Disciplina: Processamento Digital de Sinais

Material _aula 3

Ambiente Blackboard

Apresentação

- 1) Implementação de algoritmos: Média Móvel – Matlab/Python, Linguagem C
- 2) Implementação do eco – Matlab/Python
- 3) Transformação de variáveis
- 4) Convolução
- 5) Tarefas

Sistema Discreto

2.3 Discrete-time systems

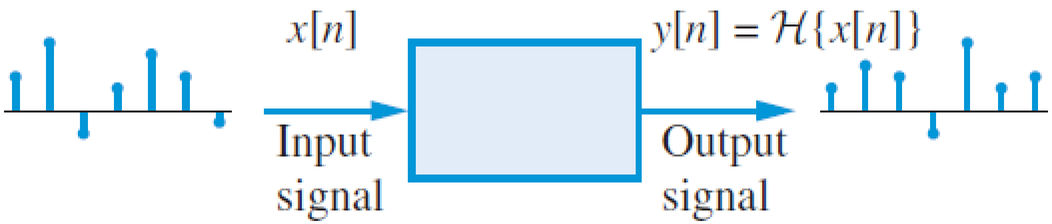


Figure 2.5 Block diagram representation of a discrete-time system.

Implementação de algoritmos

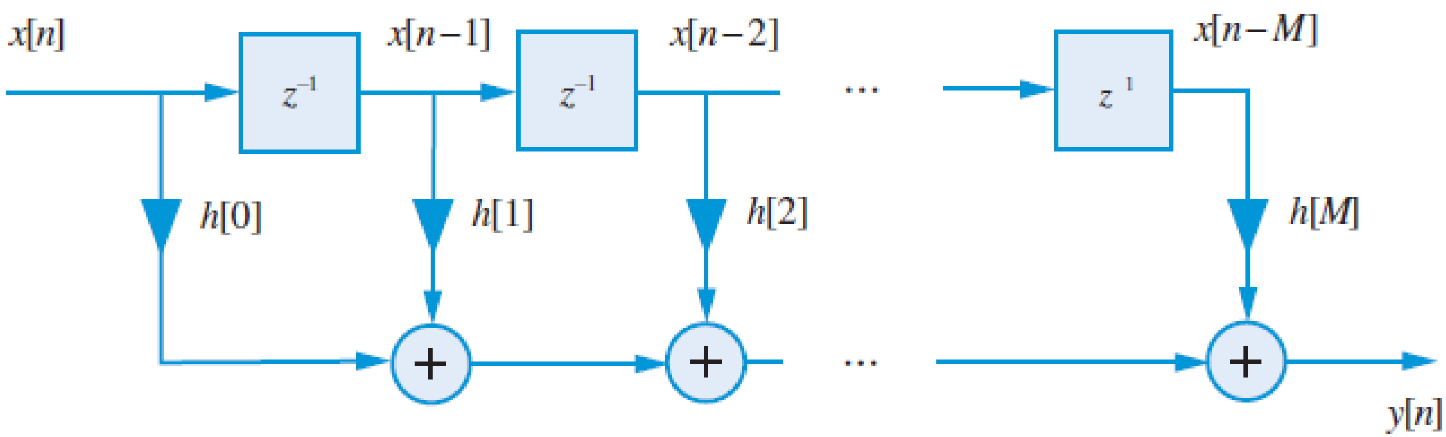
- Média Móvel de k elementos da entrada
- Por exemplo, para k = 4, temos:

$$y[n] = \frac{x[n] + x[n-1] + x[n-2] + x[n-3]}{4}$$

$$y[n] = 1/4\,.\,x[n] + 1/4\,.\,x[n-1] + 1/4\,.\,x[n-2] + 1/4\,.\,x[n-3]$$

Implementação de algoritmos

- Média Móvel de k elementos da entrada



Implementação Média Móvel

Arquivo de entrada

10
2
8
-4
-2
3

Vetor x[n] dimensão k,1

0	10	2	8	-4	-2	3
0	0	10	2	8	-4	-2
0	0	0	10	2	8	-4
0	0	0	0	10	2	8

Vetor Coefs Dimensão k,1

1/k
1/k
1/k
1/k

Arquivo de saída

2,5
3
5
4

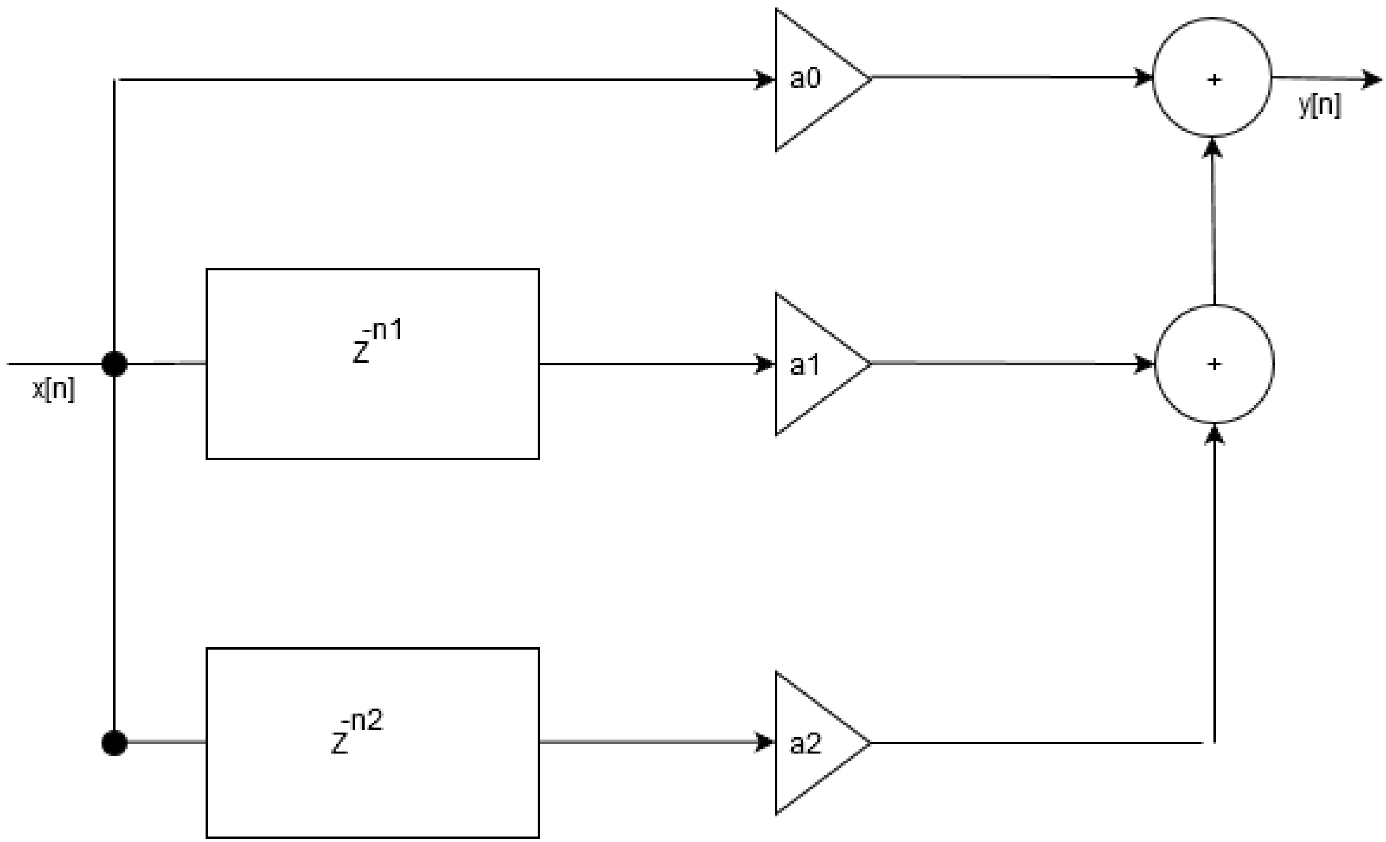
Implementação de algoritmos:

- Média Móvel - Matlab /Python
- Média Móvel - Linguagem C
- Usar como entrada um sinal de ruído branco gerado no ocenaudio
- Testar com diferentes tamanhos de média e salvar o arquivo de saída, por exemplo, k = 4, k = 8, k = 16, k = 32, etc

Implementação de algoritmos:

- Média Móvel - Matlab /Python
- Usar como entrada o impulso unitário e obtenha a correspondente saída.
- Usar como entrada o degrau unitário e obtenha a correspondente saída.

Implementação do eco



Implementação do eco

- Implemente um programa em Matlab/Python que execute a equação do diagrama de blocos do eco.

$$y[n] = a0.x[n] \; + \; a1.x[n-n1] \; + \; a2.x[n-n2]$$

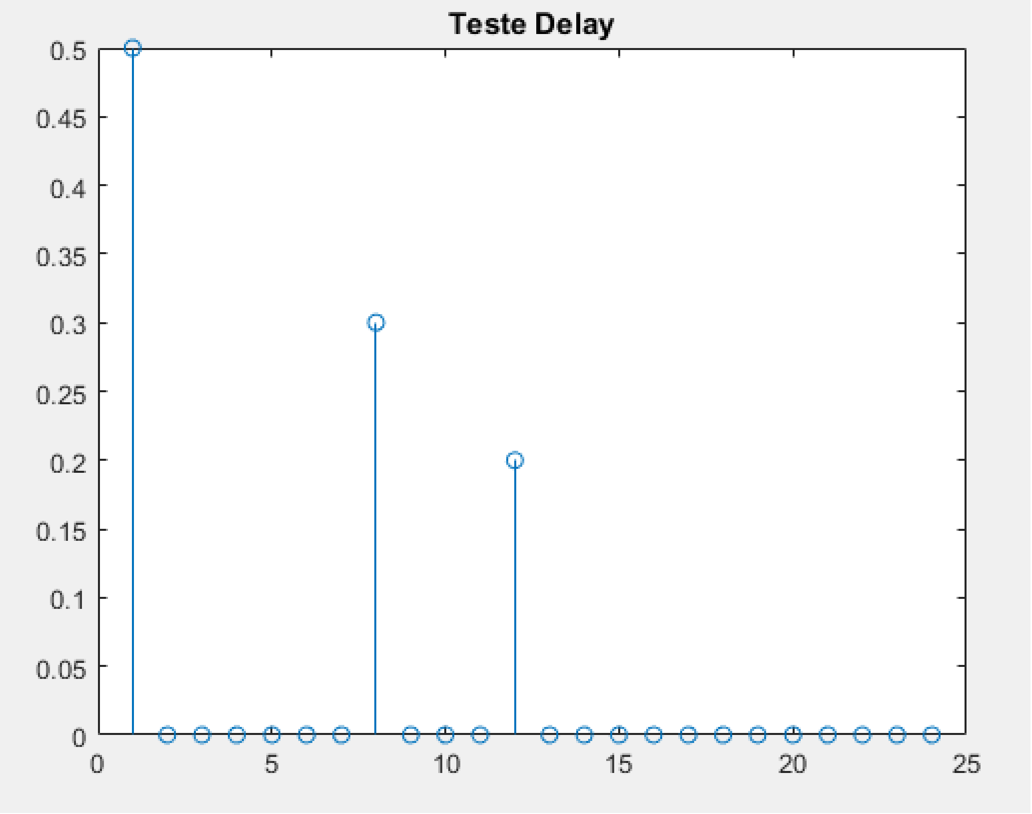
Implementação do eco

- Utilize Fs= 8k, n1 = correspondendo a um atraso de 10ms e n2 de 15ms.
- Fs = 8k, Ts = 125us.
- Portando: n1 corresponde a 80 amostras de delay e n2 120 amostras
- Use a0 = 0,5; a1 = 0,3 e a2 = 0,2.

- Valide o algoritmo para uma entrada impulso unitário

Implementação do eco

- Para uma entrada impulso unitário, n1 de 8 amostras de delay e n2 de 12 amostras, temos:



Implementação do eco

- % Exemplo delay
- clear all; close all; clc;
- Fs = 8000;
- t1 = 1.0*10^-3; t2 = 1.5*10^-3;
- n1 = t1*Fs; n2 = t2*Fs
- % Definição dos ganhos
- a0 = .5; a1 = .3; a2 = .2;
- tama_delay = n2;
- vetor_delay = zeros(tama_delay ,1);
- % Definindo a entrada
- entrada = zeros(2*tama_delay,1);
- entrada(1,1) = 1; % definindo o impulso unitário
- tama_loop = length(entrada);
- vet_saida = zeros(tama_loop,1);
- for j = 1: tama_loop
-     input = entrada(j,1);
-     vetor_delay(1,1) = input;
-     y = a0* vetor_delay(1,1) + a1 * vetor_delay(n1,1) + a2 * vetor_delay(n2,1);
-     % Desloca o vetor de delay
-     for k = 0: tama_delay-2
-             vetor_delay(tama_delay-k,1) = vetor_delay(tama_delay-k-1,1);
-     end
-   vet_saida(j,1) = y;
- end
- stem(vet_saida)
- title('Teste Delay');

Implementação do eco

- TAREFA: Modifique o programa implementado para gerar o eco em um arquivo de aúdio (Ocenaudio) com atrasos bem maiores de forma que sejam perceptiveis ao ouvir.

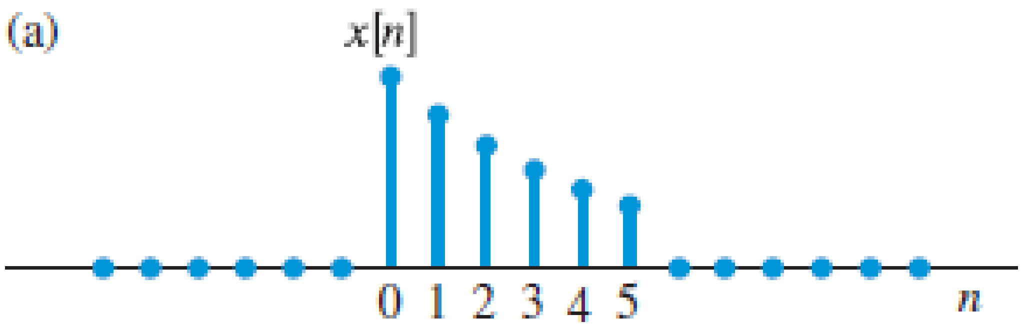
Transformação de variáveis

- Transformação na varíavel indepedente n:
- Reversão no tempo
- Deslocamento no tempo

Transformação de variáveis

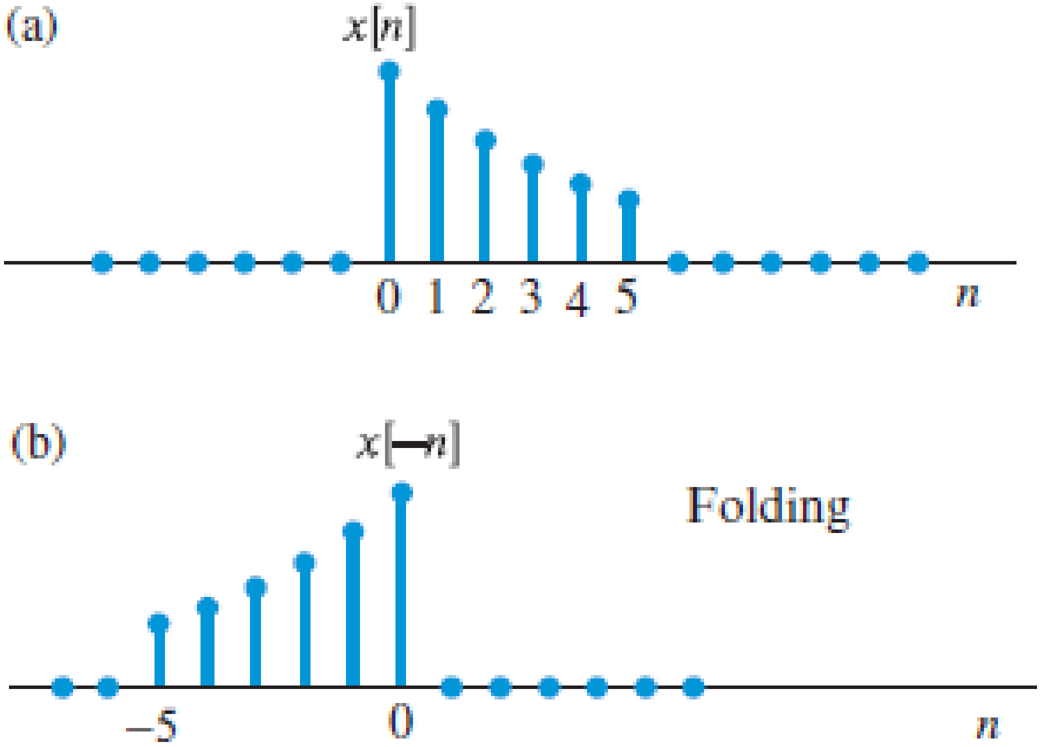
- Reversão no tempo

Time-reversal or *folding*, which is an operation defined by $y[n] = x[-n]$, reflects the sequence $x[n]$ about the origin $n = 0$. Folding a sequence in MATLAB is done using the



Transformação de variáveis

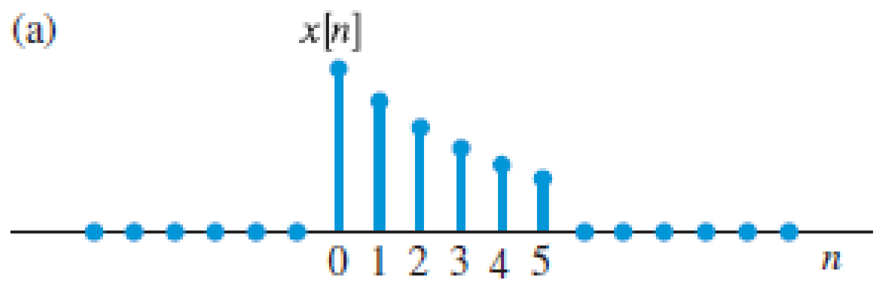
- Reversão no tempo



Transformação de variáveis

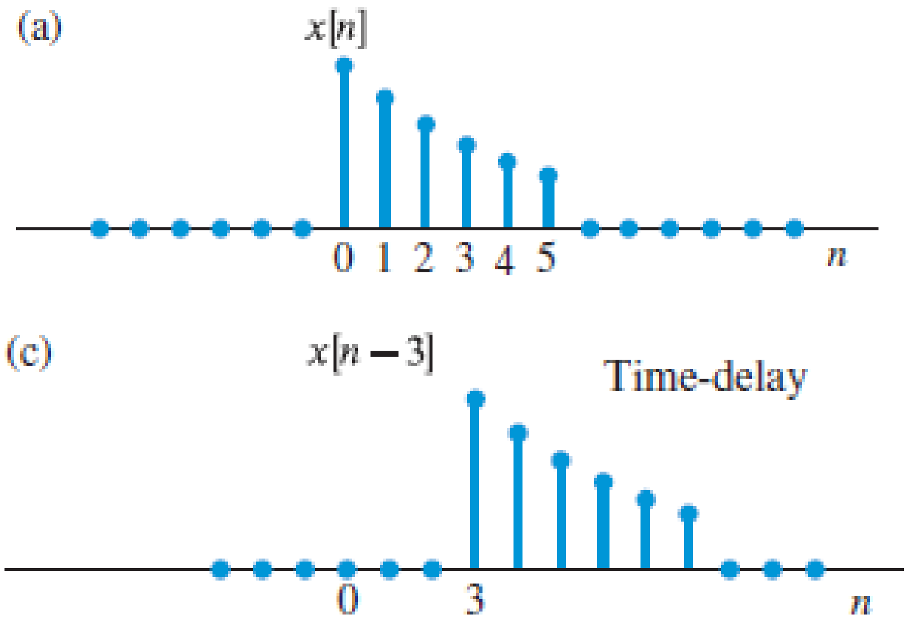
- Deslocamento no tempo

Time-shifting is defined by the formula $y[n] = x[n - n_0]$. For $n = n_0$ we have, $y[n_0] = x[0]$; thus, the sequence $x[n]$ is shifted by n_0 samples so that the sample $x[0]$ is moved to $n = n_0$. If $n_0 > 0$, the sequence $x[n]$ is shifted to the right; because the sequence "appears later," the shift corresponds to a *time-delay*. If $n_0 < 0$, the sequence is shifted to the left; because the sequence "appears earlier," the shift amounts to a *time-advance*.



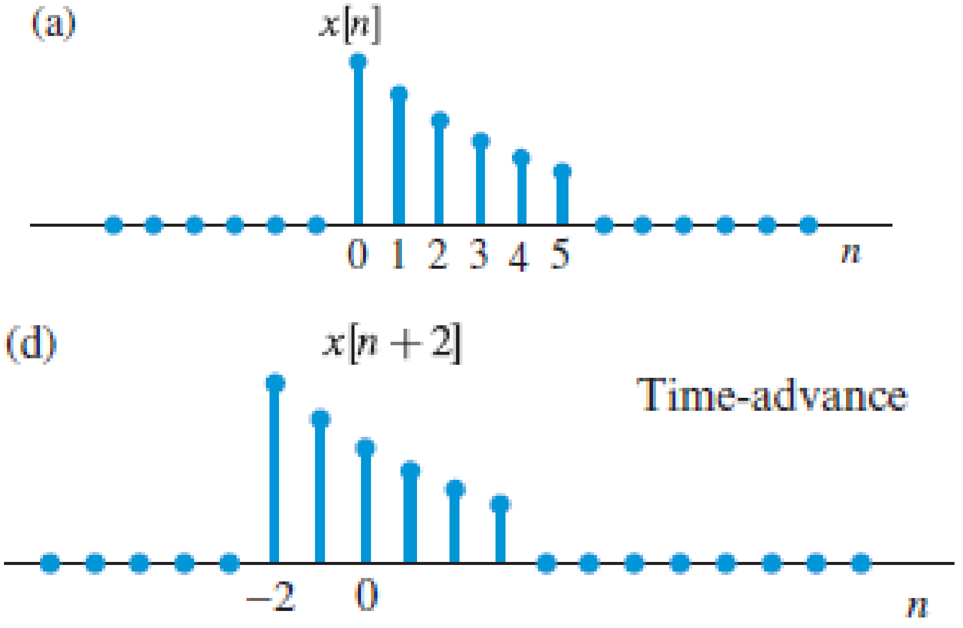
Transformação de variáveis

- Deslocamento no tempo - Atraso

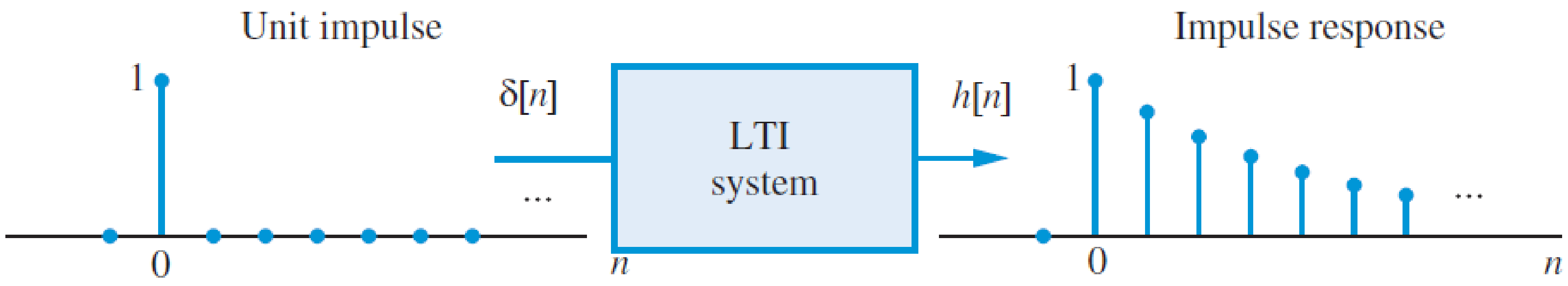


Transformação de variáveis

- Deslocamento no tempo - Avanço



Convolução



Convolução

Signal decomposition into impulses Let us define the sequence

$$x_k[n] = \begin{cases} x[k], & n = k \\ 0, & n \neq k \end{cases} \tag{2.29}$$

which consists of the sample $x[k]$ of $\{x[n]\}$ at $n = k$ and zero elsewhere. The sequence $x_k[n]$ can be obtained by multiplying the sequence $\{x[n]\}$ by the sequence

$$\delta[n-k] = \begin{cases} 1, & n = k \\ 0. & n \neq k \end{cases} \tag{2.30}$$

Hence, the sequence $\{x[n]\}$ can be expressed as

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]. \quad -\infty < n < \infty \tag{2.31}$$

Convolução

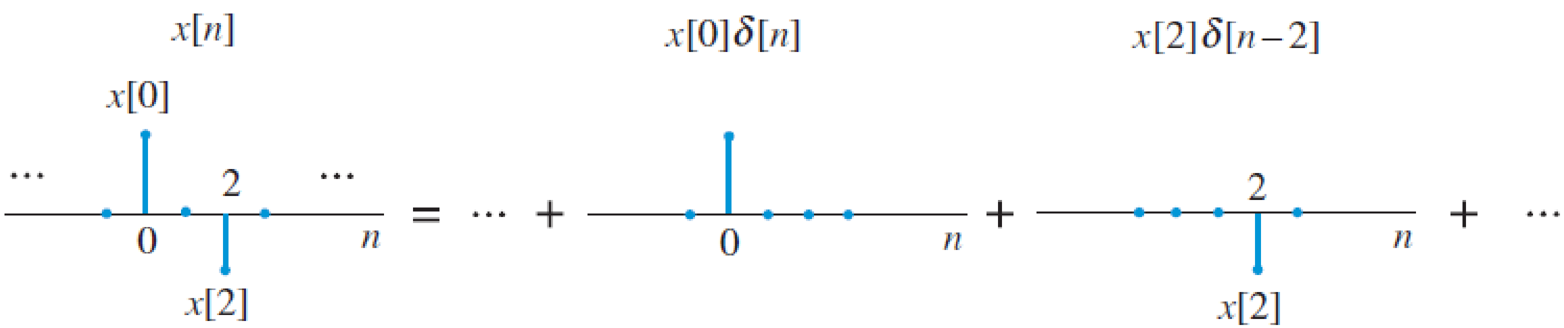


Figure 2.10 Decomposition of a discrete-time signal into a superposition of scaled and delayed unit sample sequences.

Convolução

Convolution sum We now illustrate how the properties of linearity and time-invariance restrict the form of discrete-time systems and simplify the understanding and analysis of their operation. More specifically, we show that we can determine the output of any LTI system if we know its impulse response.

We start by recalling that any sequence $x[n]$ can be decomposed into a superposition of scaled and shifted impulses as in (2.31). Consider next a linear (but possibly time-varying) system and denote by $h_k[n]$ its response to the basic signal $\delta[n-k]$. Then, from the superposition property for a linear system (see (2.27) and (2.28)), the response $y[n]$ to the input $x[n]$ is the same linear combination of the basic responses $h_k[n]$, that is,

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h_k[n], \tag{2.34}$$

Convolução

which is known as the *superposition summation* formula. Equation (2.34) provides the response of a linear time-varying system in terms of the responses of the system to the impulses $\delta[n-k]$.

If we impose the additional constraint that the system is time-invariant, we have

$$\delta[n] \xrightarrow{\mathcal{H}} h[n] \Rightarrow \delta[n-k] \xrightarrow{\mathcal{H}} h_k[n] = h[n-k]. \tag{2.35}$$

Substitution of (2.35) into (2.34) gives the formula

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]. \quad -\infty < n < \infty \tag{2.36}$$

Convolução

Equation (2.36), which is commonly called the *convolution sum* or simply *convolution* is denoted using the notation $y[n] = x[n] * h[n]$. Therefore, the response of a linear time-invariant system to any input signal $x[n]$ can be determined from its impulse response $h[n]$ using the convolution sum (2.36). If we know the impulse response of an LTI system, we can compute its response to any input without using the actual system. Furthermore,

$$y[n] = x[n] * h[n]$$

Convolução

The operation described by the convolution sum takes two sequences $x[n]$ and $h[n]$ and generates a new sequence $y[n]$. We usually say that sequence $y[n]$ is the *convolution* of sequences $x[n]$ and $h[n]$ or that $y[n]$ is obtained by convolving $x[n]$ with $h[n]$. Convolution

$$y[n] = x[n] * h[n]$$

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]. \qquad -\infty < n < \infty$$

Convolução - Cálculo

- Considere as sequências

$$x[n] = \{\underset{\uparrow}{1}\ 2\ 3\ 4\ 5\}, \quad h[n] = \{-1\ \underset{\uparrow}{2}\ 1\}.$$

x[k]	0	0	0	1	2	3	4	5	
h[k]			-1	2	1				
h[-k]			1	2	-1				

Convolução - Cálculo

x[k]	0	0	0	1	2	3	4	5	
h[-k]			1	2	-1				
n = -1 h[-k-1]		1	2	-1					y[-1] = -1
n = 0 h[-k-0]			1	2	-1				y[0] = 0
n = 1 h[-k+1]				1	2	-1			y[1] = 2
n = 2 h[-k+2]					1	2	-1		y[2] = 4

Convolução – Resumo do cálculo

In summary, the computation of convolution of two sequences involves the following steps:

1. Change the index of the sequences $h[n]$, $x[n]$ from n to k and plot them as a function of k.
2. Flip (or fold) the sequence $h[k]$ about $k = 0$ to obtain the sequence $h[-k]$.
3. Shift the flipped sequence $h[-k]$ by n samples to the right, if $n > 0$, or to the left, if $n < 0$.
4. Multiply the sequences $x[k]$ and $h[n-k]$ to obtain the sequence $z_n[k] = x[k]h[n-k]$.
5. Sum all nonzero samples of $z_n[k]$ to determine the output sample at the given value of the shift n.
6. Repeat steps $3-5$ for all desired values of n.

Convolução – Resumo do cálculo

- Considere as sequências

$$x[n] = \{\underset{\uparrow}{1}\ 1\ 1\ 1\ 1\ 1\}, \quad h[n] = \{\underset{\uparrow}{1}\ 0.5\ 0.25\ 0.125\},$$

Convolucão – Forma gráfica

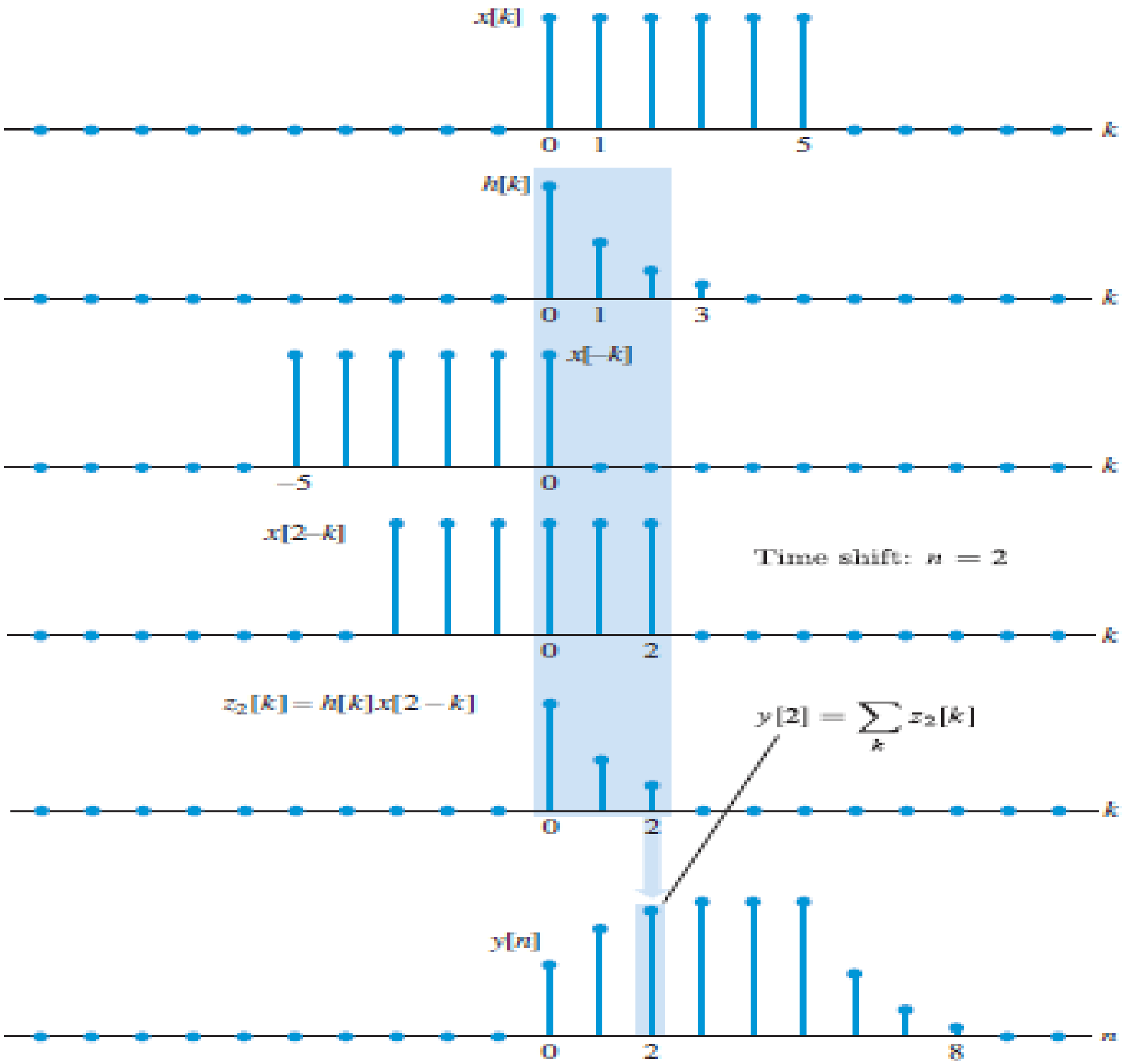


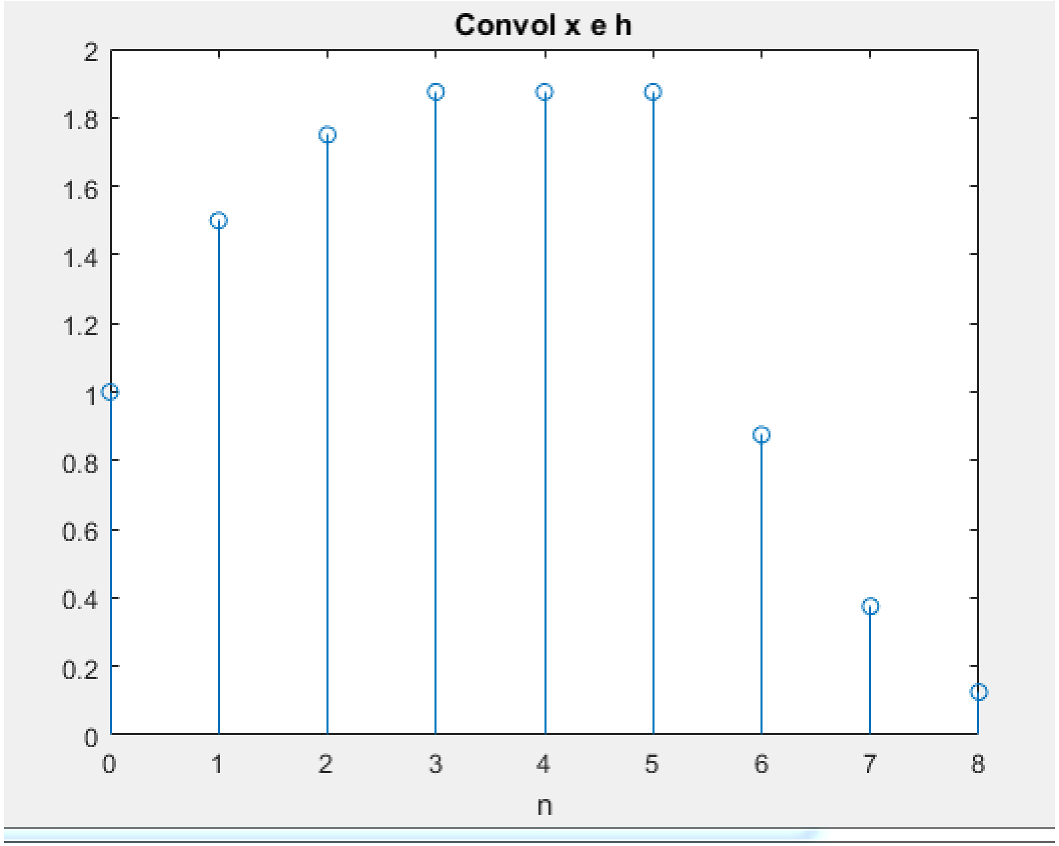
Figure 2.12 Graphical illustration of convolution as a scanning operation.

Convolução – Forma gráfica

Programa exemplo no Matlab

- % Exemplo do calculo da convolução
- clear all;close all; clc;
-
- x = [1 1 1 1 1 1];
- h = [1 .5 .25 .125];
-
- y = conv(x,h)
-
- tama = length(x) + length(h) -1;
-
- n = (0:1:tama-1);
- stem(n,y)
-
- title('Convol x e h');
- xlabel('n');

Convolução – Forma gráfica



Convolução – Forma gráfica

- **Tarefas**:
- Considere o filtro média móvel da aula passada com k = 8, obtenha:
- A saída do filtro para as seguintes entradas: impulso unitário, degrau unitário e o vetor x[n] = [1, 0.5, 0.25, 0.125]

Próxima aula

- 3) Transformada de Fourier Discreta
- 4) Transformada Z
- 5) Resposta em frequencia

Disciplina: Processamento Digital de Sinais

Material_aula 3

Ambiente Blackboard