TÉCNICO LISBOA

Principles of Biosignals and Biomedical Imaging

3rd year, P_3 (ECTS: 3.0), LEBiom
2022/2023

João Sanches
jmrs@tecnico.ulisboa.pt
Department of Bioengineering
Instituto Superior Técnico
University of Lisbon

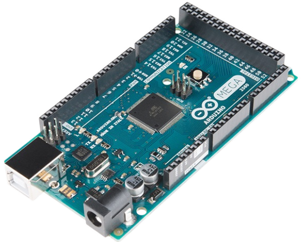
LTI systems and filtering

Discrete systems

Digital filters are algorithms implemented in digital computers with elementary arithmetic operations

$$x(n) \longrightarrow \boxed{T[.]} \longrightarrow y(n)$$

$$\begin{bmatrix} \cdots \\ x(n+1) \\ x(n) \\ x(n-1) \\ x(n-2) \\ \cdots \end{bmatrix}$$



$$\begin{bmatrix} \cdots \\ y(n+1) \\ y(n) \\ y(n-1) \\ y(n-2) \\ \cdots \end{bmatrix}$$

```
int filter(float x, float x1, float x2)
{               int out;
                out=int((x+x1+x2)/3);
                return(out);
}
```

Linear Time Invariant (LTI) Filters

$$x(n) \longrightarrow \boxed{T[.]} \longrightarrow \mathrm{y(n)}$$

Linearity: $\mathrm{y}(n) = T[\alpha x_1(n) + \beta x_2(n)] = \alpha y_1(n) + \beta y_2(n)$

Time Invariance: $\mathrm{z}(n) = T[x(n - n_0)] = y(n - n_0)$

The output of any LTI system can be obtained using the following recursive expression:

$$\mathrm{y}(n) = \underbrace{\sum_{r=0}^{q} b_r x(n-r)}_{\text{Non-recursive}} + \underbrace{\sum_{k=1}^{p} a_k y(n-k)}_{\text{Recursive}}$$

The coefficients a_k and b_r completely define the characteristics of the filter.

Linear Time Invariant (LTI) Filters

LTI representation

- Difference equation
- Impulse response
- Transfer function
- Frequency response

$$y(n) = \sum_{r=0}^{q} b_r x(n-r) + \sum_{k=1}^{p} a_k y(n-k)$$

$$h(n) = \alpha^n u(n)$$

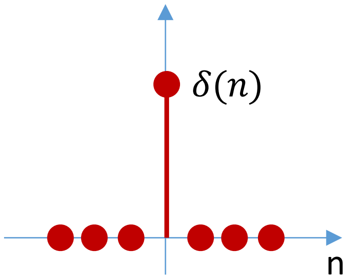
$$\mathrm{H}(z) = \frac{1}{1-\alpha z^{-1}}$$

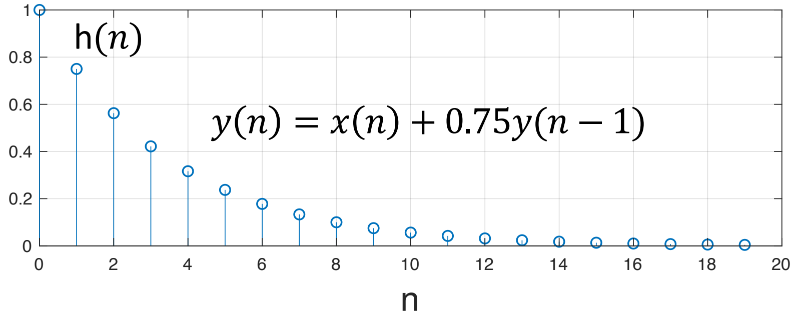
$$\mathrm{H}(\omega) = \frac{1}{1-\alpha e^{-j\omega}}$$

Linear Time Invariant (LTI) Filters

Impulse response

Impulse response: $h(n) = T[\delta(n)]$



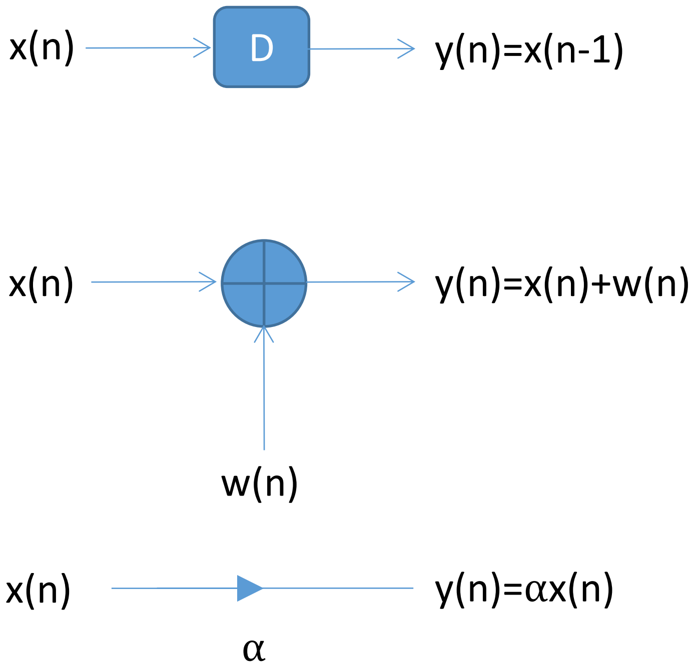


The impulse response fully describes a LTI system

Causality

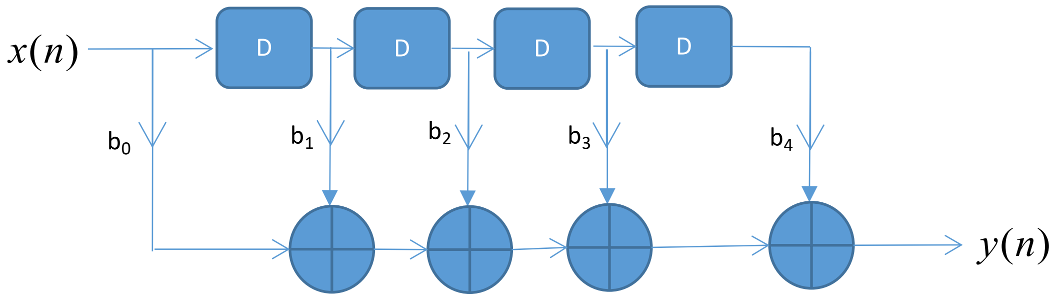
$$h(n) = 0 \; for \; n < 0$$

Basic blocks



FIR (Finite Impulse Response)

$$y(n) = \sum_{r=0}^{q} b_r x(n-r) + \cancel{\sum_{k=1}^{p} a_k y(n-k)}$$

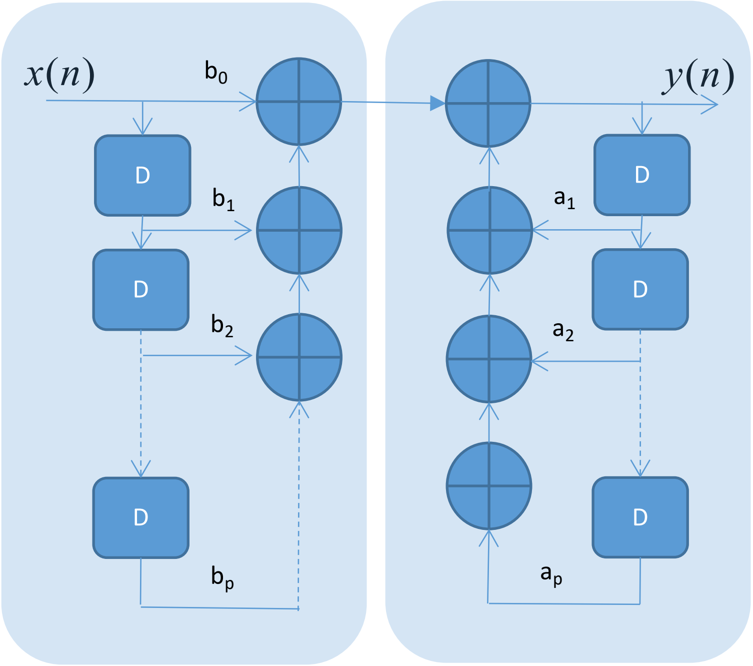


Compute impulse response

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2)$$

IIR (Infinite Impulse Response) Direct form I

$$y(n) = \sum_{r=0}^{q} b_r x(n-r) + \sum_{k=0}^{p} a_k y(n-r)$$



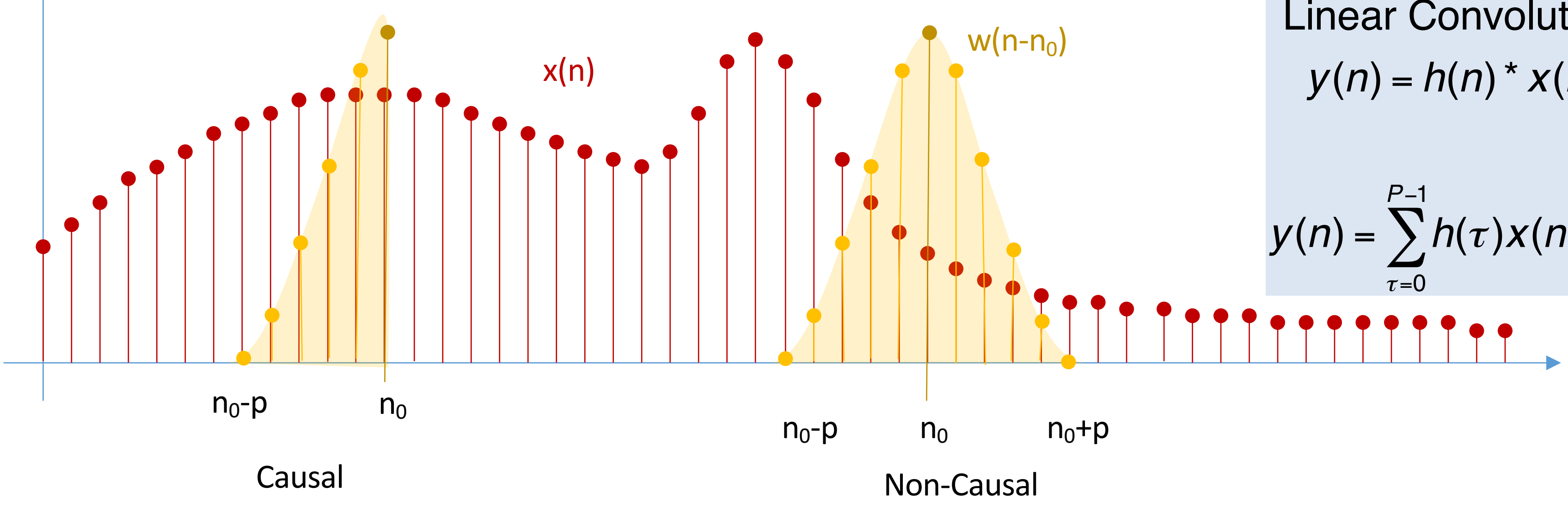
For N=M

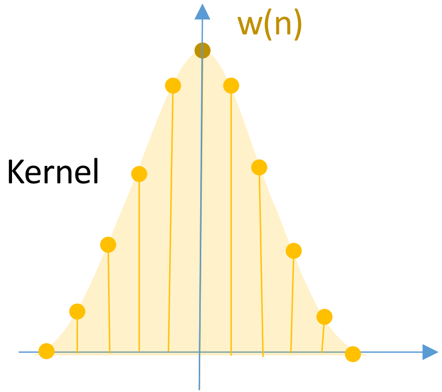
Compute impulse response

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

FIR SLIT

$$y(n) = \sum_{n-p}^{n+p} w(\tau)x(n-\tau)$$





Linear Convolution

$$y(n) = h(n) * x(n)$$

$$y(n) = \sum_{\tau=0}^{P-1} h(\tau)x(n-\tau)$$

TÉCNICO LISBOA

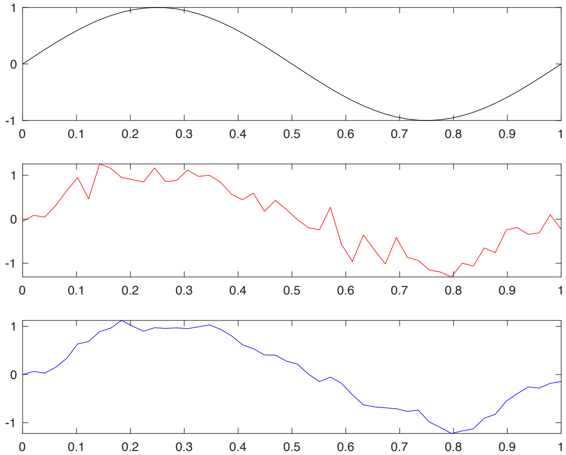
Interval

Basic input/output code

```
int OutPin    = 9;            // PWM output pin
int analogPin = 3;            // analog pin 3
float x=0.0, x1=0.0, x2=0.0;
int y=0;
int Ts= 100;                  //Sampling period ms

void setup() {          Serial.begin(9600);                          //  setup serial
                        pinMode(OutPin, OUTPUT);         // sets the pin as output
                        pinMode(analogPin, INPUT);       // sets the pin as input
                        }
void loop() {

    x2=x1;
    x1=x;
    x = float(analogRead(analogPin));  // read values (0 to 1023) from the input pin

    // Digital signal processing (Filtering)
    y=filter(x,x1,x2);

    analogWrite(OutPin, y/4);  // PWM, write values (0 to 255) to the output pin
    Serial.println(x);              // debug value

    delay(Ts); %Sampling period
    }
```

Low-pass filter (Moving average)

```
int filter(float x, float x1, float x2)
{
        int out;
        out=int((x+x1+x2)/3);
        return(out);
}
```



Filtering using vectors

FIR $y(n) = \sum_{r=0}^{q} w_r x(n-r)$

Causal

$$y(n) = [w_0 \quad w_1 \quad \dots \quad w_q] \underbrace{\begin{bmatrix} x(n) \\ x(n-1) \\ x(n-2) \\ \dots \\ x(n-q) \end{bmatrix}}_{X_n}$$

$$W = [w_0 \quad w_1 \quad \dots \quad w_q]^T$$

$$y(n) = W^T X_n$$

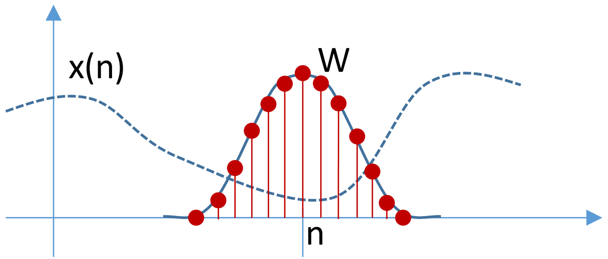
$$\overbrace{\begin{bmatrix} \vdots \\ x(n+q+1) \\ x(n+q) \\ x(n+q-1) \\ \vdots \\ x(n+1) \\ \boxed{\begin{matrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-q+1) \\ x(n-q) \end{matrix}} \\ x(n-q-1) \\ \vdots \end{bmatrix}}^{x(n)}$$

Non-causal

$$y(n) = \sum_{r=-q}^{q} w_r x(n-r)$$

$$y(n) = W^T \begin{bmatrix} x(n+q) \\ x(n+q-1) \\ \dots \\ x(n-r) \\ \dots \\ x(n-q+1) \\ x(n-q) \end{bmatrix}$$

$$W = [w_{-q} \quad w_{-q+1} \quad \dots \quad w_r \quad \dots \quad w_{q-1} \quad w_q]^T$$

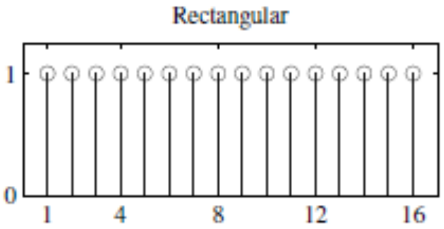


Moving average - FIR

$$y(n) = \sum_{r=-W}^{W} w_r x(n-r)$$

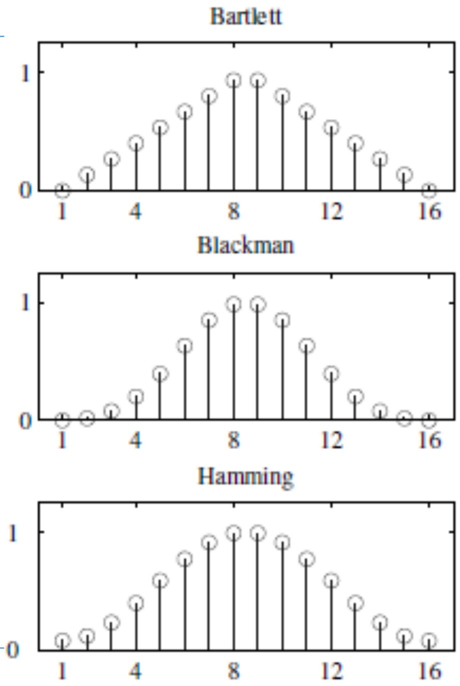
Non-causal

$$y(n) = \frac{1}{2W+1} \sum_{r=-W}^{W} x(n-r)$$



Average of the 2W+1 samples

Weighted Average of the 2W+1 samples

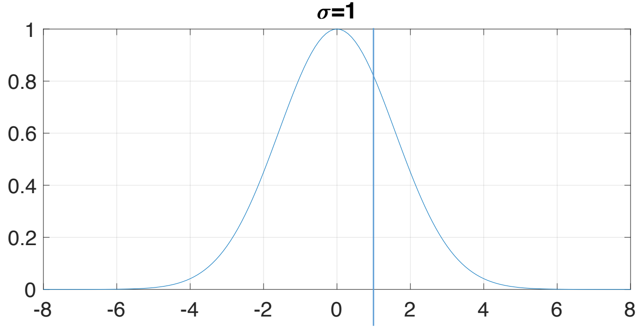


Gaussian filter

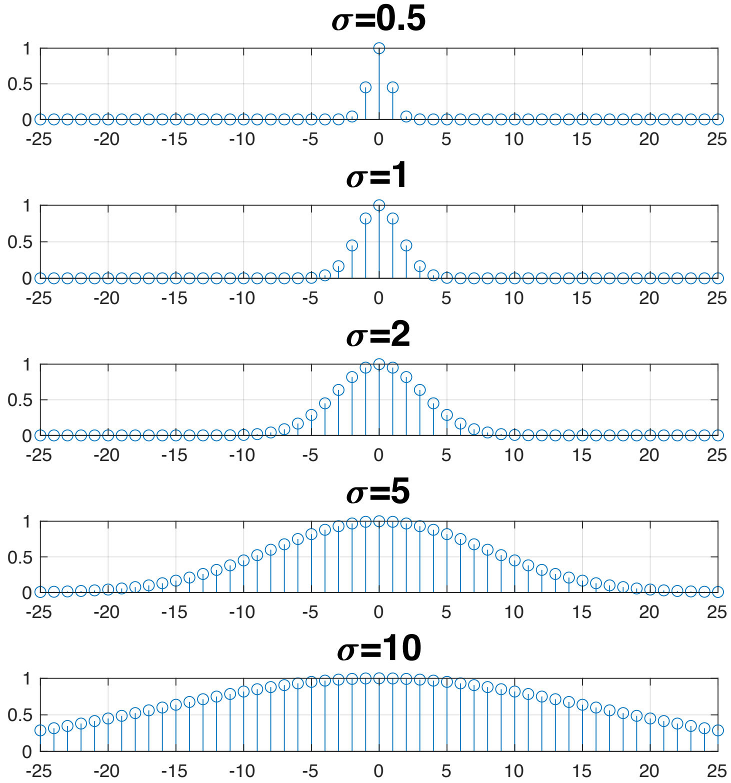
$$y(n) = \sum_{r=-W}^{W} w_r x(n-r)$$

Non-causal

$$w(n) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{n^2}{2\sigma^2}}$$



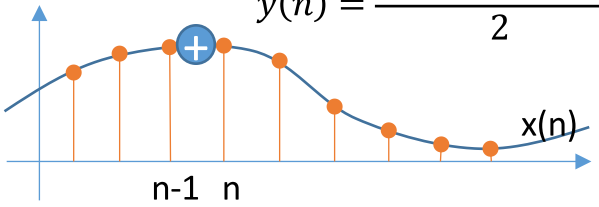
$\sigma = 1$



Derivatives

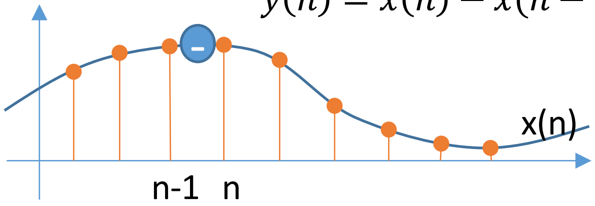
Moving Average: $W^0 = [1/2 \quad 1/2]^T$

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



First Difference/Derivative: $W^0 = [1 \quad -1]^T$

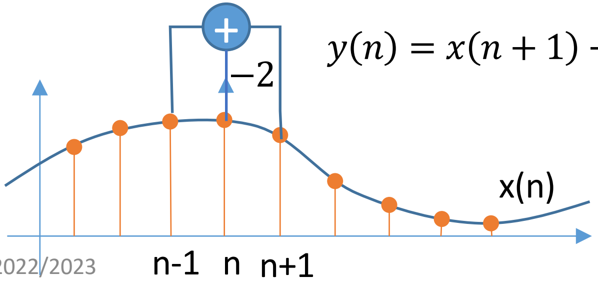
$$y(n) = x(n) - x(n-1)$$



Second difference/derivative: $W^2 = [1 \quad \underset{n=0}{-2} \quad 1]^T$

Non-causal

$$y(n) = x(n+1) - 2x(n) + x(n-1)$$



Infinite Impulse Response (IIR)

$$y(n) = \underbrace{\sum_{r=0}^{q} b_r x(n-r)}_{\text{Non-recursive}} + \underbrace{\sum_{k=0}^{p} a y(n-r)}_{\text{Recursive}}$$

Example

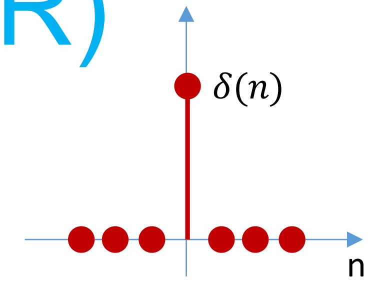
First order IIR

$$y(n) = x(n) + ay(n-1)$$

OR

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

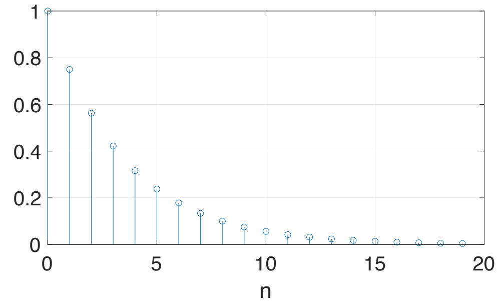
Impulse response



$$\mathrm{h}(n) = \delta(n) + ah(n-1)$$

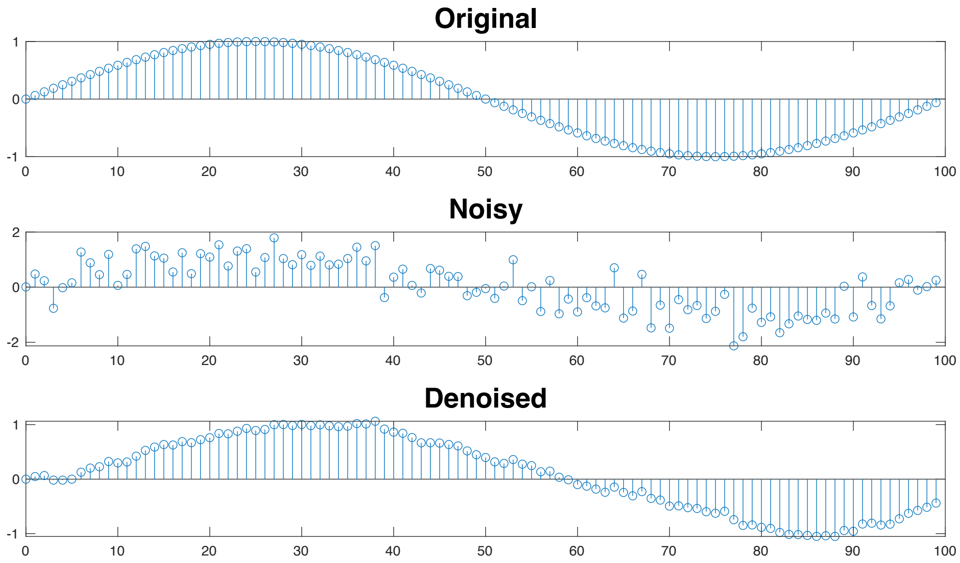
$$h(0) = \delta(0) + ah(-1) = 1 + a0 = 1$$
$$h(1) = \delta(1) + ah(0) = 0 + a * 1 = a$$
$$h(2) = \delta(2) + ah(1) = 0 + a * a = a^2$$
$$h(3) = \delta(3) + ah(2) = 0 + a * a^2 = a^3$$
$$\ldots$$
$$h(n) = \delta(n) + ah(n-1) = 0 + a * a^{n-1} = a^n$$

$$h(n) = a^n u(n)$$



Infinite Impulse Response (IIR)

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

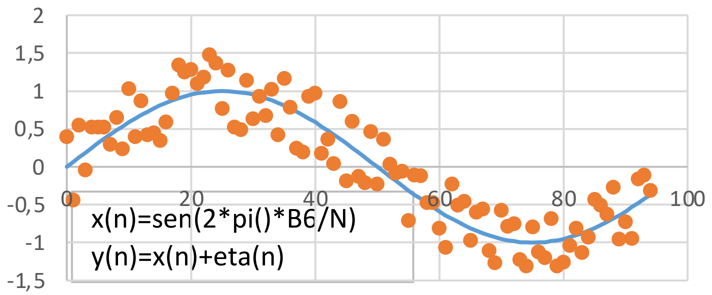


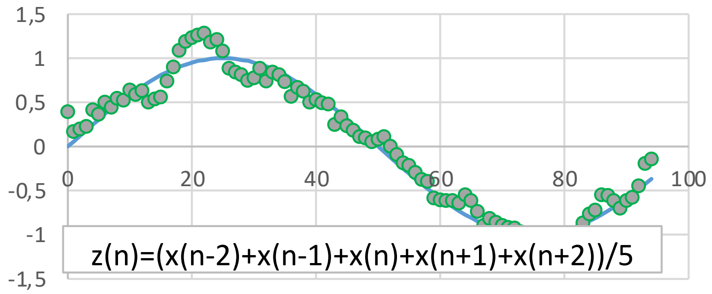

```
%%Recursive Filtering (IIR)

%Initializations
N=100;                  %dimension of the signals
n=(0:N-1)';             %Vector of discrete times
x=zeros(N,1);           %Clean
y=zeros(N,1);           %Noisy
z=zeros(N,1);           %Denoised

x=sin(2*pi*n/N);         %Original

%Noisy signal generation
sig=0.5;
z=x+sig*randn(size(x));

%Filter
a=0.9;
for l=2:N
   y(l)=(1-a)*z(l)+a*y(l-1);
end

figure(1);
subplot(3,1,1); stem(n,x); title('x','Fontsize',20);
subplot(3,1,2); stem(n,z); title('z','Fontsize',20);
subplot(3,1,3); stem(n,y); title('y','Fontsize',20);
```

Exemplo prático

Example – Noise removal

N	100			
n	x(n)	eta(n)	y(n)=x(n)+eta(n)	z(n)
0	0	0,3973648	0,397364804	0,3973648
1	0,06279052	-0,4944778	-0,431687292	0,17211419
2	0,12533323	0,42533183	0,550665067	0,20106054
3	0,18738131	-0,2259884	-0,038607092	0,22798082
4	0,24868989	0,2788773	0,527567191	0,4193918
5	0,30901699	0,22294923	0,531966228	0,36874272
6	0,36812455	0,15724305	0,525367599	0,50704609
7	0,42577929	-0,1283596	0,297419676	0,44929552
8	0,48175367	0,17115606	0,652909734	0,54994379
9	0,53582679	-0,2970124	0,238814376	0,52501884
10	0,58778525	0,44742234	1,035207588	0,63996065
11	0,63742399	-0,2366812	0,400742813	0,59537483
12	0,68454711	0,18758161	0,872128714	0,63825264
13	0,72896863	-0,298988	0,429980643	0,50178836
14	0,77051324	-0,3173098	0,453203433	0,54144781
15	0,80901699	-0,4561308	0,352886211	0,56167981
16	0,84432793	-0,2452879	0,599040063	0,74489325





TÉCNICO LISBOA