Digital Signal Processing
Lab 1: Discrete Time Signals

Matlab basics

The basic elements in Matlab are numbers, variables, and operators. The basic type of variable is the matrix. One-dimensional matrices (row or column matrices) are called vectors.

Examples of variable initialization:

```
a=3                         % is a scalar
a=[3]                       % is a scalar
x=[1 2 3]                   % is a row vector
y=[1; 2; 3]                 % is a column vector
A=[1 2 3; 4 5 6; 7 8 9]     % is a matrix
```

Note that

- Discrete-time sequences are represented in Matlab by vectors (row or column matrices);
- Numbers are represented with high precision and can be complex;
- Indexes of matrixes always start from 1 and not from 0.

The colon operator : can be used to generate sequences of numbers in a given range.

```
>> a=1:10
a =
     1     2     3     4     5     6     7     8     9    10
>> b=2:3:10   % from 2 to 10 with step 3
b =
     2     5     8
```

The colon operator is also used to access certain row/column elements of a matrix. `B=A(:,1)` is the subset of `A` given by "column 1, all the rows"; the meaning of `A(:)` is "all the elements of `A`, read column by column and regarded as a single column". See "`help :`" at the prompt of Matlab for further details.

```
>> A=[1 2 3; 4 5 6; 7 8 9];
>> A(2:3,:)
ans =
     4     5     6
     7     8     9
```

The command `size(A)` provides the size, in rows and columns, of the matrix `A`, while `length(a)` returns the length of a vector `a`.

MATLAB provides many useful functions to create special matrices. These include:

- `zeros(M,N)` for creating a matrix of all zeros,
- `ones(M,N)` for creating matrix of all ones,
- `eye(N)` for creating an N×N identity matrix.

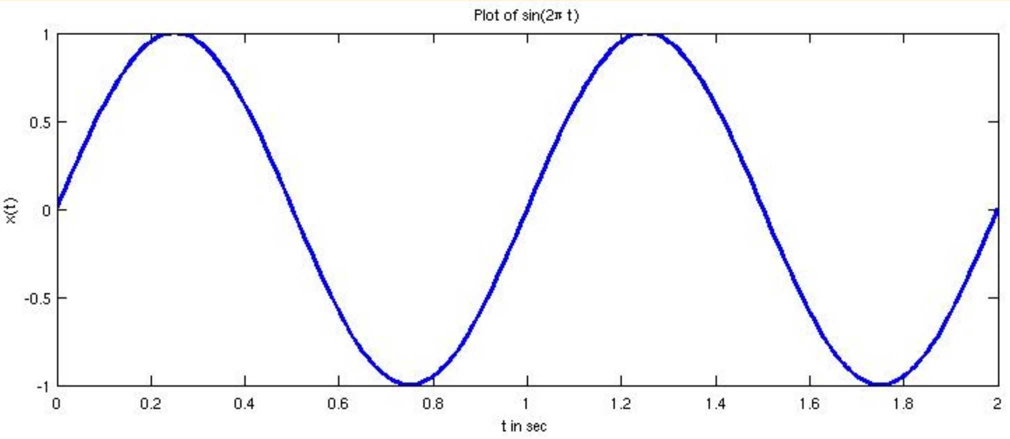
Scripts and functions

A **script** is a text file that contains a sequence of commands, one on each line of the file, that have to be executed one after the other as though they were typed at the command prompt.
Functions implement new routines, characterized by input and output parameters. A function file begins with the keyword `function` followed by an output-input variable declaration.
Both scripts and functions files must have a .m extension (*m-files*) and must be in the current directory on in a directory of the path environment of Matlab.

When time axes other than the sequence 1,2,3,...,N (or 0,1,2,...,N-1 with a unitary shift of the index) need to be represented for the sequence `x`, a time-axis vector `t` or a sample-index vector `n` will be associated to the vector `x`.

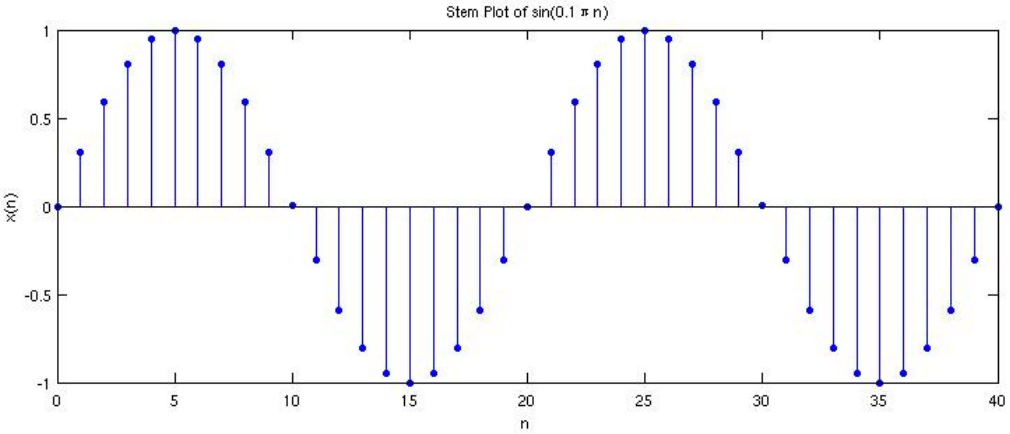
The basic plotting command is `plot(t,x)`, which generates a plot of `x` values versus `t` values.
The following set of commands creates a list of sample points, evaluates the sine function at those points, and then generates a plot of a simple sinusoidal wave, putting axis labels and title on the plot.

```
t = 0:0.01:2;          % sample points from 0 to 2 in steps of 0.01
x = sin(2*pi*t);       % Evaluate sin(2πt)
plot(t,x,'b');         % Create plot with blue line
xlabel('t in sec'); ylabel('x(t)');    % Label axes
title('Plot of sin(2\pi t)');          % Title plot
```



For plotting a discrete-time sequence we will use the `stem` command which displays each data value with a small circle at the end of a line connecting it to the horizontal axis.

```
n = 0:1:40;                         % sample index from 0 to 40
x = sin(0.1*pi*n);                  % Evaluate sin(0.1πn)
stem(n,x);                          % Stem-plot
xlabel('n'); ylabel('x(n)');        % Label axes
title('Stem Plot of sin(0.1 \pi n)');       % Title plot
```



If we want to create multiple plots within a graphical window, we can use the function `subplot`. `subplot(r,c,i)` splits the current graphical windows into a matrix of subplots (`r` rows by `c` columns) and activate the i-th subplot.

```
figure              % open new figure window
subplot(2,1,1)      % split into one column of two subplots and
                    % select the first one


plot(n,x,'-o'); % plot with line-circle style
subplot(2,1,2)  % select the second subplot
stem(n,x)       % stem the same x sequence
```

The following commands generate a **real-valued exponential sequence** $x[n] = (0.9)^n$, $0 \leq n \leq 100$

```
>> n = [0:100]; x = (0.9).^n;
```

while a **complex-valued exponential sequence** $x[n] = \exp[(2 + j3)n]$, $0 \leq n \leq 100$ is generated by

```
>> n = [0:100]; x = exp((2+3j)*n);
```

A sum of **sinusoidal signals** $x[n] = 3\cos(0.1\pi n + \pi/3) + 2\sin(0.5\pi n)$ is obtained with

```
>> n = [0:100]; x = 3*cos(0.1*pi*n+pi/3) + 2*sin(0.5*pi*n);
```

The `rand(1,N)` generates a length N **random sequence** whose elements are **uniformly** distributed between [0, 1]. The `randn(1,N)` generates a length N **Gaussian random sequence** with mean 0 and variance 1.

To generate P periods of a **periodic sequence** $\tilde{x}[n]$ from one period $\{x[n], 0 \leq n \leq N-1\}$, we can copy x[n] P times:

```
>> xtilde = [x,x,x,x,x,x];
```

but a more efficient way is to use the operator (:) on a matrix with columns that replicate the sequence `x'`

```
>> xtilde = x' * ones(1,P);     % P columns of x; x is a row vector
>> xtilde = xtilde(:);          % long column vector
>> xtilde = xtilde';            % long row vector
```

Basic sequences and operations

We define now a set of functions (from *Digital Signal Processing Using Matlab* by Vinay K. Ingle, John G. Proakis).

To implement a **unit sample sequence**

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

over the $n1 \leq n0 \leq n2$ interval, we will use the following `impseq` function

```
function [x,n] = impseq(n0,n1,n2)
% Generates x(n) = delta(n-n0); n1 <= n <= n2
% ----------------------------------------------
% [x,n] = impseq(n0,n1,n2)
%
n = [n1:n2]; x = [(n-n0) == 0];
```

To implement a **unit step sequence**

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

over the $n_1 \leq n_0 \leq n_2$ interval, we will use the following `stepseq` function

```
function [x,n] = stepseq(n0,n1,n2)
% Generates x(n) = u(n-n0); n1 <= n <= n2
% ------------------------------------------
% [x,n] = stepseq(n0,n1,n2)
%
n = [n1:n2]; x = [(n-n0) >= 0];
```

The following function `sigadd` implements the **sample-by-sample addition** of sequences. This operation requires that the sequences be defined over a common sample interval represented by `n`. This is obtained using the logical operation of intersection "&", relational operations like "<=" and "==", and the `find` function.

```
function [y,n] = sigadd(x1,n1,x2,n2)
% implements y(n) = x1(n)+x2(n)
% -----------------------------
% [y,n] = sigadd(x1,n1,x2,n2)
%
y = sum sequence over n, which includes n1 and n2
% x1 = first sequence over n1
% x2 = second sequence over n2 (n2 can be different from n1)
%
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
```

A similar function `sigmult` implements the **sample-by-sample multiplication** (with the operator "`.*`")

```
function [y,n] = sigmult(x1,n1,x2,n2)
% implements y(n) = x1(n)*x2(n)
% -----------------------------
% [y,n] = sigmult(x1,n1,x2,n2)
%
y = product sequence over n, which includes n1 and n2
% x1 = first sequence over n1
% x2 = second sequence over n2 (n2 can be different from n1)
%
n = min(min(n1),min(n2)):max(max(n1),max(n2));  % duration of y(n)
y1 = zeros(1,length(n)); y2 = y1;
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
```

The **shifting** function `sigshift` has no effect on the vector `x`, but the support vector `n` is changed by adding `k` to each element.

```
function [y,n] = sigshift(x,m,k)
% implements y(n) = x(n-k)
% -------------------------
% [y,n] = sigshift(x,m,k)
%
n = m+k; y = x;
```

The `sigfold` function implements the sign-reversal of the sample indexes `n` to get y[n]=x[-n]

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

The summation of the sample values of x[n] between n_1 and n_2 is obtained with

```
>> sum(x(n1:n2))
```

while the product of the sample values of x[n] between n_1 and n_2 is obtained with

```
>> prod(x(n1:n2))
```

The energy of a sequence x[n] is given by

$$E_x = \sum_{-\infty}^{\infty} x[n]\, x^*[n] = \sum_{-\infty}^{\infty} |\, x[n]\, |^2$$

For a finite-duration sequence it can be computed in Matlab with

```
>> Ex = sum(x .* conj(x)); % one approach
>> Ex = sum(abs(x) .^ 2); % another approach
```

Any arbitrary real-valued sequence x[n] can be decomposed into its even and odd components. This decomposition is implemented by the following `evenodd` function

```
function [xe, xo, m] = evenodd(x,n)
% Real signal decomposition into even and odd parts
% -------------------------------------------------
% [xe, xo, m] = evenodd(x,n)
%
if any(imag(x) ~= 0)
error('x is not a real sequence')
end
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));
```

Examples:

1. Generate and plot the sequence $x[n] = \delta[n+2] - 2\delta[n-3], -5 \le n \le 5$

```
n = [-5:5];
x = impseq(-2,-5,5) - 2*impseq(3,-5,5);
stem(n,x); title('Sequence in Example 1')
xlabel('n'); ylabel('x(n)');
```

2. Generate and plot the sequence
$x[n] = n\,(u[n] - u[n-10]) + 10e^{-0.3(n-10)}\,(u[n-10] - u[n-20]), 0 \le n \le 20$

```
n = [0:20];
x1 = n.*(stepseq(0,0,20)-stepseq(10,0,20));
x2 = 10*exp(-0.3*(n-10)).*(stepseq(10,0,20)-stepseq(20,0,20));
x = x1+x2;
stem(n,x); title('Sequence in Example 2')
xlabel('n'); ylabel('x(n)');
```

3. Generate and plot the sinusoidal sequence
$x[n] = 3\cos(0.04\pi n), 0 \le n \le 200$

```
n = [0:200];
x = 3*cos(0.04*pi*n);
stem(n,x); title('Sequence in Example 3')
xlabel('n'); ylabel('x(n)');
```

4. Generate and plot, for $-30 \le n \le 29$, the periodical sequence $\tilde{x}[n]$ obtained from one period
$x[n]=n+1, 0 \le n \le 9$

```
n = [-30:29];
x = [1:10];
xtilde = x' * ones(1,6);          % one period in each column
xtilde = (xtilde(:))';            % reshape into one single row
stem(n,xtilde); title('Sequence in Example 4')
xlabel('n'); ylabel('x(n)');
```

5. Let $x[n] = \{1, 2, \underline{3}, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1\}$, where the underline denotes the sample at n=0.
Determine and plot the following sequences

   A. $x_1[n] = 2x[n-5] - 3x[n+4]$
   B. $x_2[n] = x[3-n] + x[n]\,x[n-2]$

```
% A.
n = -2:10; x = [1:7,6:-1:1];
```

```
[x11,n11] = sigshift(x,n,5);
[x12,n12] = sigshift(x,n,-4);
[x1,n1] = sigadd(2*x11,n11,-3*x12,n12);
stem(n1,x1); title('Sequence in Example 5.A')
xlabel('n'); ylabel('x1(n)');
```

```
% B.
n = -2:10; x = [1:7,6:-1:1];

[x21,n21] = sigfold(x,n);
[x21,n21] = sigshift(x21,n21,3);
[x22,n22] = sigshift(x,n,2);
[x22,n22] = sigmult(x,n,x22,n22);
[x2,n2] = sigadd(x21,n21,x22,n22);

stem(n2,x2); title('Sequence in Example 5.B')
xlabel('n'); ylabel('x2(n)');
```

6. Generate and plot the sequence

$$x[n] = \sum_{k=-5}^{5} e^{|k|}\,\delta[n-2k], \qquad -10 \le n \le 10$$

```
n = [-10:10]; x = zeros(1,length(n));
for k = -5:5
   x = x + exp(-abs(k))*impseq(2*k ,-10,10);
end
stem(n,x); title('Sequence in Example 6')
xlabel('n'); ylabel('x(n)');
```

7. Generate the complex-valued signal $x[n] = e^{(-0.05+j0.3)n}$, $-30 \le n \le 30$ and plot its magnitude, phase, the real part, and the imaginary part in four separate subplots

```
n = [-30:1:30]; alpha = -0.05+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'); xlabel('n')
```

```
subplot(2,2,4);
stem(n,(180/pi)*angle(x)); title('Phase'); xlabel('n')
```

Compare the obtained plots with the graphical representation in the complex plane produced by

```
figure; polar(angle(x),abs(x),'-o')
```

8. Decompose $x[n] = u[n] - u[n-10]$ into its even and odd components

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

subplot(2,2,1); stem(n,x); title('Rectangular pulse')
xlabel('n'); ylabel('x(n)'); axis([-10,10,0,1.2])
subplot(2,2,2); stem(m,xe); title('Even Part')
xlabel('n'); ylabel('xe(n)'); axis([-10,10,0,1.2])
subplot(2,2,4); stem(m,xo); title('Odd Part')
xlabel('n'); ylabel('xe(n)'); axis([-10,10,-0.6,0.6])
```

Exercises

1. Generate and plot the following sequences

$x_1[n] = 3\delta[n+2] + 2\delta[n] - \delta[n-3] + 5\delta[n-7], \quad -5 \le n \le 15$

$x_2[n] = 10u[n] - 5u[n-5] - 10u[n-10] + 5u[n-15]$

$x_3[n] = e^{0.1n}(u[n+20] - u[n-10])$

$x_4[n] = 2(\cos(0.49\pi n) + \cos(0.51\pi n)), \quad -200 \le n \le 200$

$x_5[n] = 3\sin(0.01\pi n) \cdot \cos(0.5\pi n), \quad -200 \le n \le 200$

$x_6[n] = e^{-0.05n} \cdot \sin(0.1\pi n + \pi/3), \quad 0 \le n \le 100$

$x_7[n] = \{\ldots, 3, 1, 0, -1, 2, \ldots\}$ periodic (plot 10 periods)

2. Let $x[n] = \{2, 4, -3, \underline{1}, -5, 4, 7\}$ (underline indicates n=0). Generate and plot the following sequences

$x_1[n] = 2x[n-3] + 3x[n+4] - x[n]$

$x_2[n] = x[n+3]x[n-2] + x[1-n]x[n+1]$

$x_3[n] = 2e^{0.5n}x[n] + \cos(0.1\pi n) \cdot x[n+2], \qquad -10 \le n \le 10$

3. Decompose the following sequences into their even and odd components and plot them

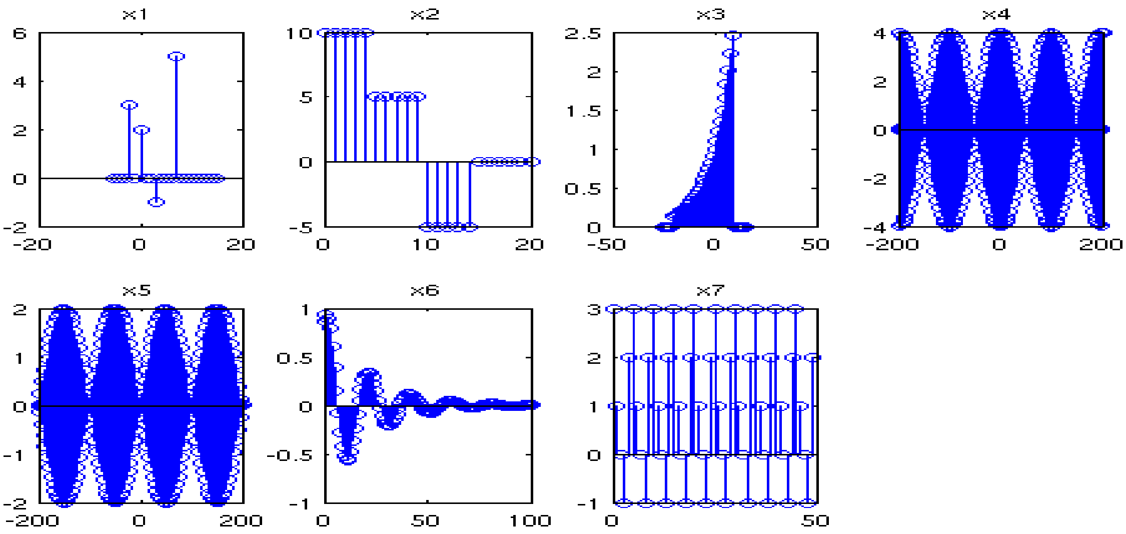
$x_1[n] = \{\underline{0}, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$x_2[n] = e^{0.1n} \cdot (u[n+5] - u[n-10])$

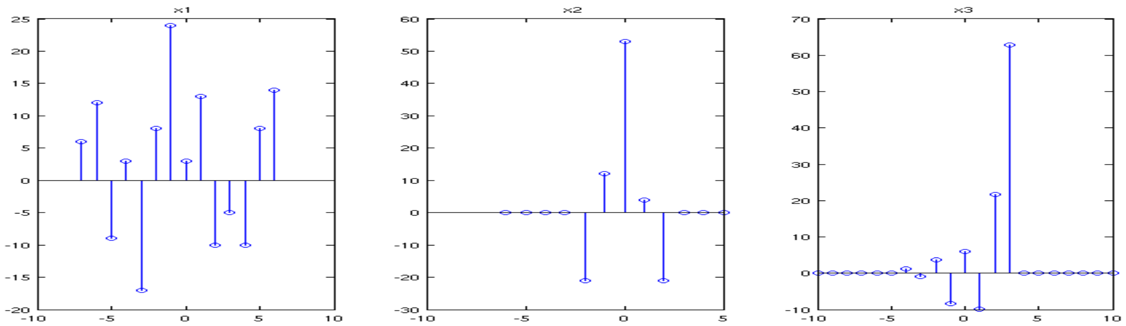
$x_3[n] = \cos(0.2\pi n + \pi/4), \quad -20 \le n \le 20$

$x_4[n] = e^{-0.05n} \cdot \sin(0.1\pi n + \pi/3) \cdot u[n]$

Plots of exercise 1.



Plots of exercise 2



Plots of exercise 3