z[n] = ax_1[n]$ into the system. Now z[n] will be transformed to $z[n] - 2z[n-1]$ →

$ax_1[n] - 2ax_1[n-1] = a(x_1[n] - 2x_1[n-1]) = ay_1$ HOMOGENEITY ✓

Formally proving that a system is linear

- We can do this process all at once, instead of doing it separately.

ALL AT ONCE:

$$z[n] = a x_1[n] + b x_2[n]$$

$$z[n] \rightarrow \boxed{H} \rightarrow z[n] - 2z[n-1]$$

$$= a x_1[n] + b x_2[n] - 2\left(a x_1[n-1] + b x_2[n-1]\right)$$

$$= a\left(x_1[n] - 2x_1[n-1]\right) + b\left(x_2[n] - 2x_2[n-1]\right)$$

$$= a y_1[n] + b y_2[n]$$ ✓ LINEARITY

- It is true that some systems can have the additive property but not the homogeneous property and vice-versa. So, it is not like one necessarily implies the other. We have to check both.

Disproving linearity with a counterexample

Example: Linear or not linear?

$y[n] = 3x[n] + 5$ **(1)**

$x_1[n] \rightarrow 3x_1[n] + 5$
$x_2[n] \rightarrow 3x_2[n] + 5$] * $\rightarrow$ $3x_2[n] + 5 + 3x_1[n] + 5$ **(2)**

Plug z[n] in **(1)**

$z[n] = x_1[n] + x_2[n]$

$z[n] \rightarrow 3z[n] + 5$

$= 3(x_1[n] + x_2[n]) + 5$ **(3)**

NOT LINEAR.

COUNTEREXAMPLE: $x_1[n] = 1$, $x_2[n] = 2$

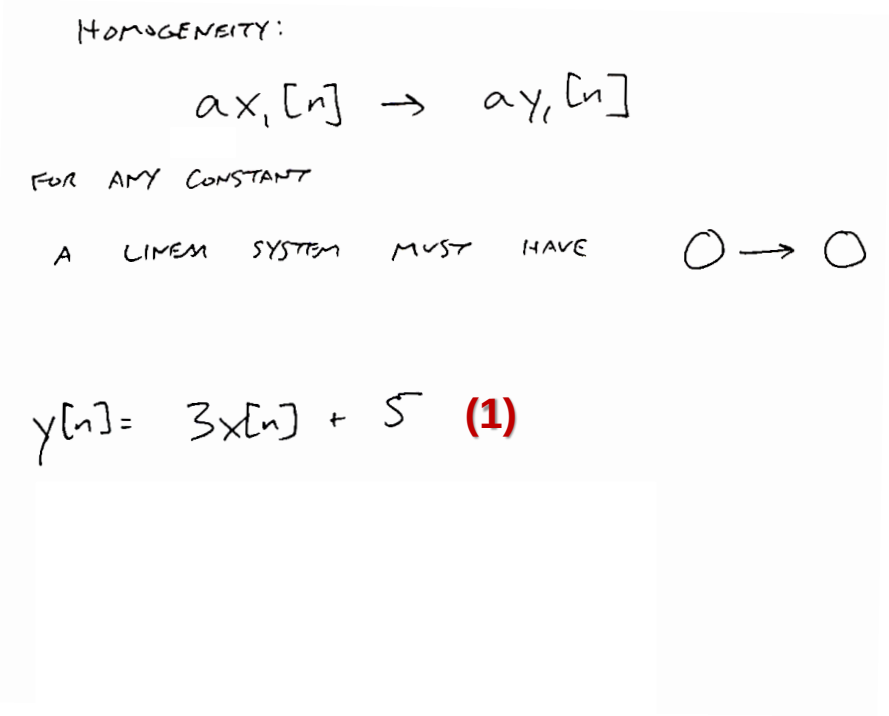
$y_1[n] = 8$, $y_2[n] = 11$ ✗

- When we add the two equations in *, we will get **$3(x_1[n] + x_2[n]) + 10$**. In **(3)**, we only have the **+5**. So, it not linear! That is, when we get to **(3)**, we ask ourself "Is this equation equal to equation **(2)**?". The "**+5**" in **(3)** says that the answer is no.

- If we are trying to show that something is not linear (or if we suspect that something is not linear), we can just prove it with a **counterexample**. To prove that something is not linear usually involves simple functions like constants or delta function.

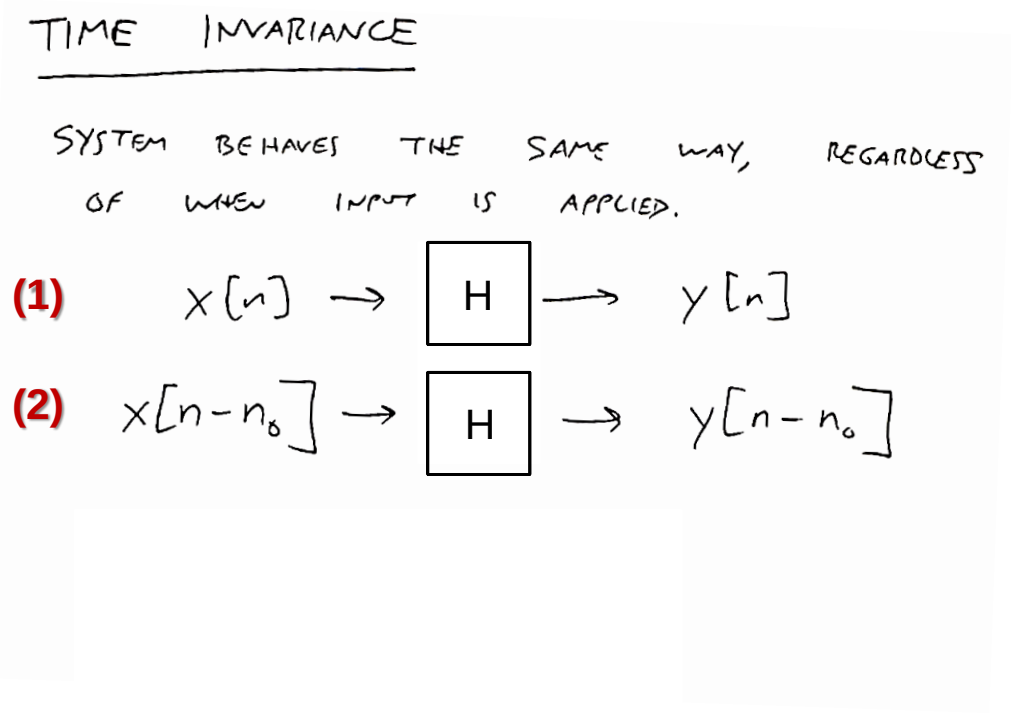
- In the above case, let us suppose that you know a counterexample. Suppose that our **x_1** of **n** is equal to **1** ***for all time*** and that **x_2** of **n** is equal to **2** ***for all time***. If we put in **1** into the system, we get **3** times **1** plus **5**, which is **8**. If we put in **2** into the system, we get **3** times **2** is **6** plus **5**, which is **11**. Since **2** is ***twice*** **1**, then for a linear system, **$y_2[n]$** should be ***twice*** **8** or **2×8 = 16** (here, we got **11**), which is not so. If the system was linear, we would expect **16** but we got **11**. So, we conclude that the system is not linear.

Disproving linearity with a counterexample



- Homogeneity is part of the linearity statement. So, if a system does not satisfy homogeneity condition, it is not linear.
- Homogeneity says that if we multiply $\mathbf{x_1[n]}$ by something Like "**a**", what we have to get out is ***the same multiple*** of the output. This has to be true for any constant. So, we could choose **0**. If we put zero in, zero has to come out. This property is often referred to as the **zero-input response**.
- As an example, if we put zero in equation **(1)** (i.e., plug in **0** for **x[n]**), we get **5** out, and NOT **0**.
- We can see that this system does not satisfy the homogeneity condition.

Time invariance



- **Time-Invariance:** Mathematically, it means that if we put a signal into the system, **(1)**, and then compare it to what we get if we delay the signal by some number of units, **(2)**, we should get out ***the same output***, delayed by some number of units.
- Lots of real world systems are not time-invariant. So, we usually have to make a model for the system as a function of time that is time-invariant and approximate the time-variant system.

Formally proving that a system is time-invariant

Example:

$y[n] = x[n] - 2x[n-1]$ (1)

TIME INVARIANT?

$z[n] = x[n-n_0]$

$z[n] \rightarrow \square \rightarrow z[n] - 2z[n-1]$ (2)

Output of dummy signal $= x[n-n_0] - 2x[n-1-n_0]$ (3)

* ?? $= y[n-n_0]$

$= x[n-n_0] - 2x[n-n_0-1]$

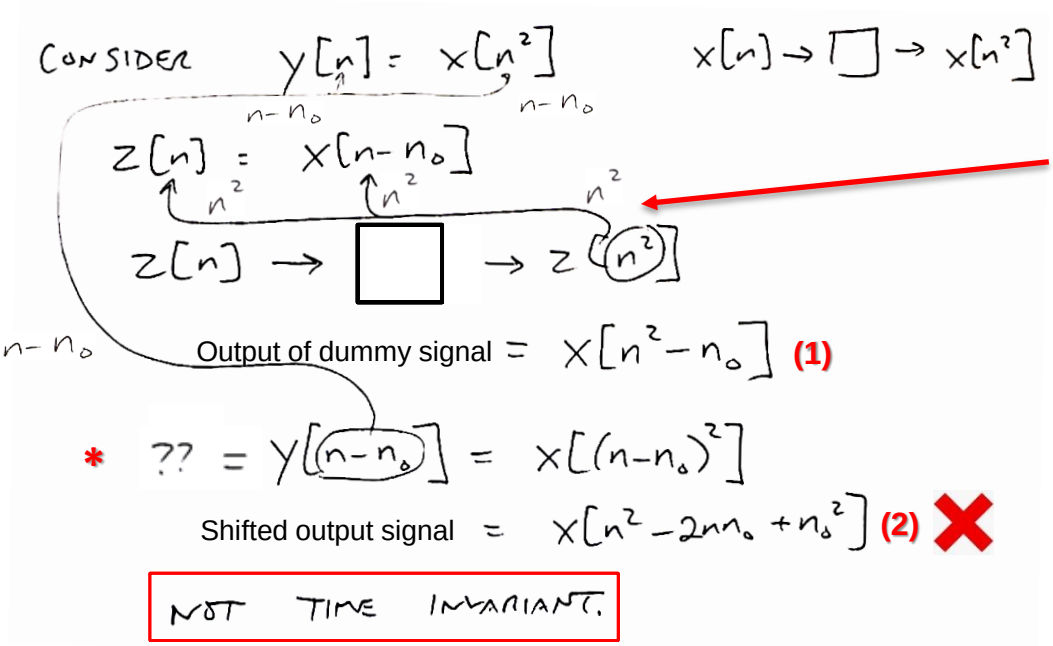
Shifted output signal $= x[n-n_0] - 2x[n-1-n_0]$ (4)

TIME INVARIANT

- **Example:** We have to prove or disprove this time-invariancy for a given system. The question here is "Is this system time-invariant?".
- We make a **dummy signal**. Let us call this signal **z[n]**. The dummy signal is going to be a shifted version of the input.
- To find **(2)**, in **(1)**, we replace **x** with **z**. To find **(3)**, we replace **z** in **(2)** with **x**, and then we shift ***the whole argument*** in the square brackets by $-n_0$.
- At step *****, we ask ourself "Is equation **(3)** equal to the output shifted by n_0 (i.e., $y[n-n_0]$) and what would that be?".
- By comparing **(3)** to **(4)**, we can say "yes", we got the correctly-shifted output that we were expecting. Here, we are comparing the output of the dummy signal, **(3)**, with the shifted output signal, **(4)**.

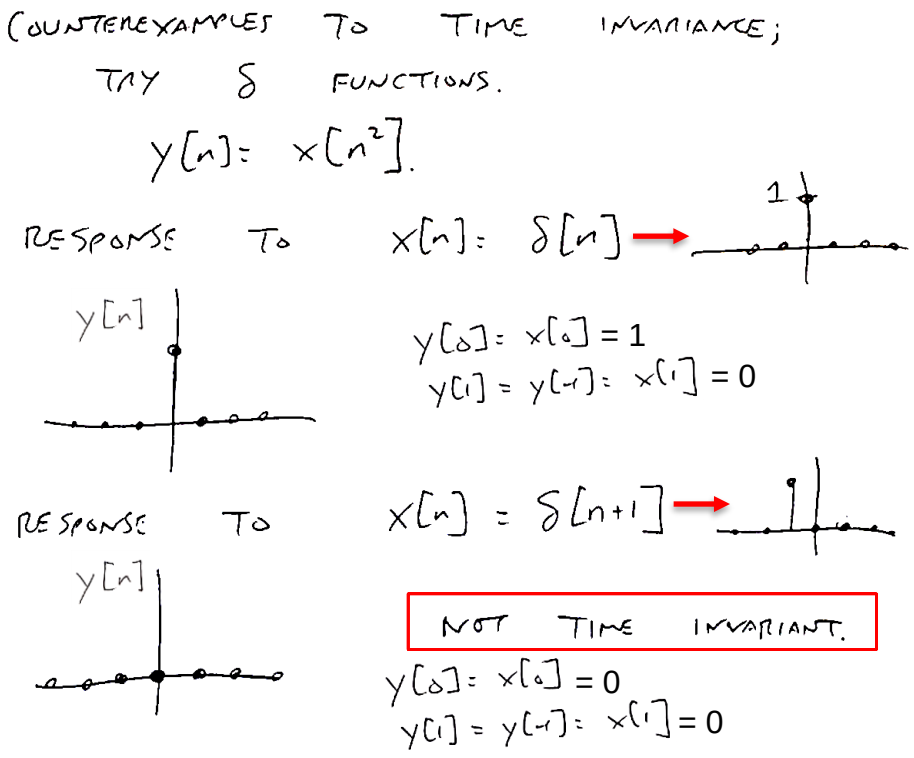
Formally proving that a system is time-invariant

Example:



- We have to plug in $\mathbf{n^2}$ whenever we see an "**n**" in **z[n]** and $\mathbf{x[n-n_0]}$.
- At step *, we ask ourself "Is equation **(1)** equal to the output shifted by $\mathbf{n_0}$ (i.e., $\mathbf{y[n-n_0]}$) and what would that be?".
- By comparing **(1)** to **(2)**, we can say "no", we do not have the correctly-shifted output that we were expecting.
- So, this ***is not*** a time-invariant system. That is, it is a time-varying system.

Disproving time-invariance with a counterexample



- **Counterexamples:** Again, in the same way that we did to prove that something is not linear is to find a counterexample. We can use the same approach for time-invariance. We may need to use a different signal to be able to get at the counterexample. Delta functions are generally a good way to start.
- Suppose that we have a delta function that is shifted by one unit to the left, $\boldsymbol{\delta[n+1]}$.
- For the response to $\boldsymbol{\delta[n+1]}$, we get **0** across the board for **y[n]**. Here, we shifted the input left by one unit, but the output did not shift left by one unit as we expected it would. So, **$y[n] = x[n^2]$** is not time-invariant.

Linear, time-invariant (LTI) systems; Superposition for LTI systems

LINEAR, TIME-INVARIANT SYSTEMS (LTI)

REAL-WORLD SYSTEMS ARE OFTEN MODELED AS LTI SINCE

a) OFTEN A GOOD APPROXIMATION
b) ANALYSIS IS VERY EASY / POWERFUL

} Main reasons

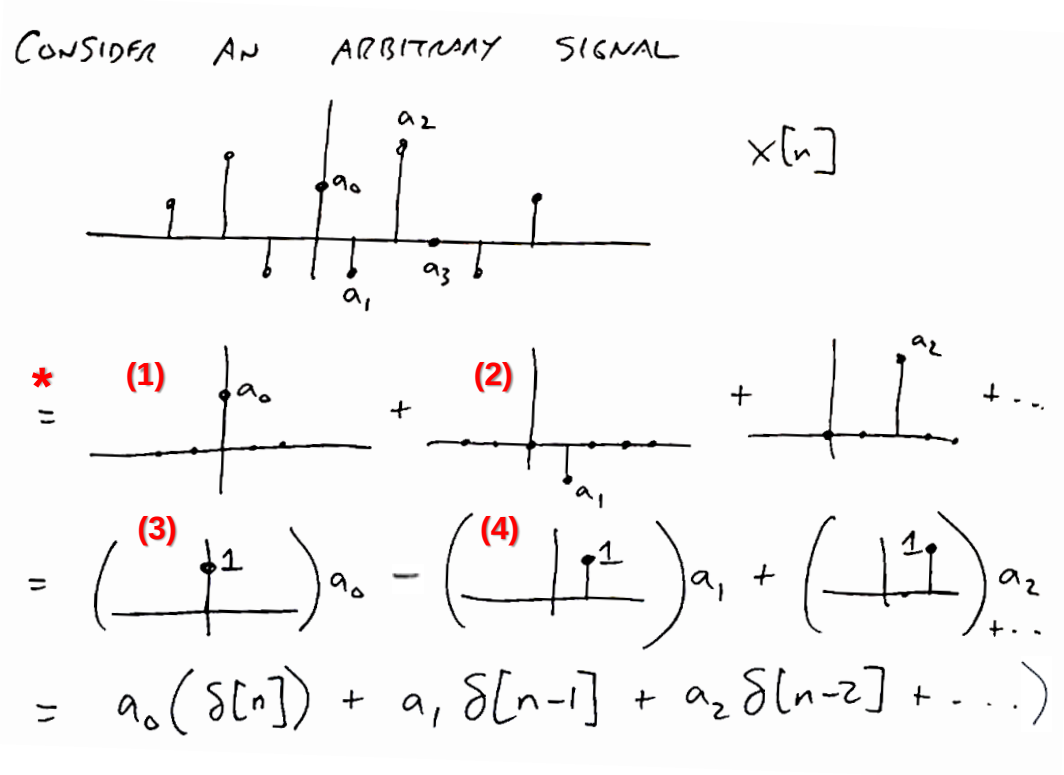
KEY CONCEPT: SUPERPOSITION FOR LTI SYSTEMS.

$$a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + \ldots \rightarrow \boxed{\text{LTI}}$$

$$\Rightarrow a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + \ldots$$

- Let us put everything we discussed together.
- **Superposition Principle:** In Linear Time-Invariant (LTI) systems, superposition is a fundamental principle that governs how the system responds to multiple inputs.
- It essentially states that the output due to a combination of multiple inputs is simply the sum of the individual outputs that would be produced by each input alone.

The response of a system to a sum of scaled, shifted delta functions



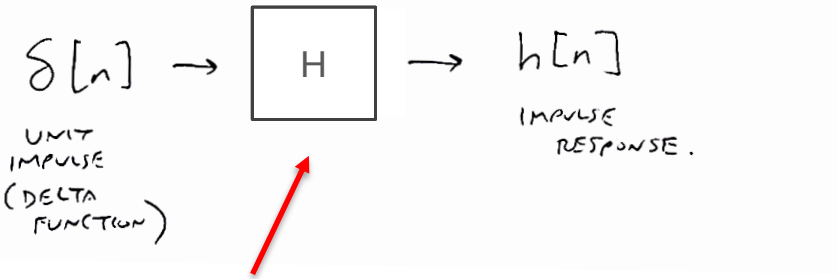
- We want to know what is the output of this signal after we put it through an LTI system.
- For step *, and for graph **(1)**, we can decompose this simple signal as graph **(3)** multiplied by $\mathbf{a_0}$. The same thing is done for **(2)**. Then we add graph **(3)** times $\mathbf{a_0}$ to graph **(4)** times $\mathbf{a_1}$, and so on. Note that graph **(4)** is the shifted version of graph **(3)**.

The impulse response

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\,\delta[n-k] \quad (1)$$

"SIFTING" PROPERTY OF δ FUNCTIONS

SUPPOSE WE HAVE AN LTI SYSTEM.



- In the above diagram, we always use lowercase "**h**" of **n** (**h[n]**) as the response to the delta function.
- In general, we can figure out what the response of the system is to any input **x[n]** by only knowing what the system did to the impulse.

- In a more mathematical way, **x[n]** is the sum of the values of **x[k]** times the corresponding shifted versions of delta functions, as shown by **(1)**.
- This delta function, $\boldsymbol{\delta[n-k]}$, only fires at **n** equals **k**. And, when it fires, its value is at whatever **x[k]** is. This means that each term **x[k]**. $\boldsymbol{\delta[n-k]}$ represents the value of the signal, **x[k]**, at **n = k**.
- The delta function $\boldsymbol{\delta[n-k]}$ "fires" or becomes **1** only at **n = k**, effectively picking out the value **x[k]** at that point.
- This property is sometimes called the **sifting property of delta functions.** Here, we have infinitely many functions added together, but only one of these functions is not zero. That is like saying the only place where the delta function is not zero is when **k = n**.

The impulse response completely characterizes an LTI system

WHAT IS RESPONSE TO $x[n]$?

$$H(x[n]) = H\left(\sum_{k=-\infty}^{\infty} x[k]\,\delta[n-k]\right)$$

BY LINEARITY, $$= \sum_{k=-\infty}^{\infty} x[k]\,H(\delta[n-k]) \quad \textbf{(1)}$$

BY TIME-INVARIANCE, $$= \sum_{k=-\infty}^{\infty} x[k]\,h[n-k]$$

- Here, **x[k]**'s are just numbers that are multiplying the heights of the delta functions. So, they are going to play the role of constants.
- Applying the **additivity** property of **linearity**, we can distribute the system **H** over the summation.
- By **homogeneity** property of **linearity**, we can immediately take the constants (i.e., **x[k]**'s) out of the sum. That is, by applying the homogeneity property, we can move the constant **x[k]** outside the system **H**.
- By **time-invariance**, we can say that the response of the system to the shifted delta function, i.e., to the **H(δ[n-k])**, is going to be the shifted version of the impulse response (i.e., **h[n-k]**). That is, since **h[n] = H(δ[n])**, we can substitute **h[n-k]** for **H(δ[n-k])** in **(1)** due to the time-invariance property of the system.

The impulse response completely characterizes an LTI system

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\,h[n-k] \qquad (1)$$

$$y[n] = x[n] * h[n] \qquad (2)$$

OUTPUT INPUT IMPULSE RESPONSE

IMPULSE RESPONSE FULLY CHARACTERIZES THE SYSTEM.

- This process tells us that if we only knew the impulse response to the system and we have some new signal, equations **(1)** or **(2)** show us how we can combine those two things to get to the output of the system.
- Equation **(2)** is called the **convolution**. So, the output is the convolution of the input and the impulse response. It means that an LTI system can be ***entirely characterized*** by what it does to the delta function.

End of Lecture 2