Multimedia Signal Processing 1st Module and Fundamentals of Multimedia Signal Processing

date: 12/02/2019

Ex.1 (Pt.12)

A signal $x(t)$, sampled at 4kHz $x(n)=[1,2,3,4,3,2,1,0,1,2,3,4]$ is filtered with a LTI filter $H(z)=1-z^{-2}$. The procedure is done in real-time and the maximum accepted delay is 1ms.

1. [6pts.] Describe all the steps in order to get the requested result following the Overlap-and-Add approach: at every step provide the output, the reminder and how it is processed at the next step.
2. [6pts.] Describe all the steps in order to get the requested result following the Overlap-and-Save approach: describe the changes at the beginning of the input signal to avoid issues related to this approach, provide the outputs for every step and their processing to get the proper result.

Ex.2 (Pt.10)

A signal is sampled at 10kHz filling all the available band. We need to upsample it to 15kHz.

1. [3Pts.] Provide the full pipeline to properly upsample the signal minimizing the presence of artifacts or aliasing and detailing the parameters of every block.
2. [7pts.] Assuming that we are working on blocks of 8 samples and we are filtering the signal in frequency domain describe a possible implementation of the previous system working in the frequency domain (using the DFT).

Ex.3 (Pt. 11 – MATLAB code)

1. Write a MATLAB function typeOfFilter(b,a) that receives as input the numerator and denominator coefficients of a filter H(z)=B(z)/A(z) (row vectors) and it returns:
   a. [3pt.] -1 if the filter is not stable
   b. [3pt.] else 1 if the filter is stable and it is minimum phase, but it is not an all-pass filter
   c. [2pt.] else 0 if the filter is stable but it is not minimum phase
2. [3pt] Call the function three times to test the three different cases

Solutions

Ex.1

Since the maximum delay is of 1ms we must process a maximum of 4 samples at each iteration (since every millisecond we acquire 4 samples)

Overlap and Add

$\bar{y}_1[n]=\{1,2,3,4\}*\{1,0,-1\}=\{1,2,2,2,-3,-4\}$; the output will be just the first 4 samples $y_1[n]=\{1,2,2,2\}$ and the reminder for the next output will be $r_1[n]=\{-3,-4\}$, this reminder will be added to the second output.

$\bar{y}_2[n]=\{3,2,1,0\}*\{1,0,-1\}=\{3,2,-2,-2,-1,0\}$ and $r_2[n]=\{-1,0\}$;

$y_2[n]=\{3,2,-2,-2\}+r_1=\{3,2,-2,-2\}+\{-3,-4,0,0\}=\{0,-2,-2,-2\}$

$r_2[n]=\{-1,0\}$

And so on:

$y_3[n]=\{0,2,2,2\}$

$y_4[n]=\{-3,-4\}$

Overlap and Save

Since we are applying a circular convolution we will affect at each iteration the first two samples (length of the filter -1) and we need to add two zeros before the first sample in order to avoid that artifacts of circular convolution. We will the get:

$\bar{y}_1[n]=\{0,0,1,2,3,4\}\otimes\{1,0,-1\}=\{-3,-4,1,2,2,2\}$ so the output after the removal of the first two samples will be: $y_1[n]=\{1,2,2,2\}$

$\bar{y}_2[n]=\{3,4,3,2,1,0\}\otimes\{1,0,-1\}=\{2,4,0,-2,-2,-2\}$

$y_2[n]=\{0,-2,-2,-2\}$

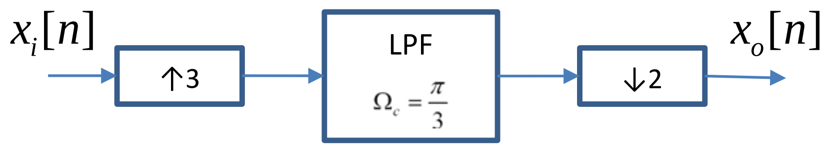
$\bar{y}_3[n]=\{1,0,1,2,3,4\}\otimes\{1,0,-1\}=\{-2,4,0,2,2,2\}$

$y_3[n]=\{0,2,2,2\}$

$\bar{y}_4[n]=\{3,4,0,0,0,0\}\otimes\{1,0,-1\}=\{3,4,-3,-4,0,0\}$

$y_4[n]=\{-3,-4,0,0\}$

Ex.2



A very simple approach could be based on extracting the DFT of the signal after upsampling (it will be made of 24 samples, 8x3), the low pass filter can then be a filter with the following DFT: $H[0..7]=1$, $H[8..15]=0$, $H[16..23]=1$.

We can multiply the DFT of the upsampled signal with this filter and then we have to take the iDFT before downsampling.

Ex.3

```
function [ typeOf ] = typeOfFilter( b,a )
%% 1. Write a MATLAB function typeOfFilter(b,a) that receives as
input the numerator and denominator coefficients of a filter
H(z)=B(z)/A(z) and it returns:
    %a. [3pt.]  -1 if the filter is not stable
    poles=roots(a);
    for p =1:length(poles)
       pole=poles(p);
       if abs(pole)>1
           typeOf= -1;
           return
       end
    end

    %b. [3pt.] else 1 if the filter is stable and it is minimum
phase, but it is not an all-pass filter
    zeros=roots(b);
    for z =1:length(zeros)
       zero=zeros(z);
       if abs(zero)>1
           typeOf= 0;
           return
       end
    end

    %c. [2pt.] else 0 if the filter is stable but it is not
minimum phase
    typeOf=1;
end
```

```
%2. [3pt] Call the function four times to test the three
different cases

% non stable
poles_ns=[0.6 1.1];
zeros_ns=[0.8];
a=poly(poles_ns);
b=poly(zeros_ns);

ns=(typeOfFilter(b,a)== -1);

% stable, non minimum phase

poles_snmp=[0.6,0.5,0.3];
zeros_snmp=[0.6,1.2];
a=poly(poles_snmp);
b=poly(zeros_snmp);
snmp=(typeOfFilter(b,a)==0);


% stable, minimum phase
poles_smp=[0.6,0.5,0.3];
zeros_smp=[0.6,0.8];
a=poly(poles_smp);
b=poly(zeros_smp);
smp=(typeOfFilter(b,a)==1);
```