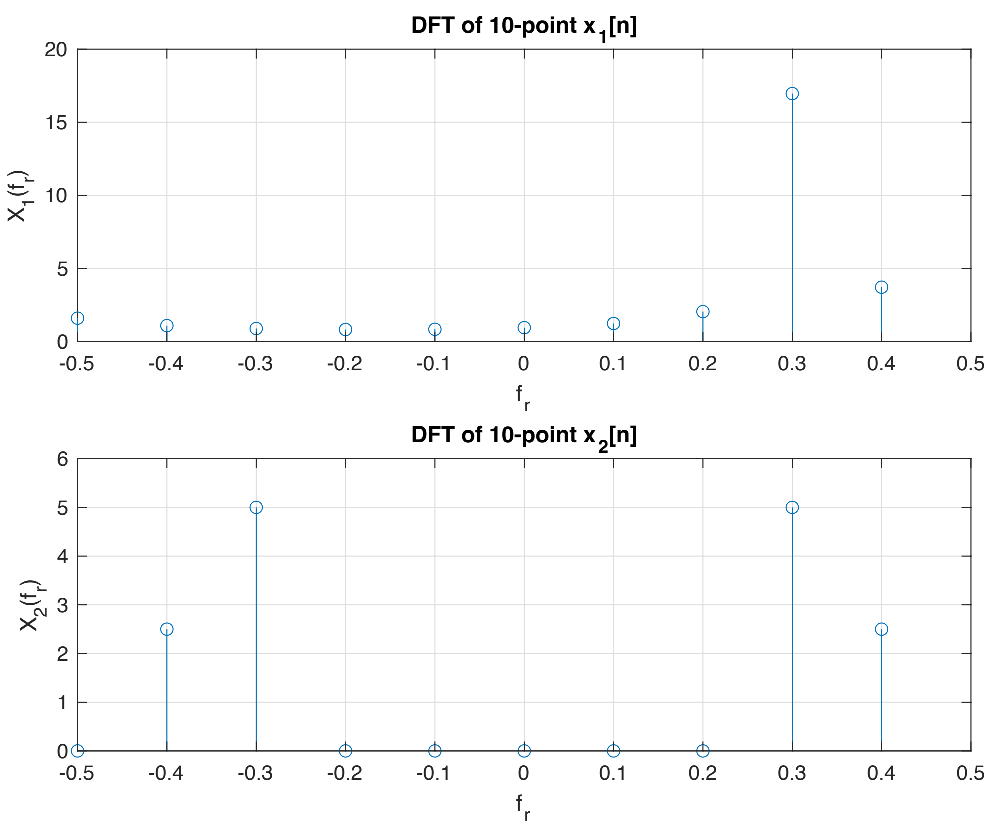
Sampling and Discrete Fourier Transform

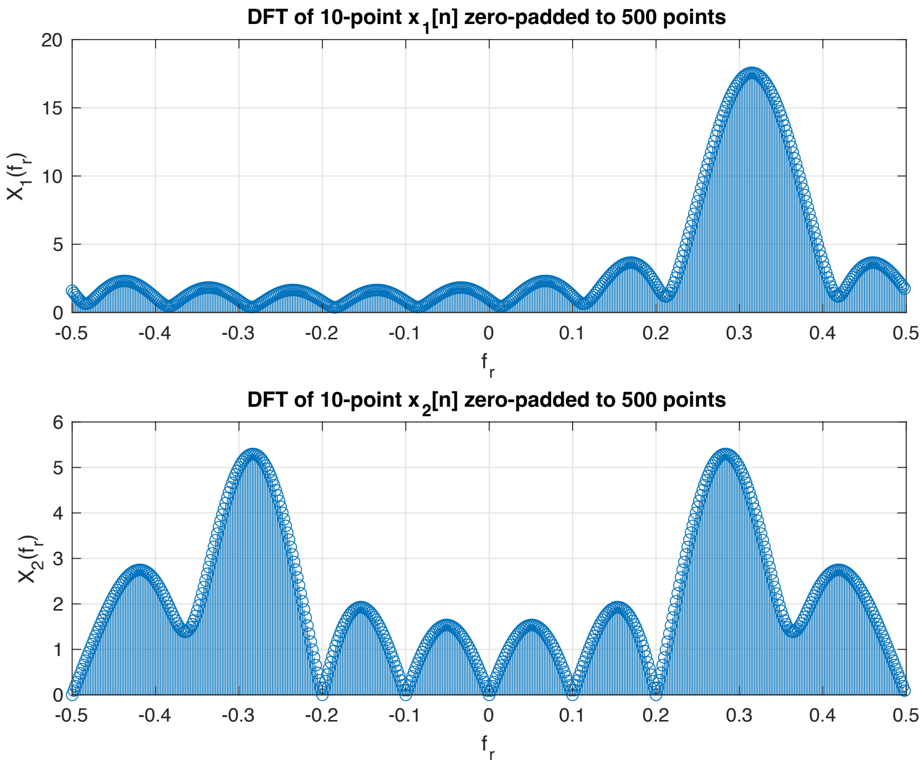
Table of Contents

A. Discrete Fourier transform (DFT)

```
% A.1

n = 0:9; % 10 samples

% Signal x[1] and x[2]
x1 = (exp(j*2*pi*n*(30/100))) + (exp(j*2*pi*n*(33/100)));
x2 = (cos(2*pi*n*(30/100))) + (0.5*cos(2*pi*n*(40/100)));

% Compute Discrete Fourier Transform for both signals
X1 = fft(x1);
X2 = fft(x2);
N = length(x1); % length(x1) = length(x2) = 10
fr = (0:N-1)/N;
figure;
subplot(2,1,1);
stem(fr-0.5,abs(fftshift(X1)));
grid;
title('DFT of 10-point x_1[n]');
xlabel('f_r');
ylabel('X_1(f_r)');
axis([-0.5 0.5 0 20]);
subplot(2,1,2);
stem(fr-0.5,abs(fftshift(X2)));
grid;
title('DFT of 10-point x_2[n]');
xlabel('f_r');
ylabel('X_2(f_r)');
axis([-0.5 0.5 0 6])
```

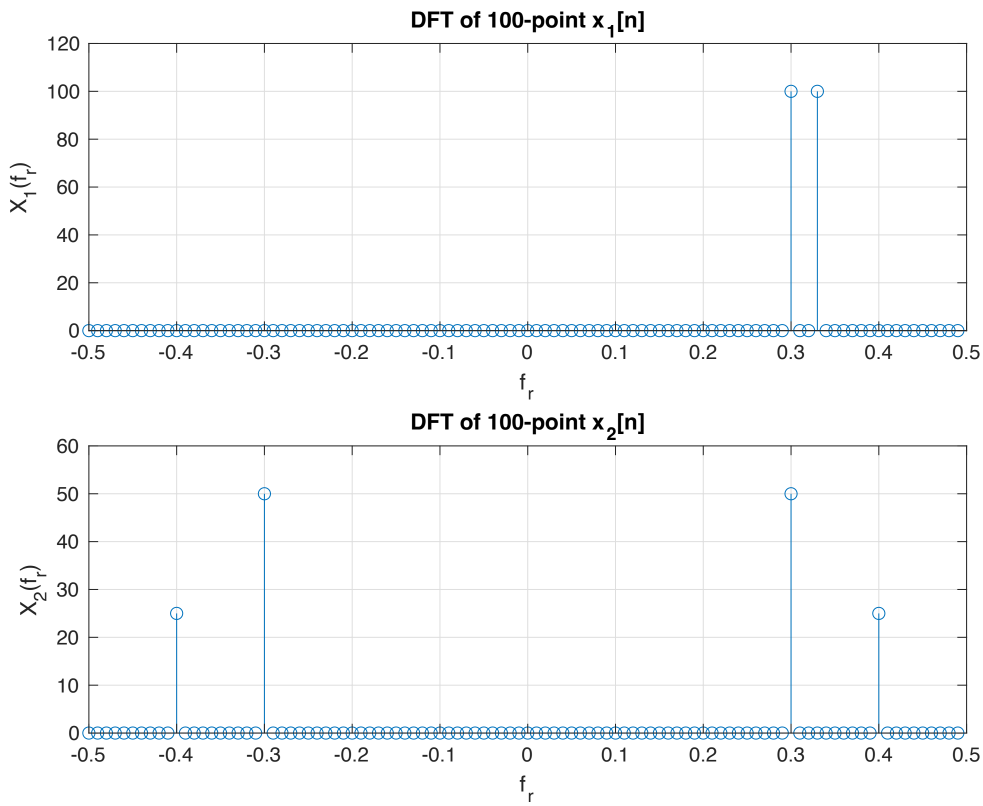


```
% A.1.i

% Signal x2[n] has a symmetric spectrum since the time-domain signal x2[n]
% is real while the time-domain signal x1[n] is complex.

% A.1.ii

% Since only 10 samples are used, the DTF has only 10 frequency bins
% uniformly spaced over the frequency interval [-0.5,0.5). Therefore,
% both frequency components of signal x2[n] at f = 0.30 and f = 0.40 are
% distinguishable. On the other hand, there is insufficient frequency
% resolution to separately identify the two components closely spaced
% at f = 0.30 and f = 0.33 for signal x1[n].

% A.1.iii

% Other frequency components occur in the plot of DTF of x1[n] because
% the frequency f = 0.33 does not lie directly on a DFT frequency bin
% which results in the effects of frequency leakage and smearing.

% A.2

n = 0:9; % 10 samples

% Signal x[1] and x[2]
x1 = (exp(j*2*pi*n*(30/100))) + (exp(j*2*pi*n*(33/100)));
x2 = (cos(2*pi*n*(30/100))) + (0.5*cos(2*pi*n*(40/100)));

% Zero-pad the signals x1[n] and x2[n] with 490 zeros
x1 = [x1,zeros(1,490)];
x2 = [x2,zeros(1,490)];

% Compute Discrete Fourier Transform for both signals
X1 = fft(x1);
X2 = fft(x2);

N = length(x1); % length(x1) = length(x2) = 500
fr = (0:N-1)/N;

figure;
subplot(2,1,1);
stem(fr-0.5,abs(fftshift(X1)));
grid;
title('DFT of 10-point x_1[n] zero-padded to 500 points');
xlabel('f_r');
ylabel('X_1(f_r)');
axis([-0.5 0.5 0 20]);

subplot(2,1,2);
stem(fr-0.5,abs(fftshift(X2)));
grid;
title('DFT of 10-point x_2[n] zero-padded to 500 points');
xlabel('f_r');
ylabel('X_2(f_r)');
axis([-0.5 0.5 0 6]);
```



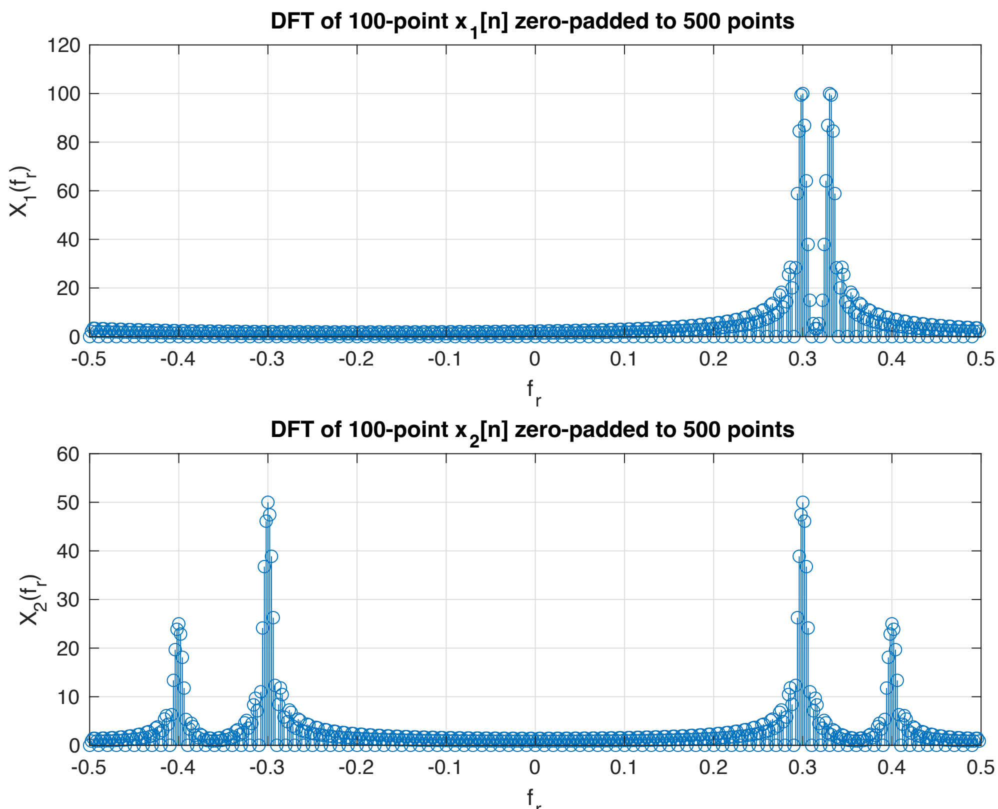

```
% Zero-padding increases the pickets in the DTF to the length of 500
% which does improve the spectrum of two signals. However, it does not
% change what lies behind the spectrum as the information about x1[n]
% to resolve the closely spaced frequency components at f = 0.30 and
% f = 0.33 is still insufficient.

% A.3

n = 0:99; % 100 samples

% Signal x[1] and x[2]
x1 = (exp(j*2*pi*n*(30/100))) + (exp(j*2*pi*n*(33/100)));
x2 = (cos(2*pi*n*(30/100))) + (0.5*cos(2*pi*n*(40/100)));

% Compute Discrete Fourier Transform for both signals
X1 = fft(x1);
X2 = fft(x2);

N = length(x1); % length(x1) = length(x2) = 100
fr = (0:N-1)/N;

figure;
subplot(2,1,1);
stem(fr-0.5,abs(fftshift(X1)));
grid;
title('DFT of 100-point x_1[n]');
xlabel('f_r');
ylabel('X_1(f_r)');
axis([-0.5 0.5 0 120]);

subplot(2,1,2);
```

```
stem(fr-0.5,abs(fftshift(X2)));
grid;
title('DFT of 100-point x_2[n]');
xlabel('f_r');
ylabel('X_2(f_r)');
axis([-0.5 0.5 0 60]);
```




```
% With 100 samples are used over the frequency interval [-0.5,0.5),
% each frequency components of both signals x1[n] and x2[n] are easily
% identified. Signal x2[n] has a symmetric spectrum since the time-domain
% signal x2[n] is real while the time-domain signal x1[n] is complex.

% A.3

n = 0:99; % 100 samples

% Signal x[1] and x[2]
x1 = (exp(j*2*pi*n*(30/100))) + (exp(j*2*pi*n*(33/100)));
x2 = (cos(2*pi*n*(30/100))) + (0.5*cos(2*pi*n*(40/100)));

% Zero-pad the signals x1[n] and x2[n] with 490 zeros
x1 = [x1,zeros(1,400)];
x2 = [x2,zeros(1,400)];

% Compute Discrete Fourier Transform for both signals
X1 = fft(x1);
X2 = fft(x2);

N = length(x1); % length(x1) = length(x2) = 500
fr = (0:N-1)/N;
```

```
figure;
subplot(2,1,1);
stem(fr-0.5,abs(fftshift(X1)));
grid;
title('DFT of 100-point x_1[n] zero-padded to 500 points');
xlabel('f_r');
ylabel('X_1(f_r)');
axis([-0.5 0.5 0 120]);

subplot(2,1,2);
stem(fr-0.5,abs(fftshift(X2)));
grid;
title('DFT of 100-point x_2[n] zero-padded to 500 points');
xlabel('f_r');
ylabel('X_2(f_r)');
axis([-0.5 0.5 0 60]);
```



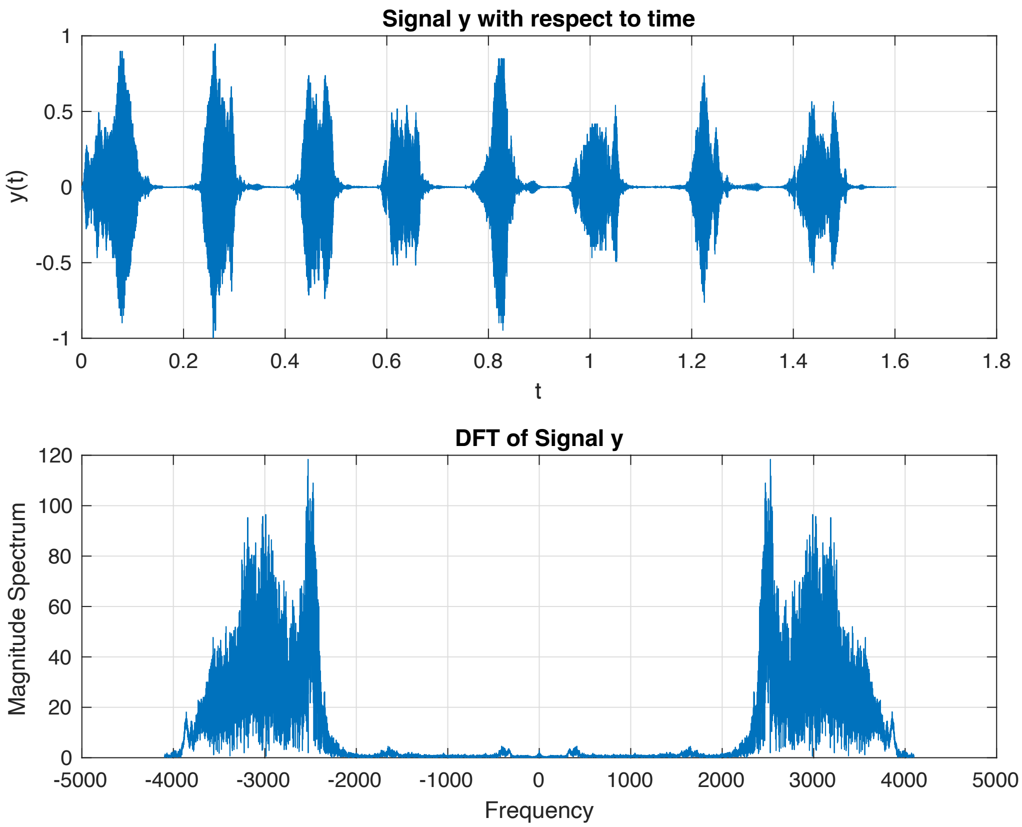

```
% Using 100 samples and zero padding to 500 points, the spectrum
% is significantly improved as each frequency component of both signal
% can be separately identified and it indicates a clearer picture of
% the data compared to the plots with 10 samples.
```

B. Sampling

```
load chirp.mat;
filename = 'chirp.wav';
audiowrite(filename,y,Fs);
clear y Fs
[y,fs] = audioread('chirp.wav');

% B.1
N_0 = length(y); % Number of samples
T_0 = N_0/fs; % Duration of the signal
T = 1/fs; % Sampling interval
```

```
N_0 =

       13129

>> T_0

T_0 =

    1.6027

>> T

T =

   1.2207e-04
```

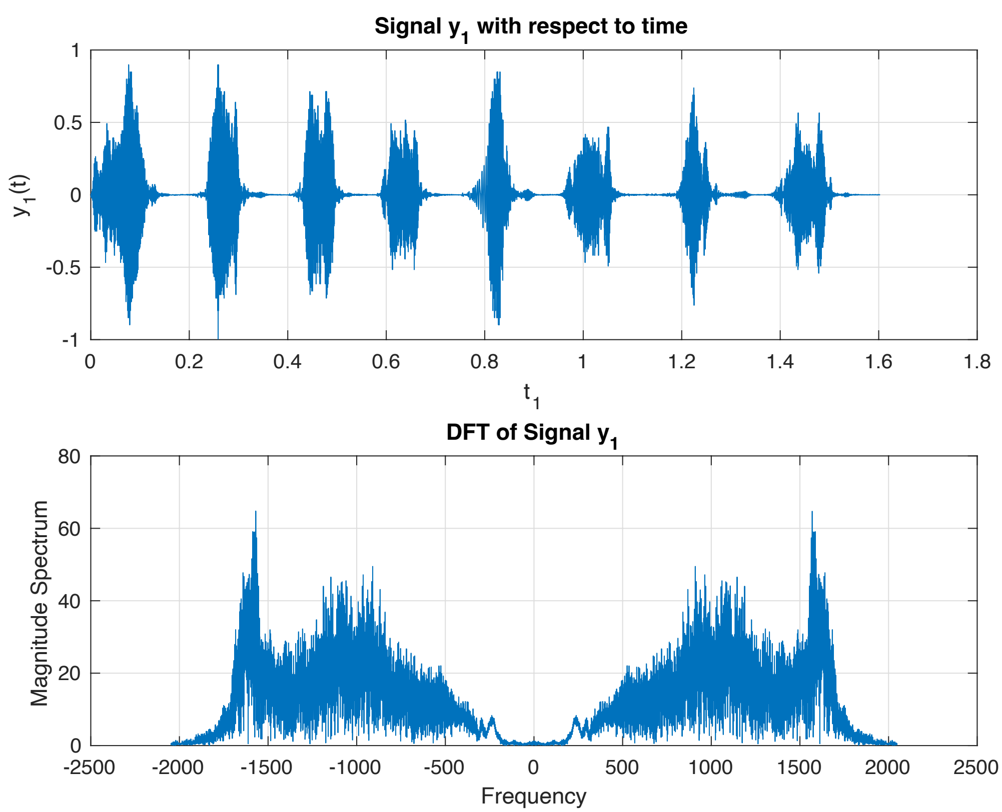
```
% B.2

t = linspace(0, T_0, N_0);

figure;
subplot(2,1,1);
plot(t, y);
grid;
title('Signal y with respect to time');
xlabel('t');
ylabel('y(t)');

% B.3

f = linspace(-fs/2, fs/2, N_0);
% Compute DFT of signal y
Y = fft(y);

subplot(2,1,2);
plot(f, abs(fftshift(Y)));
grid;
title('DFT of Signal y');
xlabel('Frequency');
ylabel('Magnitude Spectrum');
```

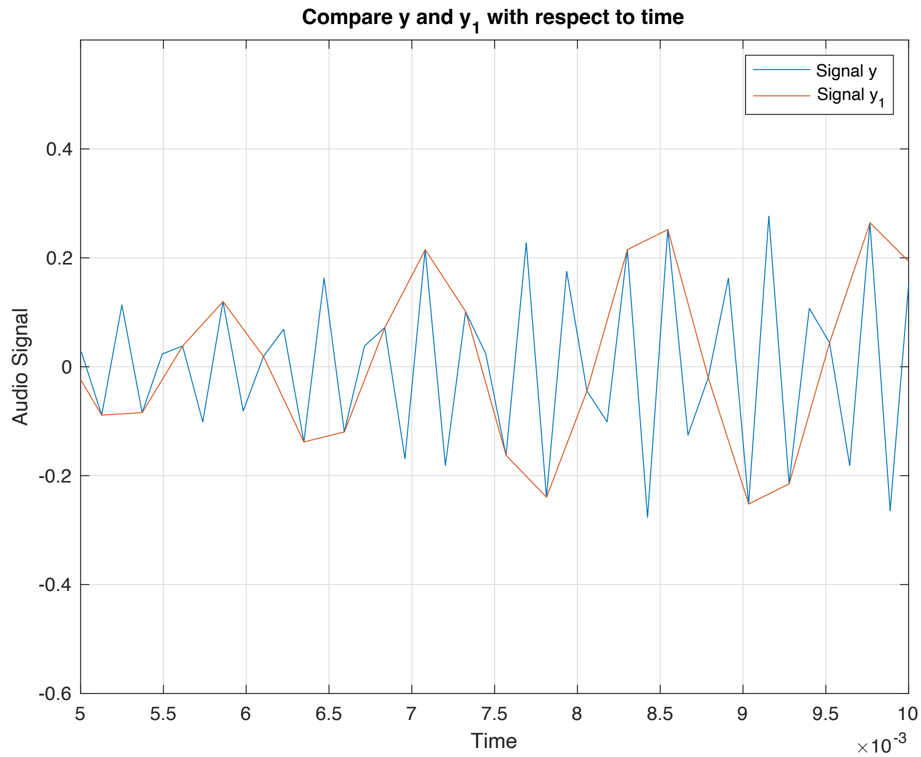



```
% B.4

% Subsampled signal y1 from signal y with rate "2"
y1 = y(1:2:length(y));

fs1 = fs/2; % Sampling frequency
N_1 = length(y1); % Number of samples
T_1 = N_1/fs1; % Duration of the signal
T1 = 1/fs1; % Sampling interval
```

```
N_1 =

        6565

>> T_1

T_1 =

    1.6028

>> T1

T1 =

   2.4414e-04
```

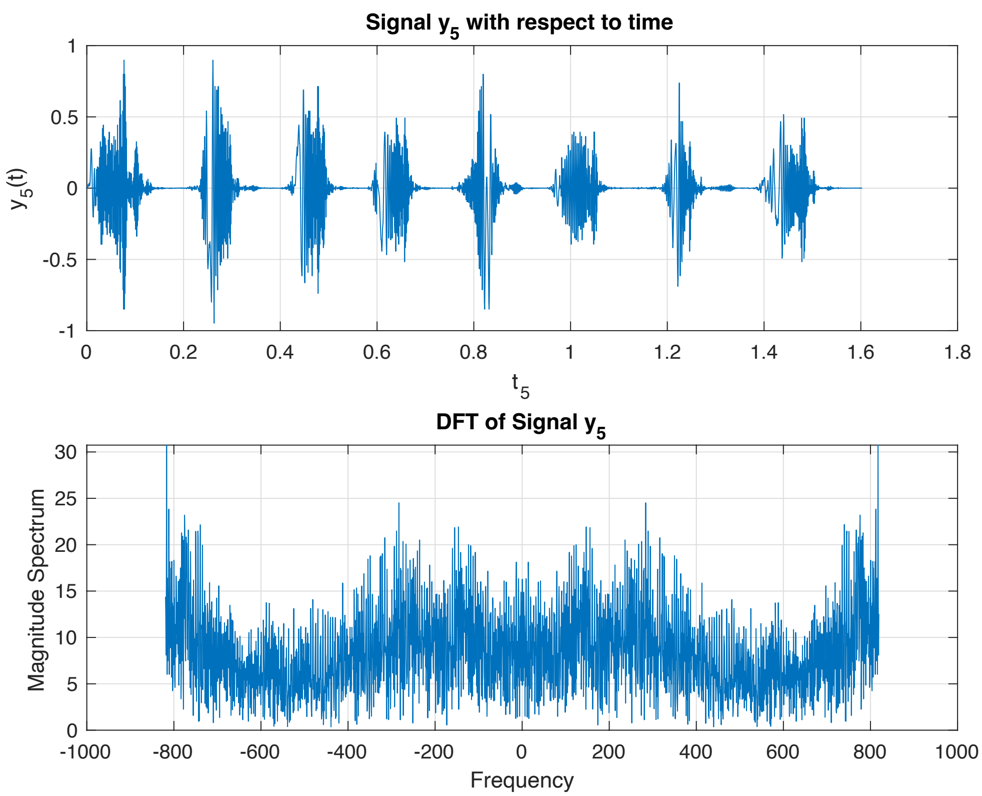
```
% B.5
t1 = linspace(0, T_1, N_1);

figure;
subplot(2,1,1);
plot(t1, y1);
grid;
title('Signal y_1 with respect to time');
xlabel('t_1');
ylabel('y_1(t)');

% B.6
f1 = linspace(-fs1/2, fs1/2, N_1);
% Compute DFT of signal y
Y1 = fft(y1);

subplot(2,1,2);
plot(f1, abs(fftshift(Y1)));
grid;
title('DFT of Signal y_1');
xlabel('Frequency');
ylabel('Magnitude Spectrum');

figure;
plot(t, y, t1, y1);
grid;
title('Compare y and y_1 with respect to time');
xlabel('Time');
ylabel('Audio Signal');
legend('Signal y', 'Signal y_1');
axis([0.005 0.01 -0.6 0.6]);
```

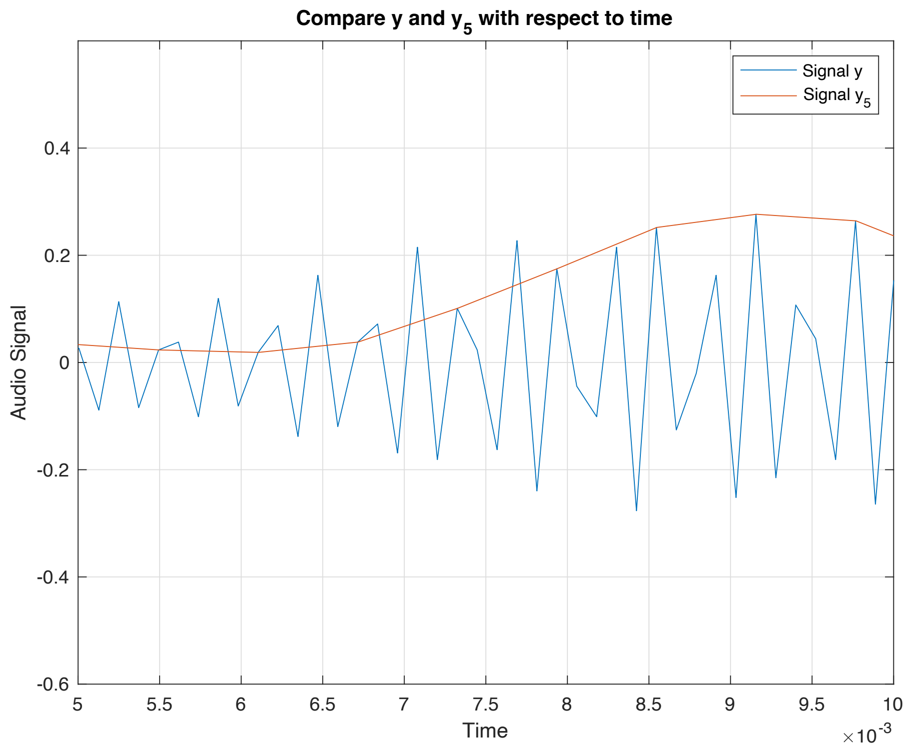





```
% The audio signal after downsampling by factor of "2" has lost half
% of the sample and cannot capture all the information of the original
% signal. In addition, the downsampling spectrum has its amplitude
% and frequency axis scaled by the factor of 1/2.

% B.7

% sound(y, fs)
% sound(y1, fs1)

% The audio after subsampling is slightly different as it is unable to
% obtain all information of the original audio with half of the samples.

% B.8

% Subsampled signal y1 from signal y with rate "5"
y5 = y(1:5:length(y));

fs5 = fs/5; % Sampling frequency
N_5 = length(y5); % Number of samples
T_5 = N_5/fs5; % Duration of the signal
T5 = 1/fs5; % Sampling interval

t5 = linspace(0, T_5, N_5);

figure;
subplot(2,1,1);
plot(t5, y5);
grid;
title('Signal y_5 with respect to time');
xlabel('t_5');
```

```
ylabel('y_5(t)');

f5 = linspace(-fs5/2, fs5/2, N_5);
% Compute DFT of signal y
Y5 = fft(y5);

subplot(2,1,2);
plot(f5, abs(fftshift(Y5)));
grid;
title('DFT of Signal y_5');
xlabel('Frequency');
ylabel('Magnitude Spectrum');

figure;
plot(t, y, t5, y5);
grid;
title('Compare y and y_5 with respect to time');
xlabel('Time');
ylabel('Audio Signal');
legend('Signal y', 'Signal y_5');
axis([0.005 0.01 -0.6 0.6]);
```



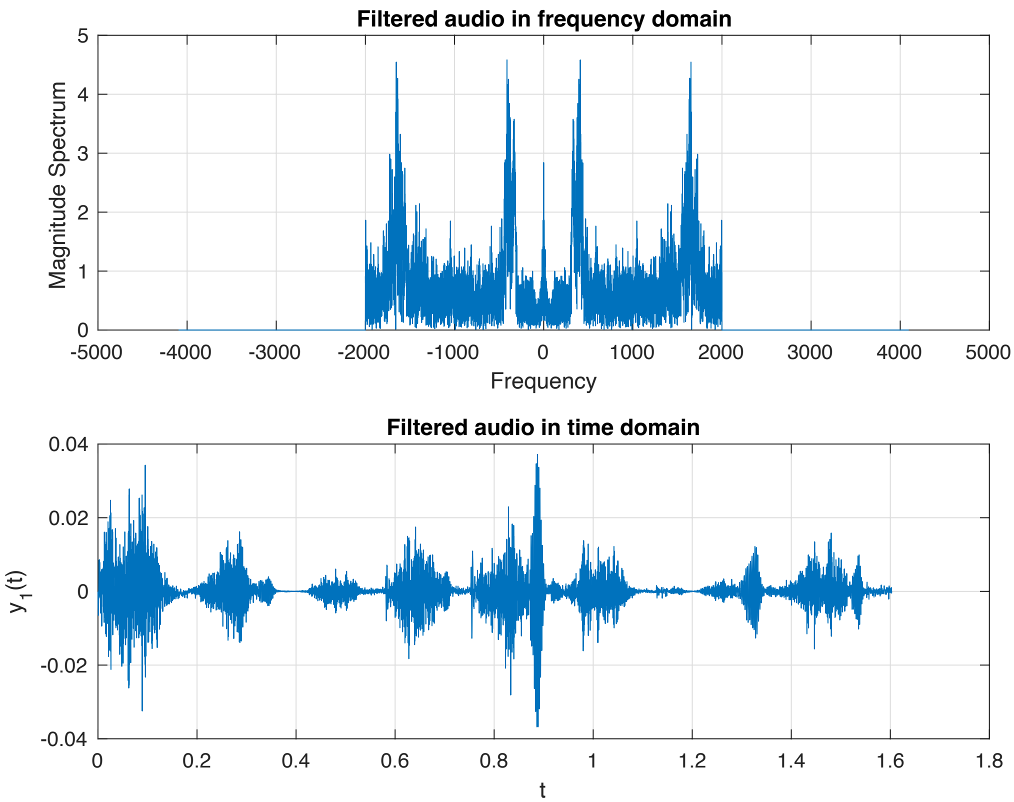



```
% sound(y5, fs5)

% The audio signal after downsampling by factor of "5" has lost one
% fifth of the sample. Hence, the downsampling signal is dramatically
% different from the actual signal as it has too few samples to recover
% all original information. Additionally, the downsampling spectrum has
% its amplitude and frequency axis scaled by the factor of 1/5.
```

C. Filter design

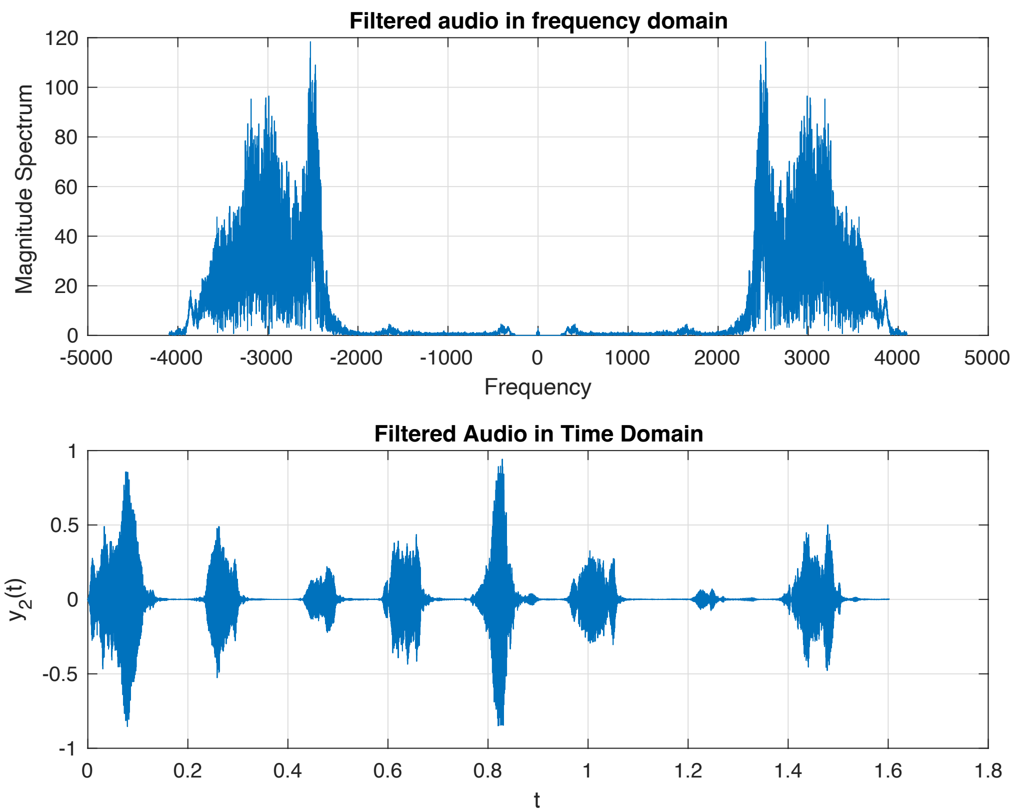
```
load chirp.mat;
filename = 'chirp.wav';
audiowrite(filename,y,Fs);
clear y Fs
[y,fs] = audioread('chirp.wav');

% C.1

N_0 = length(y); % Number of samples
T_0 = N_0/fs; % Duration of the signal
T = 1/fs; % Sampling interval

f = (0:N_0-1)/(T*N_0);
t = linspace(0, T_0, N_0);

% Compute DFT of signal y
Y = fftshift(fft(y));

% Create a rect filter that only passes frequencies less than 2000(Hz)
H1 = abs(f-fs/2) < 2000;
```

```
% Apply the filter
Y1_filtered = Y.*transpose(H1);
% Signal y after filtered
y1_filtered = real((ifft(fftshift(Y1_filtered))));

figure;
subplot(2,1,1);
plot(f-fs/2,abs(Y1_filtered));
grid;
title('Filtered audio in frequency domain');
xlabel('Frequency');
ylabel('Magnitude Spectrum');

subplot(2,1,2);
plot(t,y1_filtered);
grid;
title('Filtered audio in time domain');
xlabel('t');
ylabel('y_1(t)');
```



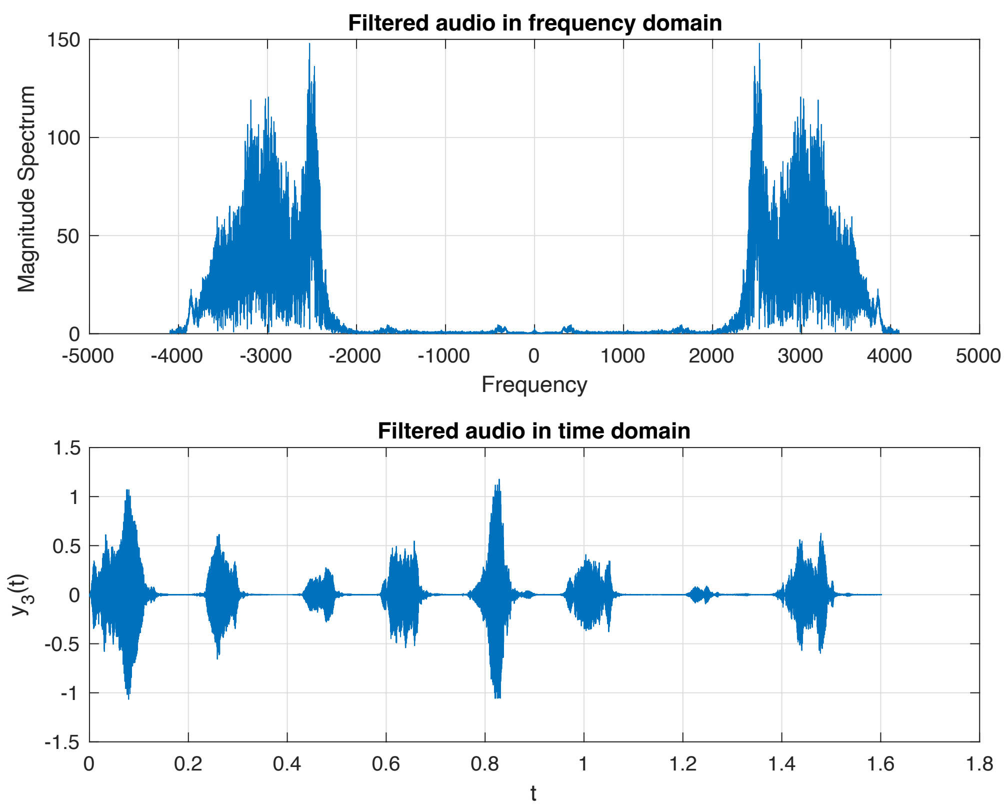

```
% C.2
% sound(y1_filtered, fs)

% The frequencies higher than +-2kHz got removed, so parts of the audio
% that contained those frequency component went silent.

% C.3

% Create a filter that cuts the bass frequencies
H2 = ~(abs(f-fs/2) > 16 & abs(f-fs/2) < 256);
```

```
% Apply the filter
Y2_filtered = Y.*transpose(H2);
% Signal y after filtered
y2_filtered = real((ifft(fftshift(Y2_filtered))));

figure;
subplot(2,1,1);
plot(f-fs/2,abs(Y2_filtered));
grid;
title('Filtered audio in frequency domain');
xlabel('Frequency');
ylabel('Magnitude Spectrum');

subplot(2,1,2);
plot(t,y2_filtered);
grid;
title('Filtered Audio in Time Domain');
xlabel('t');
ylabel('y_2(t)');
```




```
% sound(y2_filtered, fs)

% The low frequencies/bass sounds were removed, and replaced with silence.

% C.4

% Create a filter that amplifies treble frequencies
H3 = abs(f-fs/2) > 2048 & abs(f-fs/2) < 16384;
% Apply the filter
Y3_filtered = Y + Y.*transpose(H3)*0.25;
```

```
% Signal y after filtered
y3_filtered = real((ifft(fftshift(Y3_filtered))));

figure;
subplot(2,1,1);
plot(f-fs/2,abs(Y3_filtered));
grid;
title('Filtered audio in frequency domain');
xlabel('Frequency');
ylabel('Magnitude Spectrum');

subplot(2,1,2);
plot(t,y3_filtered);
grid;
title('Filtered audio in time domain');
xlabel('t');
ylabel('y_3(t)');
```




```
% sound(y3_filtered, fs)

% The parts of audio where contain treble frequencies were amplified
% and got louder.

% C.5

% In task (4), the linearity property of DFT was used which first filtered
% out the treble frequencies and multiplied by 25%. This version of treble
% frequencies was added back to the original audio spectrum. Therefore,
% it creates a new audio amplified by 25% at treble frequencies.
```