Signals and Systems I (2016506)
Faculty of Engineering
Department of Electrical and Electronics Engineering

Lecture 2 - Signals (2)

1. Basic operations with continuous-time signals

Prelude: an efficient way to deal with signal transformations is to add *local functions* to scripts. Local functions are useful if you want to reuse code within a program. They are only visible within the script they are defined, both to the code and other local functions within the file. They are equivalent to subroutines in other programming languages, and are sometimes called *subfunctions*. For example, to declare a local function (that computes the sine and cosine of the input argument) within a Matlab script, you might add the following lines to your program:

```
function [Out1,Out2] = function_name(input)
     Out1 = sin(input);
     Out2 = cos(input);
end
```

Once the local function is defined, you may call it later in the code. Functions inside the script must always be placed at the end.[a]

```
// Vector of time of 100 samples between 0 and 1
t=linspace(0,1,100);
// Computation of x1=sin(t) and x2=cos(t) using the previously declared
    local function
[x1 x2]=function_name(t);
```

[a] When using Matlab versions prior to R2016b, the function must be declared inside a dedicated script named with the same name. In order to use the function later on, the new script must reside in the same folder as the function. If the function is to be used in the command window, the path must contain the function. Evidently, this will be supported too by newer versions.

a) **Implement a function in Matlab for the unit-step signal. Save the function as a .m file (Matlab script file extension). Employ this function to plot the unit-pulse signal**

$$x_1(t) = \Pi(t) = u\left(t + \frac{1}{2}\right) - u\left(t - \frac{1}{2}\right) \tag{1}$$

in the interval $-2 \leq t \leq 2$ using a time increment of $\Delta t = 0.001$ s

Note: call this function `ctstep` and save it as a separate .m file for later use in the course.

b) **Implement a function in Matlab for the unit-ramp function. Save the function as a .m file (Matlab script file extension). Employ this function to plot the unit-triangle signal**

$$x_2(t) = \Lambda(t) = r(t+1) - 2r(t) + r(t-1) \tag{2}$$

in the interval $-2 \leq t \leq 2$ using a time increment of $\Delta t = 0.001$ s

Note: call this function `ctramp` and save it as a separate .m file for later use in the course.

c) **Let $x_3(t) = e^{-t}\cos(10t)\left[u(t) - u(t-3)\right]$. Graph the following signals in the interval $-7 \leq t \leq 7$ using a time increment of $\Delta t = 0.001$ s:**

i. $x_3(t)$ (in all plots but with a different color)

ii. $x_3(0.5t)$ (in the same window but different plot)

iii. $x_3(t+2)$ (in the same window but different plot)

iv. $x_3(-t)$ (in the same window but different plot)

v. $x_3(2t-1)$ (in the same window but different plot)

Hint: to create multiple plots within a window you may use the function `subplot`. To hold a plot within a figure you can use the command `hold on`, on the other hand, it is also possible to plot different signals using the same statement, e.g. plot(t,sig1,t,sig2). (type help plot in the command window for more information). Finally the use of title, xlabel, ylabel, and legend will make the chart more readable.

2. **Basic operations with discrete-time signals**

   a) **Implement a function in Matlab for the Kronecker delta unit-impulse, the unit-step and the unit-ramp functions. Use the definitions to plot**

$$x_1[n] = \delta[n+2] + u[n] + r[n-1] - 2r[n-2] + r[n-4] \qquad (3)$$

for the range of the sample index $-3 \leq n \leq 6$.

   b) **Let $x_2[n] = (1.2)^n (u[n+3] - u[n-7])$. Graph the following signals in the interval $-13 \leq n \leq 13$:**

i. $x_2[n]$ (in all plots but with a different color)

ii. $x_2[-(n+1)]$ (in the same window but different plot)

iii. $x_2[3n]$ (in the same window but different plot) **(Hint:** see Intermezzo**)**

iv. $x_2\left[\frac{n}{2}\right]$ (in the same window but different plot) **(Hint:** see Intermezzo**)**[1]

Intermezzo: the time index of a DT signal *must* be an integer. When we scale a DT signal $x[n]$ by a factor a such that $|a| > 1$ (i.e., we perform the signal transformation $x[an]$), every ath sample of the original signal is retained. The values of the signals between a samples are discarded. This phenomenon is known as *downsampling* and is illustrated in Figure 1 for $a = 3$.

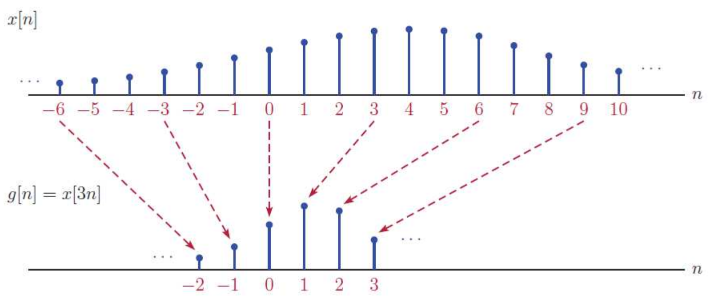

Figure 1: Downsampling.

On the other hand, if we scale a DT signal $x[n]$ by a factor $1/a$ such that $|1/a| < 1$ (i.e., we perform the signal transformation $x\left[\frac{n}{a}\right]$), the relationship between the signals $x\left[\frac{n}{a}\right]$ and $x[n]$ will be defined only for values of n that make $\frac{n}{a}$ an integer. Thus:

$$x\left[\frac{n}{a}\right] = \begin{cases} x\left[\frac{n}{a}\right] & \text{if } n/a \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

1 Notice that discrete signals are only defined for integer arguments, we can discard non-defined elements using the vector (rem(n/a,1) == 0) to determine wheter the entry in the index vector (n) is integer or not, and a .* multiplication by the signal array will discard non-defined values. In this case, rem returns the remainder of the division of n/a by 1.

This is known as *upsampling*. This is shown in Figure 2 for $a = 2$.

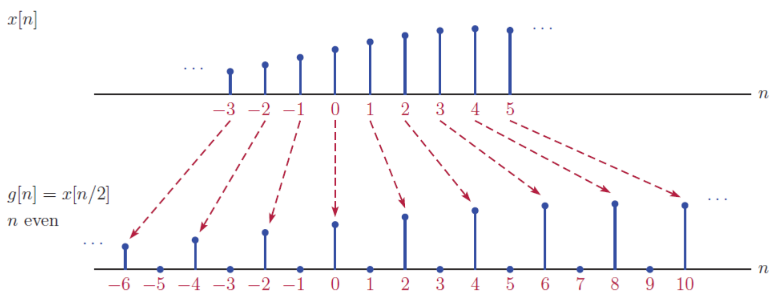

Figure 2: Upsampling.