Table of Contents

Title: SIGNAL RECONSTRUCTION THROUGH OVERSAMPLING/INTERPOLATION

Author: Filippo Valmori

Date: 29/04/2023

Reference: [1] Digital signal processing - Fundamentals and applications (Li Tan, 2013)

```
close all
clearvars
clc
```

PARAMETERS

```
Fc = 50;                                                        % Signal
 frequency [Hz]
Ncyc = 4;                                                       % Number
 of cycles to be simulated
sps = 6;                                                        % Original
 sampling factor

OvFct = 8;                                                      %
 Oversampling factor for reconstruction
Nord = 25;                                                      % Low-pass
 FIR filter order
Fedge = 2*Fc;                                                   % Low-pass
 FIR filter cut-off frequency [Hz]

Npt = 512;                                                      % Number
 of points for flilter shape representation
```

```
ScFct = 1e3;                                                        % Scaling
 factos for graphs (e.g. 1e3 = ms/kHz)
```

PROCESSING

```
Fs1 = sps*Fc;                                                       % Original
 sampling rate [Sa/s]
Time1 = 1/Fs1*(0:sps*Ncyc-1);
Sgn1 = sin(2*pi*Fc*Time1);                                          % Original
 signal

Osf2 = sps*OvFct;                                                   %
 Recontrunction oversampling factor
Fs2 = Osf2*Fc;                                                      %
 Recontrunction sampling rate [Sa/s]
Sgn2 = OvSamp(Sgn1,OvFct);
Time2 = 1/Fs2*(0:Osf2*Ncyc-1);
[FreqAxS,SpecSgn2,~] = GetSpect(Sgn2,Fs2);
B = fir1(Nord,Fedge*2/Fs2,'low');                                   % Low-pass
 FIR filter taps
[H,FreqAx] = freqz(B,1,Npt,Fs2);                                    % FIR
 filter shape
H_mgn = 20*log10(abs(H));                                           % FIR
 filter magnitude shape
Sgn3 = filter(B,1,Sgn2);                                            %
 Reconstructed signal after filtering
```

RESULTS

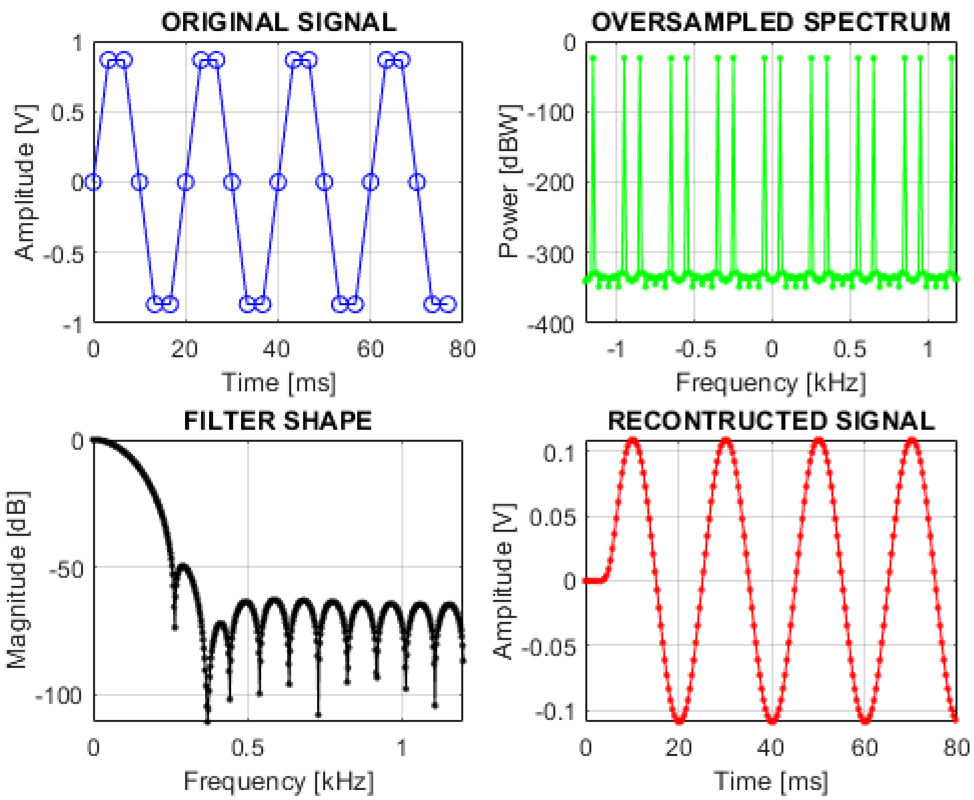
```
figure
subplot(2,2,1)
plot(Time1*ScFct,Sgn1,'bo-')
xlabel('Time [ms]')
ylabel('Amplitude [V]')
title('ORIGINAL SIGNAL')
grid on

subplot(2,2,2)
plot(FreqAxS/ScFct,SpecSgn2,'g.-')
xlabel('Frequency [kHz]')
ylabel('Power [dBW]')
title('OVERSAMPLED SPECTRUM')
grid on

subplot(2,2,3)
plot(FreqAx/ScFct,H_mgn,'k.-')
xlabel('Frequency [kHz]')
ylabel('Magnitude [dB]')
title('FILTER SHAPE')
grid on

subplot(2,2,4)
```

```
plot(Time2*ScFct,Sgn3,'r.-')
xlabel('Time [ms]')
ylabel('Amplitude [V]')
title('RECONTRUCTED SIGNAL')
grid on
```



FUNCTIONS

```
% >> Function to oversample input signal.
function [ SgnOut ] = OvSamp( SgnIn, Fct )
  LenIn = length(SgnIn);
  LenOut = LenIn*Fct;
  SgnOut = zeros(1,LenOut);
  SgnOut(1:Fct:end) = SgnIn;
end

% >> Function to get the power spectrum of the input signal.
function [ FreqAx, PwrSpect, OvPwrF ] = GetSpect( Sgn, Fs )
    Ns = length(Sgn);
  % Length (in samples) of the input waveform
    dF = Fs/Ns;
  % Discretization step for frequency axis
    FreqAx = -Fs/2:dF:Fs/2-dF;
  % Frequency axis for spectrum plot
    CpxSpect = fftshift(fft(Sgn))/Ns;
  % Complex spectrum
```

```
    PwrSpect = 20*log10(abs(CpxSpect));
  % Power spectrum [dBW/Hz]
    OvPwrF = 10*log10(sum((abs(CpxSpect)).^2));
     % Overall signal power [dBW] estimated in frequency domain (NB: do NOT
 multiply by dF, that's wrong for this descrete representation!)
    % As a check, below the overall power is estimated in time as well %
    % OvPwrT = 10*log10(sum(Sgn.^2)/length(Sgn));
     % Overall signal power [dBW] estimated in time domain (i.e. P = Vrms^2/R,
 where Vrms = sqrt(1/N*sum(Sgn[i]^2)) and assuming R = 1 Ohm)
end
```

Published with MATLAB® R2022a