Imperial College London

Lecture 11

Discrete Time Systems

Prof Peter YK Cheung

Dyson School of Design Engineering

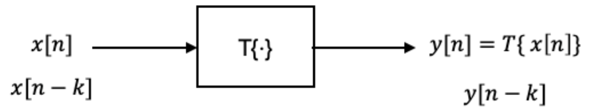
URL: www.ee.ic.ac.uk/pcheung/teaching/DE2_EE/
E-mail: p.cheung@imperial.ac.uk

In this lecture, we will explore discrete systems and how to model them in three different forms:

1. As block diagrams – this is similar to a circuit schematic. It shows how signals flows in the system and the operations being performed on the signals.
2. As **difference equation** – this relates input sample sequence to output sample sequence.
3. As **transfer function** in z-domain – this is similar to the transfer function for Laplace transform. However I will be introduce the z-transform, which is essential to represent discrete systems.

Linear Discrete time systems

- A discrete time system takes in a sequence of discrete values x[n[at the input and produces an output sequence y[n] through some internal operation or transformation T{.}



- The system is LINEAR if it obeys the principle of superposition:

$$y[n] = T\{a_1 x_1[n] + a_2 x_2[n]\} = a_1 T\{x_1[n]\} + a_2 T\{x_2[n]\}$$

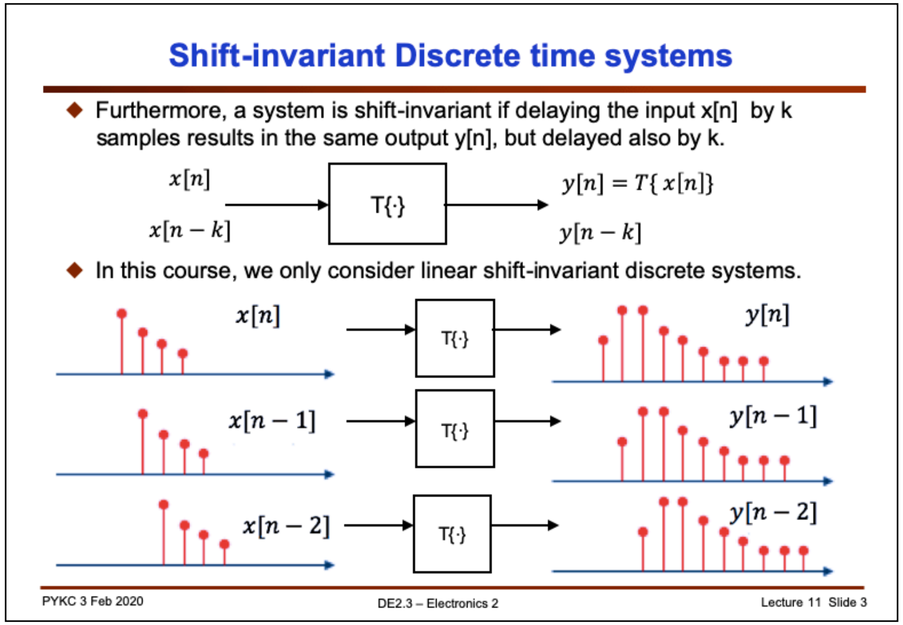
- The system is shift-invariant if:

$$y[n] = T\{a_1 x[n-k] + a_2 x_2[n-k]\} \\ = y[n-k]$$

Let us first consider a linear discrete time system shown here as T{.}. The system takes in the input sequence x[n] and produce the output sequence y[n].
A **linear system** means it obeys the principle of superposition. Remember from last year, you have learned this principle in depth. There are two separate property of a linear system:

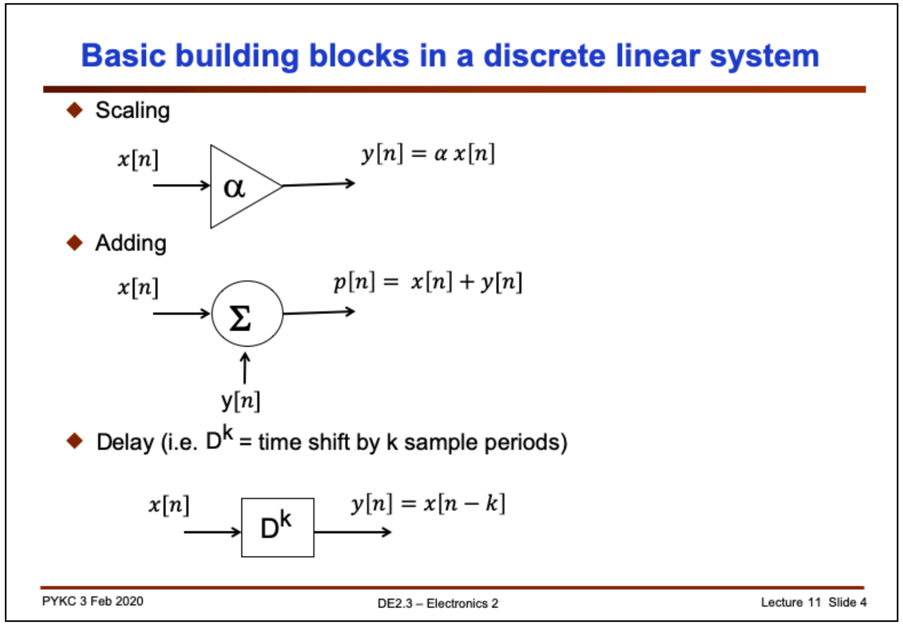
1. Response for input (A+B) = Response for input A + Response for input B.
2. If you scale input by a factor K, the output is scaled by the same factor K. For example, if you double the input (K=2), the output will also double.

The system is also **shfit invariant**. This means if input is delayed by k samples, i.e. x[n] → x[n-k] , then the output is the same but also delayed by k samples, i.e. yn] → y[n-k]. (See next slide.)



A shift-invariant system means that if you shift an input signal by k samples, the system response to x[n-k] is simply y[n-k]. It essentially means that the signal can start any time, the output remains the same but with the same delay as the input.

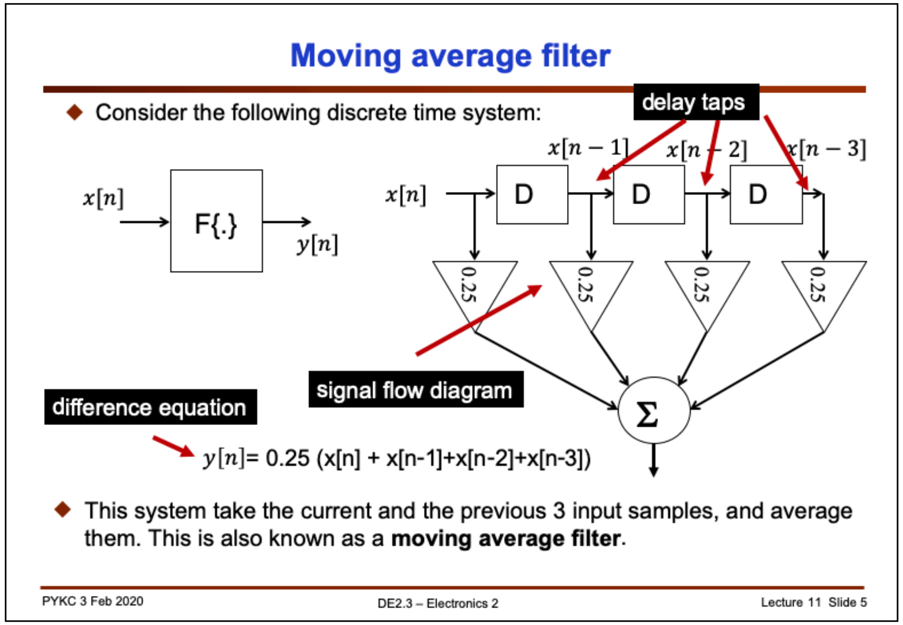
For this course, we will always assume that the discrete time system is linear, shift-invariant.



When building a linear discrete time system, we use only THREE operators:

1. **Scaling** signal by a constant. This is the same as amplifying or attenuating the signal.
2. **Adding** – this is obvious.
3. **Delay** – this is simply delaying the input sample by k sampling clock cycles (i.e. k clock period, equivalent to t = k Ts, where Ts is the sampling interval (i.e. time between two samples).

It is amazing that with only these three operators, we can construct fairly complex discrete time systems implemented on a microcontroller such as the PyBench board as used in the Lab.



Consider a simple discrete time system which takes the current and three previous three input samples, and calculate the average value:

$$y[n] = \frac{1}{4}\sum_{k=0}^{3} x[n-k]$$

This is known as a moving average filter. Since averaging reduces the effect of fast changes, this is essentially a lowpass filter.

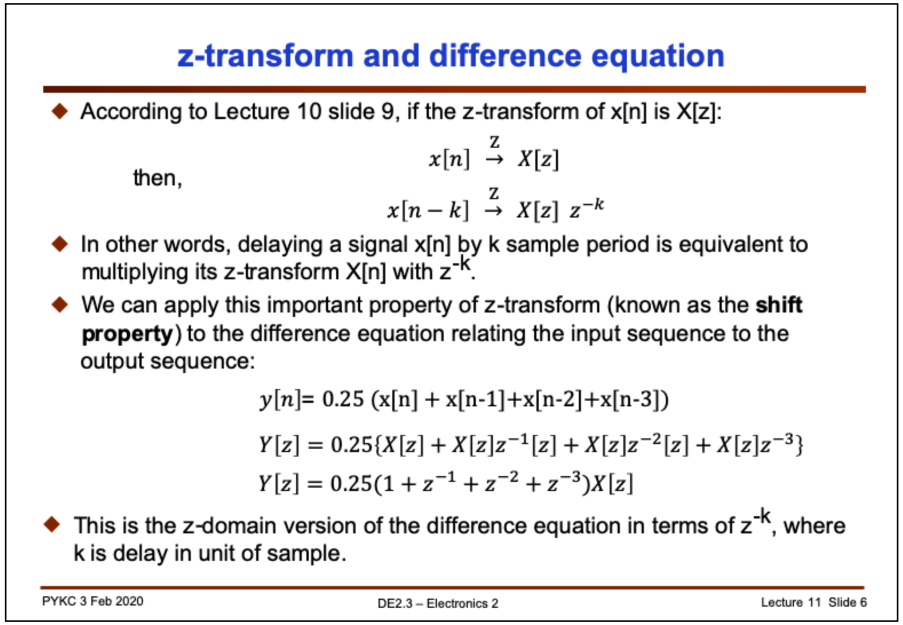
The slide here shows two different representation of the system.

1. Signal flow diagram – this is similar to a circuit schematic showing how signals are transferred to various modules. Its graphical nature allows a reader to relate to the actual physical system easier than using mathematical equations.
2. Difference equation – this is expressed in terms of a formulae defining the relationship between input samples and output samples.

Note that the input signal x[n] is passed to three unit sample delay, to obtain x[n-1], x[n-2], x[n-3]. These are called "taps" of the filter. In this case, there are four taps to this filter.

Such a sample delay can be realised in an electronic system by:

1. Using an array on a microprocessor or computer to store different x values;
2. Using D-flipflops (D-FFs) in digital hardware in a shift register configuration. In this case, assuming that all signals are 10-bit wide (from the ADC), then we need three sets of 10 D-FFs.



The third method of representing a discrete time system is to use z-transform to represent the delay operators. Again, I will discuss the mathematical basis of z-transform in a later lecture. For now all you need to know (and trust) is that delay a signal x[n] by k sampling periods to give x[n-k] is the same as multiplying the z-transform of the signal X[z] by the factor z^{-k}.

$$x[n] \overset{Z}{\rightarrow} X[z]$$

$$x[n-k] \overset{Z}{\rightarrow} X[z]\ z^{-k}$$

Using this fact, we can derive the relationship between Y[z] and X[z] directly from the difference equation:

$$y[n] = 0.25\ (x[n] + x[n\text{-}1] + x[n\text{-}2] + x[n\text{-}3])$$

$$Y[z] = 0.25\{X[z] + X[z]z^{-1}[z] + X[z]z^{-2}[z] + X[z]z^{-3}\}$$

$$Y[z] = 0.25(1 + z^{-1} + z^{-2} + z^{-3})X[z]$$

Transfer function in the z-domain

- Take the results from the previous slide and re-arrange:

$$Y[z] = 0.25\{X[z] + X[z]z^{-1}[z] + X[z]z^{-2}[z] + X[z]z^{-3}\}$$

$$Y[z] = 0.25(1 + z^{-1} + z^{-2} + z^{-3})X[z]$$

$$H[z] = Y[z]/X[z] = 0.25(1 + z^{-1} + z^{-2} + z^{-3})$$

- As in the case of Laplace transform, in the z-domain,
  transfer function = output / input
- This moving average filter takes the average of the current data sample x[i], and the previous three samples x[i-1], x[i-2] and x[i-3], to produce the output y[i].
- The averaging function has a **smoothing effect** – that is, it performs the function of a **lowpass filter**.

From the previous slide, we can compute Y[z]/X[z], and obtain the transfer function H[z]:

$$H[z] = Y[z]/X[z] = 0.25(1 + z^{-1} + z^{-2} + z^{-3})$$

Moving average filter is a lowpass filter. It has a DC gain of 1 (check this for yourself and make sure you understand why), and it reduces the amplitude of fast changing signals – i.e. at higher frequencies.

Take a sequence x[n] = {1.0, 1.0, 1.0, 1.0, 1.1, 0.8, 1.2, 0.9, 1.0, 1.2, 0.9, …}, which is a discrete signal at 1v, but with random noise added. The noise amplitude (i.e. deviation from 1.0) is $\pm$0.2v.

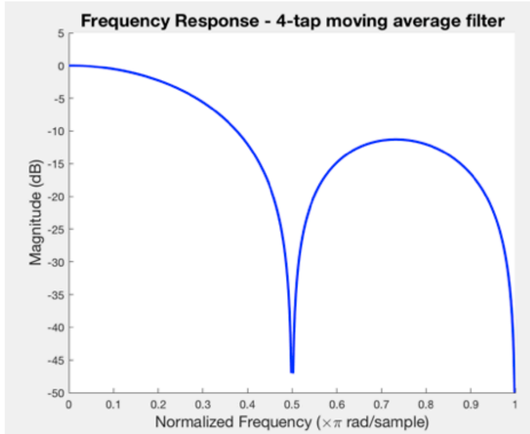
Assuming that all x[n] = 0 for n<0,

y[n] = (0.25, 0.5, 0.75, 1.0, 1.025, 0.975, 1.025, 1.0, 1.0, 1.075, 1.0, …..}

It is clear that y[n] is still centred around 1v, but the fluctuation is reduce to 0.025v!

Frequency Response of this filter

- Here is the frequency response of this moving average filter:



We can computer the frequency response of this simple 4-tap FIR or moving average filter with the following transfer function:

$$H[z] = Y[z]/X[z] = 0.25(1 + z^{-1} + z^{-2} + z^{-3})$$

I deliberately skip the mathematic derivation of the frequency response for such a filter for now. This will be covered in a later lecture. What is important to note is that the frequency response demonstrates that averaging four samples is effectively lowpass filtering the signal. The roll off (rate of attenuation as frequency increases) is very gentle.

The x-axis is the digital frequency expressed in angle increment per sample, normalized to Fs/2 (Nyquist frequency). For example, if the sampling frequency is 8kHz, the normalized frequency of 1 is at 4kHz.

There is a notch (null response) at 2kHz, when the response of the filter is zero! Same for frequency of 4kHz. Why? Test it for yourself.

General FIR filters

- Instead of using the same coefficient values in the moving average filter, one could use different coefficients at different delay taps.
- The number of delay taps can be increased to N.
- This will implement a filter function of the form as difference equation:

$$y[n] = b_0x[n] + b_1x[n-1] + b_1x[n-2] + \cdots + b_{N-1}x[n-(N-1)]$$

- In z-domain form:

$$Y[z] = (b_0 + b_1z^{-1} + b_2z^{-2} + b_3z^{-3} + \cdots + b_{N-1}z^{-(N-1)})X[z] = X[z]\sum_{k=0}^{N-1} b_kz^{-k}$$

$$H[z] = \frac{Y[z]}{X[z]} = b_0 + b_1z^{-1} + b_2z^{-2} + b_3z^{-3} + \cdots + b_{N-1}z^{-(N-1)} = \sum_{k=0}^{N-1} b_kz^{-k}$$

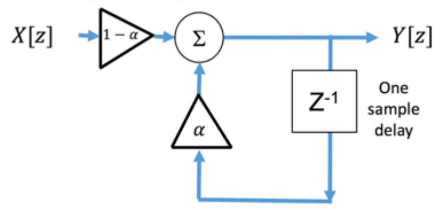
- By choosing different coefficients b_0, b_1, b_2 …., b_{N-1}, one can implement different types of filters: **lowpass**, **bandpass**, **highpass** etc.
- Such a filter will have N terms in the impulse response, where N is the number of signal taps x[n], … x[n-(N-1)]. Therefore it is also known as a **finite impulse response** filter (FIR) of order N.

The moving average filter we used previously only consider 4 input sample values (we call this a "4-tap" filter, taping into only 4 signal values). Instead of 4-tap, one can choose to perform the moving average over K-taps.
What do you expect the effect of using a higher value of K? You will be averaging over a wider time (sample) window, and therefore this will have a "stronger" lowpass filtering effect. In other words, the attenuation at high frequency will be stronger. If you measure the frequency response of the moving filter, you will find that the drop in gain as frequency increases goes up with K. Exactly how we can derive the frequency response of a K-tap moving average filter will be considered at a later lecture.
Instead of using equal coefficients on the taps in this filter, we could choose to use different coefficients. In which case, the filter you implement will have the difference equation and the transfer function as shown in the slide.
This generalised form of filter is known as FIR or finite impulse response filter. The name is due to the fact that if you apply an impulse at the input x[n] = δ[n] to a filter with N taps, the output response y[n] will have exactly N samples that is non-zero. This output y[n] for x[n] = unit impulse is known as impulse response of the system. It can be shown that for an arbitrary signal x[n], its response y[n] will always be finite – meaning that if x[n] dies down to zero, y[n] will become zero in finite time. By choosing different values for the coefficients b_i, one can implement any type of filters: lowpass, highpass, bandpass, bandstop etc.

Recursive Filter

- FIR filters derives the current output from current and previous inputs
- Such a filter does not make use of previous outputs – that is, it does not rely on past information
- Recursive filter is different – it derives the current output from both input and previous output samples.
- Here is one of the simplest recursive filter:

$$y[n] = \alpha y[n-1] + (1-\alpha)x[n]$$



$$Y[z] = \alpha\, Y[z]\, z^{-1} + (1-\alpha)X[z]$$

$$\Rightarrow Y[z] - \alpha\, Y[z]\, z^{-1} = (1-\alpha)X[z]$$

$$\Rightarrow (1-\alpha\, z^{-1})Y[z] = (1-\alpha)X[z]$$

$$\Rightarrow H[z] = \frac{Y[z]}{X[z]} = \frac{(1-\alpha)}{(1-\alpha\, z^{-1})}$$

Let us now consider a completely new class of filter. Here we have a filter that derives the output from both inputs and past output samples. The simplest form is the one shown here – known as a first-order recursive filter.

The output contains two components: 1) input x[n] multiplied by a coefficient (1-α); 2) output y[n-1] multiplied by a coefficient α.

This can be represented by the difference equation:

$$y[n] = \alpha y[n-1] + (1-\alpha)x[n]$$

Now takes the z-transform of this difference equation. We get:

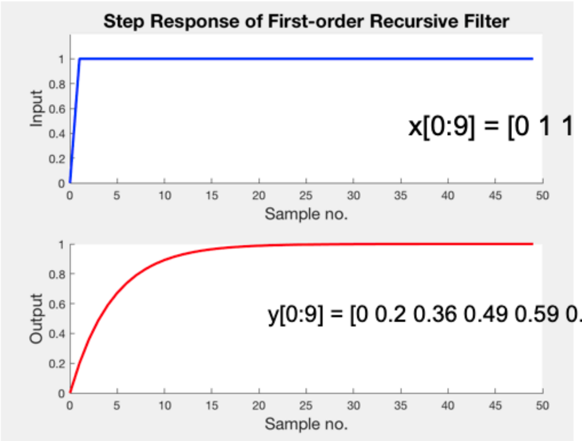
$$Y[z] = \alpha\, Y[z]\, z^{-1} + (1-\alpha)X[z]$$

From this we can derive the transfer function:

$$H[z] = \frac{Y[z]}{X[z]} = \frac{(1-\alpha)}{(1-\alpha\, z^{-1})}$$

Step response of Recursive Filter

- Let us consider the response of this filter to a step input:



x[0:9] = [0 1 1 1 1 1 1 1 1 1]

y[0:9] = [0 0.2 0.36 0.49 0.59 0.67 0.74 0.79 0.83 0.87]

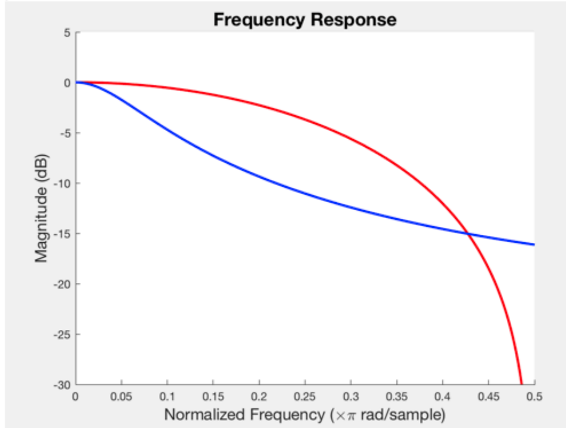
Now let us consider what output we get if we apply a unit step function at the input. The output is known as the step response of this filter.

Shown above are the outputs y in respond to the input going from 0 to 1.

The result is an exponential rise toward 1, the time constant of the rise if effective α. (Similar to what we have done before on 1st order lowpass filter and RC network.)

Frequency Response of Recursive Filter

- If we computer the magnitude response of this filter, we will get the following characteristics:



The frequency response of this filter is shown here. Again it is a simple lowpass filter.

Let us now compare this response to that of the previous 4-tap moving average filter shown in red. It is clear that the recursive filter is much more effect in removing high frequency component than the moving average filter.