Table of Contents

PART A: sampling

```
close all
clc
N=1000;
delta=1/N;
t=0:delta:(1-delta);
x=(1.5*cos(80*pi*t)+sin(240*pi*t)).*exp(-100*(t-0.5).^2);
plot_freq(x,'x[n]');
```

PART B: downsampling: factor 3

```
x3=x(1:3:end);
plot_freq(x3,'x[3n]');

% There is no any aliasing in the DTFT, as a result of
% the limited bandwidth of x[n] less than pi/3
% $$X_3(e^{j\omega}) = X(e^{j\frac{\omega}{3}})$$
% The DTFT only change its scale
```

interpolator: factor 3

```
x_inter=zeros(1,N);
x_inter(1:3:end)=x3;%zero padding
x_inter=conv(x_inter,pidiv3filt);%pi/3 filter
plot_freq(x_inter,'x_{3 interpolate}[n]');
plot_series(x,x_inter,3);
% The amplitude of the signal is reduced by a factor of 3.
```

PART C: downsampling: factor 6

```
x6=x(1:6:end);
plot_freq(x6,'x[6n]');
```

interpolator: factor 6

```
x_inter=zeros(1,N);
x_inter(1:6:end)=x6;%zero padding
```

```
x_inter=conv(x_inter,pidiv6filt);%pi/6 filter
plot_freq(x_inter,'x_{6 interpolate}[n]');
plot_series(x,x_inter,6);

% There is an aliasing in the DTFT, as a result of
% the bandwidth of x[n] that it's more than pi/6
% $$X_6(e^{j\omega}) = X(e^{j\frac{\omega}{6}})$$
% The DTFT changes significantly in addition to the decrement in
% the amplitude of the signal by a factor of 6. The shape of the
 signal
% and the DTFT is changed, and sompe peaks in the DTFT are merged.
```

PART D: Anti -Aliasing filter:

```
x_anti=conv(x,pidiv6filt);%pi/6 filter
```

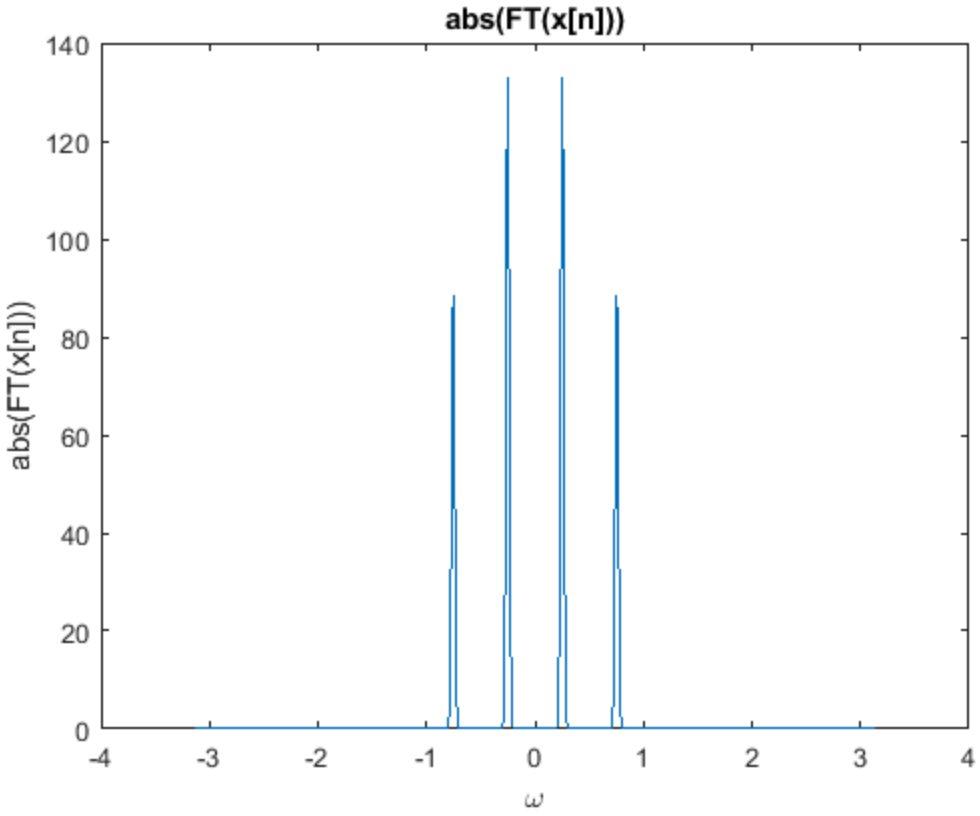
downsampling: factor 6

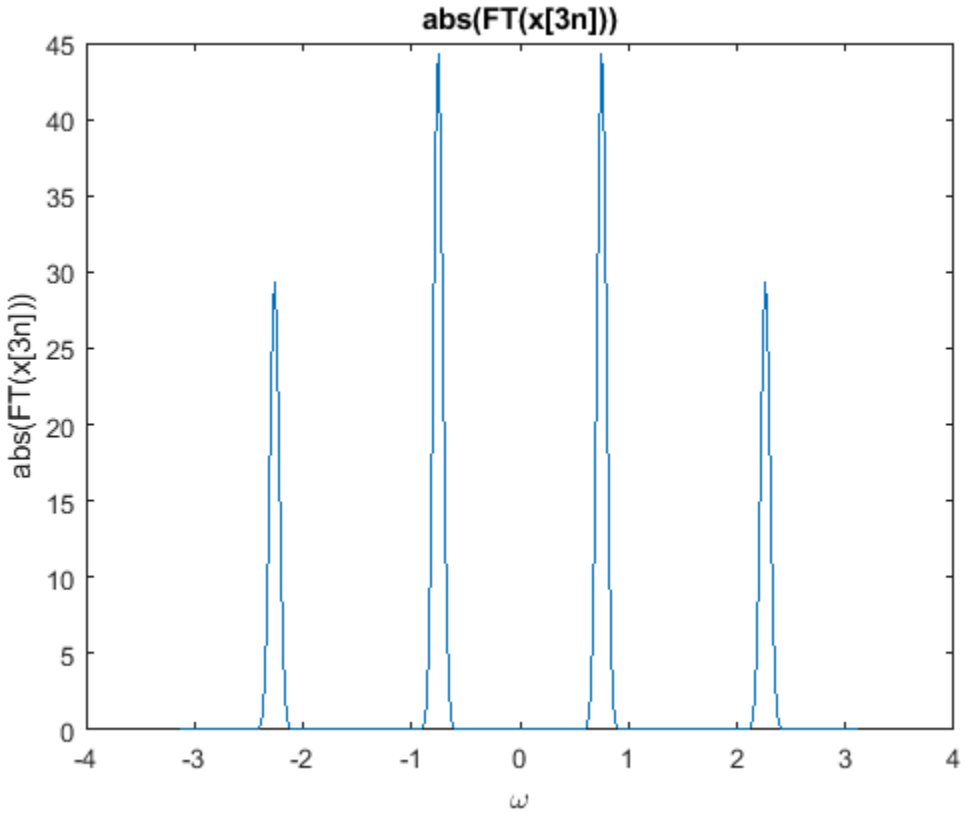
```
x_anti6=x_anti(1:6:end);
plot_freq(x_anti6,'x_{anti aliasing}[6n]');
```

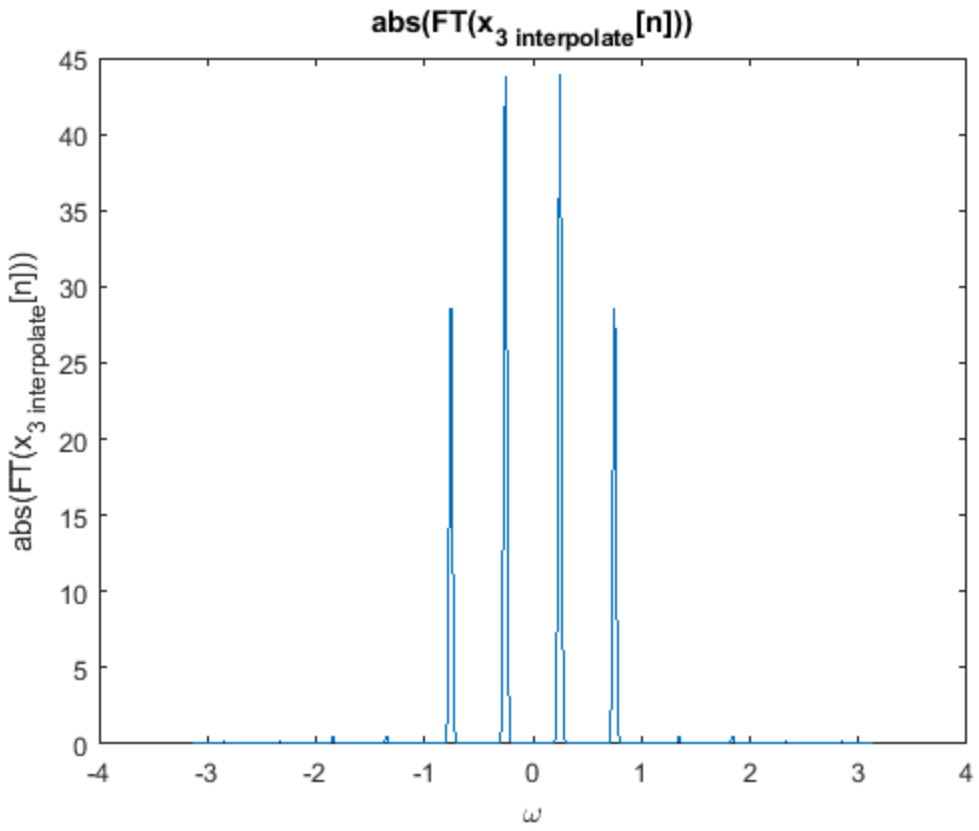
interpolator: factor 6

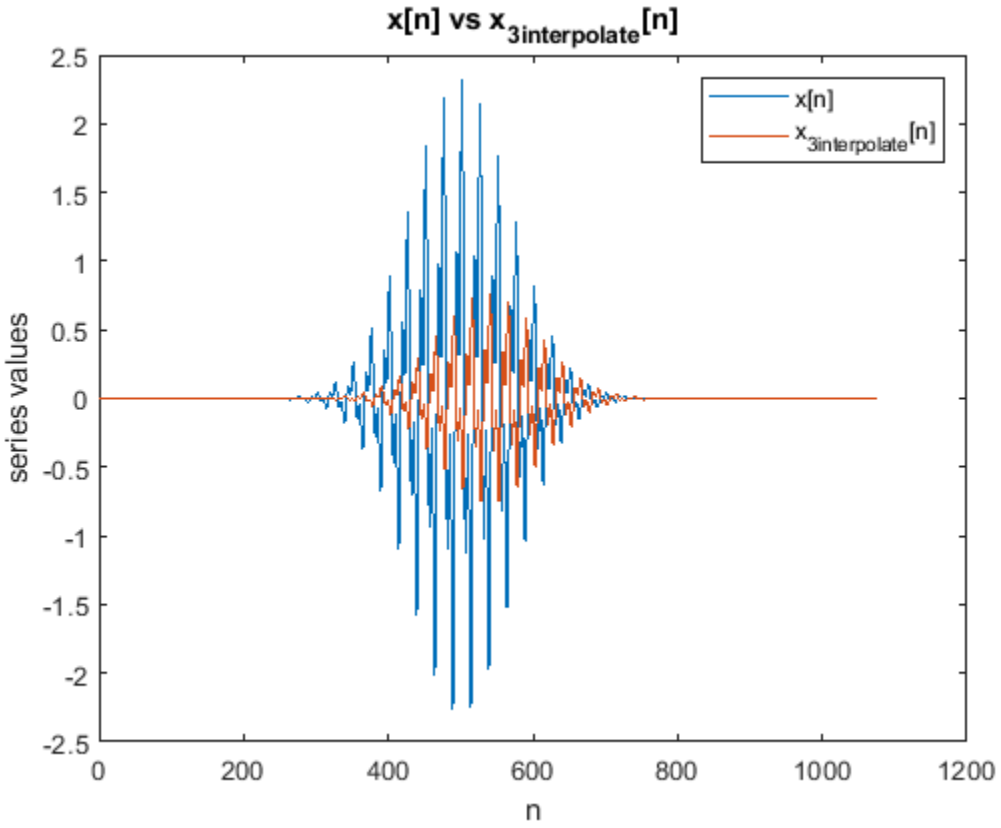
```
x_anti_inter=zeros(1,length(x_anti6)*6);
x_anti_inter(1:6:end)=x_anti6;%zero padding
x_anti_inter=conv(x_anti_inter,pidiv6filt);%pi/6 filter
plot_freq(x_anti_inter,'x_{anti-aliasing 6-interpolate}[n]');
%%%compare to PART C:
plot_compare(x6,x_anti6,pidiv6filt,'After decimation: ')
plot_compare(x_inter,x_anti_inter,pidiv6filt,'After interpolation: ')
%After the anti-aliasing filter, one remains only with the lower
 bandwidth

%Anti -Aliasing filter in PART B will not affect on the results,
% because the signal is already pi/3 band limited

function plot_freq(data,label_fig)
ft=@(x)abs(fftshift(fft(x)));

figure
delta=1/length(data);
omega=-pi:2*pi*delta:(pi-2*pi*delta);
plot(omega,ft(data));
xlabel('\omega')
ylabel(['abs(FT(',label_fig,'))'])
title(['abs(FT(',label_fig,'))'])
end


function plot_series(data,data_interpolate,val)

figure
```

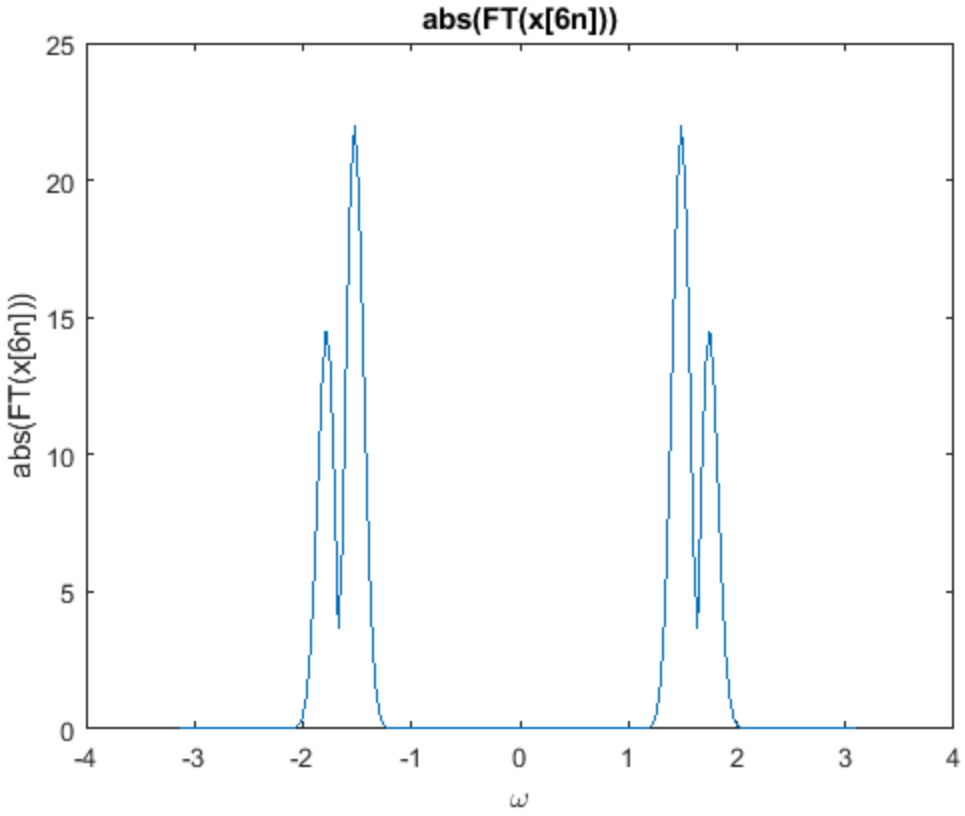
```
samples=0:length(data)-1;
plot(samples,data);
hold on
samples=0:length(data_interpolate)-1;
plot(samples,data_interpolate);
xlabel('n')
ylabel('series values')
lab=['x_{',num2str(val),'interpolate}[n]'];
title(['x[n] vs ',lab])
legend('x[n]',lab)
end


function plot_compare(data,data_anti,anti_aliasing,label_fig)

figure
samples=0:length(data)-1;
plot(samples,data);
hold on
samples=0:length(data_anti)-1;
plot(samples,data_anti);
samples=0:length(anti_aliasing)-1;
plot(samples,anti_aliasing);
xlabel('n')
ylabel('series values')
title([label_fig,' compare with and without anti-aliasing filter'])
legend('x_{6-interpolate}[n] without anti-aliasing',...
    'x_{6-interpolate}[n] anti-aliasing','h[n] anti -aliasing filter')
end
```

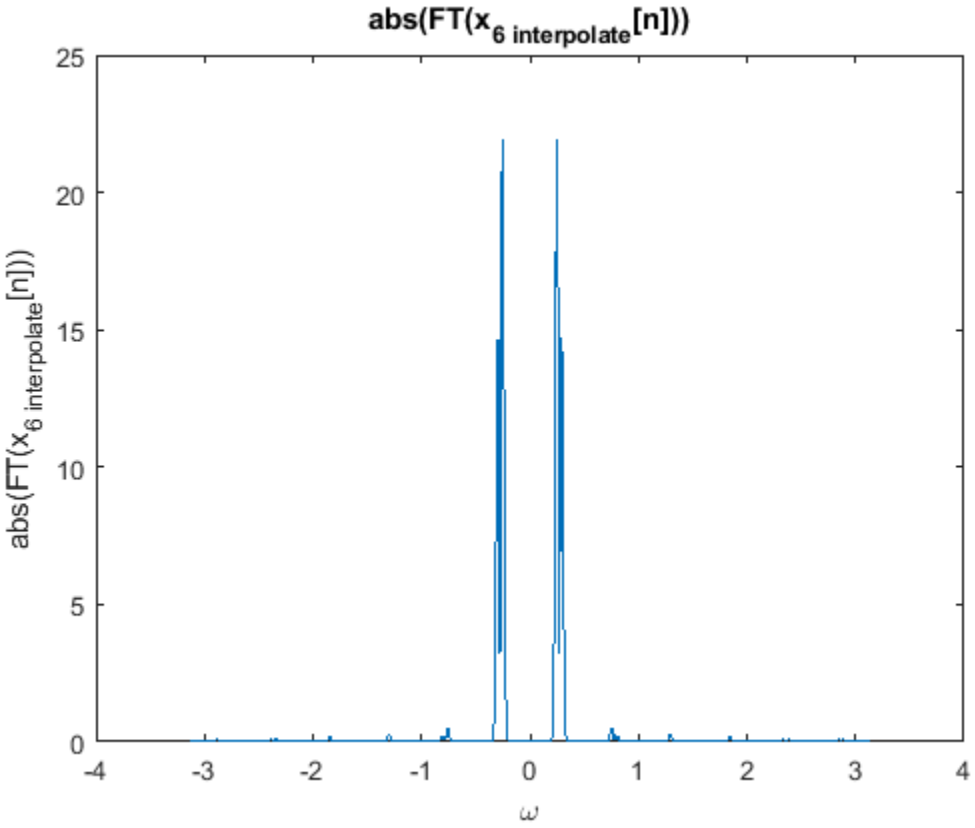

abs(FT(x[n]))
140
120
100
80
60
40
20
0
abs(FT(x[n]))
-4
-3
-2
-1
0
1
2
3
4
ω

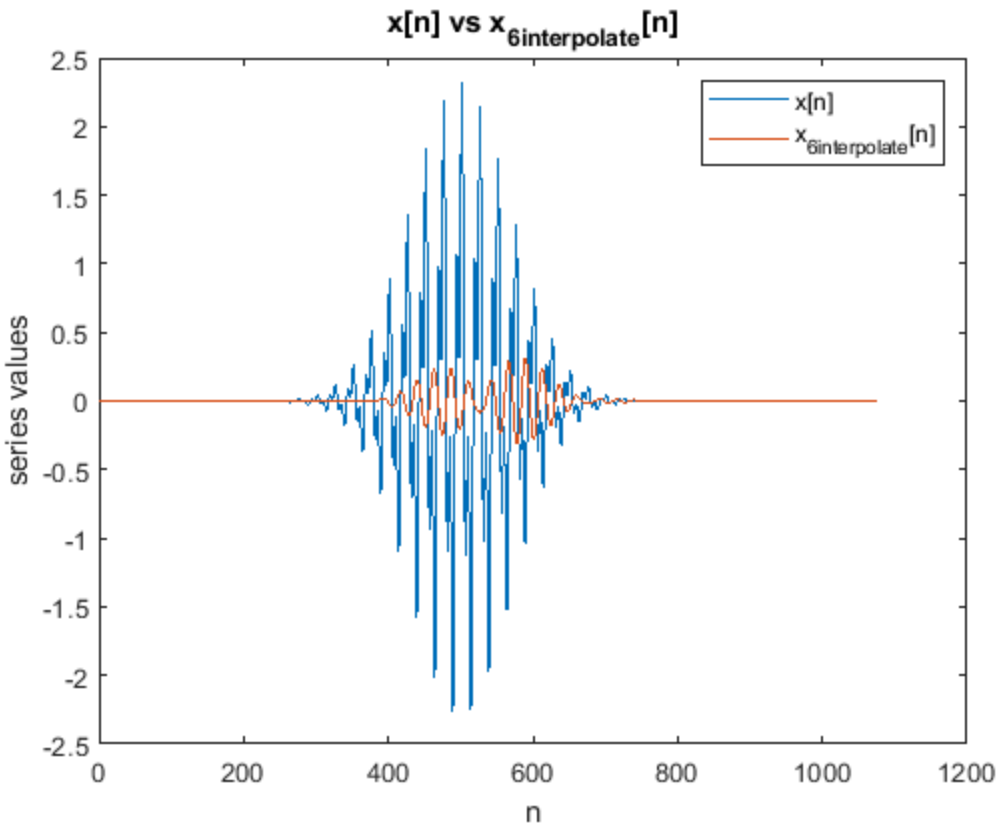

abs(FT(x[3n]))
45
40
35
30
25
20
15
10
5
0
abs(FT(x[3n]))
-4
-3
-2
-1
0
1
2
3
4
ω

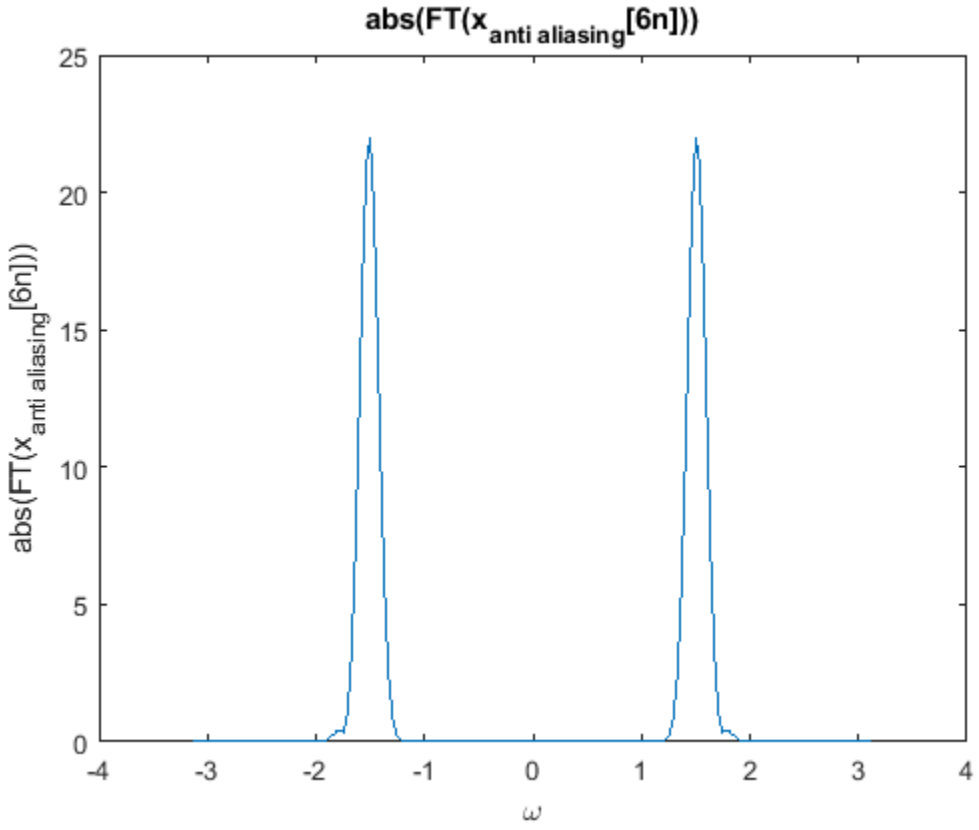

abs(FT(x3 interpolate[n]))
45
40
35
30
25
20
15
10
5
0
-4
-3
-2
-1
0
1
2
3
4
ω

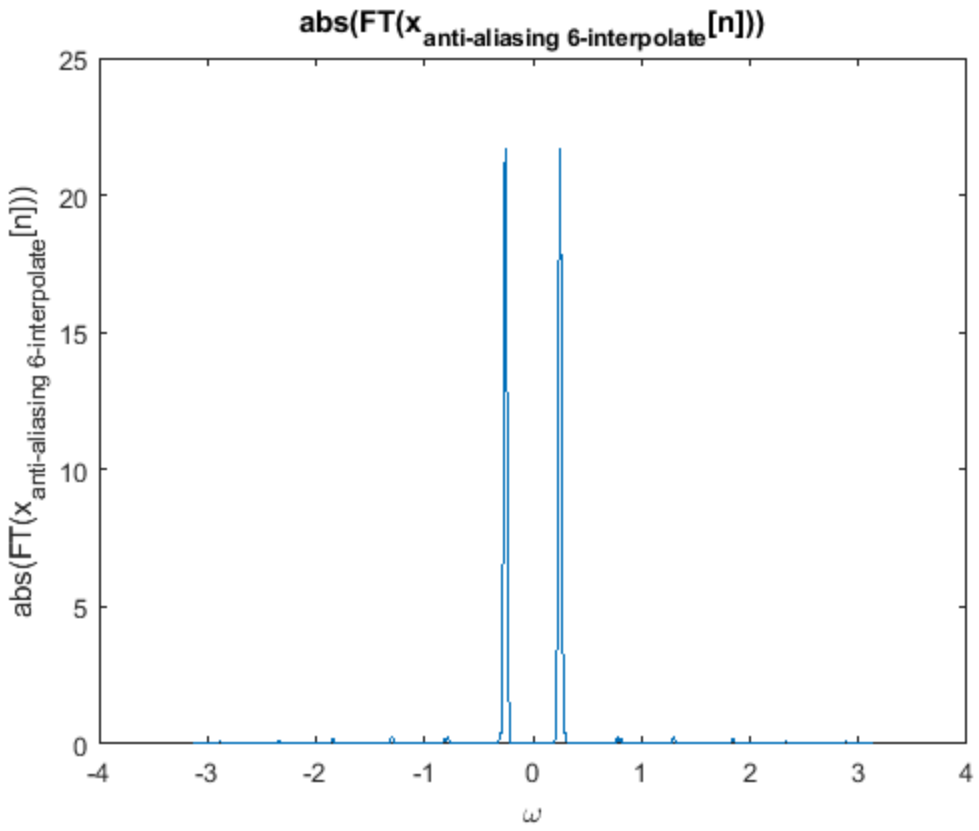

x[n] vs x3interpolate[n]
x[n]
x3interpolate[n]
series values
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
200
400
600
800
1000
1200
n

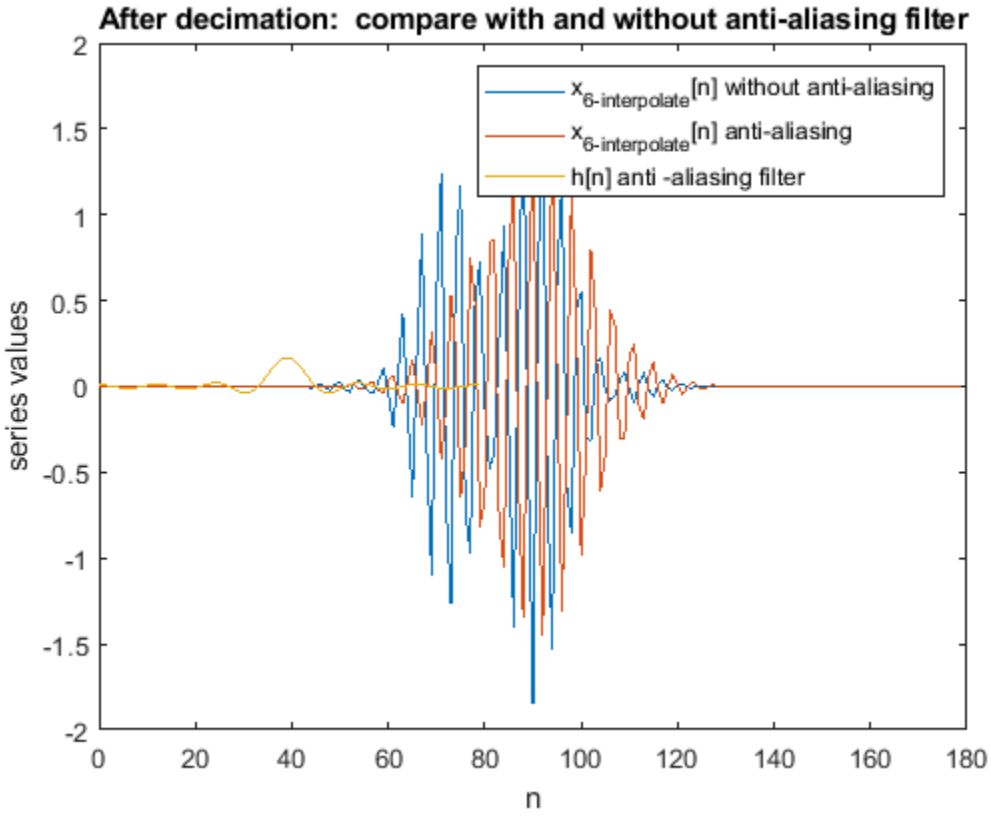

abs(FT(x[6n]))
abs(FT(x[6n]))
ω

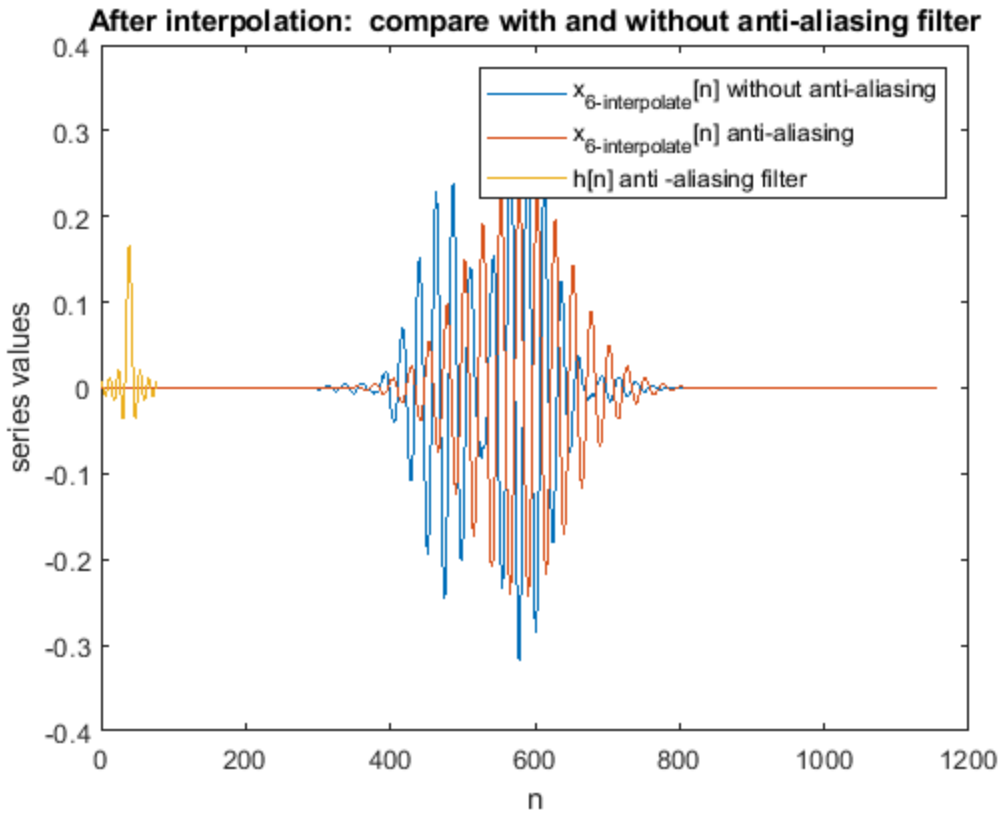

abs(FT(x6 interpolate[n]))
abs(FT(x6 interpolate[n]))
ω


x[n] vs x6interpolate[n]
x[n]
x6interpolate[n]
series values
n
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
200
400
600
800
1000
1200


abs(FT(xanti aliasing[6n]))
abs(FT(xanti aliasing[6n]))
ω
25
20
15
10
5
0
-4
-3
-2
-1
0
1
2
3
4


abs(FT(x anti-aliasing 6-interpolate[n]))
abs(FT(x anti-aliasing 6-interpolate[n]))
ω


After decimation: compare with and without anti-aliasing filter
x 6-interpolate[n] without anti-aliasing
x 6-interpolate[n] anti-aliasing
h[n] anti -aliasing filter
series values
n



Published with MATLAB® R2018b