EEE – 391

Signals and Systems

MATLAB Assignment #1

Kerem Ayöz

21501569 - Section: 2

2. Fundamental Period Calculation

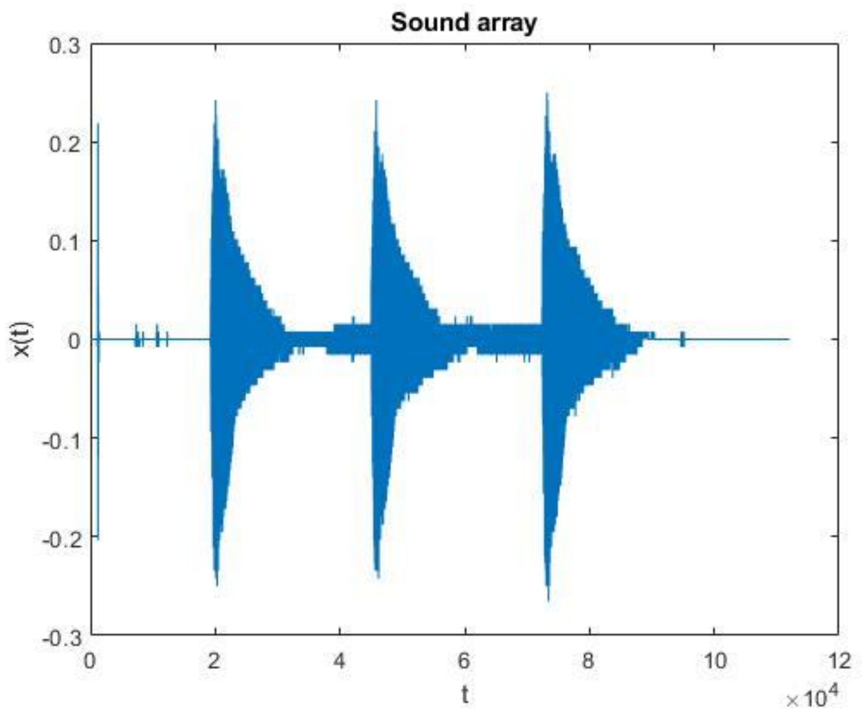

Fig.1: Original soundArray that I generated by playing the note A4.

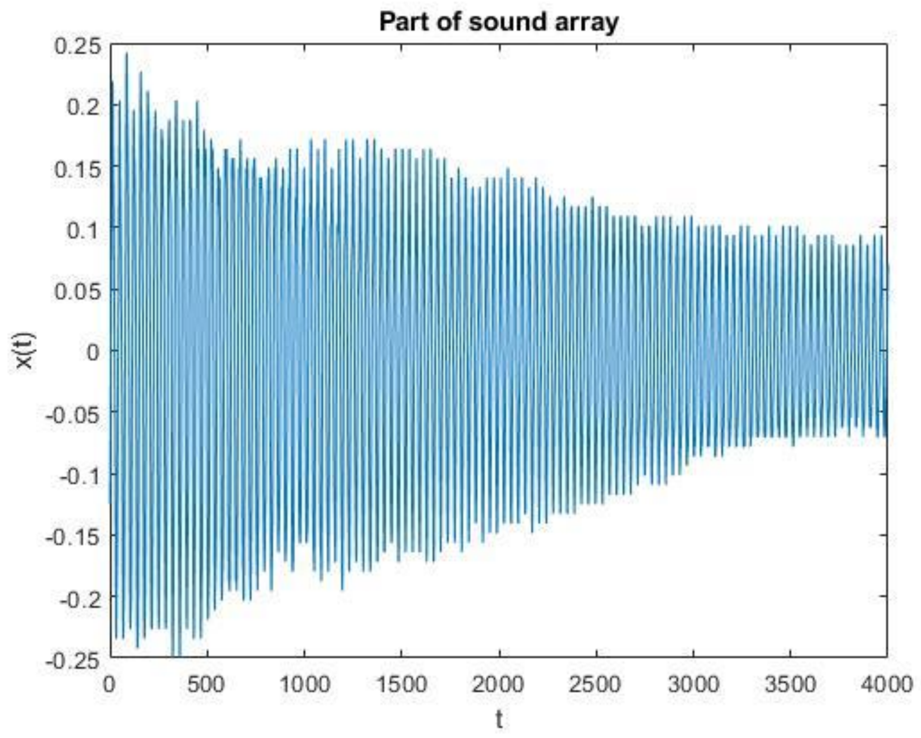

Fig.2: Cropped version of the soundArray to obtain single note with less decay.

Code:

```
recObj = audiorecorder(16000, 8, 1);
disp('Start recording.');
recordblocking(recObj, 7);
disp('End of Recording.');
play(recObj);

soundArray = getaudiodata(recObj);
%plot(soundArray)

% Crop the beginning and ending silence
partOfSoundArray = crop(soundArray);
partOfSoundArray = partOfSoundArray(1:4000);
plot(partOfSoundArray)
title('Part of sound array');
xlabel('t');
ylabel('x(t)');

% Note played: A4
fund_freq = 440;
fund_per = 1 / fund_freq;
period =  floor(16000 / fund_freq);
```

I played the note A4 from the virtual piano on the internet. After that I cropped the soundArray to get the part that I want. The fundamental frequency of the note A4 is 440.00 Hz. Fundamental period is 0.002273 seconds. However, since its sampled with the T_s = 16.000 Hz, the period size for the array is 16000 * 0.002273 which is approximately 36.

3. Fourier Series Analysis

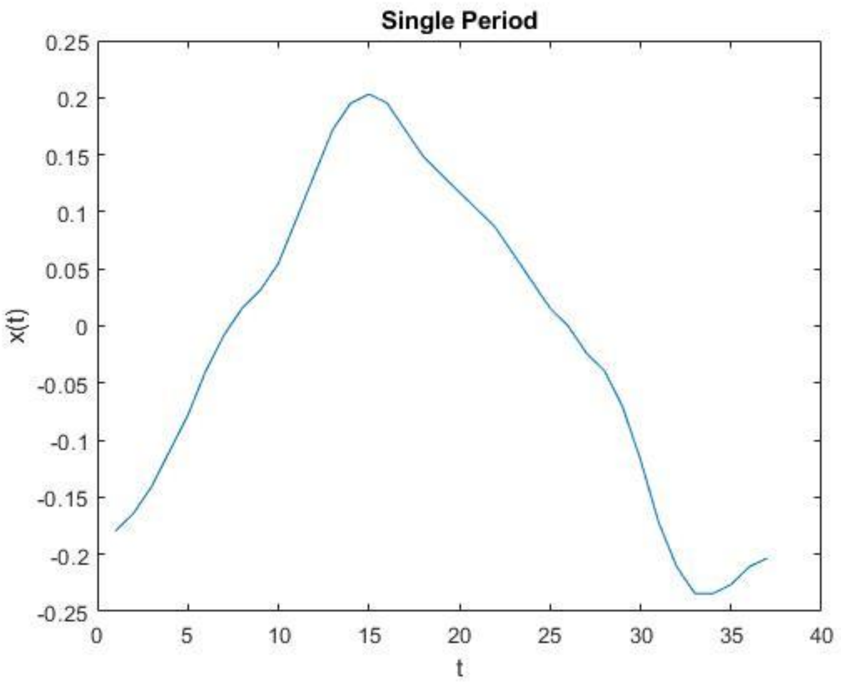

Fig.3: Signal in the one period as a function of time.

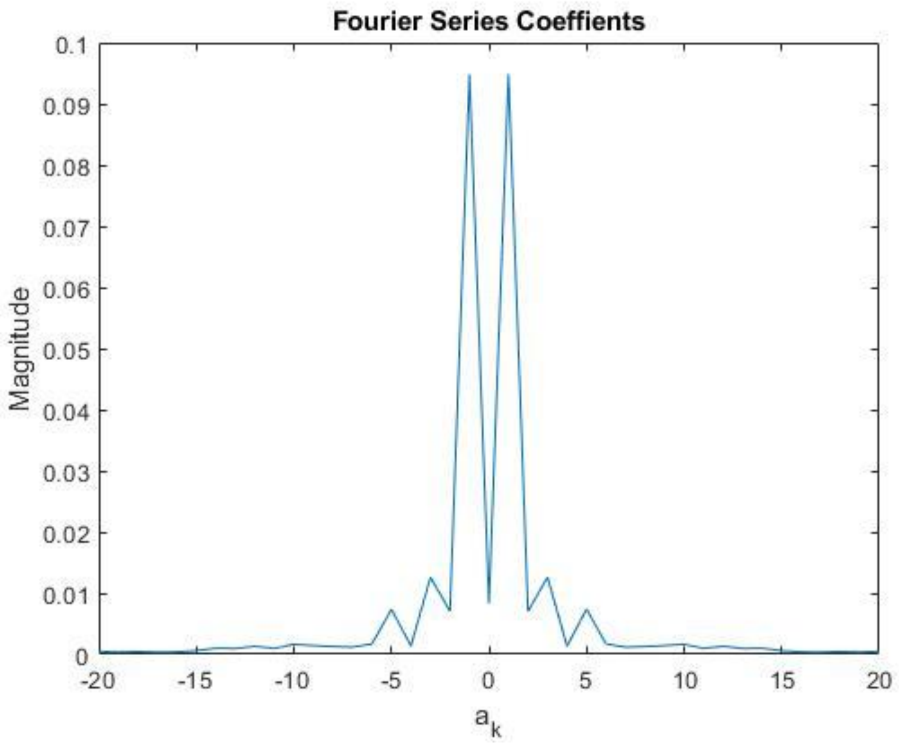

Fig.4: Coefficients found from the FCS as a functions of k.

Code:

```
singlePeriod = partOfSoundArray(1*period:2*period);
%     figure;
% plot(singlePeriod)
% title('Single Period');
% xlabel('t');
% ylabel('x(t)');



N = 20;
coef = zeros(1,2*N+1);
t = transpose(0:1/16000:(period)/16000);

for k = (-N:N)
    complexExp = exp(-1i * 2 * pi * (1 / fund_per) * k .* t);
    coef(k+N+1) = fund_freq.*trapz(t,singlePeriod.*complexExp);
end

coef_abs = abs(coef);
%     figure;
% plot(-N:N,coef_abs);
% title('Fourier Series Coeffients');
% xlabel('a_k');
% ylabel('Magnitude');
```

In a loop, I calculated the exponential term and multiplied it with the original signal. Then I used trapz which estimates the integrals in given intervals to find coefficients. Then I take the absolute value of the coefficients to calculate their magnitudes to show in the plot.

4. Fourier Series Synthesis

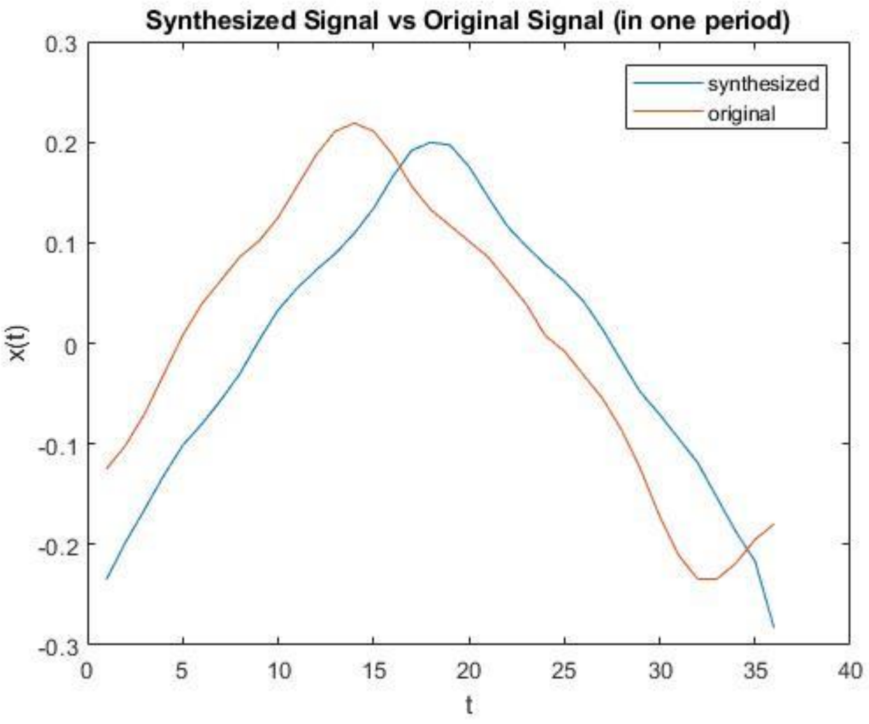

Fig.5: Synthesized signal vs Original Signal in one period time interval.

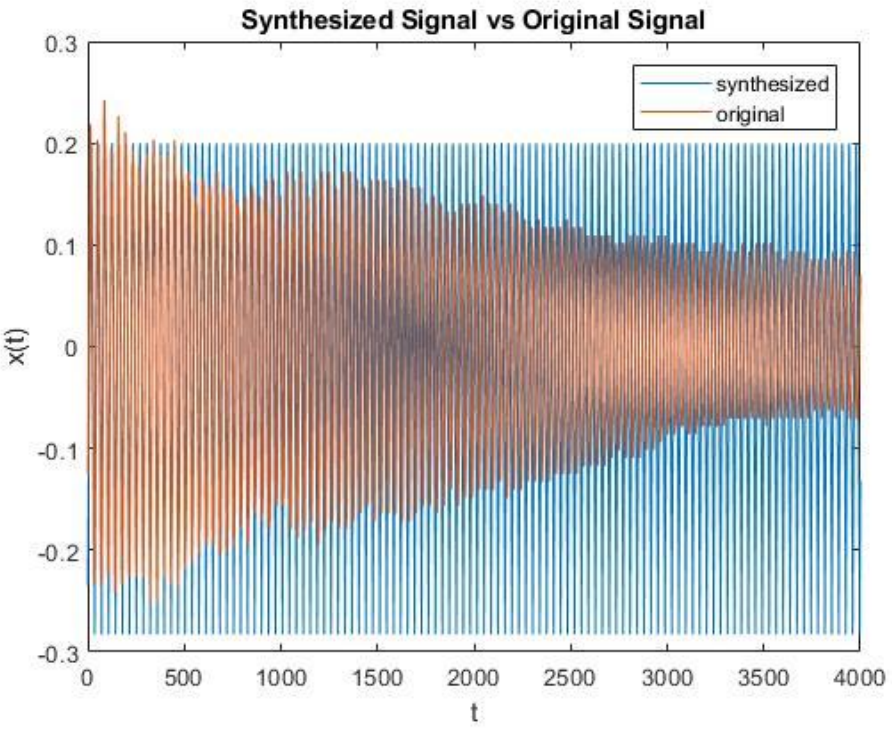

Fig.6: Synthesized signal vs Original Signal.

Code:

```
synthesized = zeros(1,period);

for ind = (1:period)
    sum_t = 0;
    for k = (-N:N)
       sum_t = sum_t + coef(k+N+1) * exp(1i * 2 * pi * (1 /
fund_per) * k * t(ind));
    end
    synthesized(ind) = real(sum_t);
end

synthSoundArray= 1:size(partOfSoundArray);
synthSoundArray = synthesized(mod(synthSoundArray,period)+1);


figure;
plot(synthSoundArray(1:4000));
hold;
plot(partOfSoundArray(1:4000));
title('Synthesized Signal vs Original Signal');
xlabel('t');
ylabel('x(t)');
legend('synthesized','original')

audiowrite('synthesized.wav',synthSoundArray,16000);
audiowrite('original.wav',partOfSoundArray,16000);
```

In order to synthesize the signal, I used the coefficients that I found in the previous part. For each unit time, I calculated the sum of the product of coefficient and the complex exponential. Then I do it for each unit time interval to obtain the synthesized sound in one period. Then I expanded the synthesized sound signal to size 4000 to match it with the original sound. Since it is one period length, I periodically expanded the signal. I also generated the voice file by using the audiowrite() function that is built in MATLAB.

6. Voice Synthesis from Partial FSCs

Code:

```
N1 = 1;
for N_i = (3:20)
    N_new =  ceil((N_i - N1) / 2.0);
    interval = -N_new : N_new;
    coef_interval = N - N_new+1 : N+1+N_new;
    coef_i = coef(coef_interval);
    synthesized = zeros(1,period);
    for ind = (1:period)
        sum_t = 0;
        for k = interval
           sum_t = sum_t + coef_i(k+N_new+1) * exp(1i * 2 * pi *
(1 / fund_per) * k * t(ind));
        end
        synthesized(ind) = real(sum_t);
    end

    synthSoundArray= 1:size(partOfSoundArray);
    synthSoundArray = synthesized(mod(synthSoundArray,period)+1);

     figure;
    plot(synthSoundArray(1:4000));
    hold;
    plot(partOfSoundArray(1:4000));
    title('Synthesized Signal vs Original Signal');
    xlabel('t');
    ylabel('x(t)');
    legend('synthesized','original')

    name = 'sythesized';
    name = strcat(name, num2str(N2));
    name = strcat(name, '.wav');
    audiowrite(name,synthSoundArray,16000);

end
```

As N2 goes 3 to 20, the sythesized sound becomes a better approximation for the original sound since we are adding more and more coefficients to estimate the original signal. Sound is changing when we add more coefficients. However, since the coefficients gets smaller and smaller, the changing effect get smaller since the most important coefficients are the one with high magnitude and in our case they are the first coefficients.

7. Voice Synthesis from FSCs with Single Magnitude

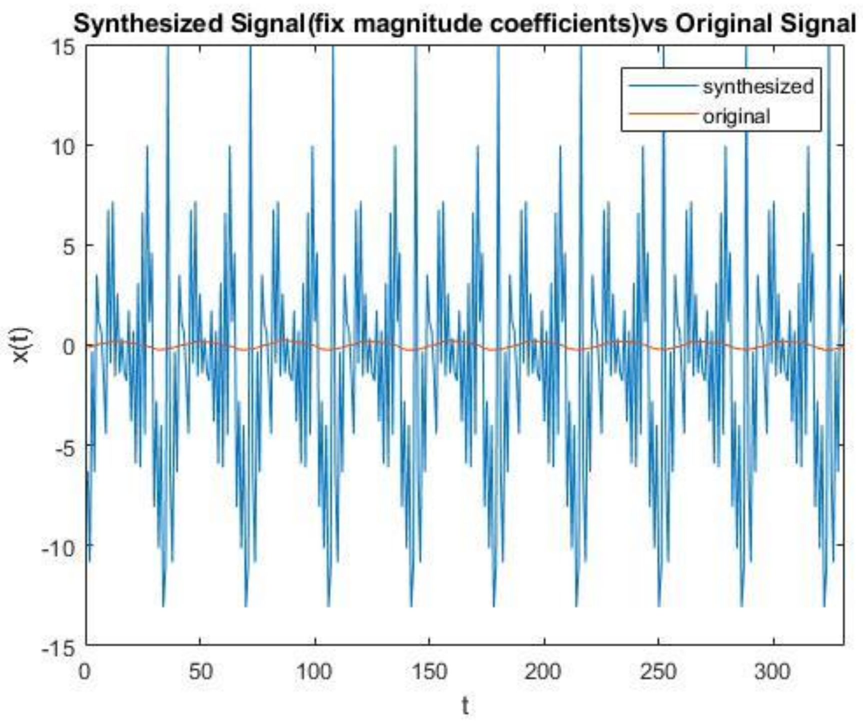

Fig.7: Sythesized signal with fixed magnitude coefficients.

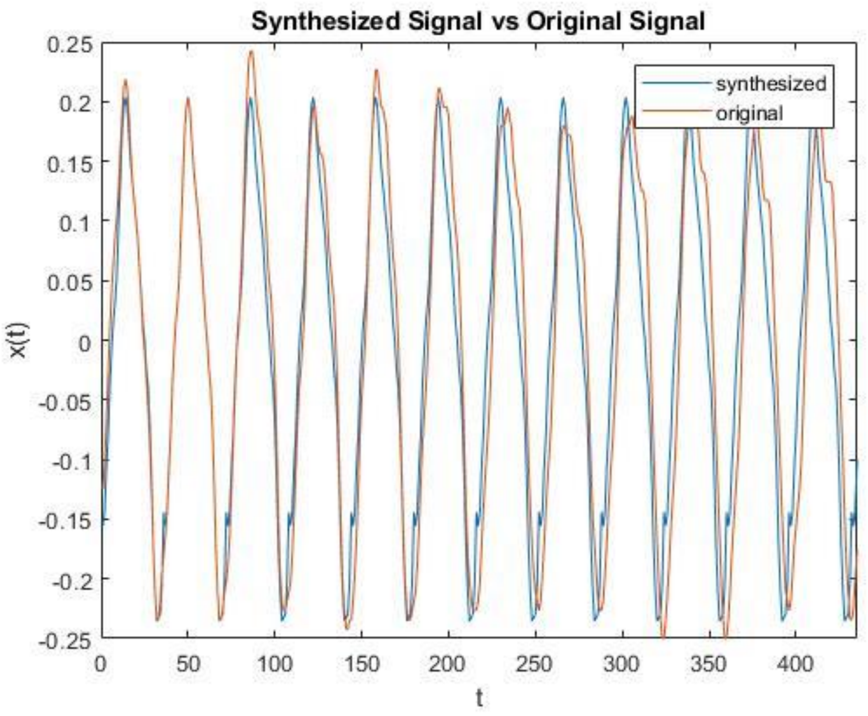

Fig.8: Sythesized signal with original coefficients.

Code:

```
coef_fix_mag = zeros(1,length(coef));
for i = (1:length(coef))
    coef_fix_mag(i) = coef(i) / abs(coef(i));
end
synthesized_fix_mag = zeros(1,period);

for ind = (1:period)
    sum_t = 0;
    for k = (-N:N)
       sum_t = sum_t + coef_fix_mag(k+N+1) * exp(1i * 2 * pi * (1
/ fund_per) * k * t(ind));
    end
    synthesized_fix_mag(ind) = real(sum_t);
end

synthSoundArray_fix_mag= 1:size(partOfSoundArray);
synthSoundArray_fix_mag =
synthesized_fix_mag(mod(synthSoundArray_fix_mag,period)+1);
figure;
plot(synthSoundArray_fix_mag(1:4000));
hold;
plot(partOfSoundArray(1:4000));
title('Synthesized Signal(fix magnitude coefficients)vs Original
Signal');
xlabel('t');
ylabel('x(t)');
legend('synthesized','original')
audiowrite('fix_mag.wav',synthSoundArray_fix_mag,16000);
```

If we change magnitudes of the coefficients to all one, we lose the information about the signal. Since, magnitudes of the coefficients differ for each sinusoidal in sum equating all of them to one cause loss of information. Synthesized signal is totally different from what it should be in the standard case. So, sound is also different from the sound that I give as input.

8. Voice Synthesis from FSCs with Zero Phase

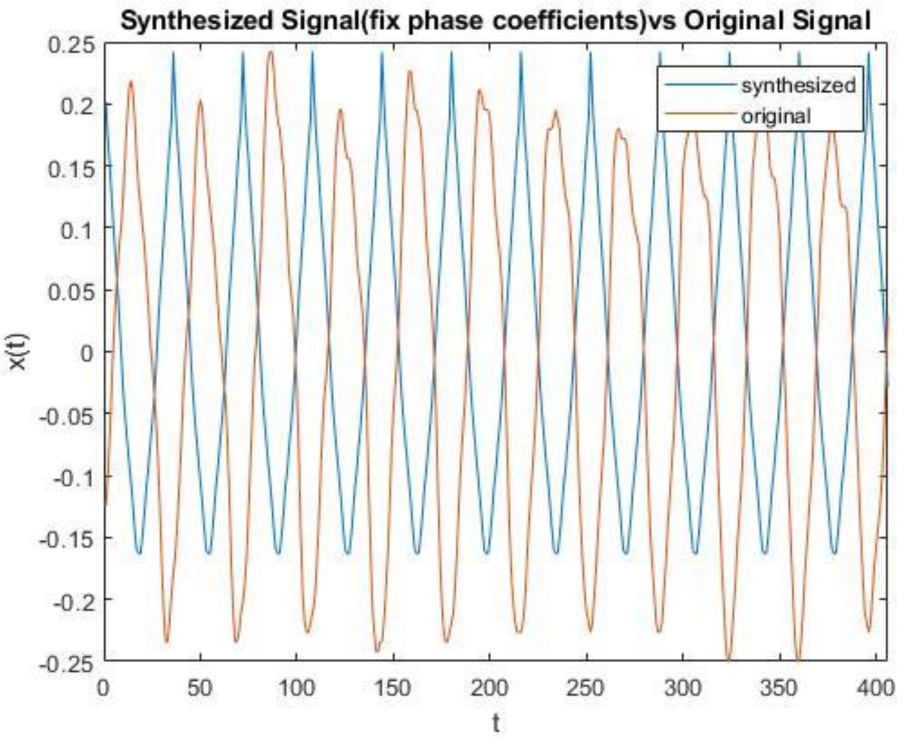

Fig.9: Sythesized signal with 0 phase coefficients.

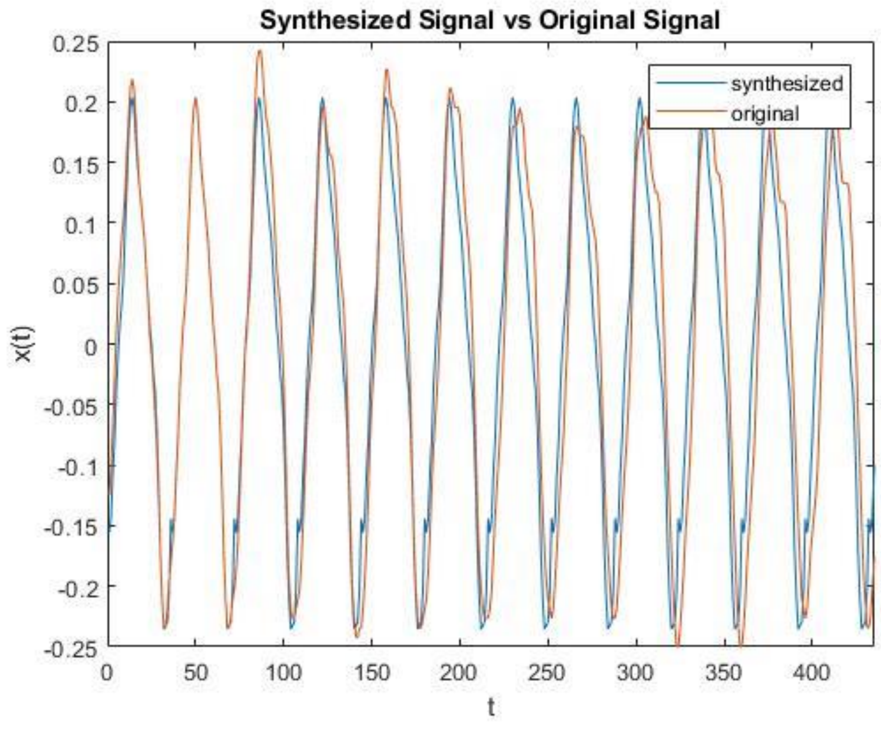

Fig.10: Sythesized signal with original coefficients.

Code:

```
coef_fix_phase = zeros(1,length(coef));
for i = (1:length(coef))
    coef_fix_phase(i) = abs(real(coef(i)));
end
angle(coef)
angle(coef_fix_phase)
synthesized_fix_phase = zeros(1,period);

for ind = (1:period)
    sum_t = 0;
    for k = (-N:N)
       sum_t = sum_t + coef_fix_phase(k+N+1) * exp(1i * 2 * pi *
(1 / fund_per) * k * t(ind));
    end
    synthesized_fix_phase(ind) = real(sum_t);
end

synthSoundArray_fix_phase= 1:size(partOfSoundArray);
synthSoundArray_fix_phase =
synthesized_fix_phase(mod(synthSoundArray_fix_phase,period)+1);
figure;
plot(synthSoundArray_fix_phase(1:4000));
hold;
plot(partOfSoundArray(1:4000));
title('Synthesized Signal(fix phase coefficients)vs Original
Signal');
xlabel('t');
ylabel('x(t)');
legend('synthesized','original')
audiowrite('fix_phase.wav',synthSoundArray_fix_phase,16000);
```

If we use 0 phase coefficients while sythesizing, we get the shifted version of the signal. Since we lose all the information about the phases of the added sinusoidals, we obtain the shifted version of the original sythesized signal. Also magnitudes change since addition of faulty phased coefficients may result in different magnitudes than the original. We obtain a wrong result however it is not as wrong as the fixed magnitude version. In my example, sythesized signal and original signal alternates. Sound is different but more similar than the fixed magnitude version.