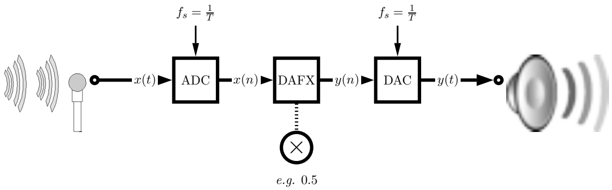
CM2208: Scientific Computing
2. Digital Signal Processing
2.2. Digital Systems

Prof. David Marshall

School of Computer Science & Informatics

Digital Systems: Representation and Definitions

Recall this Figure:



A **digital system** is represented by an algorithm which uses the input signal $x(n)$ as a sequence/stream of numbers and performs operations upon the input signal to produce and output sequence/stream of numbers — the output signal $y(n)$.

- **i.e.** the **DAFX** block in the above figure.

Classifying Digital Systems: Block v. sample-by-sample processing

We can classify the way a digital system processes the data in two ways:

Block v. sample-by-sample processing

Block Processing

Data is transferred into a **memory buffer** and then processed each time the buffer is filled with new data.

E.g. Fourier transforms (FFT), Discrete Cosine Transform (DCT), convolution — **more soon**

Sample-by-sample processing

Input is processed on individual sample data.

E.g. volume control, envelope shaping, IIR/FIR Filtering.

Linear v. Non-linear Time Invariant Systems

A second means of classification:

Linear time invariant system (LTI)

Systems that **do not change** behaviour over time and satisfy the superposition theory. The output signal is signal changed in amplitude and phase. *I.e.* A sine wave is still a sine wave just modified in amplitude and/or phase
E.g. Convolution, Filters

Non-linear time invariant system

Systems whose output is strongly shaped by non-linear processing that introduces harmonic distortion — *i.e.* harmonics that are not present in the original signal will be contained in the output.
I.e. if a sine wave is input the output may be a modified waveform or a sum of sine waves (*see Fourier Theory* later) whose frequencies may not be directly related to the input wave.
E.g. Compressors, Distortion, (Frequency) Enhancers.

Linear Time Invariant Systems

Classifying a Linear Time Invariant System

Linear time invariant systems are classified by the relation to their input/output functions, these are based on the following terms, definitions and representations:

- Impulse Response and discrete convolution
- Algorithms and signal flow graphs

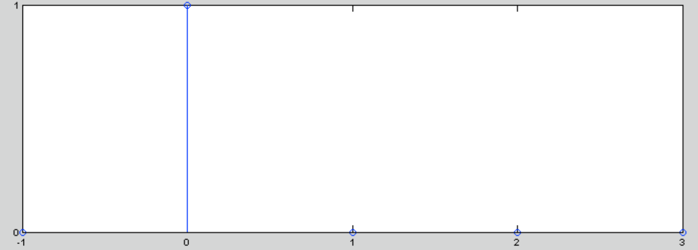
Impulse Response: Unit Impulse

Unit Impulse

- A very useful test signal for digital systems
- Defined as:

$$\delta(n) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{otherwise } (n \neq 0) \end{cases}$$

- Looks like:

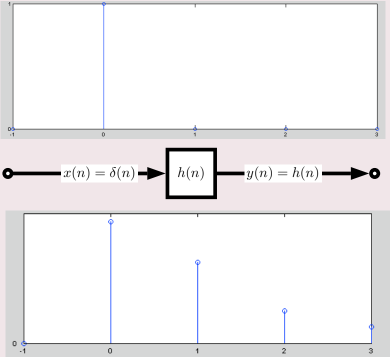


Impulse Response Definition

Impulse Response

If we apply a unit sample (impulse) function to a digital system we get an output signal $y(n) = h(n)$

- h(n) is called the **impulse response** of the digital system.



System Representation: Algorithms and Signal Flow Graphs

It is common to represent digital system signal processing routines as a visual **signal flow graphs**.

We use a simple *equation* relation to describe the **algorithm**.

Three Basic Building Blocks

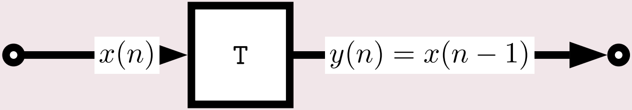
We will need to consider *three* processes:

- Delay
- Multiplication
- Summation

Signal Flow Graphs: Delay

Delay

- We represent a delay of **one sampling interval** by a block with a T **label**:



- We describe the algorithm via the equation: $y(n) = x(n-1)$

Signal Flow Graphs: Delay Example

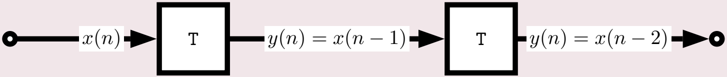
A Delay of 2 Samples

A delay of the input signal by **two** sampling intervals:

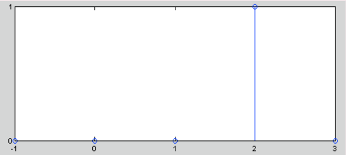
- We can describe the **algorithm** by:

$$\mathbf{y(n) = x(n-2)}$$

- We can use the block diagram to represent the **signal flow graph** as:



$x(n)$

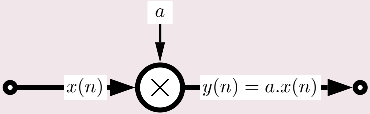


$y(n) = x(n-2)$

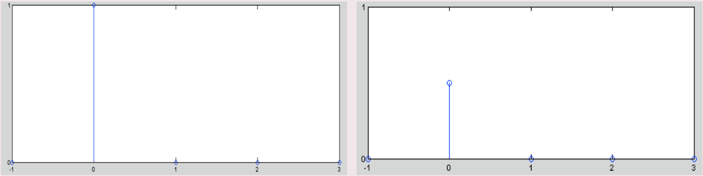
Signal Flow Graphs: Multiplication

Multiplication

- We represent a multiplication or weighting of the input signal by **a circle with a $\times$ label** .
- We describe the algorithm via the equation: $\mathbf{y(n) = a.x(n)}$



$e.g.\ a = 0.5$



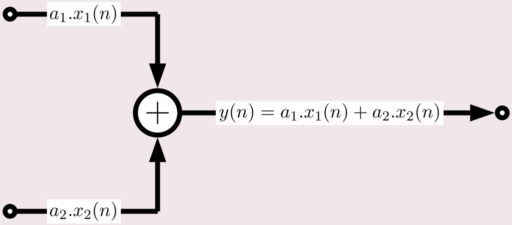
$x(n)$ $\qquad$ $y(n) = 0.5x(n)$

Signal Flow Graphs: Addition

Addition

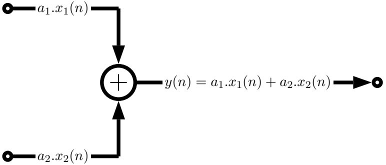
- We represent a addition of two input signal by **a circle with a + label** .
- We describe the algorithm via the equation:

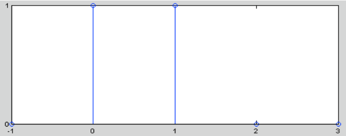
$$\mathbf{y(n) = a_1.x_1(n) + a_2.x_2(n)}$$



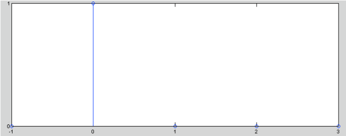
Signal Flow Graphs: Addition Example

In the example, set $a_1 = a_2 = 1$:

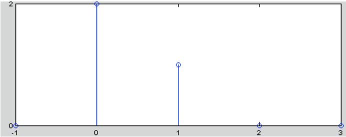




$x_1(n)$



$x_2(n)$



$y(n) = x_1(n) + x_2(n)$

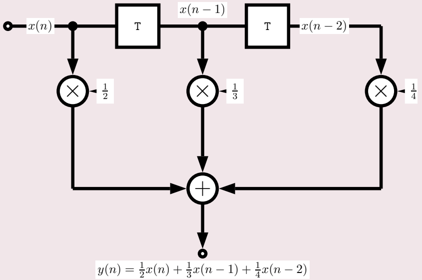
Signal Flow Graphs: Complete Example

All Three Processes Together

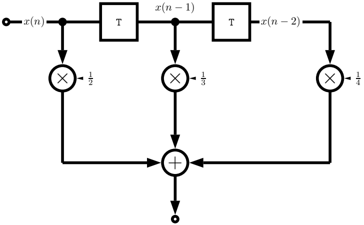
We can combine all above algorithms to build up more complex algorithms:

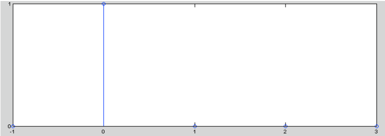
$$\mathbf{y(n) = \frac{1}{2}x(n) + \frac{1}{3}x(n-1) + \frac{1}{4}x(n-2)}$$

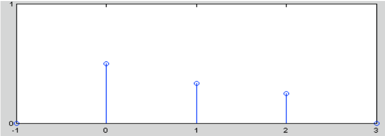
- This has the following signal flow graph:



Signal Flow Graphs: Complete Example Impulse Response







$x(n)$

$y(n) = \frac{1}{2}x(n) + \frac{1}{3}x(n-1) + \frac{1}{4}x(n-2)$

Transfer Function and Frequency Response

In a similar way to measuring the **time domain** impulse response $h(n)$ of a digital system we can measure the frequency domain response:

Frequency Domain Behaviour

The frequency domain behaviour of digital systems reflects the systems ability to **Pass**, **Reject** and **Enhance** **certain frequencies** in the input signal frequency spectrum.

We describe such behaviour with a **transfer function** $H(z)$ and the **frequency response** $H(f)$ of the digital system.

Note: To see how do this we will study the **Fourier Transform** in some detail shortly.