# Lab 3 Report: Fourier Series and Gibbs Phenomenon

Youssef Beltagy and Samuel Hunter
AUT21 BEE 235

January 6, 2022

## 1 Abstract

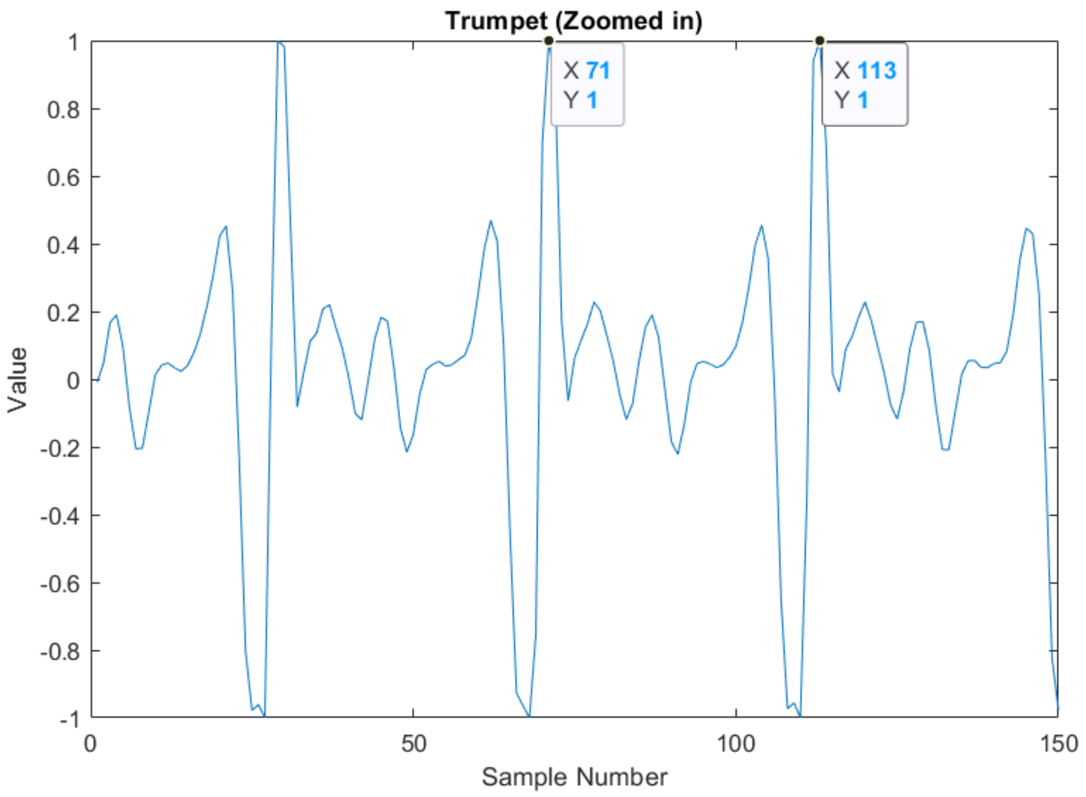
In this lab, we analyze a couple of signals and synthesize them using the fourier series. We analyzed a trumpet sound and a square wave signal. We observed and recorded how synthesized signals fare compared to original ones.

## 2 Part 1 — Signal Synthesis

In this section we analyze and synthesize a trumpet sound using fourier series.

### 2.1 Estimating the Fundamental Frequency

After loading and plotting the pre-recorded trumpet sound signal onto a graph, we measured the coordinates of two different peaks.

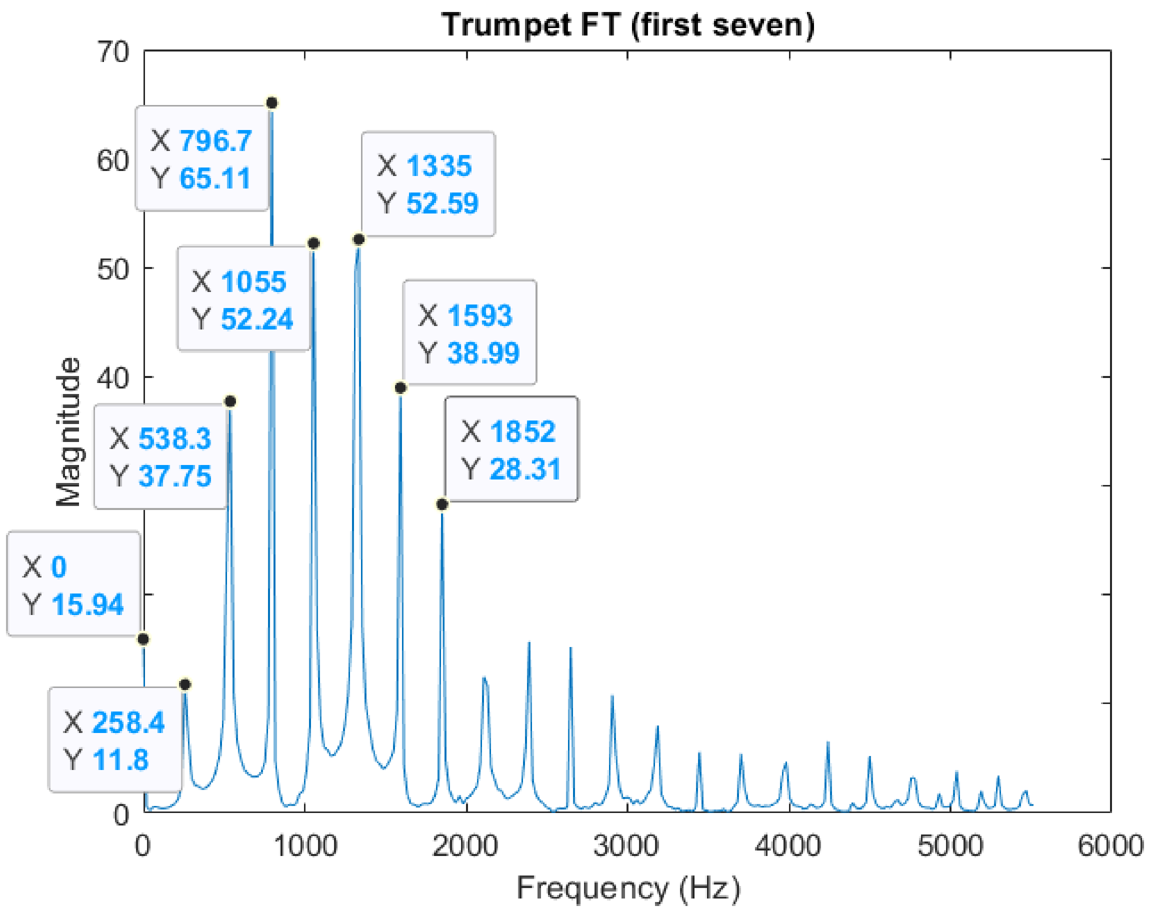


| Peak No. 1 | Peak No. 2 | Period (# of samples) | $F_0(Hz) = F_s/(\#\text{ of samples})$ |
|---|---|---|---|
| $X_1 = 71, Y_1 = 1.0000$ | $X_2 = 113, Y_1 = 1.0000$ | $\lvert X_2 - X_1 \rvert = 42$ | $F_0 = 11025/42 = 262.5\ Hz$ |

Table 1: Two sampled peaks of the trumpet sound signal

The difference between the two peaks is 42 samples. With a sample rate of $F_s = 11025\ Hz$, the difference is 262.5 $Hz$.

We then applied the Fourier transform on the trumpet signal and measured the magnitudes and frequency of the first seven harmonic peaks. We also included the 0th harmonic (the integral average) for reference.



| Harmonic | Magnitude | Frequency (Hz) | Frequency Diff (Hz) |
|---|---|---|---|
| 0 | 15.944 | 0 | N/A |
| 1 | 11.795 | 258.398 | 258.398 |
| 2 | 37.749 | 538.330 | 279.932 |
| 3 | 65.114 | 796.729 | 258.399 |
| 4 | 52.244 | 1055.13 | 258.40 |
| 5 | 52.593 | 1335.06 | 279.93 |
| 6 | 38.994 | 1593.46 | 258.40 |
| 7 | 28.305 | 1851.86 | 258.40 |

Table 2: 7 first harmoncs

The mean of these differences is $F_0 = 264.55$ Hz and the median of these differences is 258.40 Hz. The percent differences compared to our first estimation of 262.5 Hz is 0.78% and 1.6% for the mean and median respectively. The percent differences are small so the mean and median align well with our first estimate.

The harmonics all have frequencies which are multiples of $F_0 = 264.55$ Hz (approximately). So the average difference between two harmonics is 264.55. It seems musical sounds are made of multiple harmonics and not just one.

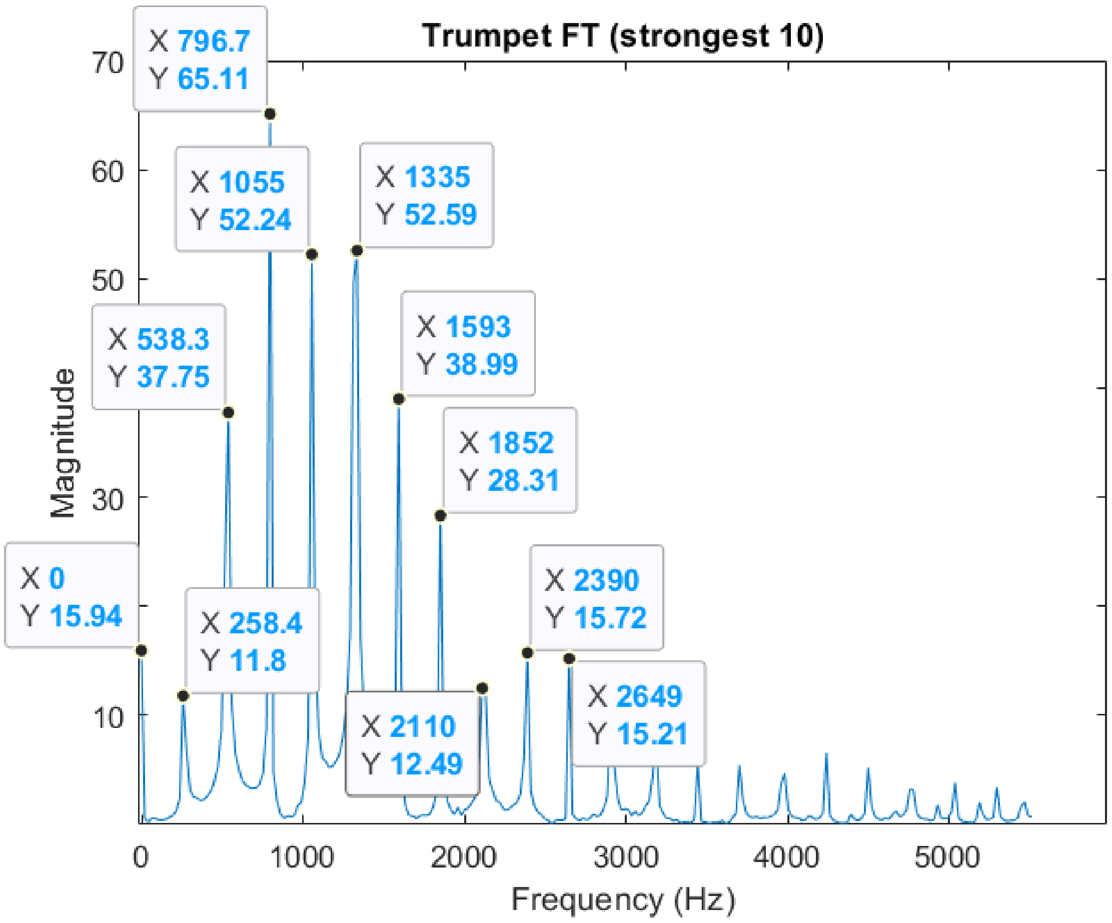
Here is the code used in analysis.

```
% Youssef Beltagy
% BEE235A, Aut 2021, Lab 3
% signal_synthesis_plot.m generates a plot of the trumpet sound and
    its
% fourier transform

load trumpet;
Fs = 11025;

% Play the trumpet -- part 2
sound(trumpet, Fs);
%pause(length(trumpet)/Fs);

% Plot the trumpet -- part 3
plot(trumpet(1:150));
title("Trumpet (Zoomed in)");
xlabel("Sample Number");
ylabel("Value");

% Plot the Frequency -- part 4 and 5
Y = fft(trumpet, 512);  % FFT (Fourier transform) of the trumpet
    sond
Ymag = abs(Y)           % take the mag of Y
f = Fs * (0:256)/512;   % get a meaningful frequency axis
plot(f, Ymag(1:257));   % plot Ymag (only half the points are needed
    )
title("Trumpet FT (first seven)");
xlabel("Frequency (Hz)");
ylabel("Magnitude");

% Plot the Frequency -- part 6
figure()
Y = fft(trumpet, 512);  % FFT (Fourier transform) of the trumpet
    sond
Ymag = abs(Y);          % take the mag of Y
```

```
f = Fs * (0:256)/512;    % get a meaningful frequency axis
plot(f, Ymag(1:257));    % plot Ymag (only half the points are needed
    )
title("Trumpet FT (strongest 10)");
xlabel("Frequency (Hz)");
ylabel("Magnitude");
```

## 2.2 Signal Synthesis

To synthesize a new trumpet signal, we recorded the 10 strongest (in magnitude) harmonic peaks. We ignored the 0th harmonic (when the frequency is 0 Hz) because it will just make the sound louder without making it more intelligible and matlab will attenuate the larger magnitude. We did try adding the 0th harmonic, but it didn't make the sound any better.



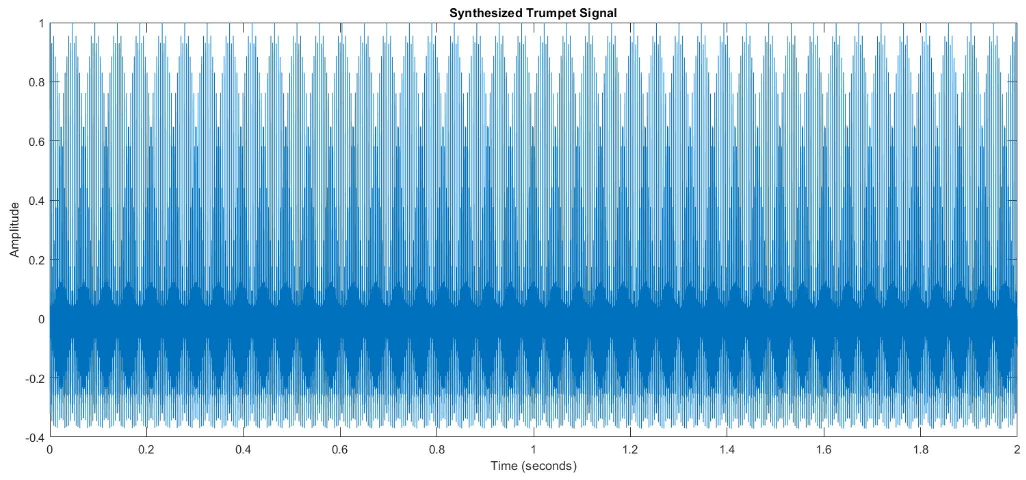
| Peak | Magnitude | Frequency (Hz) |
|---|---|---|
| 1 | 65.114 | 796.729 |
| 2 | 52.593 | 1335.06 |
| 3 | 52.244 | 1055.13 |
| 4 | 38.994 | 1593.46 |
| 5 | 37.749 | 538.33 |
| 6 | 28.305 | 1851.86 |
| 7 | 15.724 | 2390.19 |
| 8 | 15.208 | 2648.58 |
| 9 | 12.487 | 2110.25 |
| 10 | 11.795 | 258.398 |

Table 3: 10 strongest harmonic peaks

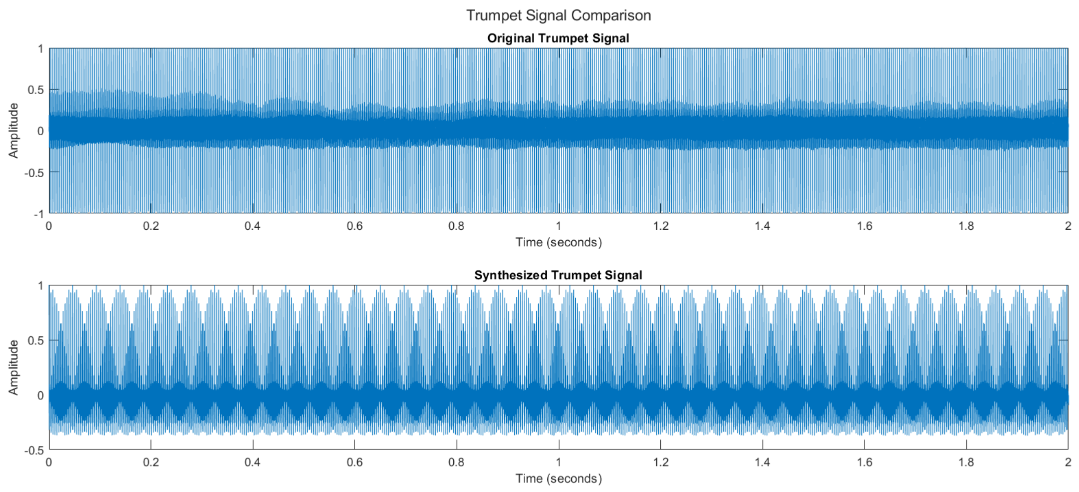
We then created a matlab script that summed up 10 sine waves into a $2 * F_s$-long vector to synthesize a 2-second long trumpet signal.

The synthesized signal sounded much like the pre-recorded trumpet signal but had zero variance in tone. The synthesized signal sounded monotonic and had a weird ringing sound. The synthesized signal may sound better if more harmonics are added.

Below is the synthesized signal. It is periodic and doesn't have any variation. It still has a sinusoidal form.

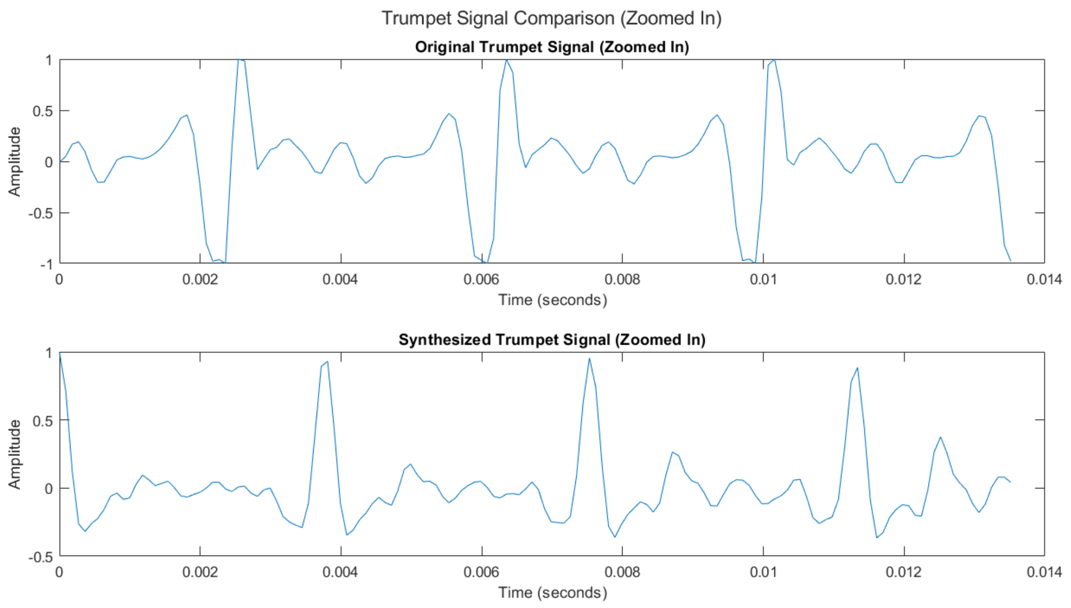


Compared to the original signal, the synthesized signal is missing components. The synthesized signal seems to mostly contain the dense central part of the original which you can see in the image below. The other frequencies are truncated.



When we zoomed into the signals, it is clear the two signals are not in phase. This is because we only used the magnitudes of the coefficients for the fourier series and didn't consider the phase.

The original signal appears smoother than the synthesized signal. The original signal has values in the range of [-1,1] but the synthesized signal only has values in the range of [-0.5,1]. The original signal has negative peaks of bigger magnitude than the synthesized signal. All of these points illustrate how the synthesized signal is missing most components of the original. Yet, the synthesized signal is adequately similar to the original for us to identify it.



```
% Samuel Hunter
% BEE235A, Au 2021, Lab 2
% signal_synthesis.m - Synthesize a trumpet sound from multiple sine
```

```
    waves.

Fs = 11025; % Sample rate of 11.025 kHz
t = 0:1/Fs:2; % Two-second time vector at Fs

% 10 most powerful sine waves from the sampled signal, sorted by
% magnitude.
freq = [
    796.729;
        1335.06;
        1055.13;
        1593.46;
        538.330;
        1851.86;
        2390.19;
        2648.58;
        2110.25;
        258.398];

mag = [
    65.114;
        52.593;
        52.244;
        38.994;
        37.749;
        28.305;
        15.724;
        15.208;
        12.487;
        11.795];

% Add up all sine waves
synthesized = zeros(size(t));
for i = 1:length(freq)
        synthesized = synthesized + mag(i)*cos( (2 * pi * freq(i))
           .* t);
end

% Normalize signal to -1:1.
synthesized = synthesized ./ max(abs(synthesized));


% Plotting the synthesized signal
plot(t, synthesized);
```

```
title('Synthesized Trumpet Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');


figure();

% Comparing the original and synthesized signal.
layout = tiledlayout(2,1);
title(layout, "Trumpet Signal Comparison");

nexttile;
plot(t, trumpet(1:length(t)));
title('Original Trumpet Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');

nexttile;
plot(t, synthesized);
title('Synthesized Trumpet Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');


figure();

% Comparing the original and synthesized signal (Zoomed In)
layout = tiledlayout(2,1);
title(layout, "Trumpet Signal Comparison (Zoomed In)");

nexttile;
plot(t(1:150), trumpet(1:150));
title('Original Trumpet Signal (Zoomed In)');
xlabel('Time (seconds)');
ylabel('Amplitude');

nexttile;
plot(t(1:150), synthesized(1:150));
title('Synthesized Trumpet Signal (Zoomed In)');
xlabel('Time (seconds)');
ylabel('Amplitude');

% Playing the sounds
sound(trumpet, Fs);
```

```
pause(length(trumpet)*2/Fs);
sound(synthesized, Fs);
```

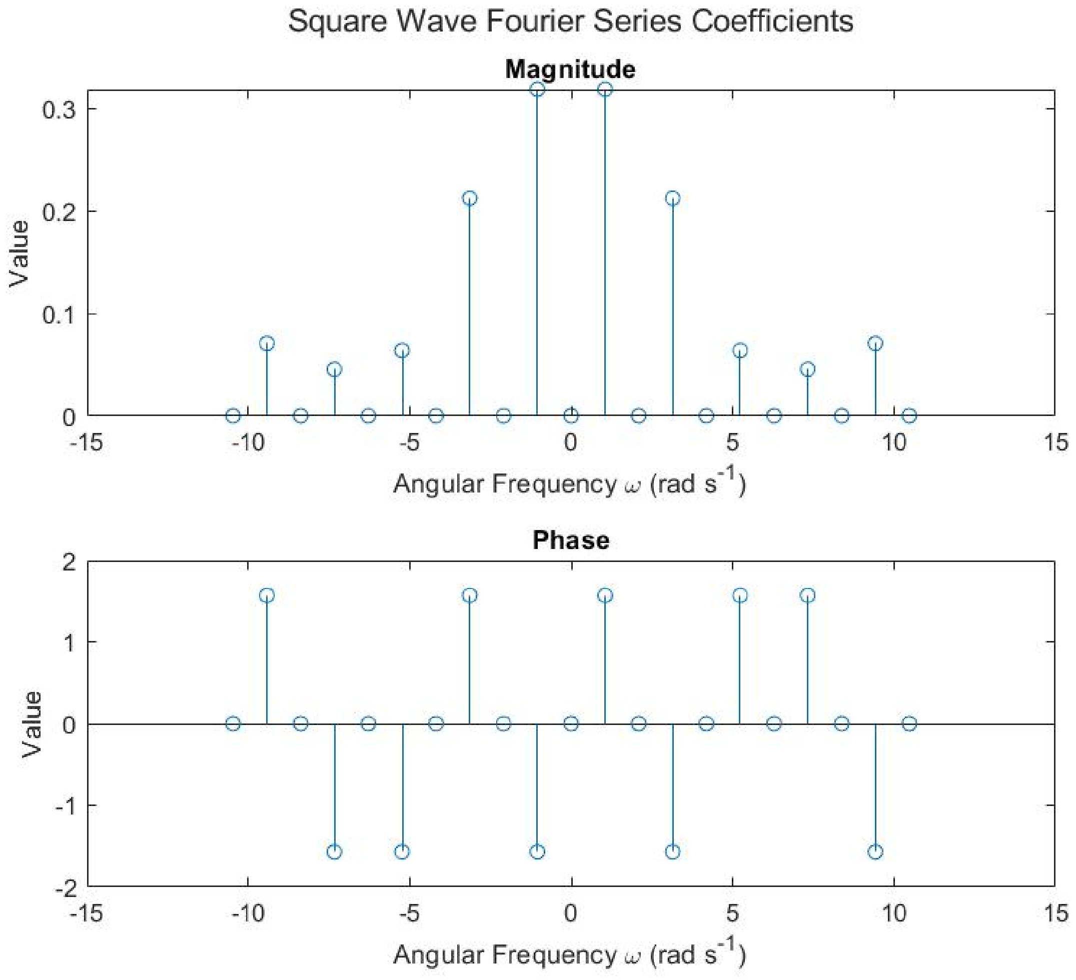
## 3 Part 2 — Fourier Series Approximation of a Square Wave

In this section, we synthesized a square wave signal using a fourier series. When the number of coefficients is low, the synthesized signal displays the Gibbs phenomenon where the signal overshoots and undershoots at sharp transitions.

### 3.1 $C_k$ Phase and Magnitude

We generated and plotted the $C_k$ coefficients of the fourier series we used to synthesize the square wave.

Below you can see the magnitudes and phases of the coefficients.



```
% Youssef Beltagy
% BEE235A, Aut 2021, Lab 3
% gibbs.m - plots coefficients of a fourier series representing a
   square wave


k = -10:1:10;
C_k = Ck(k);
w = (pi / 3) * k; % 2 pi * F == 2 pi * 1 / T == 2 pi / 6

layout = tiledlayout(2,1);
title(layout, "Square Wave Fourier Series Coefficients");

nexttile
stem(w, abs(C_k));
title("Magnitude");
ylabel("Value");
xlabel("Angular Frequency \omega (rad s^{-1})");

nexttile
stem(w, angle(C_k));
title("Phase");
ylabel("Value");
xlabel("Angular Frequency \omega (rad s^{-1})");


pause();
squarewave_plot();
```

## 3.2 Square Wave Synthesis

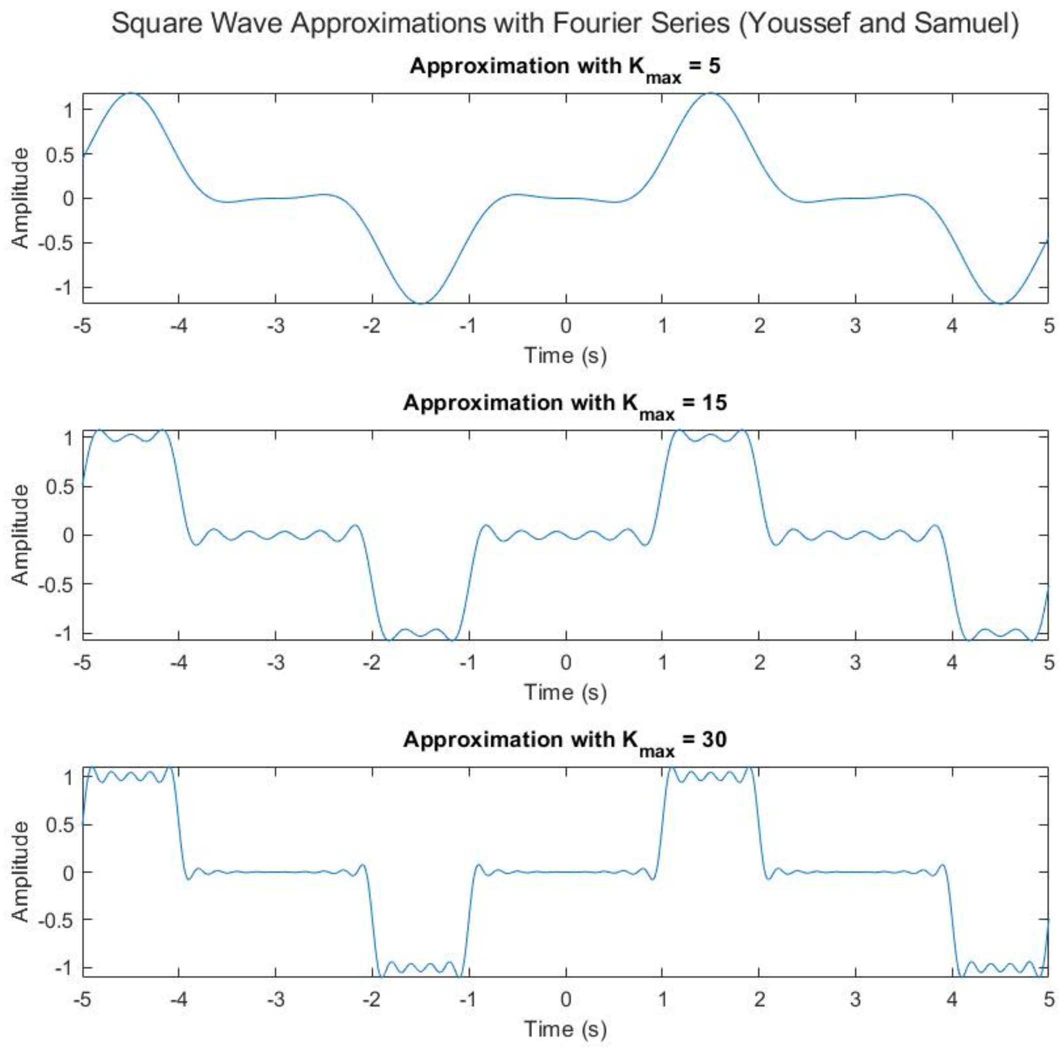
We developed a function that generates an approximation of a square wave given the time of the signal and the $K_{max}$ of the coefficients.

```
% Youssef Beltagy
% BEE235A, Aut 2021, Lab 3
% squarewave.m - Generates a Fourier Series
% approximation of a square Wave
% given the maximum number of coefficients.
% t is the input time (must be a row vector)
% max_k is the maximum number of coefficients
% This function was vectorized for efficiency.
```

```
function x=squarewave(t, max_k)

k = 0:1:max_k; % The coefficient indices
C_k = Ck(k); % The coefficient values
w_0 = pi / 3; % Angular Frequency

x = (k' * t) .* w_0; % Genrate a KxT matrix
x = x + repmat(angle(C_k'), 1, length(t)); % add angle of C_k to
    every column
x = cos(x); % Get the cosine of every element
x = x .* repmat(abs(C_k'), 1, length(t)); % bitwise multipilication
    of every column by C_k
x = sum(x, 1); % Squash the columns
x = 2 .* x; % multiply every value of x by 2

end
```

## 3.3 Square Wave Plots

We plotted the output from our synthesizing function. Clearly as $K_{max}$ increases, the synthesized signal becomes more similar to the original.



```
% Youssef Beltagy
% BEE235A, Aut 2021, Lab 3
% plots fourier series representations of a squarewave


t = -5:0.01:5; % the time
K_max = [5, 15, 30]; % all values of K_max to use
```

```
layout = tiledlayout(length(K_max),1);
title(layout, "Square Wave Approximations with Fourier Series (
    Youssef and Samuel)");

for i=1:length(K_max)


y = squarewave(t, K_max(i));

nexttile;
plot(t, y);
title(sprintf("Approximation with K_{max} = %d", K_max(i)));
ylabel("Amplitude");
xlabel("Time (s)");
end
```

### 3.4 Gibbs Phenomenon

The plots didn't demonstrate the phenomenon enough, so we wrote a script to generate a video of the synthesized signal as $k_{max}$ increases.

When $K_{max}$ is less than 500, the Gibbs phenomenon is clear. The signal rings (overshoots and undershoots) at sharp transitions.

Between $K_{max} = 400$ and $K_{max} = 500$, the ringing decreases. When $K_{max}$ rises above 500, the Gibbs phenomenon disappears. After that it appears again; then it disappears again; then it appears again, and the process repeats.

```
% Youssef Beltagy
% BEE235A, Aut 2021, Lab 3
% Generates a video of a fourier series approximation
% of a square wave as the number of coefficients increases


t = -5:0.01:5; % the time
K_max = 5000;

for k=0:1:K_max;


y = squarewave(t, k);
```

```
pause(1/100); % 60 frames per second --> will be limited by the
   squarewave generation time

plot(t, y);
title(sprintf("Approximation with K_{max} = %d", k));
ylabel("Amplitude");
xlabel("Time (s)");

end
```

## 4 Conclusion

We used the fourier transform to analyze a trumpet sound. We then used the strongest harmonics of the trumpet to synthesize a trumpet sound using the fourier series. By summing cosines mapping to the highest-magnitude harmonics, we can generate a signal that mimics the timbre of a sound piece.

We synthesized a square wave using its fourier series coefficients and compared it to the original signal. Fourier series can be used to generate signals with sharp transitions.

Because the synthesized signals are truncated fourier series, they have some issues like the Gibbs phenomenon or imperfect representations of the signal. The synthesized trumpet sound has zero variance in tone and is not able to replicate any "wibbly-wobbly" sound that appears because of smaller frequencies in the sample. The square wave displayed the Gibbs Phenomenon even when hundreds of frequencies were used.

Nonetheless, a Fourier series can generate sound tables that can be used by synthesizers and other digital audio tools as a base for instruments.