ETI 2506: TELECOMMUNICATION SYSTEMS LABS

Laboratory Exercise 2: Decimation and Expansion

Decimation

The decimation operation is performed on a discrete signal to reduce the number of samples. It is required to extract a sample each N samples of the initial signal, to form a "shorter" discrete sequence, within the same domain range. This operation may be necessary to align to the same domain (i.e. the same number and distance between values of the time domain) for signals with different resolution. If the initial signal is the discrete representation of a real continuous signal (as we do in Matlab), this operation can represent the sampling of a real signal *x(t)* to form a discrete sequence of its samples.

In this case, we can represent a real signal and a sequence on the same plot, using together a *plot* and a *stem* function, using the same domain for the samples (although being a sequence the samples are referred to integer indexes).
The extraction can be done by a *for* loop, using correct indexing to access the vector of initial values and creating a new vector for the samples. Nonetheless, extraction of values from the vector of values of the signal can be performed more efficiently by Matlab operations on matrixes. The operation to do this is recalled in the following example:

```
>> step = 5;
>> x=0:18;
>> x(1:step:end)
ans =
     0     5    10    15
```

where we extract a subset of the values x from its first one to its last one, jumping of *step*. An example of the decimation operation representing a real signal sampling is performed by executing the following commands:

```
>> dt=0.002;
>> step = 15;
>> t = [0:dt:1];
>> x = 3*sin(2*pi*5*t)+cos(2*pi*20*t);
>> n = t(1:step:end);
>> x_d = x(1:step:end);
>> plot(t,x);
>> hold on;
>> stem(n,x_d);
```

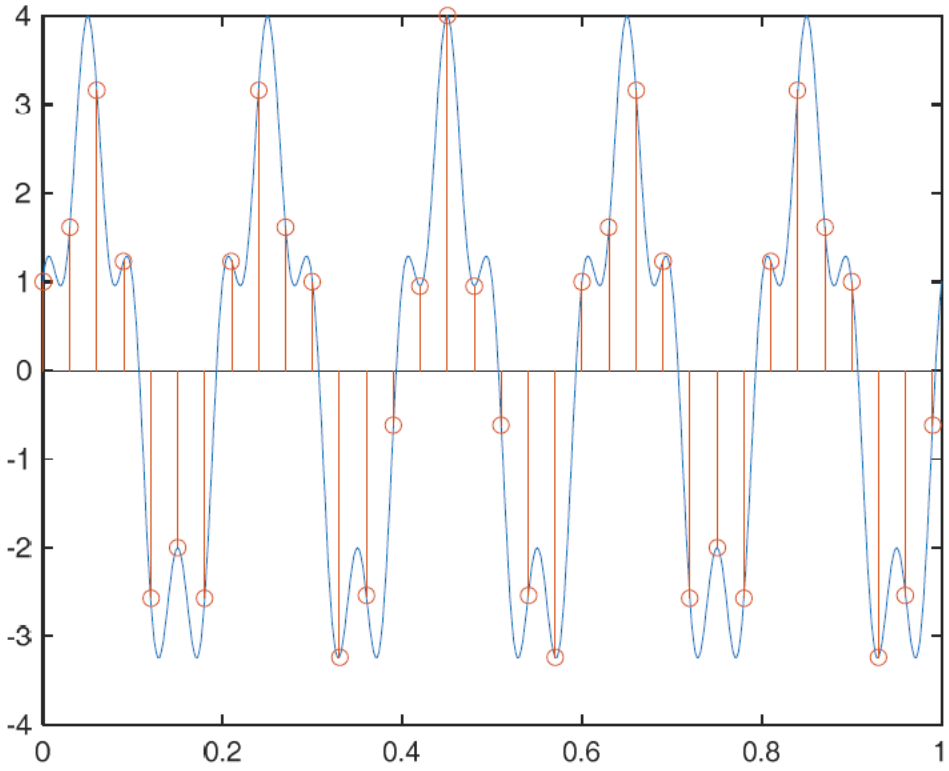

Figure 1 Sampling of a real signal by decimation

Expansion

In this case, it is required to increase the resolution of a signal, by adding intermediate values to a new signal within the same domain. Although in this case it may be sufficient to add zeros, it is also possible to replicate the same element N times, or implement complex interpolation operations (e.g., spline).

These three expansion methods are reported below by scripts and correspondent output.

expansion with zeros

```
>> dt = 0.5;
>> t = 0:dt:10;  y = cos(0.5*pi*t); % initial signal
>> N = 50; %expansion factor
>> step = dt/N; % finer step of the new signal
>> t_e = t(1):step:t(end)+(dt-step); %make it longer to host extra samples
>> y_e0 = zeros(1,length(t_e));
>> y_e0(1:N:end) = y;
>> plot(t,y,'o'); hold on;
>> plot(t_e,y_e0); title('expansion with zeros');
```

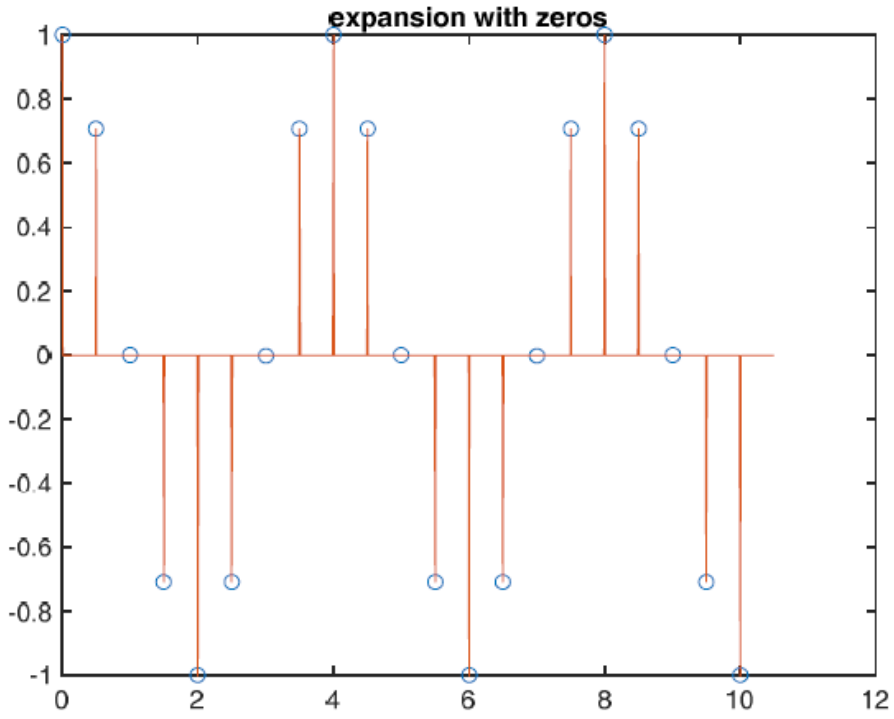

Figure 2: Expansion of a signal by adding zeros

Expansion with values

```
>> plot(t,y,'o');
>> hold on;
>> temp = [y(ones(1,N),1:end)];
>> y_e = temp(:)';
>> plot(t_e,y_e); title('expansion with values');
```

Expansion with Spline

```
>> t_sp = t(1):step:t(end);
>> y_sp = spline(t,y,t_sp);
>> plot(t,y,'o',t_sp,y_sp); title('spline');
```

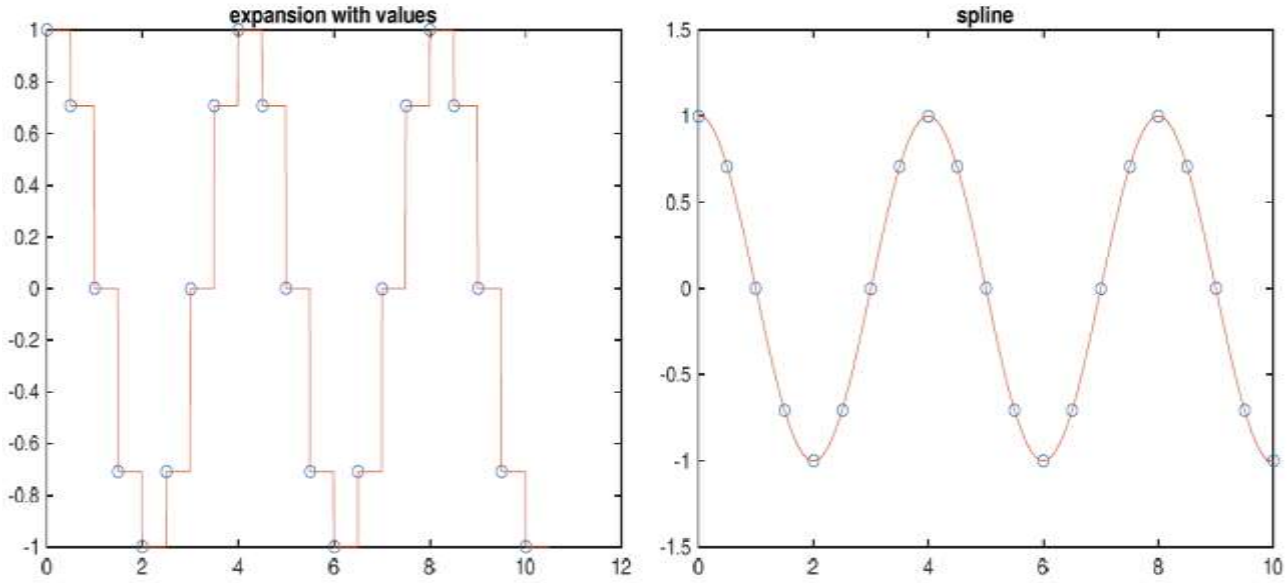

Figure 3 Expansion of a signal by replicating the value (left) and interpolation (right)

Exercise 1:

1. Plot the following x(t) real signals. For each signal, select a suitable domain to spot the main signal features (e.g., edges, periodicity, values different from 0, etc.). Use the commands xlabel, axis, legend, axis, etc. to improve the graph presentation. Concerning the resolution, typically a value of 0.01 shall be adequate, although specific consideration shall be made on each signal (i.e. smaller resolution can be enough, or higher resolution is required).

$$x(t) = \frac{1}{2}\cos\left(15\pi t + \frac{\pi}{8}\right)u(t)$$

If the function cannot be correctly represented use the above learnt methods to correct the resolution

Exercise 2:

2. Write a lab report for the above exercises and include your results and conclusions for the same.