[redacted]

EEE391 - P1

1) Using "Record.m" script, three different notes (La, Si^b, Do) are recorded and saved to the file "Notes.mat". La note is selected to be used as "sound Array" and it is plotted vs time.

2) According to the data on the website, note La (A_4) has a frequency of 440 Hz, therefore fundamental period becomes $\frac{1}{440}$ s $\approx$ 2e-3 s.

partOfSoundArray is chosen between 1.3-4 seconds of soundArray, considering decay and fundamental frequency. When zoomed in, periodic behaviour for every 2e-3 s can be seen clearly.

4) fPPSA is chosen according to $T_f = 2e\text{-}3$ and FSCs are calculated by formula; $a_k = \frac{1}{T_f} \int_0^T fPPSA \cdot e^{-j\omega_0 k t} \cdot dt$

In calculation N is selected as the length of array for convenience.

5) Using calculated a_k values, fundamental signal is resynthesized using the formula: $rFPSA = \sum_k a_k \cdot e^{j\omega_0 k t}$. Also, the amplitudes are normalized by a factor to correct amplitude change for any given N.

6) An audio file is generated and when compared with the original audio, it is heard as if the frequency increased. To investigate this problem, MATLAB function fft is used. And when looked at the plot, there is a noise around 20-25Hz which is conveniently matching with the fan speed of my laptop.

This is the only reason that the sound frequency is heard to be changed.

7) While using part of a_k s it is found that for different k s;

$$
\begin{aligned}
&1 \rightarrow a_0 \\
&2 \rightarrow a_{-1}\ a_0\ a_1 \\
&3 \rightarrow a_{-1}\ a_0\ a_1
\end{aligned}
\Bigg\}\ \text{same}
$$

$$
\begin{aligned}
&4 \rightarrow a_{-2}\ a_{-1}\ a_0\ a_1\ a_2 \\
&5 \rightarrow a_{-2}\ a_{-1}\ a_0\ a_1\ a_2
\end{aligned}
\Big\}\ \text{same}
$$

same calculations had to be repeated, instead these calculations are skipped by making increment = 2.

Then the sound files are written with their relevant names. When these sound files are listened consequtively, it can be commented that as the number of non-zero a_k increases, sound converges to original sound.

8) The sound amplitude is greatly amplified and therefore distorted in general. It is expected since we make all coefficients bigger.

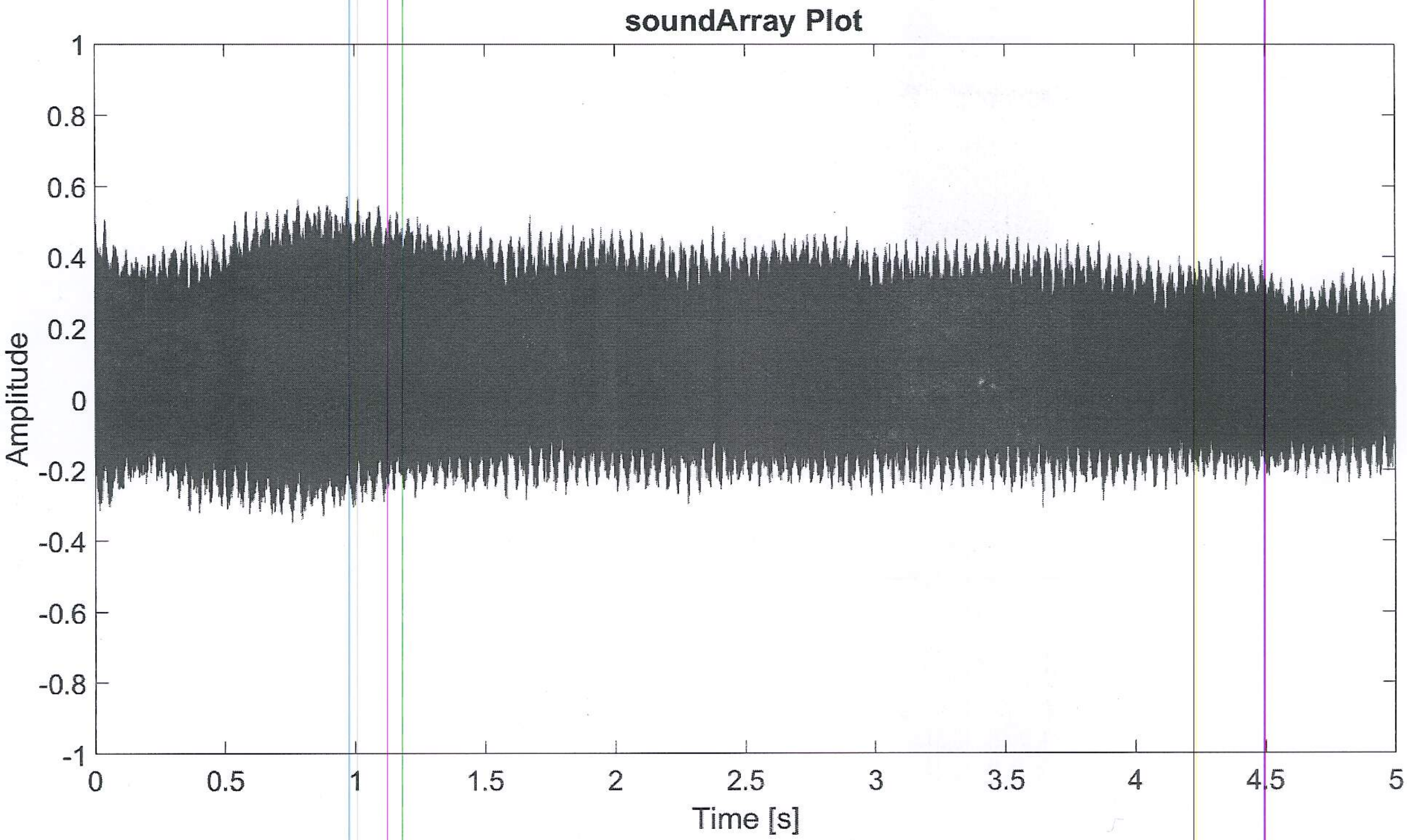
9) Since a_k's are complex magnitudes and their magnitudes are the same, sound level did not change. Moreover, sound does not seem to be changed at all when compared with section 5.

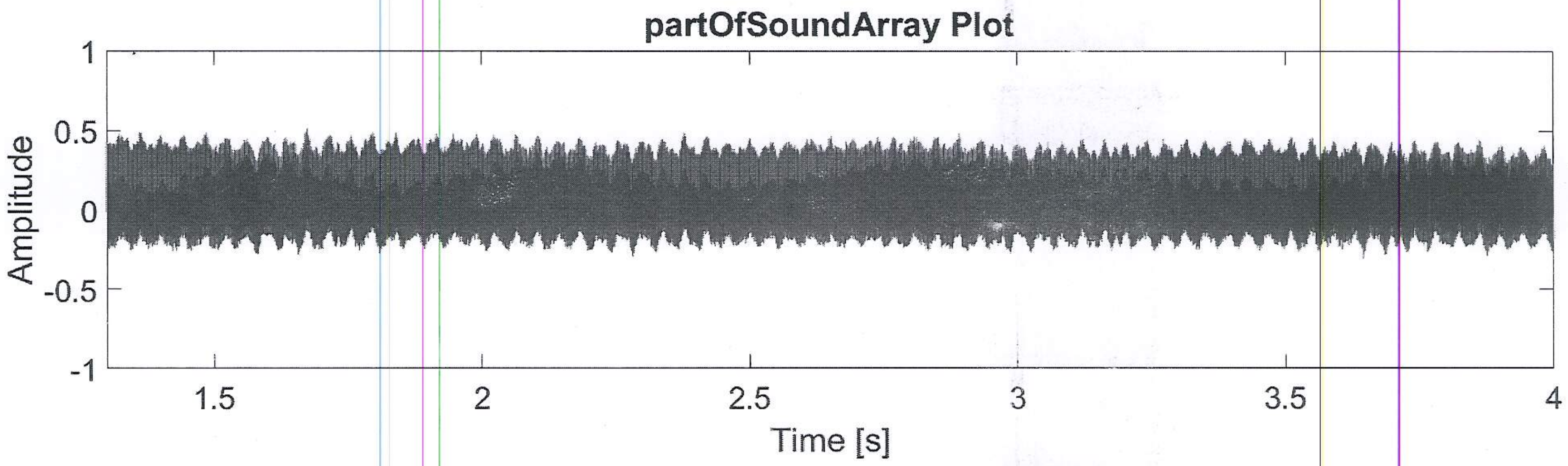
```
%% EEE391 - Assignment 1
%
%  This code is written for EEE391 - Basics of Signals and Systems course
%  Assignment 1. It takes a note as a sound array and applies processes
%  described in Assignment 1 document.
%
%  Author: Berkay Þahinoðlu
%  Update: 12.11.2017

close all
clear
clc

% Define Parameters

Fs  = 8000; % Sampling Frequency, Hz
t_r = 5;    % Recording Time, s
t_s = 1.3;  % Part starting time, s
t_e = 4.0;  % Part ending time, s

%% Voice Recording

% Use previously recorded audio
load('Notes.mat','pureLa'); % Pure La, clarinet
sA = pureLa; clear pureLa;  % Hold as sound array

L  = length(sA);     % Length of soundArray
dt = t_r/L;          % Time increment
t  = 0:dt:t_r-dt;    % Time array

% Plot soundArray
figure(1)
plot(t,sA)
xlabel('Time [s]')
ylabel('Amplitude')
title('soundArray Plot')
axis([0 t_r -1 1])
set(gca,'FontSize',24)
set(gcf,'Units','pixels','Position',[0 0 1920 1080])
print('Original Sound Array','-dpng','-r0')

%% Fundamental Period Calculation

% Extract partOfSoundArray
t_p = t_s:dt:t_e-dt; % Time array for part
i_t = round(Fs*t_p); % Array index for part, round is used to fix MATLAB
pSA = sA(i_t);       % Part of sound array

% Plot partOfSoundArray
figure(2)
subplot(2,1,1)
plot(t_p,pSA)
xlabel('Time [s]')
ylabel('Amplitude')
title('partOfSoundArray Plot')
axis([t_s t_e -1 1])
set(gca,'FontSize',24)
set(gcf,'Units','pixels','Position',[0 0 1920 1080])
print('Part of Sound Array','-dpng','-r0')
```

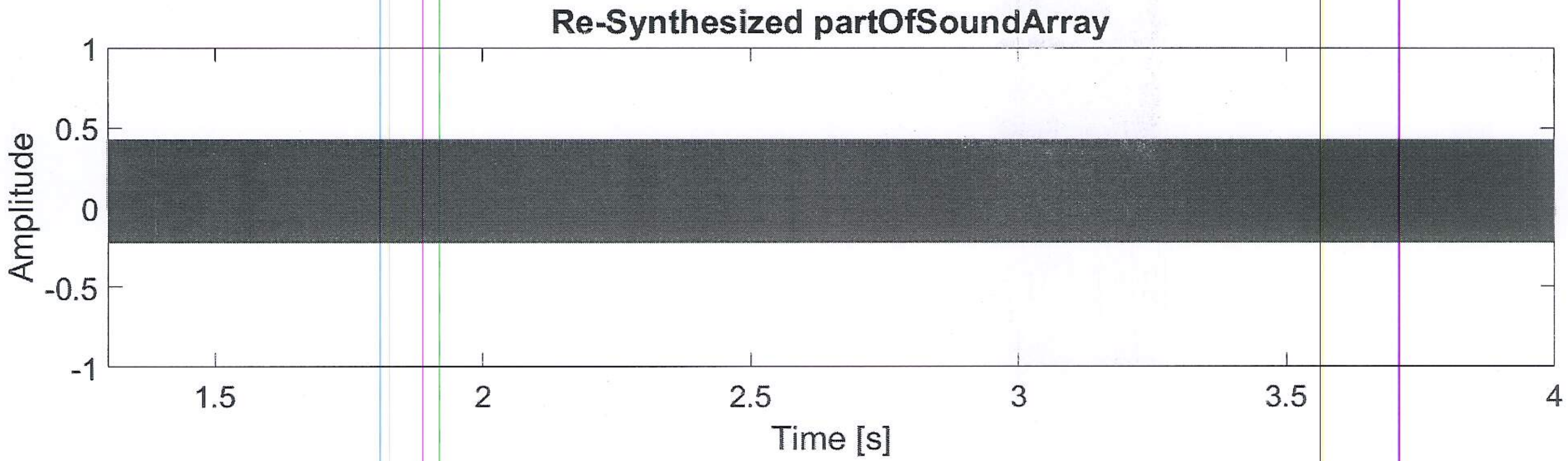
```
%% Fourier Series Analysis

% Extract firstPeriodOfPartOfSoundArray
T_f   = 0.002;                   % Fundamental Period, s
t_fp = (t_s:dt:t_s+T_f-dt)';    % Time array for first period
fPSA = pSA(1:T_f*Fs);           % firstPeriodOfPartOfSoundArray

% Plot firstPeriodOfPartOfSoundArray
figure(3)
subplot(2,1,1)
plot(t_fp,fPSA,'LineWidth',3)
xlabel('Time [s]')
ylabel('Amplitude')
title('firstPeriodOfPartOfSoundArray Plot')
axis([t_s t_s+T_f -1 1])
set(gca,'FontSize',24)
set(gcf,'Units','pixels','Position',[0 0 1920 1080])

N    = length(fPSA)/2;    % Number of coefficients
a    = zeros(2*N,1);      % Pre-allocate for speed
w_0  = 2*pi/T_f;          % Frequency in radian

for k = -N:N
    integ     = fPSA .* exp(-1j*w_0*k*t_fp);
    a(k+N+1)  = 1/T_f * trapz(t_fp,integ);
end

% Plot A_k values for firstPeriodOfPartOfSoundArray
figure(4)
stem(-N:N,abs(a),'LineWidth',3)
xlabel('Index K')
ylabel('Amplitude')
title('FSC Values')
set(gca,'FontSize',24)
set(gcf,'Units','pixels','Position',[0 0 1920 1080])
print('FSC Values','-dpng','-r0')

%% Fourier Series Synthesis

% Add terms according to series formula
rFPSA = 0;
for k = -N:N
    rFPSA = rFPSA + a(k+N+1) * exp(1j*w_0*k*t_fp);
end

rFPSA = length(fPSA)/(2*N) * real(rFPSA); % Remove imaginary and normalize

% Plot Re-Synthesized firstPeriodOfPartOfSoundArray
figure(3)
subplot(2,1,2)
plot(t_fp,rFPSA,'LineWidth',3)
xlabel('Time [s]')
ylabel('Amplitude')
title('Re-Synthesized firstPeriodOfPartOfSoundArray')
axis([t_s t_s+T_f -1 1])
set(gca,'FontSize',24)
set(gcf,'Units','pixels','Position',[0 0 1920 1080])
print('Fundamental Comparison','-dpng','-r0')
```

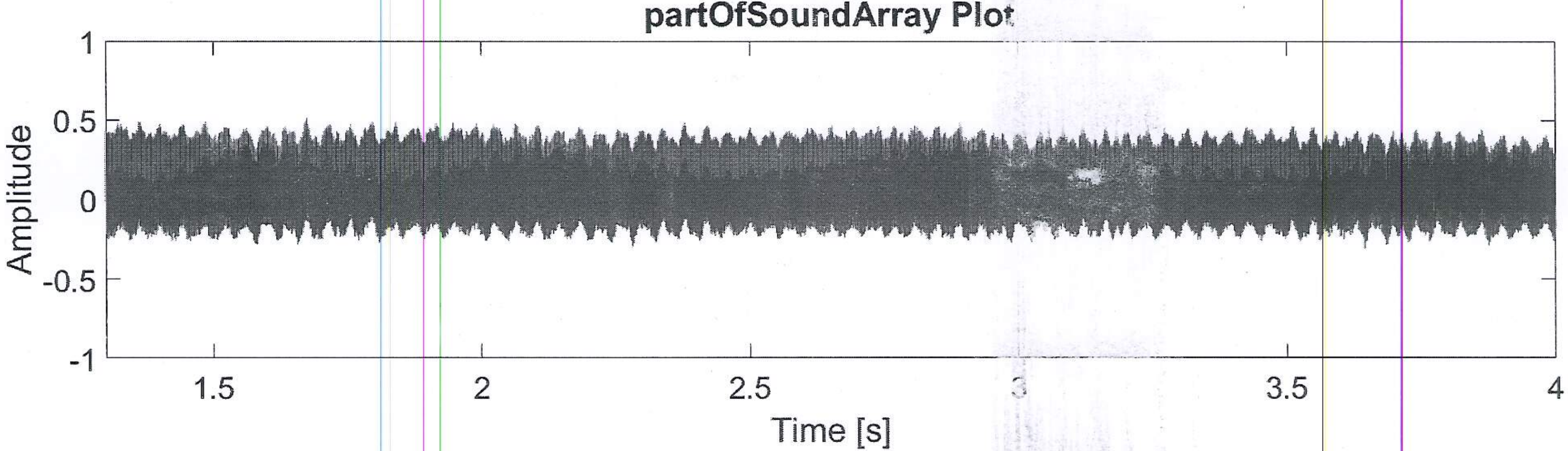
```
% Extend signal to cover partOfSoundArray
rPSA = repmat(rFPSA,ceil(length(pSA)/length(rFPSA)),1);
rPSA = rPSA(1:length(t_p));

% Plot Re-Synthesized partOfSoundArray
figure(2)
subplot(2,1,2)
plot(t_p,rPSA)
xlabel('Time [s]')
ylabel('Amplitude')
title('Re-Synthesized partOfSoundArray')
axis([t_s t_e -1 1])
set(gca,'FontSize',24)
set(gcf,'Units','pixels','Position',[0 0 1920 1080])
print('Part Comparison','-dpng','-r0')

%% Voice File Generation

% If you wished to listen sound on the go
% sound(pSA,Fs,16)
% pause(5)
% sound(rPSA,Fs,16)

audiowrite('originalSound.flac',sA,Fs,'BitsPerSample',16)
audiowrite('reconstructedSound.flac',rPSA,Fs,'BitsPerSample',16)

% Find and Plot Two Sided Fourier Transform to See Noise
dF  = Fs/length(sA);                   % Frequency increment
f   = -Fs/2:dF:Fs/2;                   % Frequency range of array
FFT = fftshift(fft(sA,length(f)));     % Obtain and shift FFT

% For comparison
figure(6)
plot(f,abs(FFT)/length(f))
title('FFT by MATLAB')
xlabel('Frequency [Hz]')
ylabel('Amplitude')
set(gca,'FontSize',24)
set(gcf,'Units','pixels','Position',[0 0 1920 1080])
print('MATLAB FFT','-dpng','-r0')
```

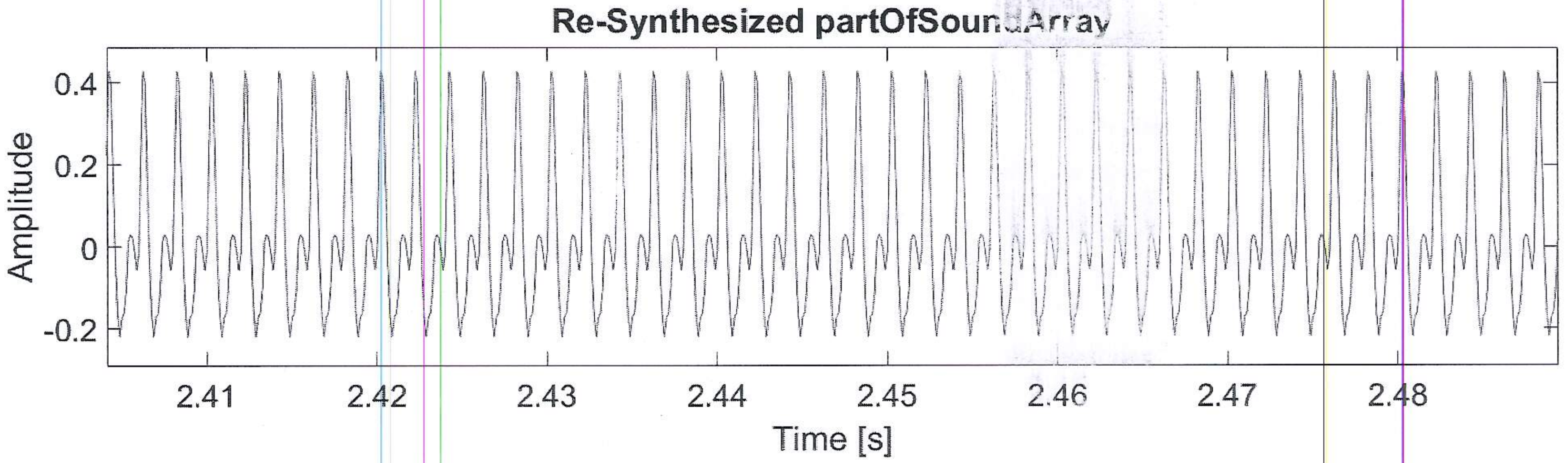
```
%% Synthesis of the Voice from Partial FSCs

for i = 2:2:2*N % Since for k(N_i=2n+1) and k(N_i=2n+2) are the same

   k     = unique(fix(-(i-1)/2:(i-1)/2));  % Find the indeces
   a_p   = [zeros((length(a)-i+1)/2,1);...
            a(k+floor(length(a)/2)+1);...
            zeros((length(a)-i+1)/2,1)];   % Get partial FSCs

   % Calculate reconstructed partial fundamental part of sound array
   rPFPSA = 0;
   for l = -N:N
       rPFPSA = rPFPSA + a_p(l+N+1) * exp(1j*w_0*l*t_fp);
   end

   % Remove imaginary and normalize
   rPFPSA = length(fPSA)/(2*N) * real(rPFPSA);

   % Construct part of sound array
   rPPSA = repmat(rPFPSA,ceil(length(pSA)/length(rPFPSA)),1);
   rPPSA = rPPSA(1:length(t_p));

   name = ['Num_Ak_' num2str(length(k)) '.flac']; % Generate a name
   audiowrite(name,rPPSA,Fs,'BitsPerSample',16)    % Write audio
end

%% Synthesis of the Voice from FSCs whose Magnitudes are equated to one

s = 1./abs(a);  % Scaling factor to equate magnitudes to 1
a_m1 = s.*a;    % Scale the a_k values

rM1FPSA = 0;
for l = -N:N
   rM1FPSA = rM1FPSA + a_m1(l+N+1) * exp(1j*w_0*l*t_fp);
end

% Remove imaginary and normalize
rM1FPSA = length(fPSA)/(2*N) * real(rM1FPSA);

% Construct part of sound array
rM1PSA = repmat(rM1FPSA,ceil(length(pSA)/length(rM1FPSA)),1);
rM1PSA = rM1PSA(1:length(t_p));

audiowrite('Magnitude_1.flac',rM1PSA,Fs,'BitsPerSample',16) % Write audio

%% Synthesis of the Voice from FSCs whose Phases are equated to zero

a_p0 = abs(a);  % Phase = 0 mean only the magnitude remains

rP0FPSA = 0;
for l = -N:N
   rP0FPSA = rP0FPSA + a_p0(l+N+1) * exp(1j*w_0*l*t_fp);
end

% Remove imaginary and normalize
rP0FPSA = length(fPSA)/(2*N) * real(rP0FPSA);

% Construct part of sound array
rP0PSA = repmat(rP0FPSA,ceil(length(pSA)/length(rP0FPSA)),1);
rP0PSA = rP0PSA(1:length(t_p));

audiowrite('Phase_0.flac',rP0PSA,Fs,'BitsPerSample',16) % Write audio
```

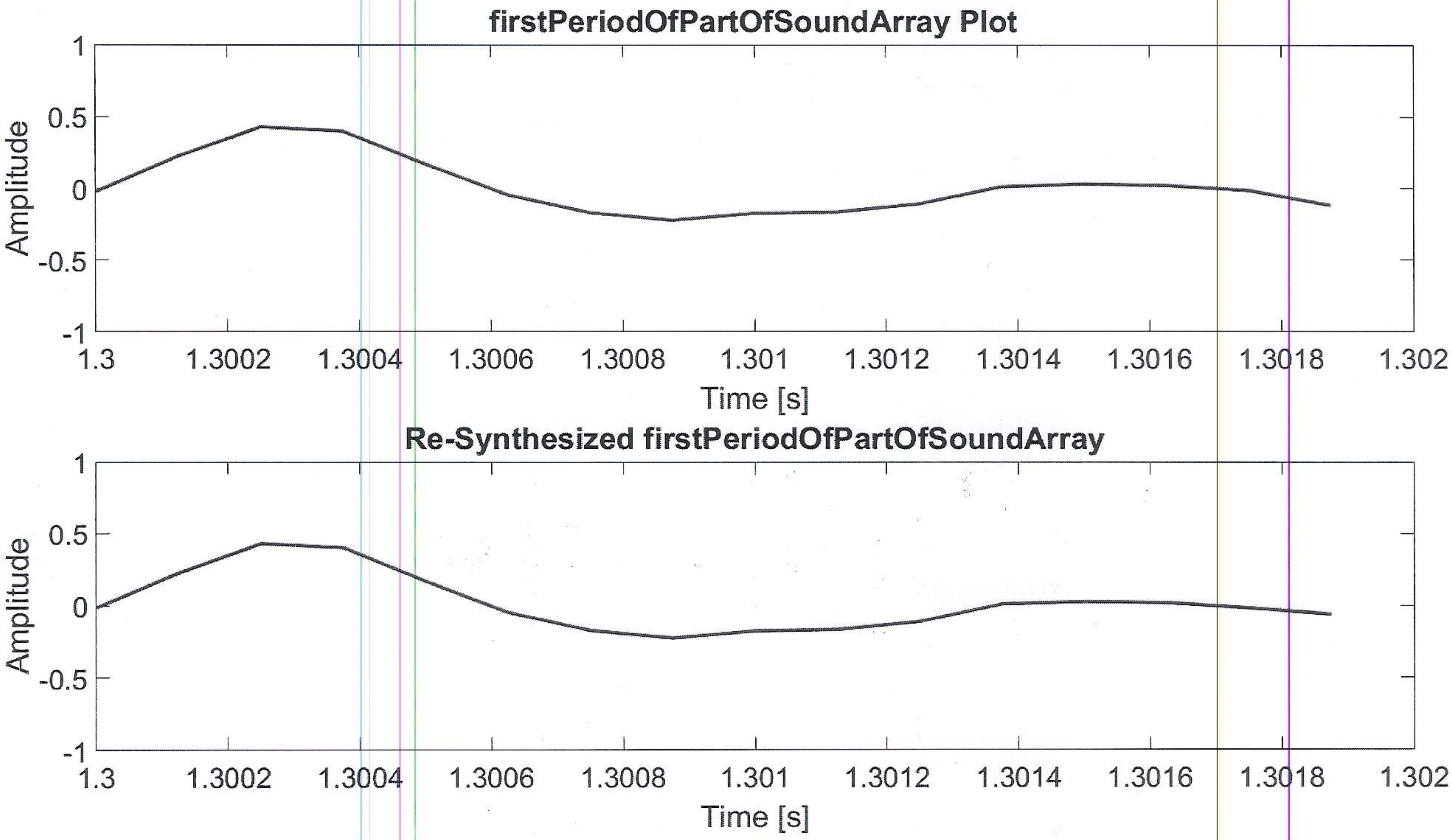

soundArray Plot
Amplitude
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time [s]

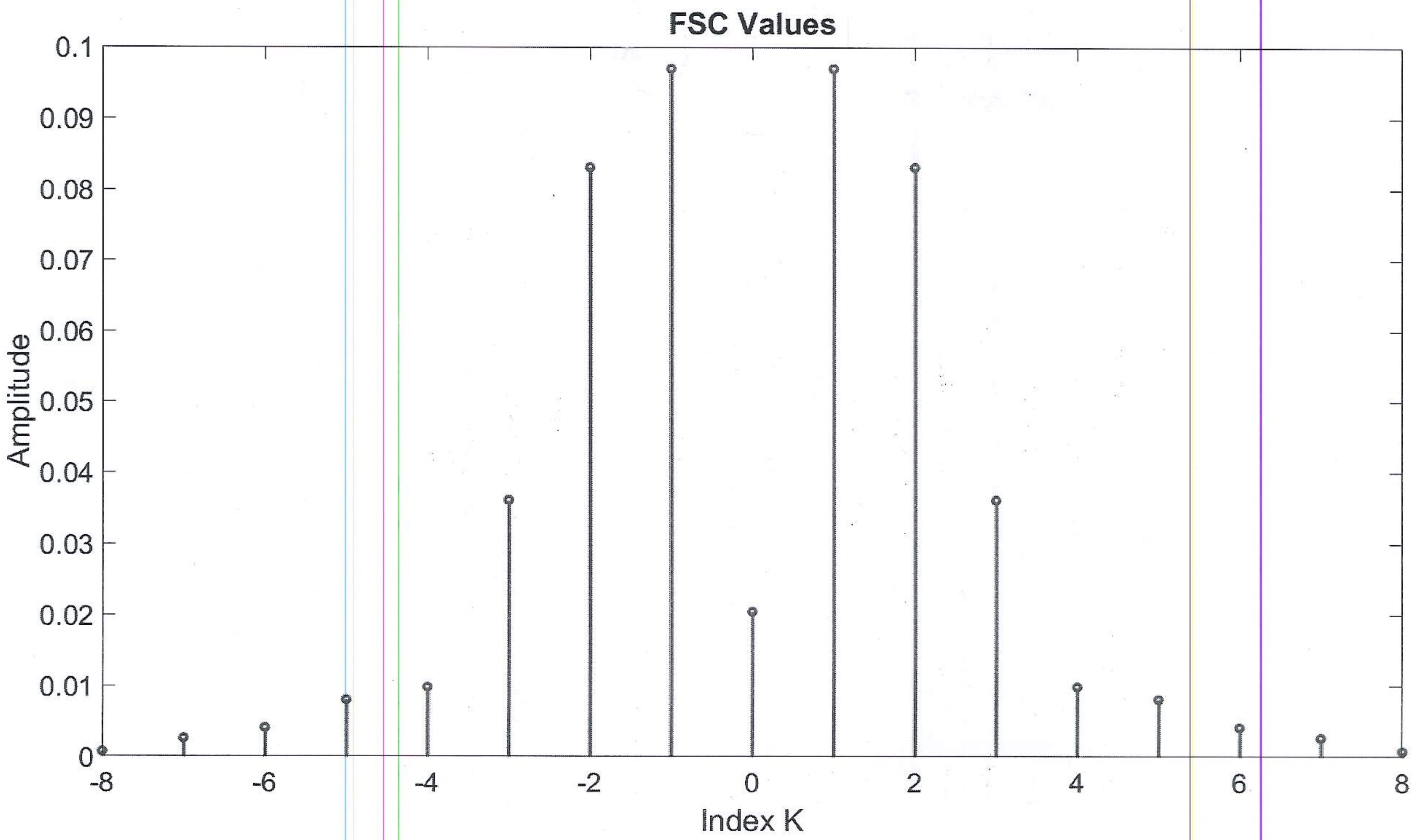

partOfSoundArray Plot
Amplitude
1
0.5
0
-0.5
-1
1.5
2
2.5
3
3.5
4
Time [s]

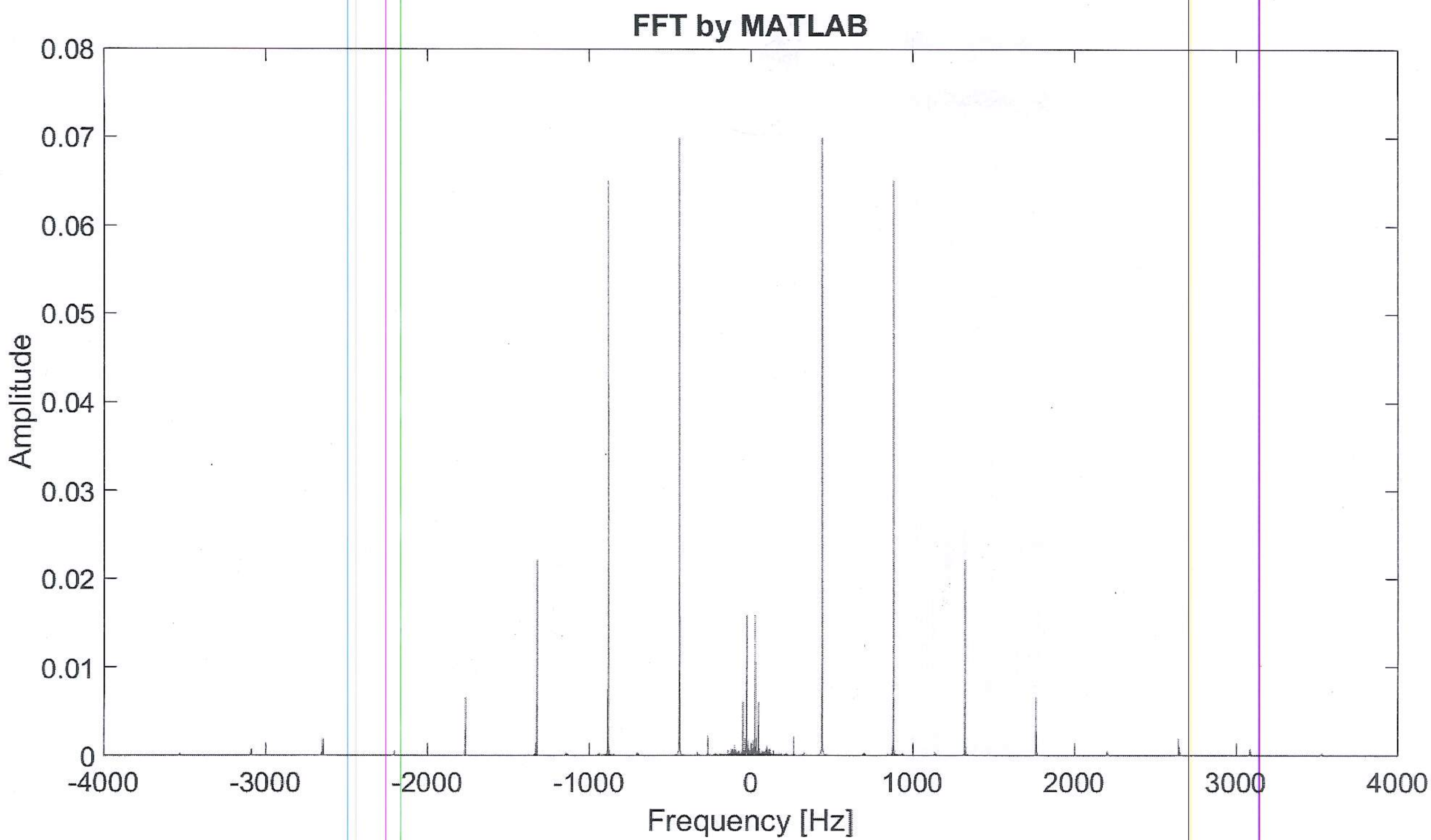

Re-Synthesized partOfSoundArray
Amplitude
1
0.5
0
-0.5
-1
1.5
2
2.5
3
3.5
4
Time [s]


partOfSoundArray Plot
Amplitude
1
0.5
0
-0.5
-1
1.5
2
2.5
3
3.5
4
Time [s]


Re-Synthesized partOfSoundArray
Amplitude
0.4
0.2
0
-0.2
2.41
2.42
2.43
2.44
2.45
2.46
2.47
2.48
Time [s]


firstPeriodOfPartOfSoundArray Plot
Amplitude
1
0.5
0
-0.5
-1
1.3
1.3002
1.3004
1.3006
1.3008
1.301
1.3012
1.3014
1.3016
1.3018
1.302
Time [s]
Re-Synthesized firstPeriodOfPartOfSoundArray
Amplitude
1
0.5
0
-0.5
-1
1.3
1.3002
1.3004
1.3006
1.3008
1.301
1.3012
1.3014
1.3016
1.3018
1.302
Time [s]


FSC Values
Amplitude
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
-8
-6
-4
-2
0
2
4
6
8
Index K


FFT by MATLAB
Amplitude
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
-4000
-3000
-2000
-1000
0
1000
2000
3000
4000
Frequency [Hz]