EBU4375 Signals and Systems: Working with signals in Matlab

Dr Jesús Requena Carrión

Queen Mary
University of London

Working with signals in computing environments

There exist several numerical computing environments that can be used for Signals and Systems, such as Matlab or Python.

In this module, we will use Matlab:

- Matlab is a numerical computing environment that allows vector and matrix manipulations, representation of data and implementation of algorithms.
- We will use Matlab to define, plot and manipulate signals.

Representing signals with Matlab

Matlab uses **vectors** to represent signals. A vector is defined by placing a sequence of numbers within square braces:

```
>> v = [1 2 3 4]

v =

     1     2     3     4

>> w = [0.1 0.2 0.3 0.4]

w =

    0.1000    0.2000    0.3000    0.4000
```

Representing signals with Matlab

To represent a signal in Matlab, we use:

- One vector for representing time.
- One vector for representing the value of the signal.

Once defined, a signal can be plotted by using the command `stem` (for DT signals) and `plot` (for CT signals).

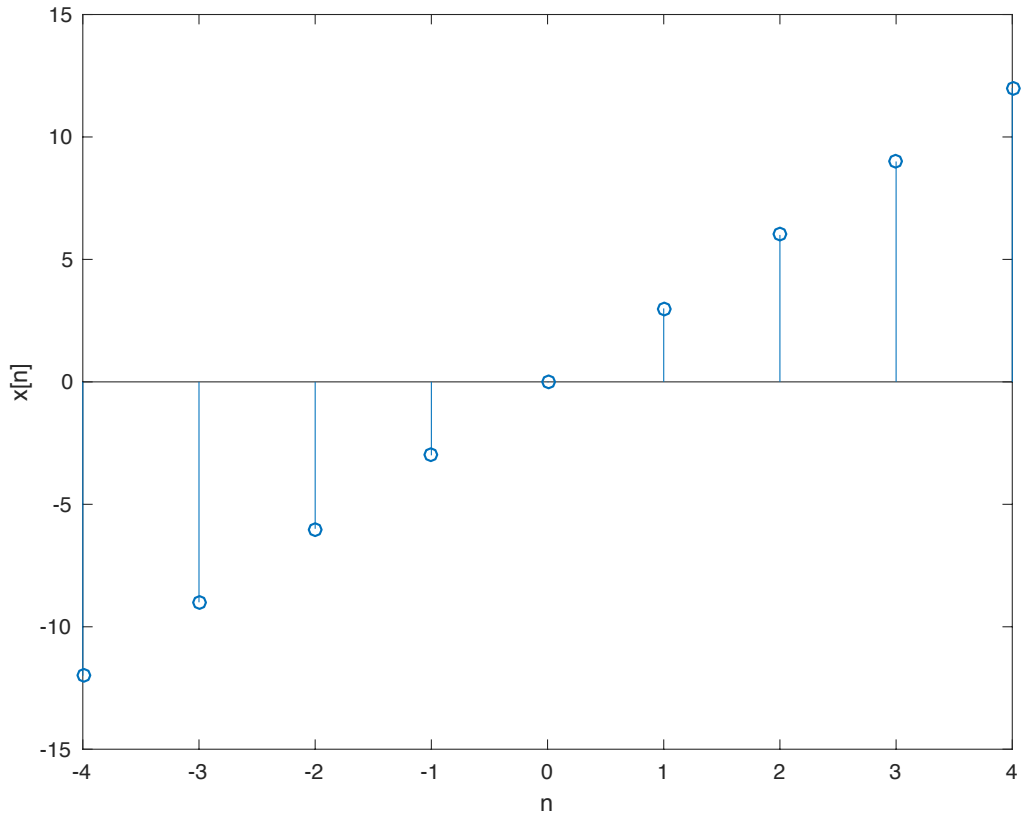
The following is a Matlab script that defines and represent the DT signal $x[n] = 3n$ in the time interval $-4 \leq n \leq 4$:

```
n = [-4 -3 -2 -1 0 1 2 3 4]; % Variable n denotes time
x = 3*n; % Variable x is the signal value

stem(n,x) % plots x agains n
xlabel('n') % adds text below the X-axis
ylabel('x[n]') % adds text beside the Y-axis
```

Representing signals with Matlab

The result of the previous Matlab script is the following figure:



Representing signals with Matlab

In order for us to represent a CT signal $x(t)$ in Matlab, we would need to define a vector `t` that contains *all the values* of the independent variable. This is impossible.

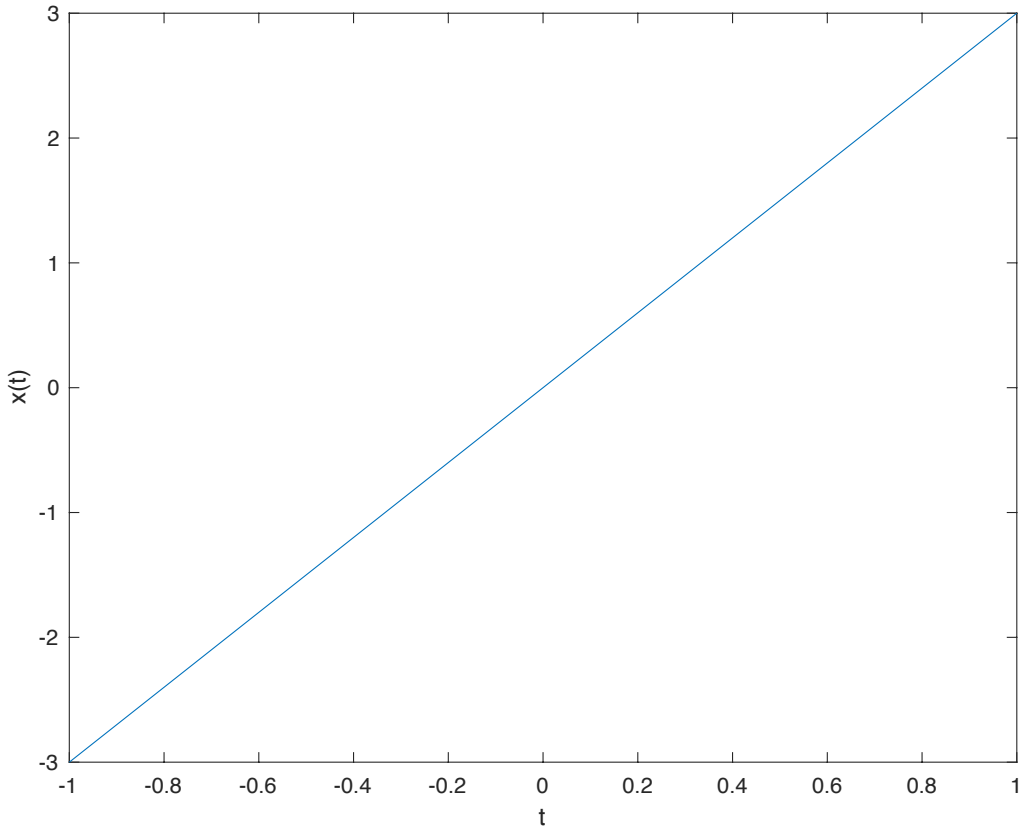
Instead, we only use the values at a finite number of time instants, which we call **samples**. For instance, we can define and represent the signal $x(t) = 3t$ by taking small steps in time, say $\Delta t = 0.25$:

```
t = [-1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 1]; % Variable t ...
    denotes time
x = 3*t; % Variable x is the signal value

plot(t,x) % plots x agains n
xlabel('t') % adds text below the X-axis
ylabel('x(t)') % adds text beside the Y-axis
```

Representing signals with Matlab

The CT signal $x(t)$ is plotted as follows:



Basic DT signals in Matlab

In Matlab, a useful way of defining the time corresponding to DT signals is using

```
n = n_s:1:n_e
```

where `n_s` is the start of the sequence and `n_e` is the end of the sequence.

For instance:

```
>>  n = -2:1:8

n = -2   -1   0    1    2    3    4    5    6    7    8
```

Basic DT signals in Matlab

Here are several examples of DT signals in Matlab. As an exercise, define them and plot them by using the `stem` command.

```
n_s=0;
n_e=20;
n = n_s:1:n_e; % Time

x1 = exp(-0.2*n); % exponential signal
x2 = cos(2*pi*n/10); % sinusoidal signal
x3 = exp(j*2*pi*n/10); % complex exponential signal

x3r = real(x3); % real part of x3
x3i = imag(x3); % imaginary part of x3
x3a = abs(x3); % magnitude of x3
x3p = angle(x3); % phase of x3
```

Basic CT signals in Matlab

We will define the independent variable of CT signals in Matlab by using

```
t = t_s:dt:t_e
```

where `dt` is the time difference between two consecutive time instants, and t_s and t_e are, respectivey, the first and last time instants.

For instance:

```
>>  t = -1:0.5:1

t = -1.0000    -0.5000     0     0.5000     1.0000
```

Basic CT signals in Matlab

Here are several examples of CT signals in Matlab. As an exercise, define them and plot them by using the `plot` command.

```
t_s=0;
t_e=20;
dt=0.001;
t = t_s:dt:t_e; % Time

x1 = exp(-0.2*t); % exponential signal
x2 = cos(2*pi*t/10); % sinusoidal signal
x3 = exp(j*2*pi*t/10); % complex exponential signal

x3r = real(x3); % real part of x3
x3i = imag(x3): % imaginary part of x3
x3a = abs(x3); % magnitude of x3
x3p = angle(x3): % angle of x3
```

Basic operations with signals in Matlab

- Operating with signals in Matlab means operating with the vectors that represent them.
- Mathematically, signals extend from $-\infty$ to ∞. However, in Matlab we can only represent a **finite number of samples**.

The following are some examples of operations with DT signals in Matlab:

```
n = -10:1:10; % definition of n
x = ones(size(n)); % x is a vector with the same size as n ...
    and all ones
y = 2*x; % y is a scaled version of x
z = x + y; % z is the sum of x and y
v = y.*z ; % v is the product of x and y, DO NOT FORGET THE DOT!
w = zeros(size(n)); % w is a vector with the same size as n ...
    and all zeros
w(11:end) = 1; % The samples 11 to 21 of w are set to 1
```

Plot `x`, `y`, `z`, `v` and `w` and make sure you understand these operations.

Computations with DT signals in Matlab

Calculating the area, average value, energy and mean power of DT signals involves adding samples or the square of the samples.

In Matlab, we use `sum` to a add up the samples in a vector, `length` to obtain the number of samples in a vector and `x.^2` to square each sample in `x`. Here is how you calculate area, average value, energy and mean power of a DT signal `x`:

```
Ar_x = sum(x); % Area of x
Av_x = sum(x)/length(x); % Average value of x
E_x = sum(x.^2); % Energy of x, DO NOT FORGET THE DOT!
P_x = sum(x.^2)/length(x); % Average value of x, DO NOT ...
    FORGET THE DOT!
```

Apply these operations on the examples that we have seen before and compare the numerical result with the theoretical one.

Computations with CT signals in Matlab

Calculating the area, average value, energy and mean power of CT signals involves integrating. In Matlab we only represent a finite number of samples and the integral will be approximated by a sum. Given a CT signal `x` in Matlab and a time step between `dt`, area, average value, energy and mean power can be calculated as follows:

```
Ar_x = sum(x)*dt; % Area of x
Av_x = (sum(x)*dt)/(length(x)*dt); % Average value of x
E_x = sum(x.^2)*dt; % Energy of x, DO NOT FORGET THE DOT!
P_x = (sum(x.^2)*dt)/(length(x)*dt); % Average value of x, ...
    DO NOT FORGET THE DOT!
```

Apply these operations to the CT signals that we have defined in our previous slides and compare the obtained values with the theoretical ones.