به نام خدا

گزارش پروژه

نام و نام‌خانوادگی و شماره‌دانشجویی اعضای گروه:

مهتا رنجبر دامغانی (۴۰۱۱۸۸۱۳) و پریا خان‌جان (۴۰۱۱۷۷۳۳)

پروژه درس: سیگنال‌ها و سیستم‌ها

نام استاد: دکتر فاطمه رضایی

گزارش Part1:

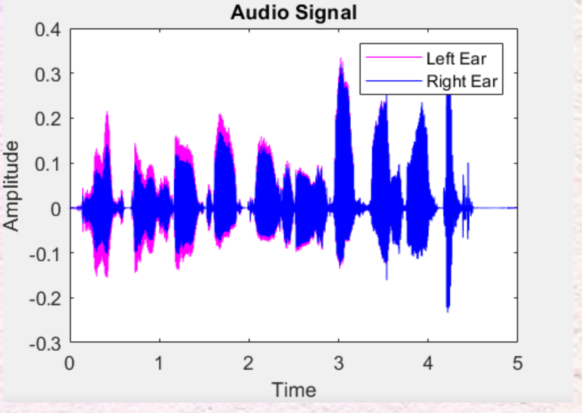
- در این قسمت ابتدا فایل صوتی voice.wav را لود کرده. می‌دانیم که صدا سیگنالی برحسب متغیر time می‌باشد، پس این متغیر را به صورت پیوسته در خط بعدی با تابع linspace تعریف می‌کنیم.

```
%%% PART 1 %%%

[audio, Fs] = audioread("voice.wav");
time = linspace(0, length(audio)/Fs, length(audio));
```

- صوت داخل فایل صوتی دارای ۲ کانال است که کانال اول، صدا در گوش سمت چپ است و کانال دوم، صدا در گوش سمت راست است. شکل کانال اول را با رنگ صورتی، و شکل کانال دوم را با رنگ آبی نشان می‌دهیم و با ترکیب این دو کانال، شکل سیگنال صوت را رسم می‌کنیم.

```
figure;
plot(time, audio(:,1), 'm');
hold on;
plot(time, audio(:,2), 'b');
xlabel('Time');
ylabel('Amplitude');
title('Audio Signal');
legend('Left Ear', 'Right Ear');
```



- سپس نرخ نمونه‌برداری صوت را دو برابر می‌کنیم و صوت جدید با این نرخ نمونه‌برداری را در فایل صوتی out_doubled.wav ذخیره می‌کنیم.

```
Fs_2 = 2 * Fs;
audiowrite("./out_doubled.wav", audio, Fs_2);
```

- با دو برابر کردن نرخ نمونه‌برداری، تعداد نمونه‌های گرفته‌شده در هر ثانیه را افزایش می‌دهیم. در پی این موضوع، دامنه سیگنال متراکم‌تر می‌شود و این موضوع باعث افزایش دامنه فرکانس می‌شود.

گزارش Part2:

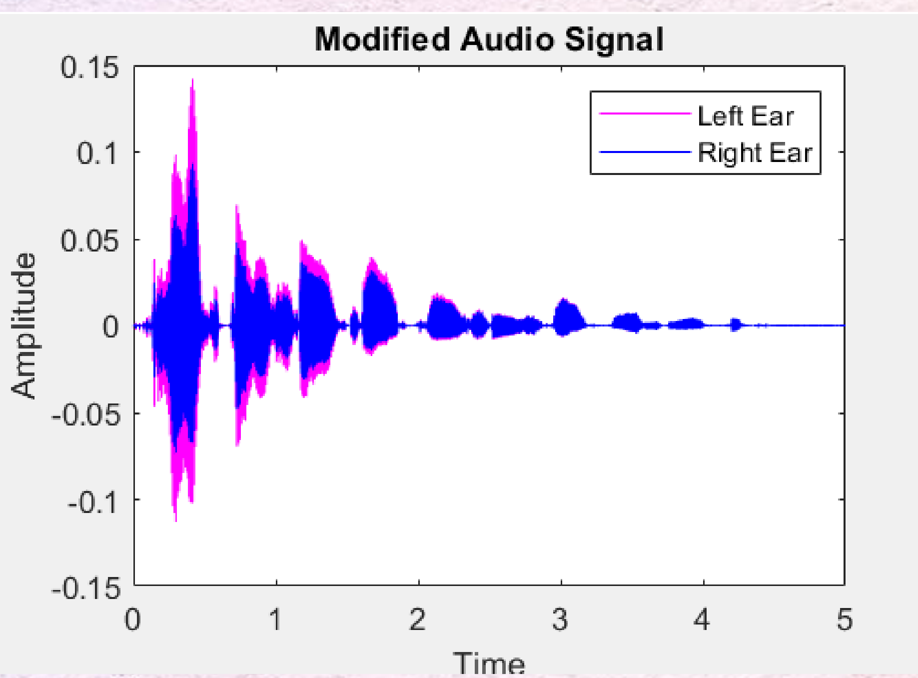
- در این قسمت، سیگنال نمایی منفی (e^{-t}) را درست کرده و با ضرب عنصری این سیگنال در سیگنال صوتی که در قسمت part1 گرفتیم، صوت جدیدی را می‌سازیم. (ضرب سیگنال نمایی منفی باعث روند کاهشی صوت می‌شود.)

```
%%% PART 2 %%%

exp_signal = exp(-1*time);
modified_audio = audio .* exp_signal';
```

- سپس شکل سیگنال صوت جدید مانند part1 رسم می‌کنیم. (یعنی با ترکیب کانال ۱ (صورتی) و کانال ۲ (آبی) شکل سیگنال صوت جدید به‌دست می‌آید) و صوت جدید را در فایل صوتی modified_audio_signal.wav ذخیره می‌کنیم.

```
figure;
plot(time, modified_audio(:,1), 'm');
hold on;
plot(time, modified_audio(:,2), 'b');
xlabel('Time');
ylabel('Amplitude');
title('Modified Audio Signal');
legend('Left Ear', 'Right Ear');
audiowrite("./modified_audio_signal.wav", modified_audio, Fs);
```



نمودار سیگنال صوت جدید (صوت voice و تابع نمایی منفی با هم ضرب شده‌اند)

گزارش Part3:

- در این قسمت، متغیرهایی برای اکو از جمله تاخیر و شدت آن را که به ترتیب ۱ ثانیه و ۸۰٪ است، را مقداردهی می‌کنیم. همچنین برای اکو دوم این متغیرها برای تاخیر با مقدار ۲ ثانیه و شدت ۵۰٪ نیز مقداردهی می‌کنیم.

```
%%% PART 3 %%%

first_echo_delay = 1.0 * Fs;
second_echo_delay = 2.0 * Fs;

first_echo_intensity = 0.8;
second_echo_intensity = 0.5;
```

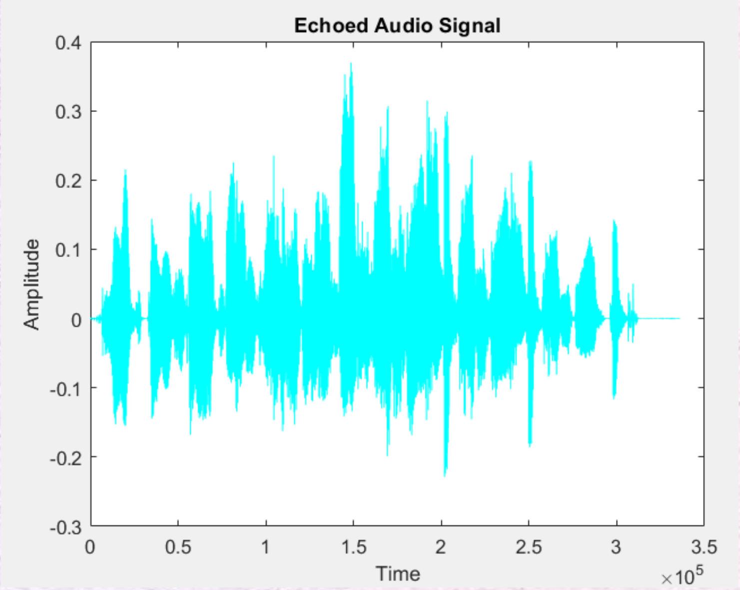
برای مقداردهی تاخیر این دو اکو، تاخیر را از ثانیه به نمونه تبدیل می‌کنیم.

- سپس طول پاسخ ضربه را براساس حداکثر تاخیر تعریف می‌کنیم. سپس پاسخ ضربه‌ای درست می‌کنیم که نشان‌دهنده اثر اکو با تنظیم مقادیر در موقعیت‌های تاخیر، با شدت‌های مربوطه است. (یعنی با استفاده از پاسخ ضربه فیلتر اکویی درست می‌کنیم که در موقعیت‌های تاخیر هر اکو، شدت آن اکو را دارد.)

```

max_delay = max( first_echo_delay, second_echo_delay);
impulse_response = zeros(max_delay + 1, 1);
impulse_response(1) = 1;
impulse_response(first_echo_delay + 1) = first_echo_intensity;
impulse_response(second_echo_delay + 1) = second_echo_intensity;
```

- در آخر آن پاسخ ضربه‌ای (فیلتر) که درست کردیم را با صوتی که در part1 گرفتیم، عمل کانولوشن را انجام می‌دهیم که باعث می‌شود صوت جدیدی دارای دو اکو با تاخیر و شدت‌های متفاوت همراه صوت اصلی، ایجاد شود. در آخر شکل صوت جدید را رسم می‌کنیم و آن را در فایل صوتی out_with_echo.wav ذخیره می‌کنیم.




```
output_audio = conv2(audio, impulse_response);

figure;
plot(output_audio, 'c');
title('Echoed Audio Signal');
xlabel('Time');
ylabel('Amplitude');

audiowrite("./out_with_echo.wav", output_audio, Fs);
```

گزارش Part4:

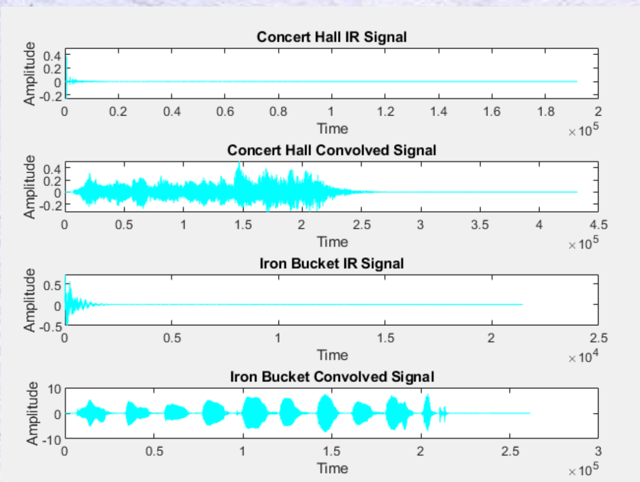
- در این قسمت، ابتدا فایل صوتی concert_hall_IR.wav را لود می‌کنیم و شکل سیگنال آن را رسم می‌کنیم. سپس این صوت را با صوتی که در قسمت part1 گرفتیم، عمل کانولوشن را برایشان انجام می‌دهیم و صوت جدیدی ایجاد می‌شود. این صوت ایجاد شده براثر عمل کانولوشن انگار فایل صوتی voice در محیط فایل صوتی concert_hall_IR پخش می‌شود. برای این صوت جدید، شکل سیگنالش را رسم می‌کنیم و این صوت جدید را در فایل صوتی out_concert_hall ذخیره می‌کنیم.

```
%%% PART 4 %%%

IR_concert_hall = audioread("concert_hall_IR.wav");
figure;
subplot(4,1,1);
plot(IR_concert_hall, 'c');
title('Concert Hall IR Signal');
xlabel('Time');
ylabel('Amplitude');

output_concert_hall = conv2(audio, IR_concert_hall);
subplot(4,1,2);
plot(output_concert_hall, 'c');
title('Concert Hall Convolved Signal');
xlabel('Time');
ylabel('Amplitude');

audiowrite("./out_concert_hall.wav", output_concert_hall, Fs);
```



- سپس، فایل صوتی iron_bucket_IR.wav را لود می‌کنیم و شکل سیگنال آن را رسم می‌کنیم. سپس این صوت را با صوتی که در قسمت part1 گرفتیم، عمل کانولوشن را برایشان انجام می‌دهیم و صوت جدیدی ایجاد می‌شود. این صوت ایجاد شده براثر عمل کانولوشن انگار فایل صوتی voice در محیط فایل صوتی iron_bucket_IR پخش می‌شود. برای این صوت جدید، شکل سیگنالش را رسم می‌کنیم و این صوت جدید را در فایل صوتی out_iron_bucket ذخیره می‌کنیم.

```
subplot(4,1,3);
IR_iron_bucket = audioread("iron_bucket_IR.wav");
plot(IR_iron_bucket, 'c');
title('Iron Bucket IR Signal');
xlabel('Time');
ylabel('Amplitude');

output_iron_bucket = conv2(IR_iron_bucket, audio);

subplot(4,1,4);
plot(output_iron_bucket, 'c');
title('Iron Bucket Convolved Signal');
xlabel('Time');
ylabel('Amplitude');

audiowrite("./out_iron_bucket.wav", output_iron_bucket/max(abs(output_iron_bucket)), Fs);
```

گزارش Part5:

```
%%% PART 5 %%%

figure;
subplot(5,1,1);
t = linspace(-4, 4, 1000);
y = square(2*pi*(t+1)/4);
plot(t, y, 'b');
ylim([-1.5 1.5]);
xlabel('Time');
ylabel('Amplitude');
title('X(t)');
grid on;
```

- در این قسمت، ابتدا سیگنال x(t) را محاسبه کریم و تابع آن را برحسب متغیری از t بدست می‌آورم. این سیگنال، سیگنال موج مربعی است که یک واحد به سمت چپ شیفت خورده است. سپس شکل این سیگنال را رسم می‌کنیم.

- سپس، ضرایب سری فوریه آن را محاسبه می‌کنیم. می‌دانیم که ضرایب سری فوریه از فرمول $\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)e^{-j\frac{2\pi}{T}kt}dt$ به دست می‌آید. براساس سیگنال x(t) مشاهده می‌کنیم که در بازه‌ی [-2,-1]، تابع مقدار ۱- را دارد و در بازه‌ی [-1,1] مقدار ۱ و در بازه‌ی [1, 2] مقدار منفی ۱- دارد. همچنین در این سیگنال مشاهده می‌شود که T (دوره تناوب) ۴ است. پس ما باید سه انتگرال را برای محاسبه ضرایب فوریه‌اش حساب کنیم. برای محاسبه انتگرال‌ها از تابع trapezoidal استفاده می‌کنیم. ورودی‌های این تابع، سیگنالی برحسب t است که براساس بازه هایی که مشخص کردیم یا $e^{-j\frac{2\pi}{4}kt}$ یا $-e^{-j\frac{2\pi}{4}kt}$ است. با محاسبه این انتگرال متوجه می‌شویم که a_k همان $sinc\left(\frac{k}{2}\right)$ است. و سپس این ضرایب فوریه را رسم می‌کنیم.

```
ak = @(k) 0.25 * (trapezoidal_rule(@(t) -exp(-0.5*1i*pi*k*t), -2, -1, 1000) ...
    +trapezoidal_rule(@(t) exp(-0.5*1i*pi*k*t), -1, 1, 1000) + ...
    trapezoidal_rule(@(t) -exp(-0.5*1i*pi*k*t), 1, 2, 1000));

subplot(5,1,2);
fplot(@(k)ak);
xlabel('k');
ylabel('ak');
title('Fourier Series Coefficents');
grid on;

```

- سپس، براساس فرمول $\sum_{-\infty}^{+\infty} a_k e^{j\frac{2\pi}{T}kt}$ سری فوریه این تابع را محاسبه می‌کنیم. ابتدا این مقدار را برای $k\epsilon[-20,20]$ محاسبه می‌کنیم. برای این کار، تابعی از t و k تعریف می‌کنیم و بازه‌ی ۲۰- تا ۲۰ را در یک متغیر قرار می‌دهیم که حالت صفرش را از آن جدا کرده و بصورت جداگانه محاسبه می‌کنیم. می‌دانیم جمله اول، برابر صفر است. سپس تابع تعریف شده را برابر با $a_k e^{j\frac{2\pi}{4}kt}$ قرار می‌دهیم. سپس با حلقه‌ای مقدار k را از ۲۰- تا ۲۰ تغییر می‌دهیم و در فرمول جایگذاری می‌کنیم و جواب به دست آمده با آن k را با کل تابع جمع می‌کنیم. با این کار سری فوریه سیگنال را به دست می‌آوریم و شکل آن را رسم می‌کنیم.
- همین روند را برای بازه‌های $k\epsilon[-100,100]$ و $k\epsilon[-500,500]$ تکرار می‌کنیم.

```
syms X(t,k);
size_of_k = 20;
K1 = -1*size_of_k:size_of_k;
K = nonzeros(K1);
K = K';
x = 0;
X(k) = ak.* exp(0.5*1i*pi*k*t);

for k = K
    x = x + X(k);
end

subplot(5,1,3);
fplot(x, [-4,4]);
xlabel('t');
ylabel('x');
title('Fourier Series When -20 < k < 20');
grid on;
```

کد $k\epsilon[-20,20]$

```
syms X1(t,k);
size_of_k_new = 100;
K2 = -1*size_of_k_new:size_of_k_new;
K = nonzeros(K2);
K = K';
x = 0;
X1(k) = ak.* exp(0.5*1i*pi*k*t);
for k = K
    x = x + X1(k);
end

subplot(5,1,4);
fplot(x, [-4,4]);
xlabel('t');
ylabel('x');
title('Fourier Series When -100 < k < 100');
grid on;
```

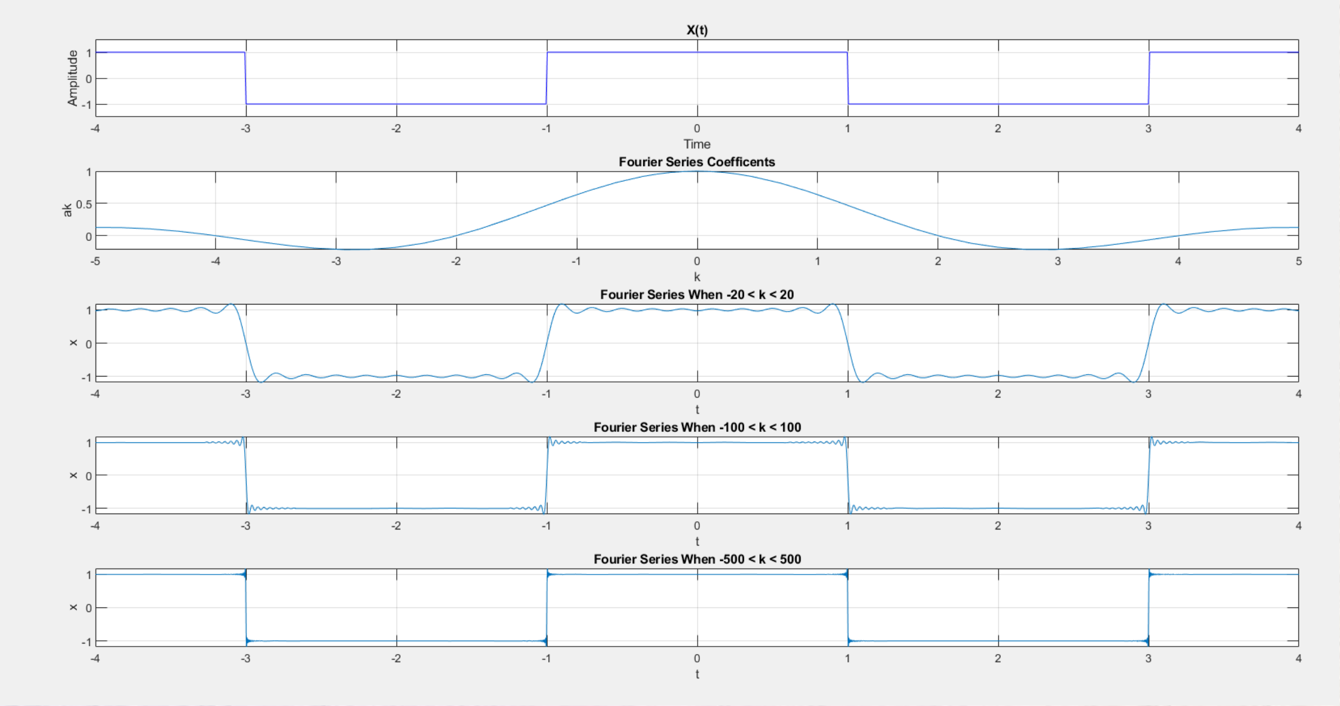
کد $k\epsilon[-100,100]$

```
syms X2(t,k);
size_of_k_newer = 500;
K3 = -1*size_of_k_newer:size_of_k_newer;
K = nonzeros(K3);
K = K';
x = 0;
X2(k) = ak.* exp(0.5*1i*pi*k*t);
for k = K
    x = x + X2(k);
end

subplot(5,1,5);
fplot(x, [-4,4]);
xlabel('t');
ylabel('x');
title('Fourier Series When -500 < k < 500');
grid on;
```

کد $k\epsilon[-500,500]$

- می‌دانیم در صورت برقراری شرایط کافی همگرایی دیریکله در نقاط پیوسته سری فوریه با سیگنال x(t) برابر است، ولی در نقاط ناپیوسته میل می‌کند به $\frac{1}{2}\left(x(t^+) + x(t^-)\right)$. (میانگین حد چپ و راست) هرچقدر مقدار k به بینهایت میل کند، خطای تقریب کمتر می‌شود و سری فوریه در نقاط ناپیوسته به سیگنال اصلی نزدیکتر می‌شود. این موضوع در شکل هم نمایان است.




```
figure;
subplot(3,1,1);
t = linspace(-4, 4, 1000);
y_new = square(2*pi*t/4);
plot(t, y_new, 'b');
ylim([-1.5 1.5]);
xlabel('Time');
ylabel('Amplitude');
title('NEW X(t)');
grid on;
```

- سپس سراغ سیگنال دیگر می‌رویم. ابتدا سیگنال x(t) را محاسبه کریم و تابع آن را برحسب متغیری از t بدست می‌آورم. این سیگنال، سیگنال موج مربعی است که یک واحد به سمت راست نسبت به سیگنال قبلی شیفت خورده است. سپس شکل این سیگنال را رسم می‌کنیم.

• سپس، ضرایب سری فوریه آن را محاسبه می‌کنیم. می‌دانیم که ضرایب سری فوریه از فرمول $\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)e^{-j\frac{2\pi}{T}kt}dt$ به دست می‌آید. براساس سیگنال x(t) مشاهده می‌کنیم که در بازه‌ی [-2,0]، تابع مقدار -۱ را دارد و در بازه‌ی [0,2] مقدار ۱ دارد. همچنین در این سیگنال مشاهده می‌شود که T (دوره تناوب) ۴ است. پس ما باید دو انتگرال را برای محاسبه ضرایب فوریه‌اش حساب کنیم. برای محاسبه انتگرال‌ها از تابع trapezoidal استفاده می‌کنیم. ورودی‌های این تابع، سیگنالی برحسب t است که براساس بازه هایی که مشخص کردیم یا $e^{-j\frac{2\pi}{4}kt}$ یا $-e^{-j\frac{2\pi}{4}kt}$ است. با محاسبه این انتگرال متوجه می‌شویم که همان c_k $sinc\left(\frac{k}{2}\right)$ $-\mathrm{jsin}\left(\frac{k\pi}{2}\right)$ است. و سپس این ضرایب فوریه را رسم می‌کنیم.

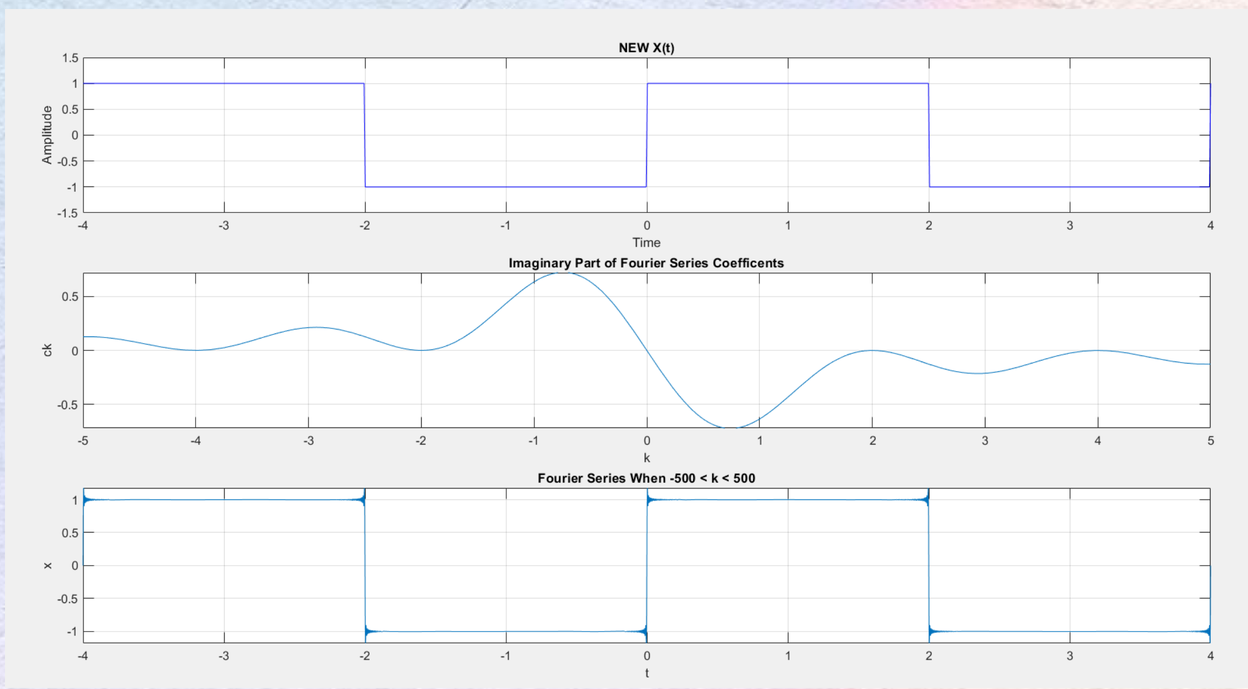
```
ck = @(k) 0.25 * (trapezoidal_rule(@(t) -exp(-0.5*1i*pi*k*t), -2, 0, 1000) ...
    +trapezoidal_rule(@(t) exp(-0.5*1i*pi*k*t), 0, 2, 1000));

subplot(3,1,2);
fplot(@(k)ck);
xlabel('k');
ylabel('ck');
title('Imaginary Part of Fourier Series Coefficents');
grid on;
```

• سپس، براساس فرمول $\sum_{-\infty}^{+\infty} c_k e^{j\frac{2\pi}{T}kt}$ سری فوریه این تابع را محاسبه می‌کنیم. این مقدار را برای $k\epsilon[-500,500]$ محاسبه می‌کنیم. برای این کار، تابعی از t و k تعریف می‌کنیم و بازه‌ی -۵۰۰ تا ۵۰۰ را در یک متغیر قرار می‌دهیم که حالت صفرش را از آن جدا کرده و بصورت جداگانه محاسبه می‌کنیم. می‌دانیم جمله اول، برابر صفر است. سپس تابع تعریف شده را برابر با $c_k e^{j\frac{2\pi}{4}kt}$ قرار می‌دهیم. سپس با حلقه‌ای مقدار k را از -۵۰۰ تا ۵۰۰ تغییر می‌دهیم و در فرمول جایگذاری می‌کنیم و جواب به دست آمده با آن k را با کل تابع جمع می‌کنیم. با این کار سری فوریه سیگنال را به دست می‌آوریم و شکل آن را رسم می‌کنیم.

```
syms X_NEW(t,k);
size_of_k_newer = 500;
K_NEW = -1*size_of_k_newer:size_of_k_newer;
K = nonzeros(K_NEW);
K = K';
x = 0;
X_NEW(k) = ck.*exp(0.5*1i*pi*k*t);
for k = K
    x = x + X_NEW(k);
end

subplot(3,1,3);
fplot(x, [-4,4]);
xlabel('t');
ylabel('x');
title('Fourier Series When -500 < k < 500');
grid on;
```



- سپس، رابطه بین a_k و c_k را با گرفتن نسبت c_k بر a_k برای k=3 بدست می‌آوریم. در آخر آن را نشان می‌دهیم.

```
a_3 = ak(3);
c_3 = ck(3);
coefficients_relation = c_3 / a_3;
disp('ck / ak for k = 3: ');
disp(coefficients_relation);
```

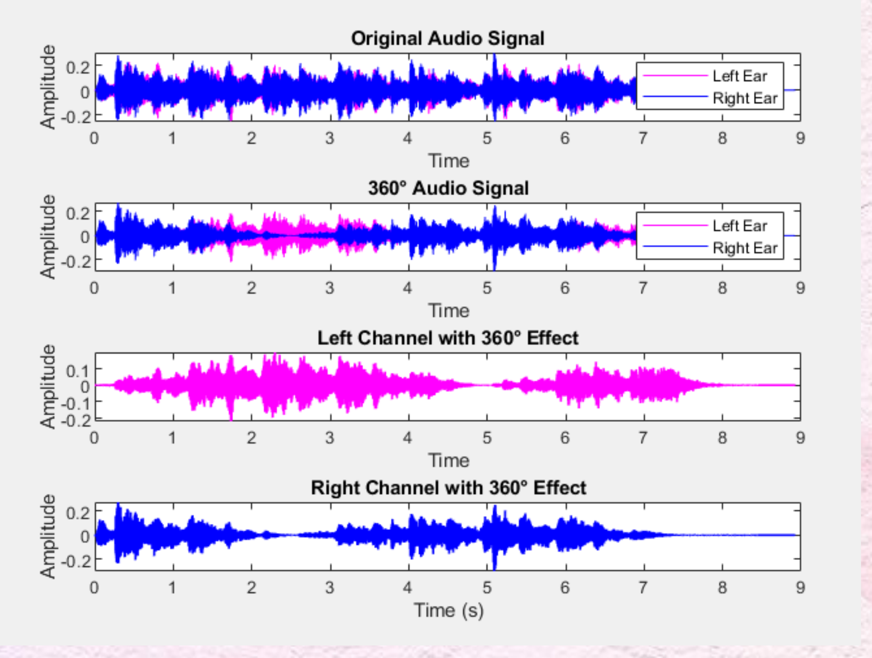
```
>> part_five
ck / ak for k = 3:
   0.0000 + 1.0000i

>>
```

- سیگنال دوم همانطور که گفتیم یک واحد نسبت به سیگنال اولی شیفت خورده بود. که طبق خواص سری فوریه این موضوع موجب ضرب $e^{-j\frac{2\pi}{T}kt_0}$ در ضرایب سری فوریه می‌شود. که در اینجا t_0 برابر با ۱ می‌باشد. (T=4) در نتیجه توقع داریم که: $c_k = a_k * e^{-j\frac{2\pi}{4}k}$ باشد. که با قرار دادن k=3 به $c_k = a_k * e^{-j\frac{3\pi}{2}}$ که با روابط اویلر ($e^{-j\theta} = \cos(\theta) - j\sin(\theta)$) و با قرار دادن $\theta=\frac{3\pi}{2}$ به $c_k = a_k * (0 + 1j)$ می‌رسیم که در خروجی متلب به آن رسیده بودیم.

گزارش Part6:

- در این قسمت ابتدا فایل صوتی original_audio.wav را لود می‌کنیم و داده‌های صوتی و فرکانس نمونه‌برداری را استخراج می‌کنیم. سیگنال‌های تغییر فاز را برای کانال‌های چپ و راست با استفاده از توابع سینوسی و کسینوسی براساس فرکانس زاویه‌ای خاص (omega)، محاسبه می‌کنیم. سپس برای به وجود آوردن صدای ۳۶۰ درجه، سیگنال‌های تغییر فاز را با صدای اصلی ضرب می‌کنیم تا جلوه صوتی ۳۶۰ درجه ایجاد شود. سپس شکل سیگنال صوتی اصلی، سیگنال صوتی ۳۶۰ درجه و کانال‌های چپ و راست تغییر فاز را رسم می‌کنیم. در آخر، سیگنال صوتی ۳۶۰ درجه را در یک فایل صوتی جدید 360_audio.wav ذخیره می‌کنیم.



```
%%% PART 6 %%%

[original, Fs] = audioread('original_audio.wav');

duration = size(original, 1) / Fs;
time = (0:size(original, 1)-1) / Fs;
omega = 2 * pi * 0.1;
left_phase_shift = sin(omega * time)';
right_phase_shift = cos(omega * time)';
left_channel = original(:, 1) .* left_phase_shift;
right_channel = original(:, 2) .* right_phase_shift;
audio_360 = [left_channel, right_channel];

figure;
subplot(4,1,1);
plot(time, original(:,1), 'm');
hold on;
plot(time, original(:,2), 'b');
xlabel('Time');
ylabel('Amplitude');
title('Original Audio Signal');
legend('Left Ear', 'Right Ear');

subplot(4,1,2);
plot(time, audio_360(:,1), 'm');
hold on;
plot(time, audio_360(:,2), 'b');
xlabel('Time');
ylabel('Amplitude');
title('360° Audio Signal');
legend('Left Ear', 'Right Ear');

subplot(4, 1, 3);
plot(time, left_channel, 'm', 'LineWidth', 1);
title('Left Channel with 360° Effect');
xlabel('Time');
ylabel('Amplitude');

subplot(4, 1, 4);
plot(time, right_channel, 'b', 'LineWidth', 1);
title('Right Channel with 360° Effect');
xlabel('Time (s)');
ylabel('Amplitude');

audiowrite('360_audio.wav', audio_360, Fs);
```