Introduction to Discrete Signal Analysis

Multimedial Signal Processing 1st Module

Politecnico di Milano –
Polo regionale di Como

Particulars

- Eliana Frigerio
  - efrigerio@elet.polimi.it
  - Phone: 02 2399 9653
  - Send to me an email in order to organize conferences and for any question

Summary:

- Ex1: Represent Discrete Time Signals:
  - Unit Sample Sequence
  - Unit Step Sequence
  - Real-valued Exponential Sequence
  - Complex-valued Exponential Sequence
  - Sinusoidal Sequence
  - Periodic Sequence
- Ex2: Random Sequence
- Ex3: Even and Odd Synthesis

Summary:

- Ex4: Operations on Sequences
  - Signal Addition
  - Signal multiplication
  - Scaling
  - Shifting
  - Folding
  - Sample Summation
  - Sample products
  - Signal energy
  - Signal power
- Ex5: Convolution
- Ex6: Correlation

Exercise 1 (1/8)

- Goal: represent finite-duration sequences in Matlab:
  - Using a row vector of appropriate values does not bring information about smple position n
  - A correct representation of x(n) would require two vectors, one for x and one for n:

    n=[-3 -2 -1 0 1 2 3 4];

    x=[2 1 -1 0 1 4 3 7];

Exercise 1 (2/8)

- Unit sample sequence over the $n_1 \le n \le n_2$ interval:

$$\delta(n-n_0)=\begin{cases}1 & n=n_0\\ 0 & n\neq n_0\end{cases}$$

  - n =[n1:n2];
  - x = [(n-n0)==0];
  - % if n-n0 is equal to 0 --> then x(n) is equal to 1
  - % else if n-n0 is different from 0 --> then x(n)=0

Exercise 1 (3/8)

- Unit step sequence over the $n_1 \leq n \leq n_2$ interval:

$$u(n-n_0)=\begin{cases}1 & n\geq n_0\\ 0 & n<n_0\end{cases}$$

  - n =[n1:n2];
  - x = [(n-n0)>=0];
  - % if n-n0 is major or equal to 0 --> then x(n) is equal to 1
  - % else if n-n0 is minor than 0 --> then x(n)=0

Exercise 1 (4/8)

- Real-valued exponential sequence:

$$x(n) = \alpha^n \quad \forall n; \alpha \in \Re$$

  - n =[n1:n2];
  - x = (a).^n;

- HINT: Z = X.^Y denotes element-by-element powers. X and Y must have the same dimensions unless one is a scalar.

Exercise 1 (5/8)

- Complex-valued exponential sequence:

$$x(n) = e^{(\sigma + j\omega_0)n} \quad \forall n$$

  - Where σ is the attenuation and
    ω is the frequency in radiant

  - n =[n1:n2];
  - x=exp((2+3j).*n);

Exercise 1 (6/8)

- Sinusoidal :

$$x(n) = \cos(\omega_0 n + \vartheta) \quad \forall n$$

  - Where ϑ is the initial phase in radiants and
    ω is the frequency in radiants

  - n =[n1:n2];
  - x=3*cos(0.1*pi*n+pi/3) + 2*sin(0.5*pi*n);

Exercise 1 (7/8)

- Periodic sequence: A sequence x(n) is periodic if

$$x(n) = x(n+N) \quad \forall n$$

N = fundamental period

  - Generate P periods of $\tilde{x}(n)$ from one period {x(n), 0≤n≤N-1};
  - x = [(n-n0)==0];
  - % if n-n0 is equal to 0 --> then x(n) is equal to 1
  - % else if n-n0 is different from 0 --> then x(n)=0

Exercise 1 (8/8)

- Generate P periods of $\tilde{x}(n)$ from one period {x(n), 0≤n≤N-1};

  - xtilde= x'*ones(1,P)

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \\ x(N-1) \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

% P columns of x' (matrix of NxP elements),

x is a row vector (we need a column vector)

  - xtilde=xtilde(:); % reads matrix by column obtaining a long column vector
  - xtilde=xtilde'; % row vector

Exercise 2 (1/10)

- Random sequence: these sequences are characterized by parameters of the associated probability density functions.

- In Matlab two types of (pseudo-) random sequences are available:

Exercise 2 (2/10)

- Matlab provides the functions:
- "x=rand(M,N)" that produces a M-by-N matrix with random entries, chosen from a uniform distribution on the interval (0.0,1.0).
- "m=mean(x)" that compute the mean value:
  - For vectors, "m" is the mean value of the elements in "x"
  - For matrices, "m" is a row vector containing the mean value of each column.
- "s=var(x)" that compute the variance:
  - For vectors "s" returns the variance of the values in "x".
  - For matrices, "s" is a row vector containing the variance of each column of "x".

Exercise 2 (3/10)

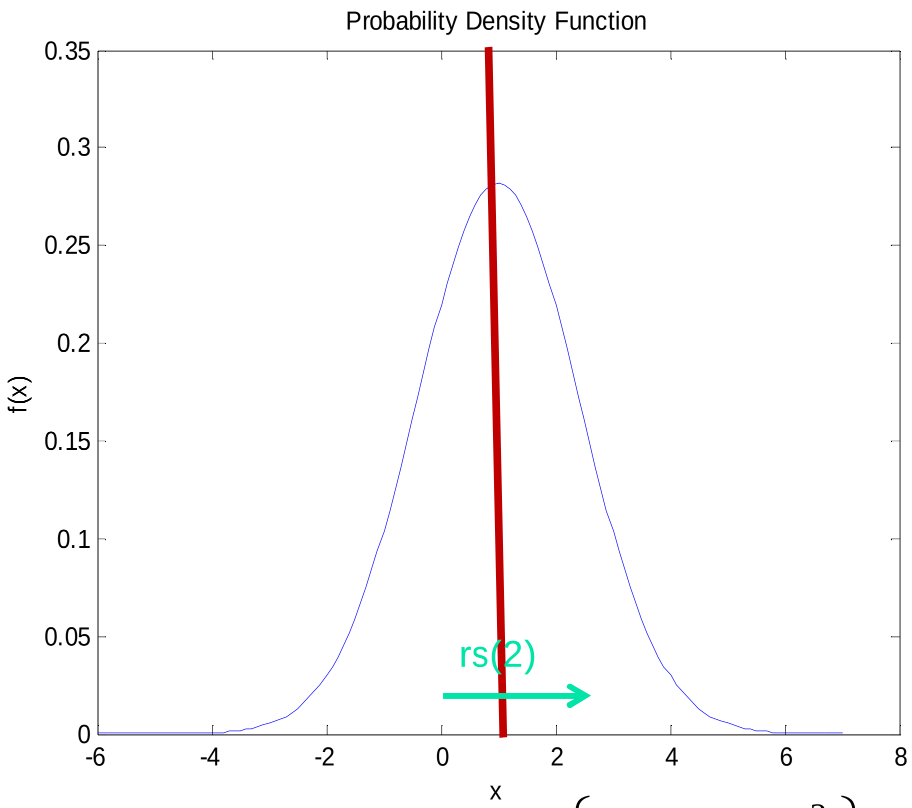
- Pseudocode:
  - generate a iid sequence of 100 samples with uniform distribution on the interval (0.0 , 1.0).
    - x = rand(1,100);
  - generate a iid sequence of 100 samples with uniform distribution on the interval (2.0 , 3.0).
    - x =(rand(1,100))+2;

Exercise 2 (4/10)

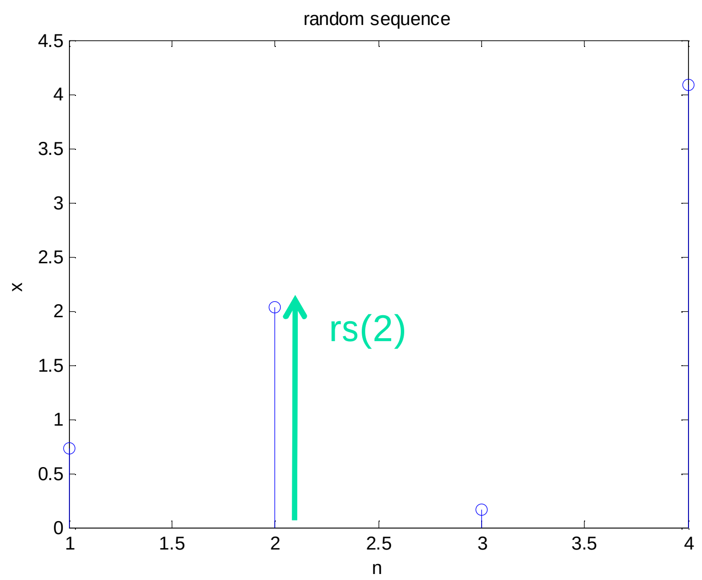
- Pseudocode:
  - generate a iid sequence of 100 samples with normal distribution
    - x = randn(100,1);
  - set variance = s
    - x = sqrt(s)*randn(100,1);
  - set mean = m
    - x = sqrt(s)*randn(100,1)+m;

- N.B. For a gaussian distribution, 99,7% of possible value are inside the interval $m \pm 3\sqrt{s}$

Exercise 2 (5/10)



rs = [0.7360   2.0264   0.1680   4.0875]



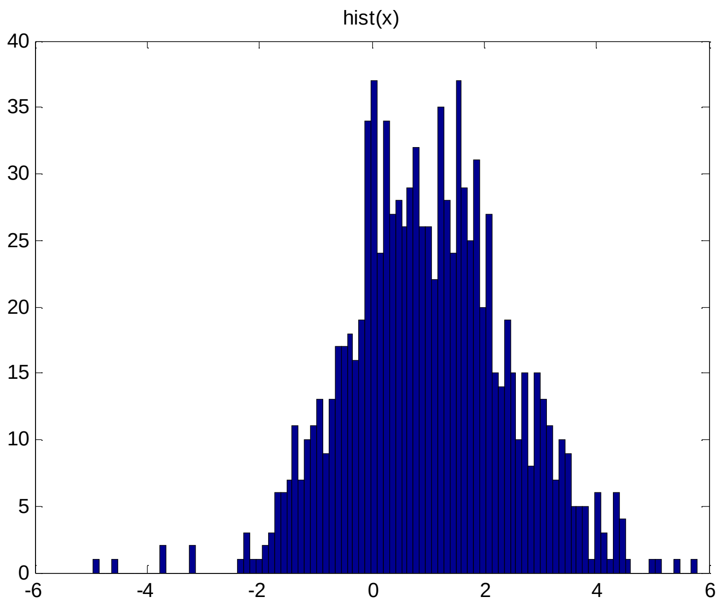
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-m)^2}{2\sigma^2}\right\}$$

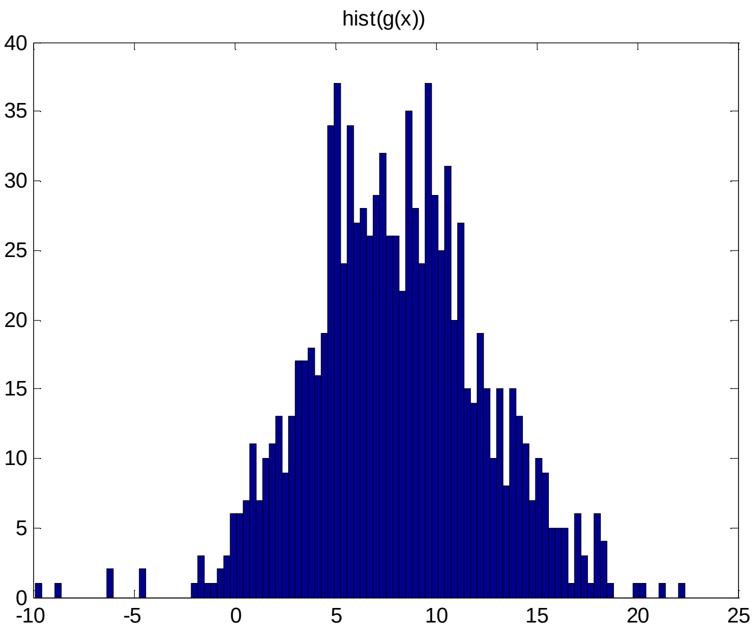
Exercise 2 (6/10)

- Hint: Remember:
  - if x is a gaussian variable $x \approx N(\eta_x, \sigma_x^2)$

then g(x)=ax+b is a guassian variable

$$y = g(x) \approx N(a\eta_x + b, a^2\sigma_x^2)$$





Exercise 2 (7/10)

- Matlab provides the functions:
- “x=randn(M,N)” that produces a M-by-N matrix with random entries, chosen from a normal distribution with mean zero, variance one and standard deviation one.

- “n=hist(x,M)” that bins the elements of “x” into M equally spaced containers and returns the number of elements in each container “n”. If “x” is a matrix, “hist” works down the columns.

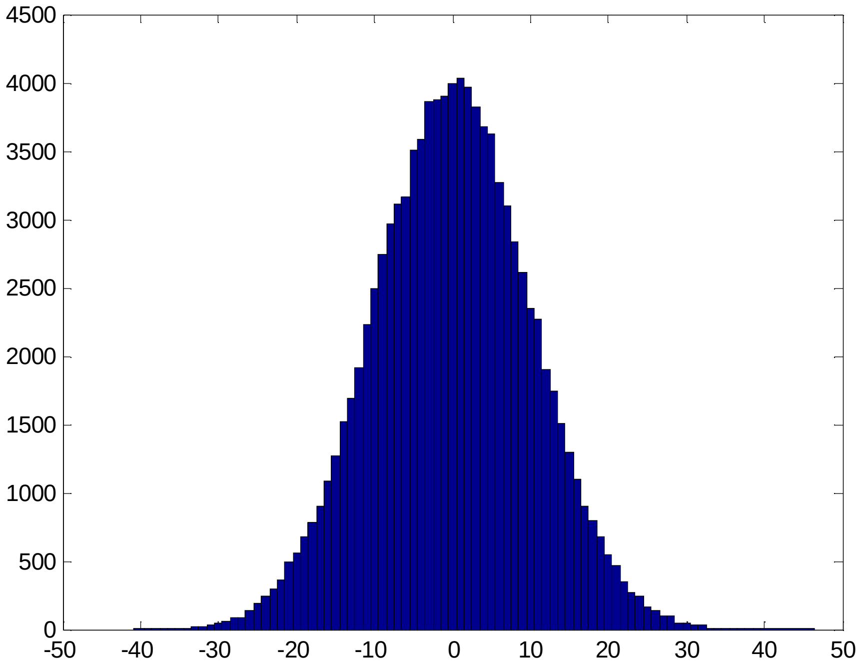
Exercise 2 (8/10)

- HINT: "n=hist(x,b)" where b is a vector, returns the distribution of x among bins with centers specified by b.
- In order to approximate the probability distribution, it is necessary to divide n by the number of trials and the cell dimension:
  - frR=hist(x,b)/N/bin;
  - bar(b,frR);
- Ex 2.b: See the difference between:
  - bar(b,frR);

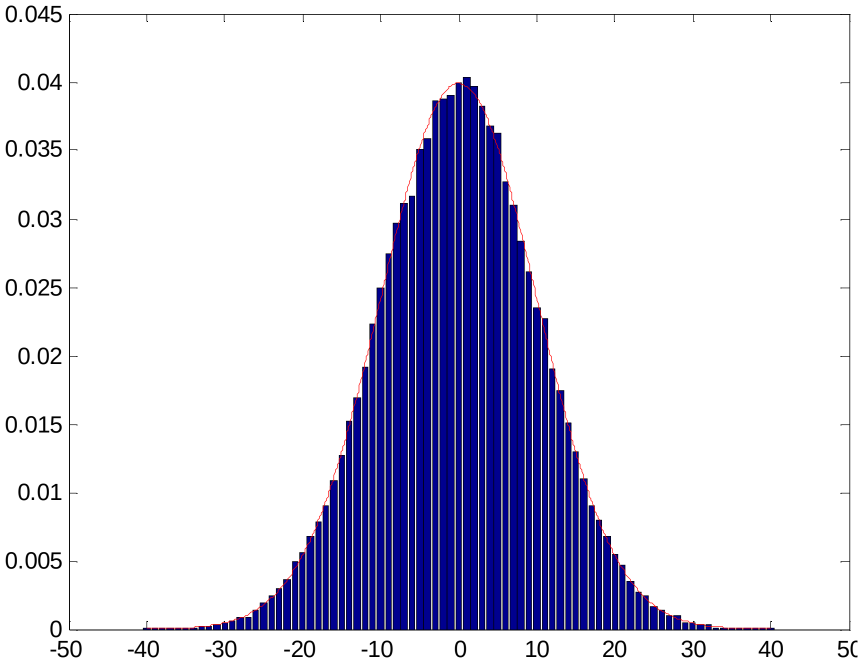
  and
  - hist(x,b)

Exercise 2 (9/10)

hist(x,b)



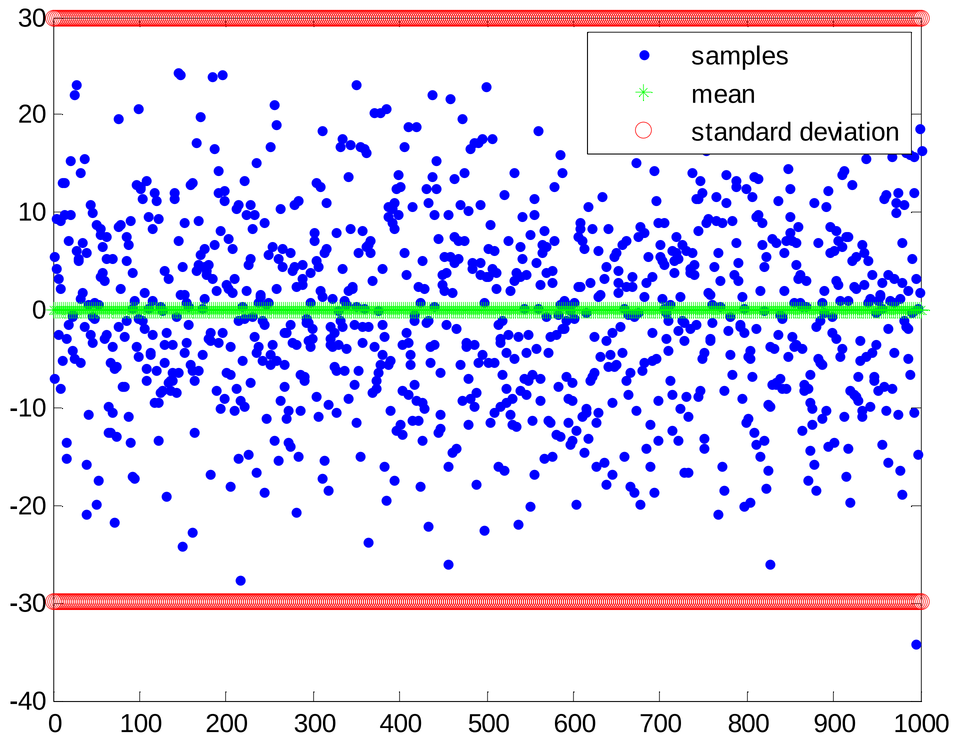
bar(b,frR)



$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(x-m)^2}{2\sigma^2} \right\}$$

Exercise 2 (10/10)

- Estimate mean and standard deviation:
- m = mean (y)
- s = std (y)



- mv = m.*ones(1,N);
- sp = 3*s.* ones(1,N);

- plot([1:N],y,'.',[1:N],mv,'g*',[1:N],mv+sp,'ro',[1:N],mv-sp,'ro')

Exercise 3 (1/6)

- EVEN AND ODD SYNTHESIS: Any arbitrary *real-valued* sequence x(n) can be decomposed into its even and odds components:

$$x(n) = x_e(n) + x_o(n)$$

Where the even and odds parts are given by:

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

Exercise 3 (2/6)

- Remember:
  - A real-valued sequence $x_e(n)$ is called even (symmetric) if:

$$x_e(-n) = x_e(n)$$

  - A real-valued sequence $x_o(n)$ is called odd (antisymmetric) if:

$$x_o(-n) = -x_o(n)$$

Exercise 3 (3/6)

- Design a Matlab function to decompose a given sequence in its even and odd parts:
  - **Function [xe, xo, m] = evenodd(x,n)**
  - m = -fliplr(n);
  - m1 = min([m,n]); m2 = max([m,n]); m = m1:m2;
  - nm = n(1)-m(1); n1 = 1:length(n);
  - x1 = zeros(1,length(m));
  - x1(n1+nm) = x; x = x1;
  - xe = 0.5*(x + fliplr(x));
  - xo = 0.5*(x - fliplr(x));

Exercise 3 (4/6)

□ **EXAMPLE 2.4** Let $x(n) = u(n) - u(n-10)$. Decompose $x(n)$ into even and odd components.

Solution The sequence $x(n)$, which is nonzero over $0 \le n \le 9$, is called a *rectangular pulse*. We will use MATLAB to determine and plot its even and odd parts.

```
>> n = [0:10]; x = stepseq(0,0,10)-stepseq(10,0,10);
>> [xe,xo,m] = evenodd(x,n);
```

Exercise 3 (5/6)

- Step by step:
  - x = stepseq(0,0,10)-stepseq(10,0,10)
    = [1 1 1 1 1 1 1 1 1 1 0]
  - n = [0 1 2 3 4 5 6 7 8 9 10]
  - m = -fliplr(n) = [-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0]
  - m1 = min([m,n]) =-10;
  - m2 = max([m,n]) = 10;
  - m = m1:m2 =-10:10; % new sampling interval

Exercise 3 (6/6)

- Step by step:
  - nm = n(1)-m(1) =10;
  - n1 = 1:length(n) = 1:11;
  - x1 = zeros(1,length(m));   % x must have the same length of m
  - x1(n1+nm) = x;   % x over m starts at nm sample plus 1
  - x = x1 = [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0]   % x over m
  - xe = 0.5*(x + fliplr(x)); $x_e(n) = \frac{1}{2}\left[x(n) + x(-n)\right]$
  - xo = 0.5*(x - fliplr(x)); $x_o(n) = \frac{1}{2}\left[x(n) - x(-n)\right]$

Exercise 4 (1/13)

- Signal Addition: it is a sample by sample addition:

$$\{x_1(n)\}+\{x_2(n)\}=\{x_1(n)+x_2(n)\}$$

- Hint: be carefull with sequnces of unequal lengths or different sample positions:
  - function [y,n] = sigadd(x1,n1,x2,n2)
  - Input: % x1 = first sequence over n1

    % x2 = second sequence over n2
  - Output:

    % y = sum sequence over n, which includes n1 and n2

Exercise 4 (2/13)

function [y,n] = sigadd(x1,n1,x2,n2)

- n = [min(min(n1),min(n2)) : max(max(n1),max(n2))];
  % duration of y(n)
- y1 = zeros(1,length(n));
- y2 = y1; % initialization
- y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
  % x1 with duration of y
- y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
  % x2 with duration of y
- y = y1+y2; % sequence addition

Exercise 4 (3/13)

- Signal Multiplication: it is a sample by sample multiplication:

$$\{x_1(n)\}\cdot\{x_2(n)\} = \{x_1(n)\, x_2(n)\}$$

- Hint: be carefull with sequnces of unequal lengths or different sample positions:
  - function [y,n] = sigmult(x1,n1,x2,n2)
  - Input: % x1 = first sequence over n1

    % x2 = second sequence over n2
  - Output:

    % y = productsequence over n, which includes n1 and n2

Exercise 4 (4/13)

function [y,n] = sigmultd(x1,n1,x2,n2)

- n = [min(min(n1),min(n2)) : max(max(n1),max(n2))];
  % duration of y(n)
- y1 = zeros(1,length(n));
- y2 = y1; % initialization
- y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
  % x1 with duration of y
- y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
  % x2 with duration of y
- y = y1.*y2; % sequence multiplication

Exercise 4 (5/13)

- Scaling: each sample is multiplied by a scalar α:

$$\alpha \cdot \{x_1(n)\} = \{\alpha\, x_1(n)\}$$

- Hint: The sample positions remain the same:
  - y = a*x1
  - N.B. y is defined over n1 as x1

Exercise 4 (6/13)

- Shifting: each sample of x(n) is shifted by k:

$$y(n) = \{x(n-k)\}$$

- It's equivalent to: $y(m+k) = x(m)$
  - The vector x does not change
  - The vector n is changed by adding k to each element

  - function [y,n] = sigshift(x,m,k)
  - n = m+k
  - y = x

Exercise 4 (7/13)

- Folding: each sample of x(n) is flipped around n=0:

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

- Matlab provides the functions:
- "y=fliplr(X)" that returns X with row preserved and columns flipped in the left/right direction.

  - function [y,n] = sigfold(x,n)
  - y = fliplr(x);
  - n = -fliplr(n);

Exercise 4 (8/13)

- Sample summation: it adds all sample values of x(n):

$$y = \sum_{n \in n_1} x(n)$$

- Matlab provides the function:
- “s=sum(X)” that sums the elements of the vector X. If X is a matrix, s is a row vector with the sum over each column.

Exercise 4 (9/13)

- Sample products: it multiplies all sample values of x(n):

$$y = \prod_{n \in n_1} x(n)$$

- Matlab provides the function:
- "p=prod(X)" that multiplies the elements of the vector X. If X is a matrix, p is a row vector with the product over each column.

Exercise 4 (10/13)

- Signal energy: it adds all sample values of x(n):

$$E_x = \sum_{n \in n_1} x(n) \cdot x^*(n) = \sum_{n \in n_1} |x(n)|^2$$

  - Ex=sum(x.*conj(x)) % first way
  - Ex=sum(abs(x).^2) % second way

- Matlab provides the functions:
- "c=conj(X)" that computes the complex conjugate of each element of X: c = real(X) - i*imag(X).
- "a=abs(X)" that computes the absolute value (modulus) of each element of X.

Exercise 4 (11/13)

- Signal power: the average power of a periodic sequence with fundamental period N is given by:

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1}|x(n)|^2$$

  - N=length(x);
  - Px=(sum(abs(x).^2))./N

- Matlab provides the function:
- "l=length(X)" that returns the length of vector X.

Exercise 4 (12/13)

EXAMPLE 2.1 Generate and plot each of the following sequences over the indicated interval.

a. $x(n) = 2\delta(n+2) - \delta(n-4), \quad -5 \le n \le 5.$
b. $x(n) = n\,[u(n) - u(n-10)] + 10e^{-0.3(n-10)}\,[u(n-10) - u(n-20)]\,,$
$0 \le n \le 20.$
c. $x(n) = \cos(0.04\pi n) + 0.2w(n), \; 0 \le n \le 50$, where $w(n)$ is a Gaussian random sequence with zero mean and unit variance.
d. $\tilde{x}(n) = \{\ldots, 5, 4, 3, 2, 1, \underset{\uparrow}{5}, 4, 3, 2, 1, 5, 4, 3, 2, 1, \ldots\}; \; -10 \le n \le 9.$

EXAMPLE 2.2 Let $x(n) = \{1, 2, \underset{\uparrow}{3}, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1\}$. Determine and plot the following sequences.

a. $x_1(n) = 2x(n-5) - 3x(n+4)$
b. $x_2(n) = x(3-n) + x(n)\,x(n-2)$

Exercise 4 (13/13)

- **EXAMPLE 2.3** Generate the complex-valued signal

$$x(n) = e^{(-0.1+j0.3)n}, \quad -10 \le n \le 10$$

and plot its magnitude, phase, the real part, and the imaginary part in four separate subplots.

Solution MATLAB Script

```
>> n = [-10:1:10]; alpha = -0.1+0.3j;
>> x = exp(alpha*n);
>> subplot(2,2,1); stem(n,real(x));title('real part');xlabel('n')
>> subplot(2,2,2); stem(n,imag(x));title('imaginary part');xlabel('n')
>> subplot(2,2,3); stem(n,abs(x));title('magnitude part');xlabel('n')
>> subplot(2,2,4); stem(n,(180/pi)*angle(x));title('phase part');xlabel('n')
```

ANGLE(H) returns the phase angles, in radians, of a matrix with complex elements.

Exeicise 5 (1/13)

- LINEAR TIME INVARIANT SYSTEM

$$x(n) \longrightarrow \boxed{h(n)} \longrightarrow y(n) = x(n) * h(n)$$

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k)$$

the output y(n) is given by the linear convolution between the input x(n) and the system impulse response h(n)

Exeicise 5 (1/13)

- Properties:

$$x_1(n) * x_2(n) = x_1(n) * x_2(n) \quad \text{: Commutation}$$

$$[x_1(n) * x_2(n)] * x_3(n) = x_1(n) * [x_2(n) * x_3(n)] \quad \text{: Association}$$

$$x_1(n) * [x_2(n) + x_3(n)] = x_1(n) * x_2(n) + x_1(n) * x_3(n) \quad \text{: Distribution}$$

$$x(n) * \delta(n - n_0) = x(n - n_0) \quad \text{: Identity}$$

Exeicise 5 (2/13)

- Graphical interpretation: $y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k)$

h(n-k) is the folded and shifted version of h(k).

y(n) is obtained as a sample sum under the overlap of x(k) and h(n-k).

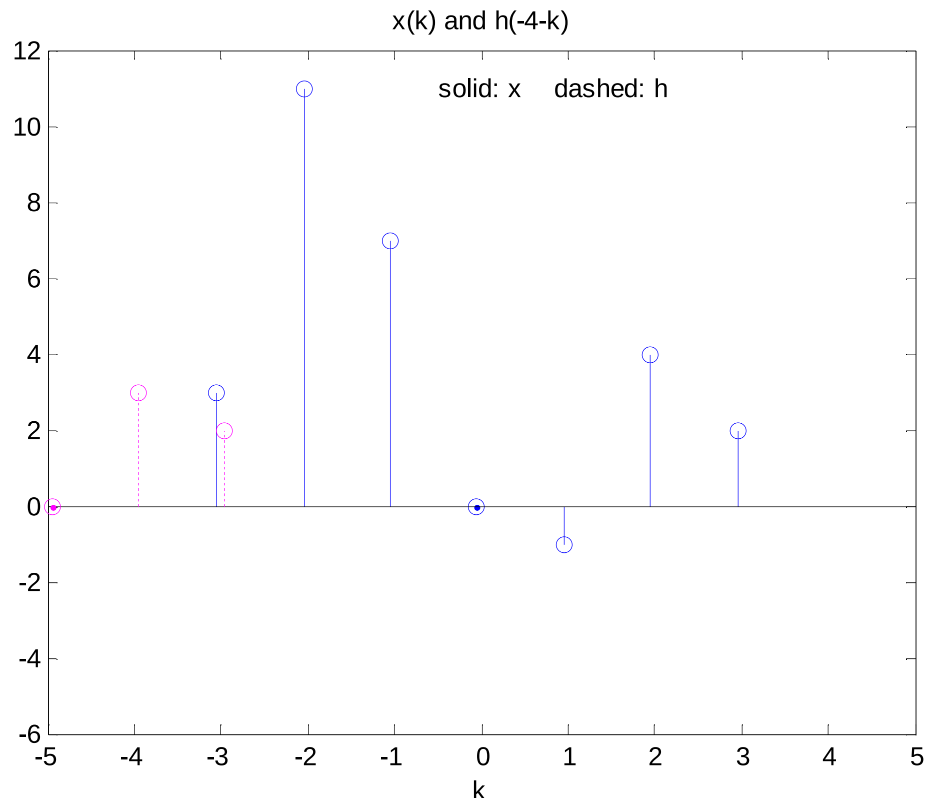
EXAMPLE 2.6 Given the following two sequences

$$x(n) = \left[3, 11, 7, \underset{\uparrow}{0}, -1, 4, 2\right], \quad -3 \le n \le 3; \qquad h(n) = \left[2, \underset{\uparrow}{3}, 0, -5, 2, 1\right], \quad -1 \le n \le 4$$

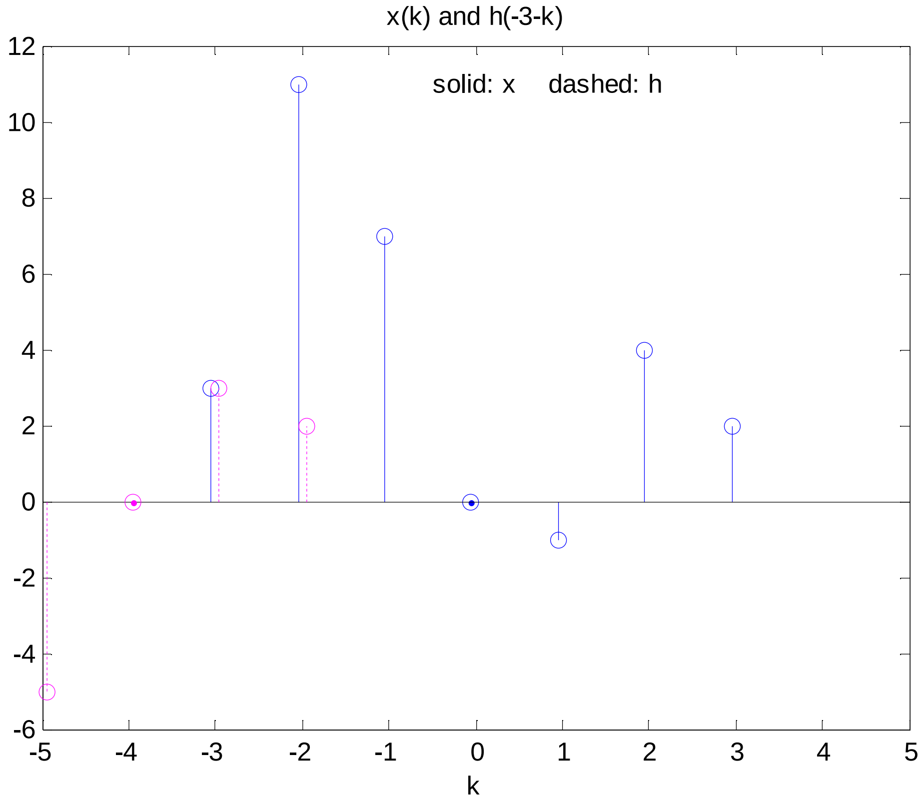
determine the convolution $y(n) = x(n) * h(n)$.

Exeicise 5 (3/13)

$$y(-4)=\sum_{k=-\infty}^{\infty} x(k)\,h(-4-k)=x(-3)h(-4-(-3))=3\cdot 2=6$$

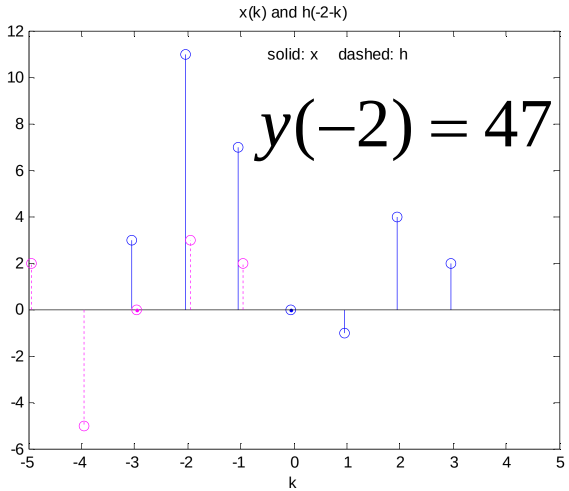


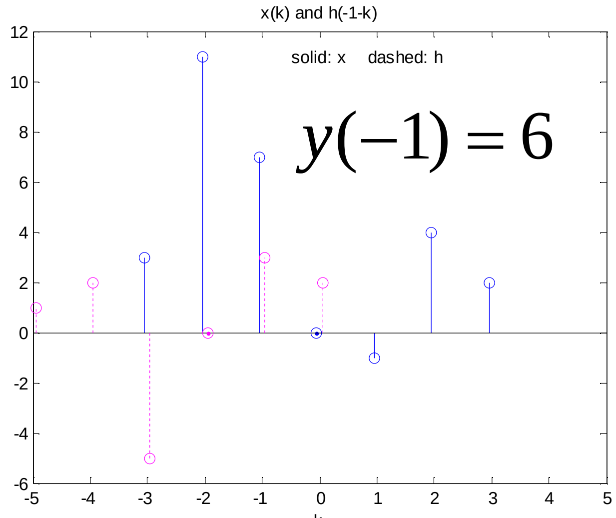
Exeicise 5 (4/13)

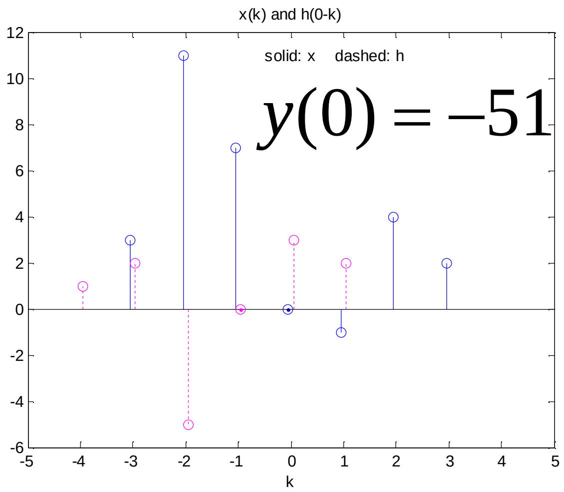


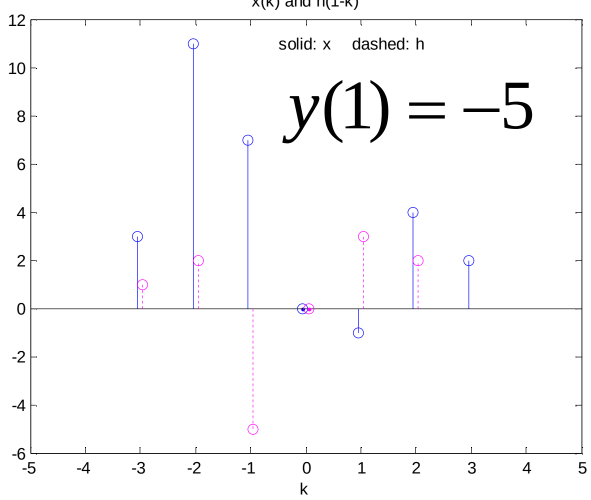
$$y(-3) = x(-3)h(-3-(-3)) + x(-2)h(-3-(-2)) = 3 \cdot 3 + 11 \cdot 2 = 31$$

Exeicise 5 (5/13)





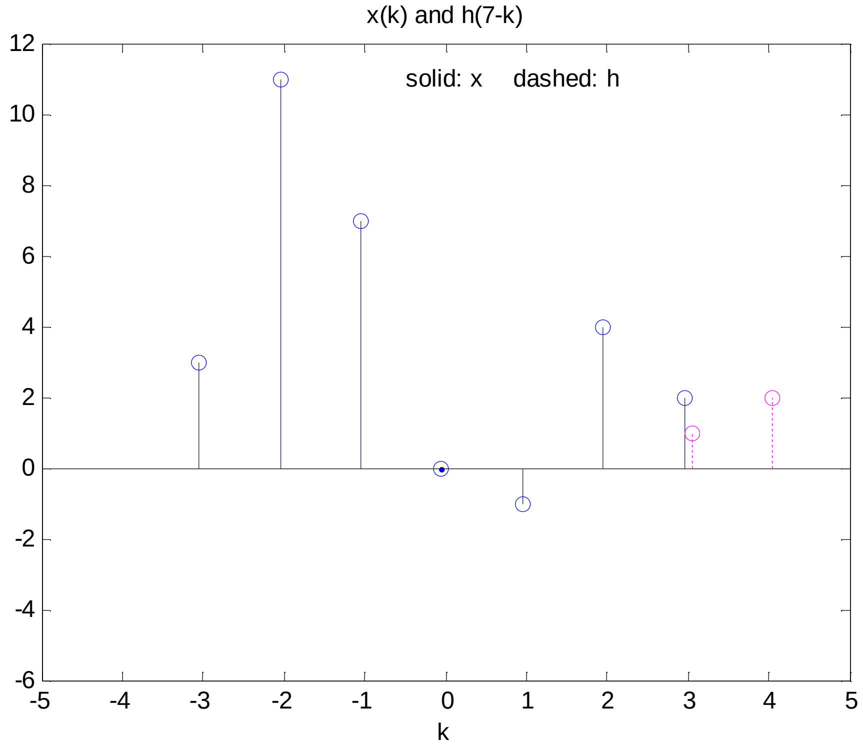




Exeicise 5 (6/13)

... and so on ...

$$y(7) = 2$$



$$y(n) = \left\{ 6, 31, 47, 6, \underset{\uparrow}{-51}, -5, 41, 18, -22, -3, 8, 2 \right\}$$

Exeicise 5 (7/13)

- Goal: design a function that compute the convolution:
- Matlab provides the function:
  - "C = conv(A, B)" that convolves vectors A and B. The resulting vector is length max([length(A)+length(B)-1,length(A),length(B)]).
- However, the conv function assumes that the two sequences begin at n=0 and desn't provides any timing information if the sequences have arbitrary support.

Exeicise 5 (8/13)

- If $\{x(n); n_{xb} \le n \le n_{xe}\}$ and $\{h(n); n_{hb} \le n \le n_{he}\}$
- then $n_{yb} = n_{xb} + n_{hb}$ and $n_{ye} = n_{xe} + n_{he}$

- We can define:
  - **function [y,ny] = conv_m(x,nx,h,nh)**
  - nyb = nx(1)+nh(1);
  - nye = nx(length(x)) + nh(length(h));
  - ny = [nyb:nye];
  - y = conv(x,h);

EXAMPLE 2.7 Perform the convolution in Example 2.6 using the conv_m function.

Exeicise 5 (9/13)

- An alternate method can be used to perform the convolution: a matrix-vector multiplication:

$$\vec{y} = H\vec{x}$$

  - where linear shift in h(n-k) for n=0, ..., Nh-1 are arranged as row in the matrix H
  - length(y)=length(x)+length(h)-1
  - H must be a length(y) x length(x) matrix

Exeicise 5 (10/13)

$$\vec{y} = H\vec{x}$$

- Looking at the figures in the 2.6 example we can say that H must be:

$$H = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 & 0 & 0 & 0 \\ -5 & 0 & 3 & 2 & 0 & 0 & 0 \\ 2 & -5 & 0 & 3 & 2 & 0 & 0 \\ 1 & 2 & -5 & 0 & 3 & 2 & 0 \\ 0 & 1 & 2 & -5 & 0 & 3 & 2 \\ 0 & 0 & 1 & 2 & -5 & 0 & 3 \\ 0 & 0 & 0 & 1 & 2 & -5 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & -5 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$h(n) = \left[2, \underset{\bullet}{3}, 0, -5, 2, 1\right], \quad -1 \leq n \leq 4$$

Exeicise 5 (11/13)

- H is a Toeplitz matrix (each descending diagonal from left to right is constant)

- Matlab provides the function:
  - "toeplitz(C,R)" that buids a non-symmetric Toeplitz matrix having C as its first column and R as its first row.
- We can define a new function:
  - function [y,ny,H]=conv_tp(x,nx,h,nh)
  - that computes the convolution with a matrix-vector multiplication

Exeicise 5 (12/13)

- function [y,ny,H]=conv_tp(x,nx,h,nh)
  - Nx = length(x); Nh = length(h);
  - nyb = nx(1)+nh(1); nye = nx(Nx) + nh(Nh);
  - ny = [nyb:nye];

  - hc=[h; zeros(Nx-1, 1)];
  - hr=[h(1),zeros(1,Nx-1)];
  - H=toeplitz(hc,hr);
  - y=H*x;

Exeicise 5 (13/13)

P2.14 MATLAB provides a function called **toeplitz** to generate a Toeplitz matrix, given the first row and the first column.

a. Using this function and your answer to Problem 2.13 part d, develop an alternate MATLAB function to implement linear convolution. The format of the function should be

```
function [y,H]=conv_tp(h,x)
% Linear Convolution using Toeplitz Matrix
% ----------------------------------------
% [y,H] = conv_tp(h,x)
% y = output sequence in column vector form
% H = Toeplitz matrix corresponding to sequence h so that y = Hx
% h = Impulse response sequence in column vector form
% x = input sequence in column vector form
```

b. Verify your function on the sequences given in Problem 2.13.

Exercise 6 (1/3)

- CORRELATION OF SEQUENCES:
  - Crosscorrelation: it is a measure of the degree to wich two sequences are similar:

$$r_{x,y}(l) = \sum_{n=-\infty}^{\infty} x(n)\, y(n-l) = y(l) * x(-l)$$

  - Autocorrelation: it provides a measure of self-similarity between different alignments of the sequence:

$$r_{x,x}(l) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-l) = x(l) * x(-l)$$

Exercise 6 (2/3)

- The correlation can be computed using the conv function:
  - [x,nx] = sigfold(x,nx);   % obtain x(-n)
  - [rxy,nrxy] = conv_m(y,ny,x,nx);

- Hint: Matlab provides the function:
  - “C = xcorr(A,B)”, where A and B are length M vectors (M>1), that returns the length 2*M-1 cross-correlation sequence C. If A and B are of different length, the shortest one is zero-padded.
  - N.B. There is not timing information!!

Exercise 6 (3/3)

EXAMPLE 2.8 In this example we will demonstrate one application of the crosscorrelation sequence. Let

$$x(n) = [3, 11, 7, \underset{\uparrow}{0}, -1, 4, 2]$$

be a prototype sequence, and let $y(n)$ be its noise-corrupted-and-shifted version

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

where $w(n)$ is Gaussian sequence with mean 0 and variance 1. Compute the crosscorrelation between $y(n)$ and $x(n)$.