امیرعلی دهقانی | ریاضی مهندسی | دکتر طالع ماسوله
۸۱۰۱۰۲۴۴۳ | تمرین کامپیوتری اول | تاریخ تحویل : ۱۴۰۳/۷/۱۳

۱ سری فوریه

۱.۱ زلزله‌نگار

در این پروژه، داده‌های ثبت‌شده توسط یک زلزله‌نگار که به‌صورت سری زمانی ذخیره شده است، مورد بررسی قرار می‌گیرد. فایل داده (data.mat) شامل اطلاعاتی از سیگنال زلزله با نرخ نمونه‌برداری ۱۰۰۰ هرتز می‌باشد. داده‌های شبیه‌سازی‌شده بیانگر وقوع یک زلزله در یک شهر زلزله‌خیز هستند. هدف اصلی این پروژه محاسبه ضرایب سری فوریه داده‌های ثبت‌شده و تحلیل تأثیر مؤلفه‌های فرکانسی مختلف بر ساختمان‌ها با ارتفاع‌های متفاوت است.

یکی از نکات مهم در این بررسی، در نظر گرفتن فرکانس طبیعی ساختمان‌ها و مقایسه آن با فرکانس مؤلفه‌های زلزله است. اگر فرکانس طبیعی یک ساختمان با فرکانس زلزله در یک محدوده قرار گیرد، پدیده تشدید رخ می‌دهد که می‌تواند آسیب‌های شدیدی به ساختمان وارد کند. بنابراین، تحلیل دقیق این داده‌ها می‌تواند به ارائه توصیه‌هایی برای طراحی ساختمان‌هایی با ارتفاع مناسب کمک کند تا در صورت وقوع زلزله، کمترین آسیب ممکن به ساختمان‌ها وارد شود.

برای تحلیل این موضوع، از اطلاعات موجود در جدول زیر استفاده می‌شود:

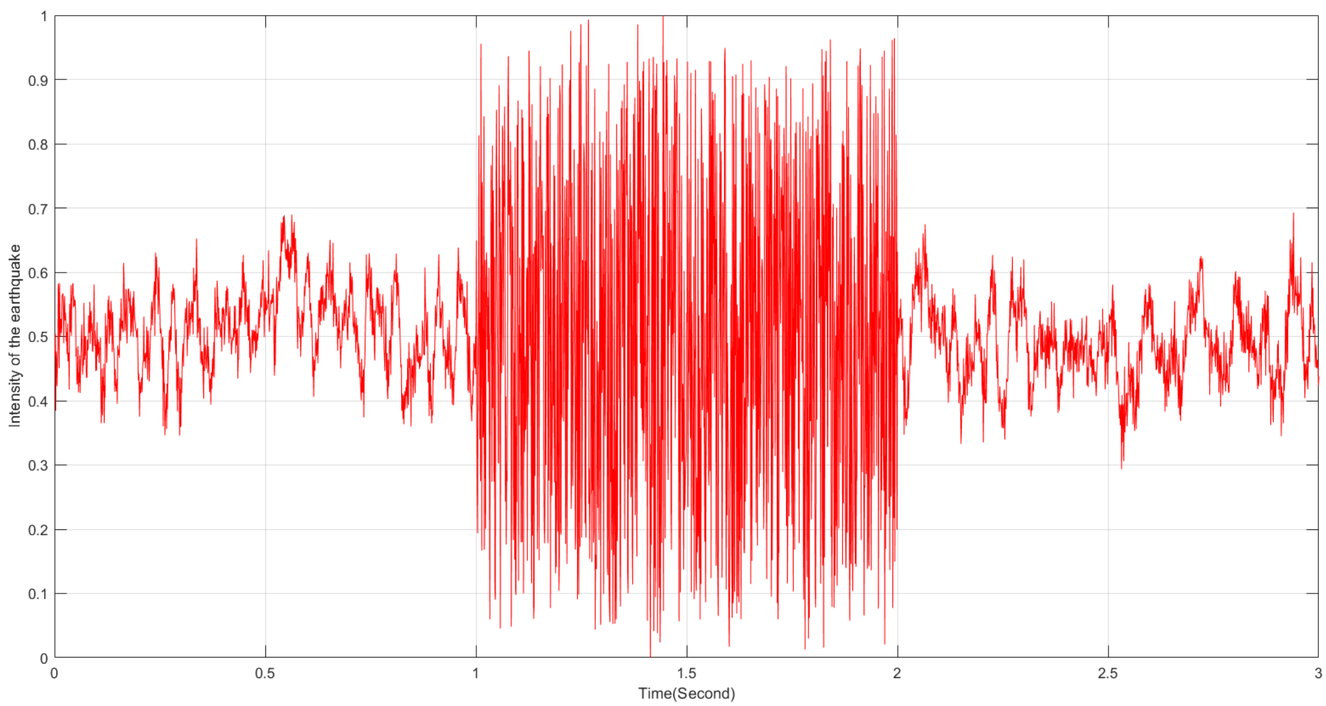
جدول ۱: ارتباط بین ارتفاع ساختمان و فرکانس طبیعی

ارتفاع ساختمان	فرکانس طبیعی (Hz)
۱-۴ طبقه (کوتاه)	۱۰-۳
۵-۱۰ طبقه (متوسط)	۲-۱
بیشتر از ۱۰ طبقه (بلند)	۱-۱.۰

در ادامه، بخش‌های مختلف پروژه شرح داده شده‌اند:

۱. بارگذاری و نرمال‌سازی داده‌ها: داده‌های ثبت‌شده را بارگذاری کرده و به مقادیر نرمال (بین ۰ و ۱) تبدیل کنید. برای این کار، داده‌ها را بر بزرگ‌ترین مقدار آن تقسیم نمایید. سپس سیگنال نرمال‌شده را با نرخ نمونه‌برداری ۱۰۰۰ هرتز رسم کنید.

```
dataFile = load('data.mat');
data = dataFile.time_series;
frequency = 1000;
lenData = length(data);
normalizedData = (data - min(data)) / (max(data) - min(data));
time = (1:lenData) / frequency;

figure;
plot(time, normalizedData, Color = 'Red');
xlabel("Time(Second)")
ylabel("Intensity of the earthquake")
grid on;
```



طبق چیزی که گفته شد دیتاها نرمال شده و با نرخ نمونه برداری ۱۰۰۰ هرتز، به ۳ ثانیه می‌رسیم که دیتاها را روی نمودار نمایش می‌دهیم.

۲. محاسبه ضرایب سری فوریه: تابعی بنویسید که سیگنال و مقدار n را به‌عنوان ورودی دریافت کرده و ضرایب سری فوریه (به فرم نمایی) را محاسبه کند. با توجه به اینکه داده‌ها گسسته هستند، به جای استفاده از انتگرال، از جمع مقادیر استفاده کنید.

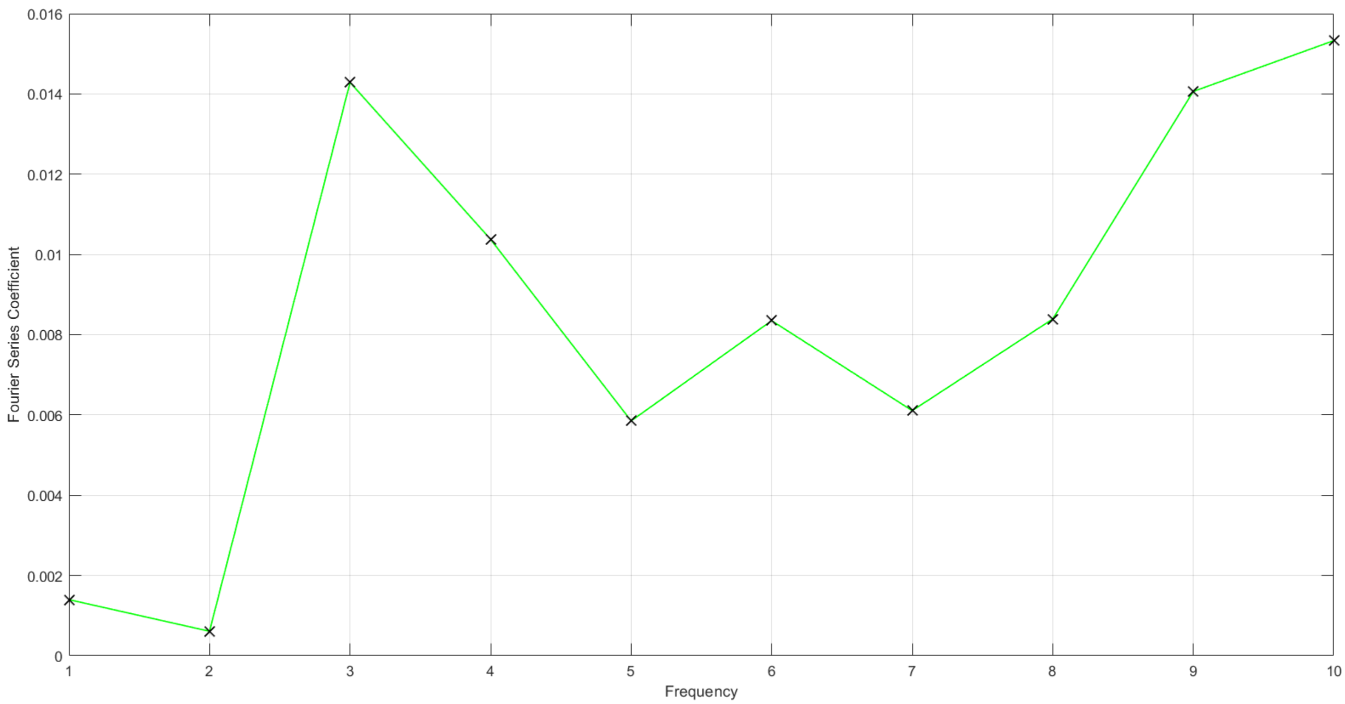
```
function fourierCoeff = calculateFourierCoeff(data, k)
    N = length(data);
    n = (1:N)';
    sigma = sum((exp(-1 * 1i * 2 * pi * n * k / N) .* data));
    fourierCoeff = sigma / N;
end
```

این تابع طبق فرمول $C_k = \frac{1}{N}\sum_{n=1}^{N} e^{-\frac{2k\pi n}{N}j} . x_n$ ضریب سری فوریه برای K مورد نظر را روی داده ها محاسبه می‌کند و خروجی می‌دهد.

۳. تحلیل بخش زلزله: بخش مرکزی داده‌ها که شامل ۱۰۰۰ نمونه می‌شود (داده های زمان وقوع زلزله) را جدا کرده و ضرایب سری فوریه برای فرکانس‌های کمتر از ۱۰ هرتز محاسبه کنید. این ضرایب را رسم کنید.

```
coeffs = [];
centerData = normalizedData(lenData / 3 + 1 : lenData * 2 / 3);
N = length(centerData);
coeffs = [coeffs, abs(calculateFourierCoeff(centerData, 1:10))];

figure;
plot(coeffs, Color = 'Green', LineWidth = 1, Marker = 'X',
MarkerEdgeColor='Black', MarkerSize=10);
xlabel("Frequency")
ylabel("Fourier Series Coefficient")
grid on;
```

برای به دست آوردن بخش مرکزی که همان زمان وقوع زلزله است، در واقع تعداد داده‌ها را به ۳ تقسیم می‌کنیم و بخش دوم ان را برمیداریم که همان بخش مرکزی آن است. سپس بر اساس فرمول $f_i = i\left(\frac{fs}{n}\right)$ برای محاسبه‌ی داده‌هایی که فرکانس زیر ۱۰ دارند در بخش مرکزی با توجه به اینکه با ۱۰۰۰ داده مواجه هستیم و نرخ نمونه برداری ۱۰۰۰ هرتز هست، در اینجا فقط گفته شده که برای $k = 1{:}10$ ضرایب سری فوریه را محاسبه کند.



۴. تحلیل آسیب‌پذیری ساختمان‌ها: با استفاده از نتایج بخش قبلی، توضیح دهید که کدام ساختمان‌ها در معرض خطر بیشتری قرار دارند و برای کاهش آسیب، بهتر است ساختمان‌ها چه ارتفاعی داشته باشند.

با توجه به ضرایب به دست آمده و طبق جدولی که داده شده، ساختمان‌هایی که ارتفاع کمتر دارند در معرض خطر بیشتری قرار دارند و با توجه به ضریب سری فوریه برای فرکانس ۲ هرتز، ساختمان‌های متوسط (۵-۱۰ طبقه) بهترند.

۲.۱ تشخیص نوت موسیقی

در این پروژه، یک سیگنال صوتی با نام song.wav بررسی می‌شود که مربوط به قطعه‌ای موسیقی شامل چندین نوت است. این نوت‌ها به‌صورت متوالی و با مدت‌زمان برابر 0.4 ثانیه در سیگنال ظاهر می‌شوند. نرخ نمونه‌برداری این سیگنال 44100 هرتز است. هدف پروژه، تحلیل این سیگنال برای استخراج نوت‌های موسیقی و شناسایی نام و فرکانس هر نوت است. هر نوت موسیقی به‌صورت یک موج سینوسی تولید شده که فرکانس آن متناظر با فرکانس مشخص نوت است. فرکانس هر نوت نشان‌دهنده زیر و بمی صدای آن بوده و ویژگی اصلی تمایز نوت‌ها محسوب می‌شود. در این پروژه، با استفاده از تحلیل سری فوریه، ساختار فرکانسی سیگنال بررسی شده و بر اساس ضرایب فوریه، نوت‌های موجود در آهنگ شناسایی خواهند شد. این روش به ما امکان می‌دهد تا از یک سیگنال صوتی پیچیده، اطلاعات مربوط به اجزای فرکانسی آن را استخراج کرده و به تفکیک نوت‌ها بپردازیم.

جدول ۲: فرکانس‌ها و نام نوت‌های موسیقی موجود در فایل صوتی

نام نوت	فرکانس (Hz)
C (Do)	523.25
D (Re)	587.33
E (Mi)	659.26
F (Fa)	698.46
G (Sol)	784.00

بخش‌های مختلف این پروژه به شرح زیر است:

۱. بارگذاری داده‌ها: سیگنال song.wav را بارگذاری کنید .

```
song = audioread('song.wav');
```

۲. محاسبه ضرایب سری فوریه: تابعی طراحی کنید که سیگنال و مقدار n را به‌عنوان ورودی دریافت کرده و ضرایب سری فوریه (به فرم نمایی) را محاسبه کند. برای این کار، با توجه به گسسته بودن داده‌ها، به‌جای انتگرال، از جمع مقادیر استفاده نمایید.

```
function fourierCoeff = calculateFourierCoeff(data, k)
    N = length(data);
    n = (1:N)';
    sigma = sum((exp(-1 * 1i * 2 * pi * (n-1) * k / N) .* data));
    fourierCoeff = sigma / N;

end
```

همان تابع در سوال قبلی است.

۳. جداسازی نوت‌ها: با توجه به مدت‌زمان هر نوت (0.4 ثانیه)، سیگنال را به نوت های جداگانه تقسیم کنید و ضرایب سری فوریه هر بخش را محاسبه کنید. فقط ضرایب مربوط به فرکانس‌های در ناحیه نوت های عنوان شده را استخراج کنید.

۴. رسم ضرایب سری فوریه: ضرایب سری فوریه محاسبه‌شده برای هر نوت را رسم کنید و تمام نمودارها را با استفاده از subplot در یک figure نمایش دهید. هر نمودار ضرایب سری فوریه یک نوت را نشان می‌دهد و محورهای آن فرکانس و مقدار ضریب را مشخص می‌کنند.

۵. تشخیص نوت‌ها: فرکانس نوت‌های داده‌شده (C، D، E، F، G) و اندیس ضرایب متناظر آن‌ها را محاسبه کنید. سپس برای هر نوت، ضریب سری فوریه با بیشترین مقدار (ضریب غالب) و اندیس آن را پیدا کنید. اگر اختلاف بین اندیس ضرایب و اندیس‌های تئوری نوت‌ها برابر با 0 یا 1 باشد، نام نوت مربوطه را به‌درستی تشخیص دهید. برای تمام نوت‌های سیگنال، نام نوت‌های تشخیص‌داده‌شده را به‌ترتیب پرینت کنید و نتیجه نهایی را گزارش دهید

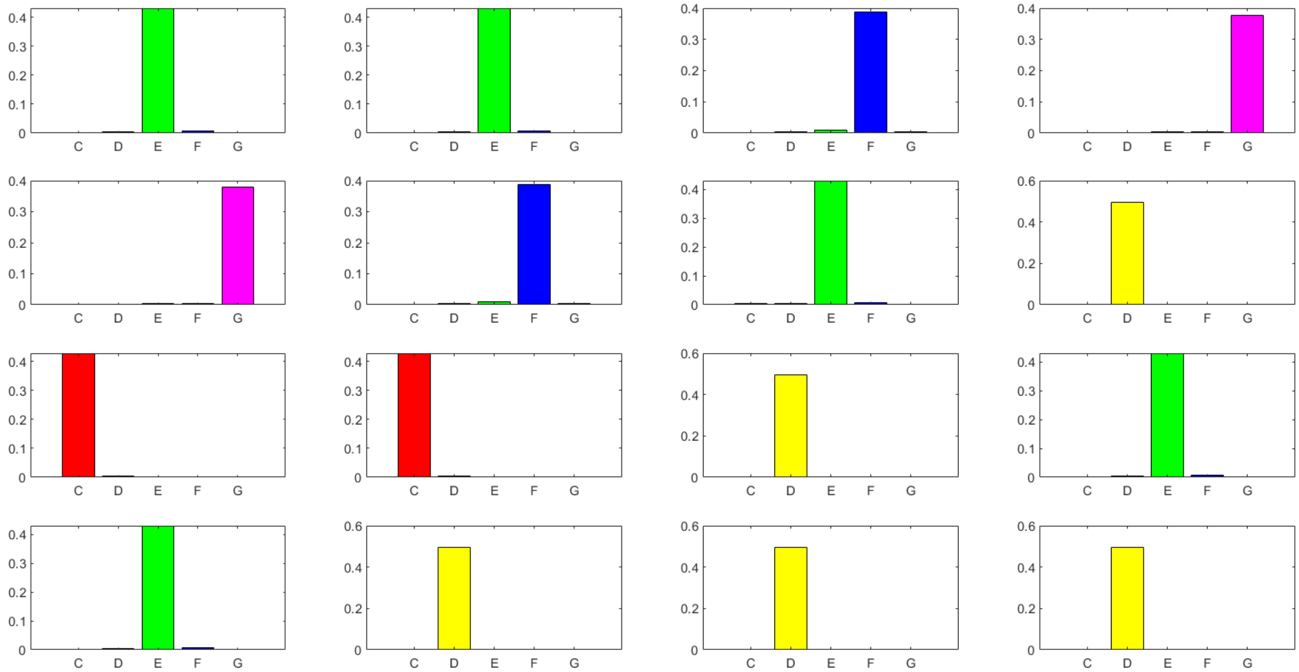
```
%% Process
fs = 44100;
segmentSize = 0.4 * fs;
numOfSegments = floor(length(song) / segmentSize);
segments = reshape(song(1:numOfSegments * segmentSize), segmentSize,
numOfSegments);
noteFrequencies = [523.25, 587.33, 659.26, 698.46, 784.00];
notes = {'C', 'D', 'E', 'F', 'G'};
recognizedNotes = cell(numOfSegments, 1);
coeffs = zeros(length(noteFrequencies), numOfSegments);

for i = 1:numOfSegments
    segment = segments(:,i);
    meanAmplitude = mean(abs(segment));
    for j = 1:length(noteFrequencies)
        k = round(noteFrequencies(j) * segmentSize / fs);
        coeff = calculateFourierCoeff(segment, k);
        coeffs(j, i) = abs(coeff);
        if abs(coeff) > 0.5 * meanAmplitude
            recognizedNotes{i} = notes{j};
        end
    end
end
```

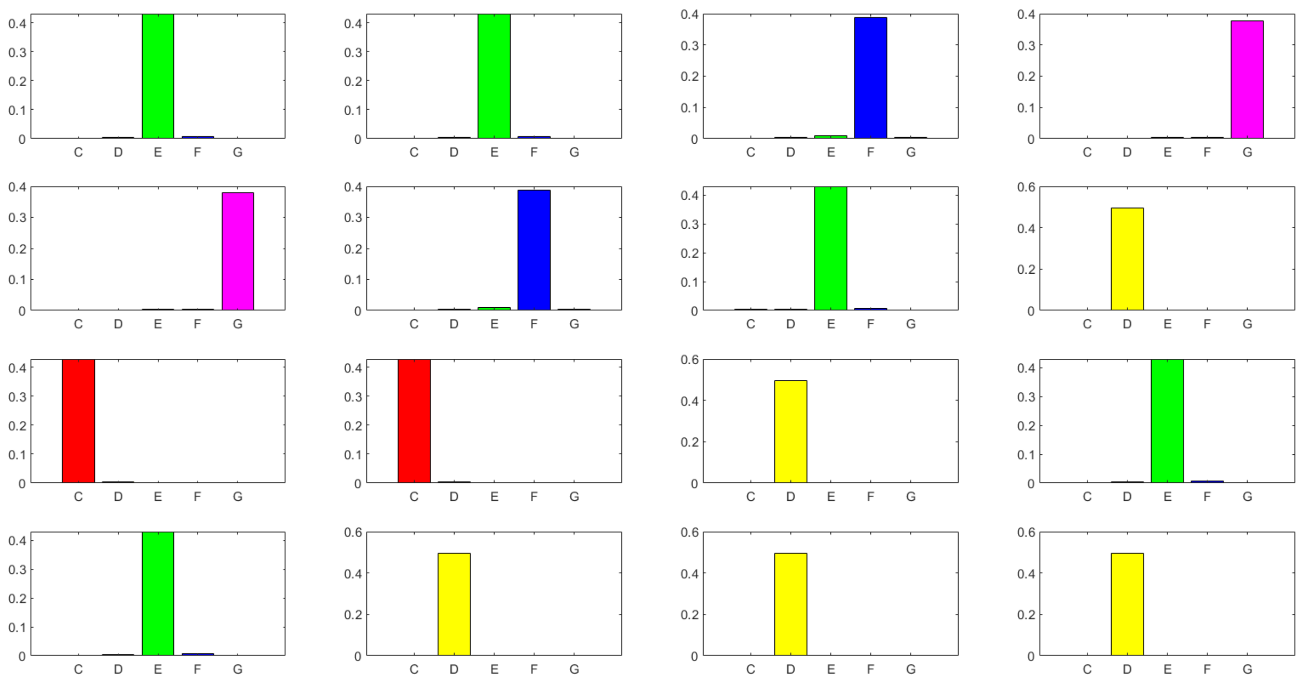
نرخ نمونه‌برداری موسیقی ما ۴۴۱۰۰ هرتز است و مدت زمان هر نوت ۰.۴ ثانیه است، اندازه‌ی هر پارت را جدا می‌کنیم و در نهایت با تقسیم کردن کل داده‌ها به این، تعداد نت‌ها را به دست می‌آوریم که در `numOfSegments` قرار می‌دهیم. علت استفاده از `floor` این است که موسیقی کاملا دقیق فقط شامل نوت نیست و بعضی جاها (بین دو نوت) صدایی دریافت نمی‌کنیم و در نهایت تقسیم کردن آن به ما یک عدد اعشاری می‌دهد که باید از `floor` استفاده کنیم.

حال برای هر بخش ضرایب سری فوریه ۵ نوت داده شده را محاسبه می‌کنیم. مشابه چیزی که در سوال زلزله گفته شد، برای `k` از فرمول $f_i = f\left(\frac{f_s}{n}\right)$ استفاده می‌کنیم تا ببینیم برای هر نوت باید ضریب فوریه به ازای چه `k` ای را باید محاسبه کنیم.

حال برای تشخیص دادن هر نوت، بعد از اینکه ضرایب فوریه ۵ نوت را برای هر بخش به دست آوردیم، بررسی می‌کنیم که اگر ضریب فوریه یکی از نت‌ها از نصف میانگین داده‌ها در آن بازه بیشتر بود، آن نوت نوت مد نظر ماست. مقدار نصف به صورت آزمون و خطا به دست آمد به صورتی که از ۱ شروع به کم شدن شد و در نهایت با ۰.۵، نت های درست و صحیح به دست آمد.

```
%% Plot
colors = [1 0 0;
          1 1 0;
          0 1 0;
          0 0 1;
          1 0 1];
for page = 1:2
    figure(page);
    for plotIdx = 1:16
        subplot(4, 4, plotIdx);
        segmentIdx = (page - 1) * 16 + plotIdx;
        if segmentIdx <= numOfSegments
            b = bar(coeffs(:, segmentIdx));
            b.FaceColor = 'flat';
            for barIdx = 1:length(notes)
                b.CData(barIdx, :) = colors(mod(barIdx - 1, size(colors,
1)) + 1, :);
            end
            set(gca, 'XTickLabel', notes);
        end
    end
end
```





همانطور که در نمودارها مشاهده می‌شود، نت صحیح ضریب سری فوریه‌ی بیشتری دارد.

```
%% Display recognized notes
disp(recognizedNotes');
```

```
Columns 1 through 11
    {'E'}    {'E'}    {'F'}    {'G'}    {'G'}    {'F'}    {'E'}    {'D'}
{'C'}    {'C'}    {'D'}
  Columns 12 through 22
    {'E'}    {'E'}    {'D'}    {'D'}    {'D'}    {'E'}    {'E'}    {'F'}
{'G'}    {'G'}    {'F'}
  Columns 23 through 32
    {'E'}    {'D'}    {'C'}    {'C'}    {'D'}    {'E'}    {'D'}    {'C'}
{'C'}    {'C'}
```

خروجی به شکل بالا است که با نت‌های این آهنگ در واقعیت همخوانی دارد.

۲ تبدیل فوریه

۱.۲ پیاده سازی محاسبه گر تبدیل فوریه

تابعی به نام general_fourier_transform بنویسید که تبدیل فوریه یک تابع دلخواه محاسبه کند. راهنمایی: برای محاسبه انتگرال در MATLAB، می‌توانید از گزینه‌های زیر استفاده کنید:

- برای محاسبه انتگرال بر روی توابع پیوسته، از دستور integral می‌توانید استفاده کنید. اگر بخواهید انتگرال تابعی گسسته‌زمانی را با استفاده از تابع integral محاسبه کنید، ابتدا باید از کد زیر استفاده کنید: این کد مقادیر تعریف‌نشده تابع در نقاط زمانی را با استفاده از درون‌یابی خطی محاسبه می‌کند.
- برای محاسبه انتگرال بر روی توابع گسسته، می‌توانید از تابع trapz استفاده کنید.

در باقی سوالات تمرین در صورت نیاز به محاسبه تبدیل فوریه از تابعی که در این بخش نوشتید می توانید استفاده کنید.

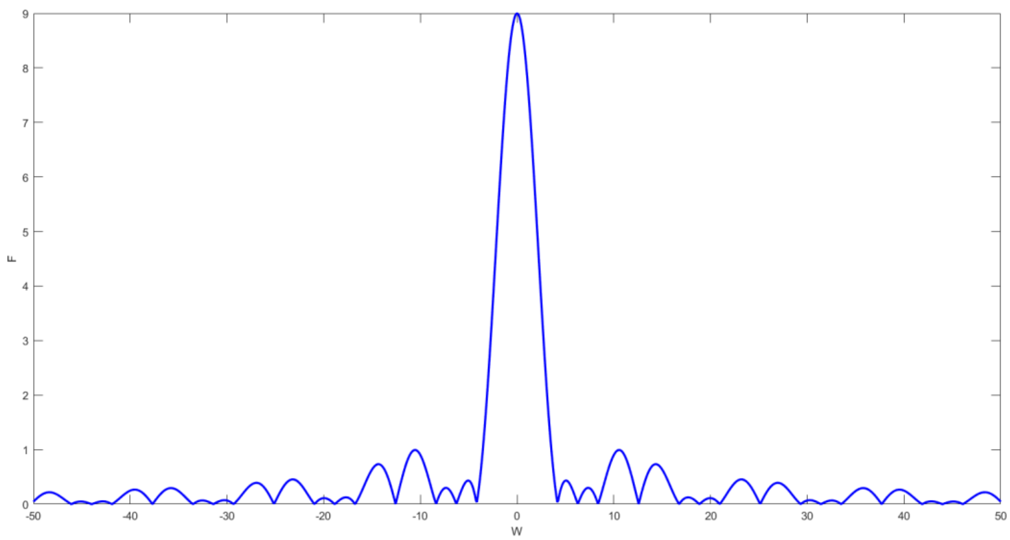
```
f = @(t_query) interp1(t, f_t, t_query, 'linear', 0);  % Interpolates
    f_t values
```

```
function F = general_fourier_transform(f, omega)
    F = arrayfun(@(w) integral(@(t) f(t) .* exp(-1j * w * t), -10, 10),
omega);
end
```

با توجه به فرمول تبدیل فوریه $\int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,dt$، تابع فوق را نوشته‌ایم. به دلیل اینکه محاسبات برنامه طولانی می‌شود به جای ∞- و ∞+ از 10- و 10+ استفاده می‌کنیم. درواقع arrayfun تمام عضوهای لیست را در تابع قرار می‌دهد و خروجی را به صورت یه یک لیست می‌دهد.

۱. تبدیل فوریه $f(t)$ را محاسبه کرده و نمودار آن را در MATLAB رسم کنید.

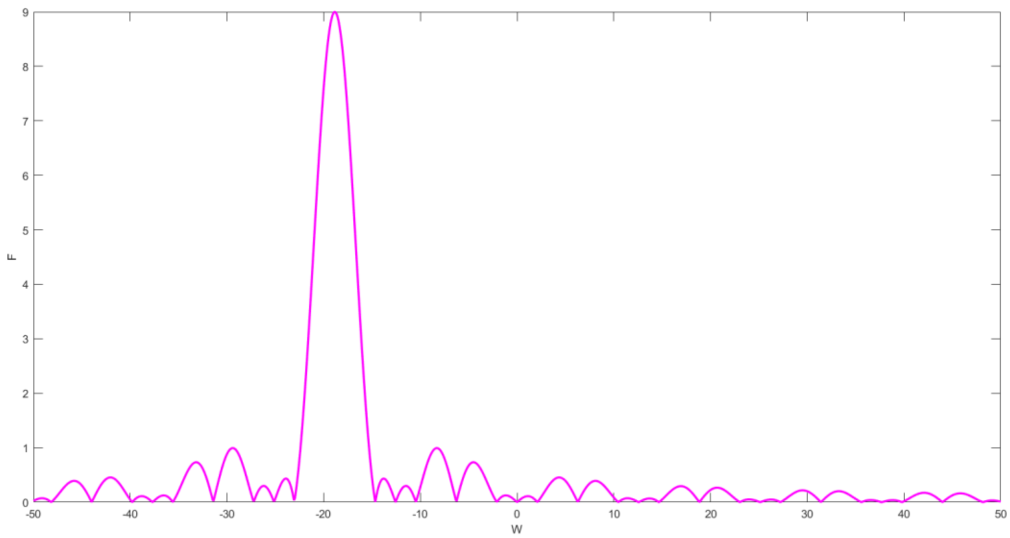
```
f = @(t) 3*((abs(t/2)<=0.5)+(abs(t)<=0.5));
omega=linspace(-50,50,1000);
F_f = general_fourier_transform(f, omega);

figure;
plot(omega, abs(F_f), Color = "blue", LineWidth = 2);
xlabel('W');
ylabel('F');
```



دامنه‌ی امگا از 50- تا 50+ به صورت هزار داده تعریف شده است که تبدیل فوریه تابع $f(t) = 3 \cdot \Pi\left(\frac{t}{2}\right) + \Pi(t)$ روی این دامنه انجام می‌شود و خروجی آن نمودار بالا است.

۲. تابع $g(t) = e^{-j2\pi 3t} f(t)$ را تعریف کنید. تبدیل فوریه $g(t)$ را محاسبه کرده و نمودار آن را نیز در MATLAB رسم کنید.

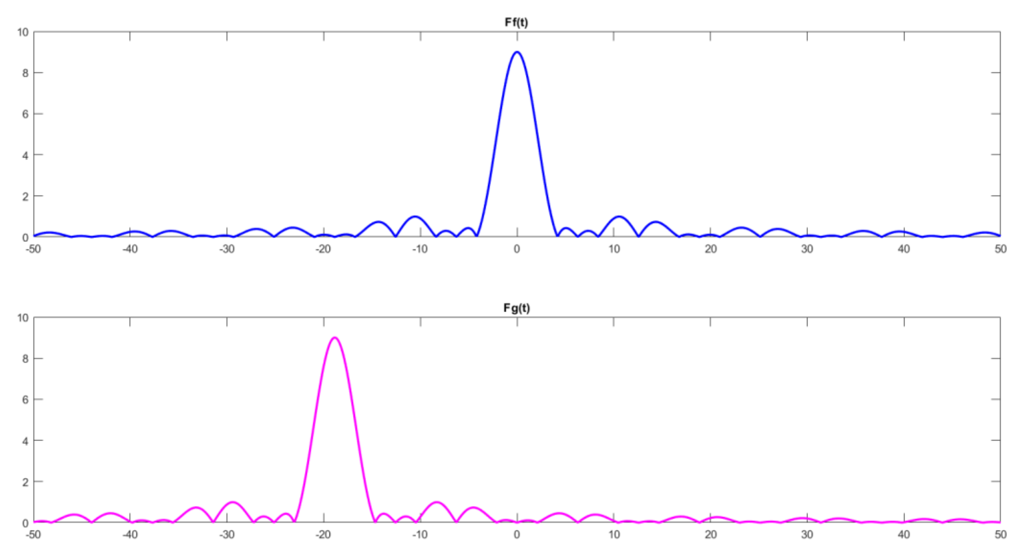
```
g = @(t) exp((-1i)*2*pi*3.*t) .* f(t);
F_g = general_fourier_transform(g,omega);
figure;
plot(omega, abs(F_g), Color = "magenta", LineWidth = 2);
xlabel('W');
ylabel('F');
```



مشابه قسمت قبلی

۳. نمودارهای حاصل از تبدیل فوریه $f(t)$ و $g(t)$ را مقایسه کرده و تحلیل کنید که چگونه ضرب $f(t)$ در $e^{-j2\pi 3t}$ بر تبدیل فوریه آن تأثیر گذاشته است.

```
figure;
subplot(2, 1, 1);
plot(omega, abs(F_f), Color = "blue", LineWidth = 2);
title('F{f(t)}');
subplot(2, 1, 2);
plot(omega, abs(F_g), Color = "magenta", LineWidth = 2);
title('F{g(t)}');
```



برای راحتی در مقایسه کردن این دو تابع، دو ساب پلات از نمودار های قبلی رسم شد. همانطور که مشاهده می‌شود، در بخش دوم 6π شیفت داشتیم که به این علت نمودار به سمت چپ به این اندازه حرکت کرده است.

۲.۲ توابع پنجره‌بندی

در تحلیل سیگنال، بررسی سیگنالی مانند $f(t)$ در بازه‌ی $-\infty < t < \infty$ واقع‌بینانه نیست. جای آن، یک بخش محدودی از سیگنال، مثلاً از $t = -\frac{1}{2}$ تا $t = \frac{1}{2}$ در نظر گرفته می‌شود. این فرآیند به عنوان پنجره‌بندی شناخته می‌شود و با ضرب سیگنال در یک تابع پنجره‌بندی $w(t)$ انجام می‌گیرد.

در این تمرین قصد داریم توضیحات فوق را با محاسبه و تحلیل مثال‌هایی عملی پیاده‌سازی کنیم.

با فرض اینکه تابع پنجره‌بندی به صورت زیر تعریف شده است:

$$w(t) = \Pi(t) = \begin{cases} 1 & |t| \leq \frac{1}{2}, \\ 0 & |t| > \frac{1}{2}. \end{cases}$$

برای هر یک از توابع زیر:

$$f_1(t) = \text{sinc}^2(t),$$

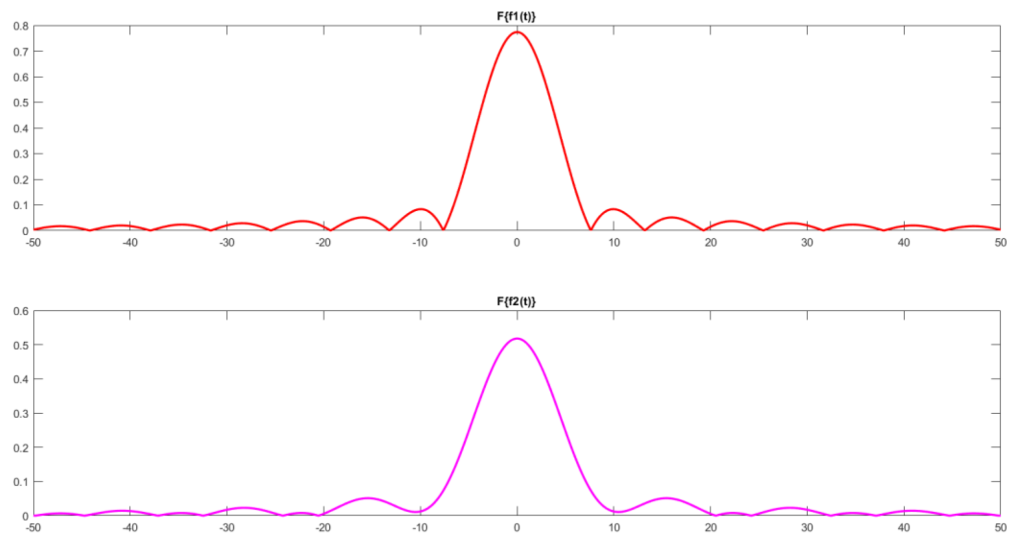
$$f_2(t) = e^{-3|t|},$$

مراحل زیر را انجام دهید:

۱. ابتدا تابع $f(t)$ را در تابع پنجره‌ای $w(t)$ ضرب کنید. سپس تبدیل فوریه حاصل را محاسبه کرده و نمودار آن را در MATLAB رسم کنید.

```
f1 = @(t) sinc(t).^2;
f2 = @(t) exp(-3 * abs(t));
w = @(t) abs(t) <= 0.5;

omega = linspace(-50, 50, 1000);
F1 = general_fourier_transform((@(t) f1(t) .* w(t)), omega);
F2 = general_fourier_transform((@(t) f2(t) .* w(t)), omega);

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

```
plot(omega, abs(F1), Color = 'Red', LineWidth= 2);
title('F\{f1(t)\}');

subplot(2, 1, 2);
plot(omega, abs(F2), Color = 'Magenta', LineWidth= 2);
title('F\{f2(t)\}');
```

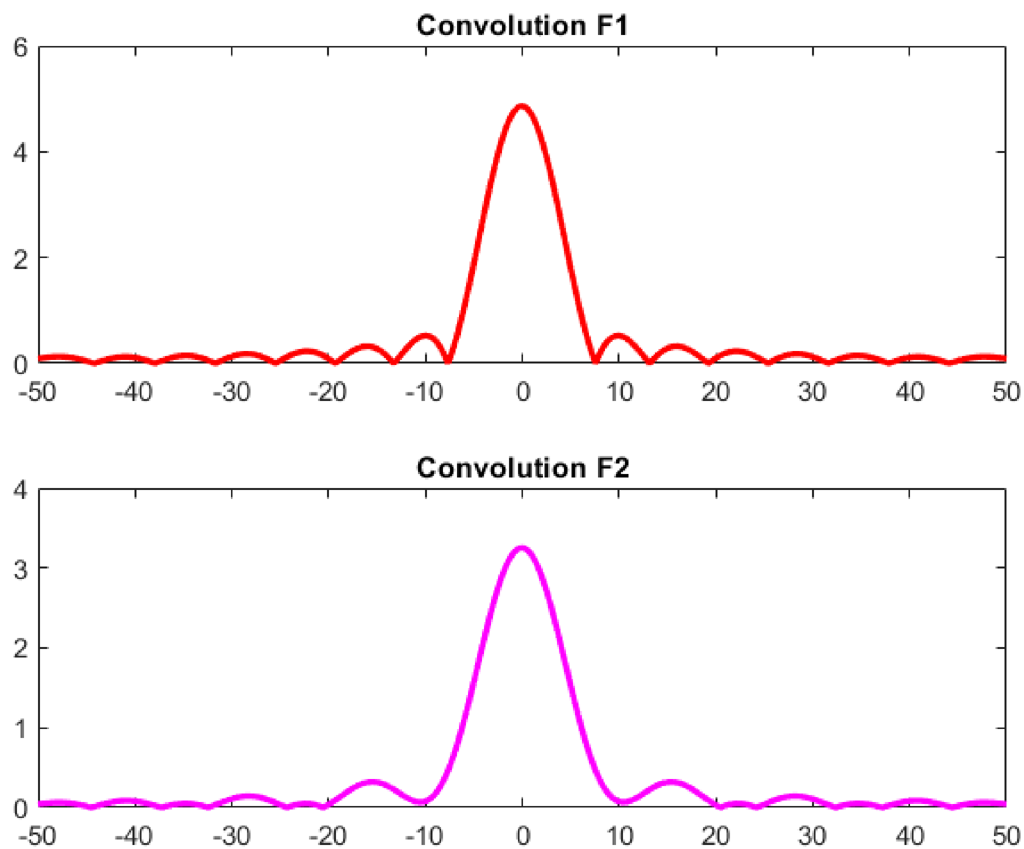


در ابتدا هر تابع در تابع پنجره‌بندی ضرب شده و سپس تبدیل فوریه آن گرفته شده و نمایش داده شده است.

۲. تبدیل فوریه تابع $w(t)$ و $f(t)$ را به صورت جداگانه حساب کنید. سپس کانولوشن این دو تبدیل فوریه را در حوزه فرکانس محاسبه کنید. نتایج را در MATLAB رسم کنید.

```
%%
F1 = general_fourier_transform(f1, omega);
F2 = general_fourier_transform(f2, omega);
W = general_fourier_transform(w, omega);

F1Convolution = conv(F1, W,'same') * (omega(2) - omega(1));
F2Convolution = conv(F2, W,'same') * (omega(2) - omega(1));
figure;
subplot(2, 1, 1);
plot(omega, abs(F1Convolution), Color = 'Red', LineWidth= 2);
title('Convolution F1');
```

```
subplot(2, 1, 2);
plot(omega, abs(F2Convolution), Color = 'Magenta', LineWidth= 2);
title('Convolution F2');
```



اینجا ابتدا تبدیل فوریه هر تابع گرفته شده و سپس کانولوشن تبدیل فوریه تابع و تابع پنجره‌بندی گرفته شده و نمایش داده شده است.

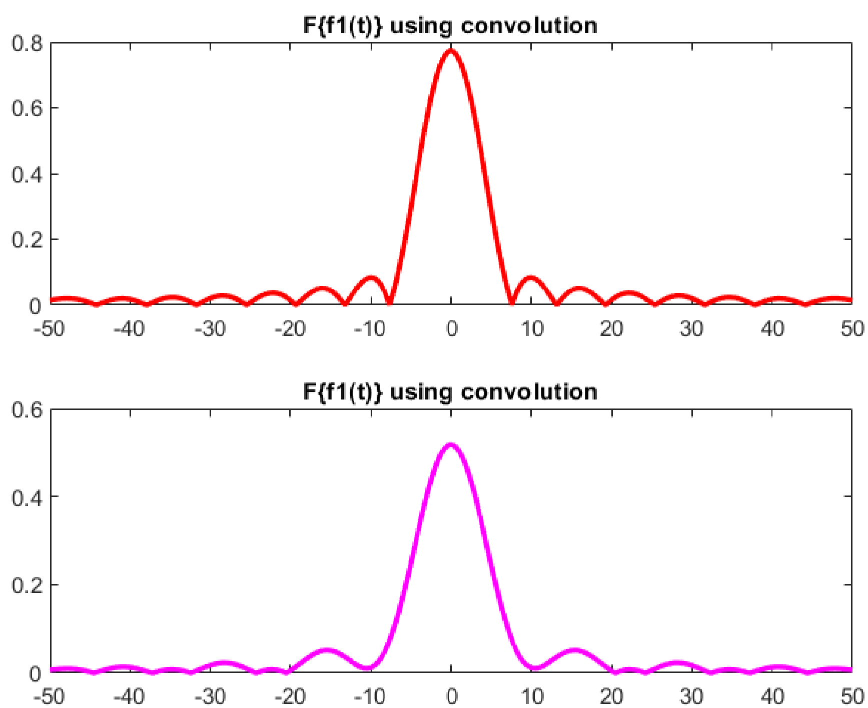
۳. خروجی‌های مراحل بالا را با یکدیگر مقایسه کنید. بر اساس خواص تبدیل فوریه، تفاوت یا شباهت نتایج را توجیه کنید.

تبدیل فوریه حاصل ضرب دو تابع به صورت زیر است:

$$F\{f_1(t) \cdot f_2(t)\} = \frac{1}{2\pi}[F_1(\omega) \cdot F_2(\omega)]$$

هدف در این سوال اثبات عبارت بالا است. نکته‌ای که وجود دارد این است که نمودار کانولوشن به اندازه 2π اختلاف دارد. برای اینکه دقیقا ببینیم که دو عبارت بالا برابر است، کد زیر را اجرا می‌کنیم که نتیجه آن به شرح زیر است :

```
ffigure;
subplot(2,1,1)
plot(omega, abs(F1Convolution / (2 * pi)), Color = 'Red', LineWidth= 2);
title('F\{f1(t)\} using convolution');

subplot(2, 1, 2);
plot(omega, abs(F2Convolution / (2 * pi)), Color = 'Magenta', LineWidth=
2);
title('F\{f1(t)\} using convolution');
```



همانطور که مشاهده می‌شود نمودار بالا با نمودار تبدیل فوریه به دست آمده از تابع نوشته شده دقیقا یکی است.

۳.۲ تخمین تبدیل فوریه با تابع مثلثی

۱.۳.۲ تابع مثلثی

$$\Lambda(\frac{t}{T}) = \begin{cases} 1 - \frac{|t|}{T}, & \text{if} |t| < T, \\ 0, & \text{otherwise.} \end{cases}$$

تابعی به نام triangular_function بنویسید که رابطه‌ی ریاضی تابع مثلثی را در لحظات t خروجی دهد:

- t: بردار زمانی
- T: نصف پهنای زمانی تابع مثلثی
- Shift: مقدار شیفت سیگنال در زمان

```
function A = triangular_function(t, T, Shift)
    A = (abs(t - Shift) < T) .* (1 - abs(t - Shift) / T);
end
```

تابع مثلثی به این صورت نوشته شده است که اگر شرط تابع رعایت شود، حاصل ۱ است و ضرب در مقدار خروجی ممکن می‌شود و اگر شرط رعایت نشود، حاصل ۰ است که خروجی تابع را نیز صفر می‌کند.

۲.۳.۲ تبدیل فوریه تابع مثلثی

تابعی به نام triangular_fourier_transform بنویسید که تبدیل فوریه تابع مثلثی را محاسبه کند. ورودی تابع به صورت زیر خواهد بود:

- T: نصف پهنای زمانی تابع مثلثی
- w_vals: برداری از مقادیر فرکانس

برای محاسبه‌ی انتگرال از دستور integral در MATLAB استفاده کنید. این دستور برای محاسبه‌ی انتگرال روی توابع پیوسته تعریف شده است. در نتیجه، تابع مثلثی را به صورت پیوسته با کمک دستور @ در MATLAB تعریف کرده و آن را به دستور integral ورودی دهید.

```
function F = triangular_fourier_transform(T, omega)
    A = @(t) triangular_function(t, T, 0);
    F = arrayfun(@(w) integral(@(t) A(t) .* exp(-1j * w * t), -T, T),
omega);
end
```

این تابع تبدیل فوریه تابع مثلثی را در تمام امگاهای داده‌شده محاسبه می‌کند و خروجی می‌دهد. در اینجا بازه انتگرال گیری از T- تا T است که به عنوان ورودی از کاربر به عنوان نصف پهنای زمانی تابع مثلثی می‌گیرد.

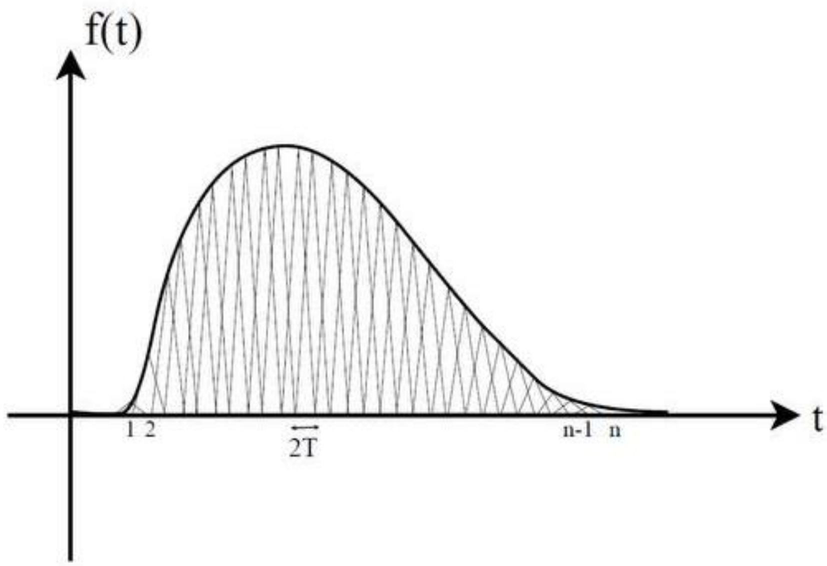
۳.۳.۲ تخمین تبدیل فوریه با تابع مثلثی

در این بخش می خواهیم با استفاده از توابع بخش های قبل، تابعی به نام approximate_fourier_transform بنویسیم که تبدیل فوریه تابعی دلخواه را با کمک توابع مثلثی به صورت تقریبی محاسبه کند. ورودی تابع به فرم زیر خواهد بود :

- f: تابعی دلخواه
- t: بردار زمانی
- w_vals: برداری از مقادیر فرکانس

برای استفاده از این روشِ تخمین، این تابع دلخواه می‌بایست شرایط زیر را داشته باشد تا بتوانیم آن را به صورت مجموع توابع مثلثی شیفت‌یافته تخمین بزنیم:

- در بازه‌ای پیوسته از زمان تعریف شده باشد و در خارج از این بازه صفر باشد.
- مقدار تابع در نقاط ابتدایی و انتهایی این بازه صفر باشد.



شکل ۱: تقریب مثلثی از تابع

راهنمایی: نقاط ابتدایی و انتهایی بازه‌ای را که تابع در آن مقدار غیر صفر دارد را پیدا کنید. برای این کار می‌توانید از دستور find استفاده کنید. این بازه را به تعداد زیادی زیربازه‌هایی با طول یکسان تقسیم کنید که، طبق شکل هر یک طول T را خواهند داشت. حال به ازای هر w_vals، تبدیل فوریه تابع را با پیاده‌سازی روابط ریاضی زیر به دست آورید.
تابع $f(t)$ به صورت زیر تخمین زده می شود:

$$f(t) = \sum_{k=1}^{n-1} f(kT) \cdot \Lambda_i(t - kT).$$

حال، تبدیل فوریه تابع فوق را محاسبه می‌کنیم:

$$F(\omega) = \mathcal{F}\left\{\sum_{k=1}^{n-1} f(kT)\cdot\Lambda_i(t-kT)\right\} = \sum_{k=1}^{n-1} e^{-i\omega kT}\cdot f(kT)\cdot\mathcal{F}\{\Lambda_i(t)\}.$$

```
function F = approximate_fourier_transform(f, t, w_vals)
    idx = f(t) ~= 0;
    T = t(2) - t(1);
    tNew = t(idx):T:t(end);
    FTriangular = triangular_fourier_transform(T, w_vals);
    F = arrayfun(@(w, i) sum(exp(-1j * w * tNew) .* f(tNew) .*
FTriangular(i)), w_vals, 1:length(w_vals));
end
```

این تابع در ابتدا نقاط غیر صفر تابع اصلی را برمی‌دارد تا فقط با اون داده‌ها کار کند. بعد تبدیل فوریه تابع مثلثی را محاسبه می‌کند تا بخش سمت راست فرمول نهایی که در بالا هست، به دست بیاید. بعد در نهایت خود تابع برای هر داده با توجه به فرمول داده شده محاسبه می‌شود و خروجی می‌دهد.

```
w = @(t) abs(t) <= 0.5;
```

برای بررسی صحت عملکرد توابعی که در بخش‌های قبل نوشتید، تبدیل فوریه تابعِ

$$f_1(t) = \Pi(\frac{t}{5}).(1-cos(\frac{2\pi t}{5}))$$
$$f_2(t) = \Lambda(t) + \Lambda(t-2) + \Lambda(t-4) + \Lambda(t-6)$$

را به سه روش محاسبه کنید:

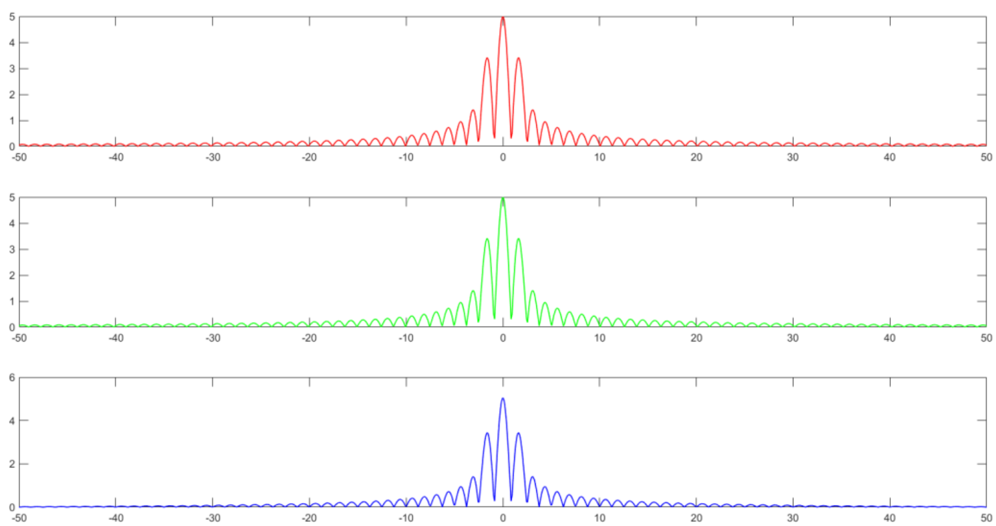
۱. به صورت تئوری و مشابه مثال حل‌شده در جزوه

۲. با کمک تابع general_fourier_transform

۳. با کمک تابع approximate_fourier_transform

با کمک دستور subplot، خروجی هر روش را زیر یکدیگر plot کرده و یکسان بودن آن‌ها را نتیجه بگیرید.

```
f = @(t) w(t/5).*(1-cos(2*pi*t/5));
Ff = @(omega) 5 * (sinc(5/(2*pi) * omega) - 1/2 * sinc(5/(2*pi)*(omega -
2*pi/5)) - 1/2 * sinc(5/(2*pi)*(omega + 2*pi/5)));

FfTheory = Ff(omega);
FfGeneral=general_fourier_transform(f , omega);
FfApproximate=approximate_fourier_transform(f,t,omega);

figure;
subplot(3,1,1)
plot(omega,abs(FfTheory), Color = 'Red', LineWidth = 1)

subplot(3,1,2)
plot(omega,abs(FfGeneral), Color = 'Green', LineWidth = 1)

subplot(3,1,3)
plot(omega,abs(FfApproximate), Color = 'Blue', LineWidth = 1)
```



در اینجا ۳ نمودار که به ترتیب تبدیل فوریه به روش تئوری، با استفاده از تابع general_fourier_transform و تابع approximate_fourier_transform برای تابع f1 هست را داریم. که بخش دوم و سوم نتایج یکسانی دارند.

محاسبات برای بخش تئوری :

$$f_1(t) = \sqcap\left(\frac{t}{5}\right)\cdot\left(1-\cos\left(\frac{2\pi t}{5}\right)\right)$$

$$\hookrightarrow \; F\{f_1(t)\} = F\left(\sqcap\left(\frac{t}{5}\right)\cdot\left(1-\cos\left(\frac{2\pi t}{5}\right)\right)\right)$$

$$= \frac{1}{2\pi}\left[F\left(\sqcap\left(\frac{t}{5}\right)\right) * F\left(1-\cos\left(\frac{2\pi t}{5}\right)\right)\right]$$

$$\hookrightarrow \; F\left\{\sqcap\left(\frac{t}{5}\right)\right\} = \underline{5\,\mathrm{Sinc}(5\omega)}$$

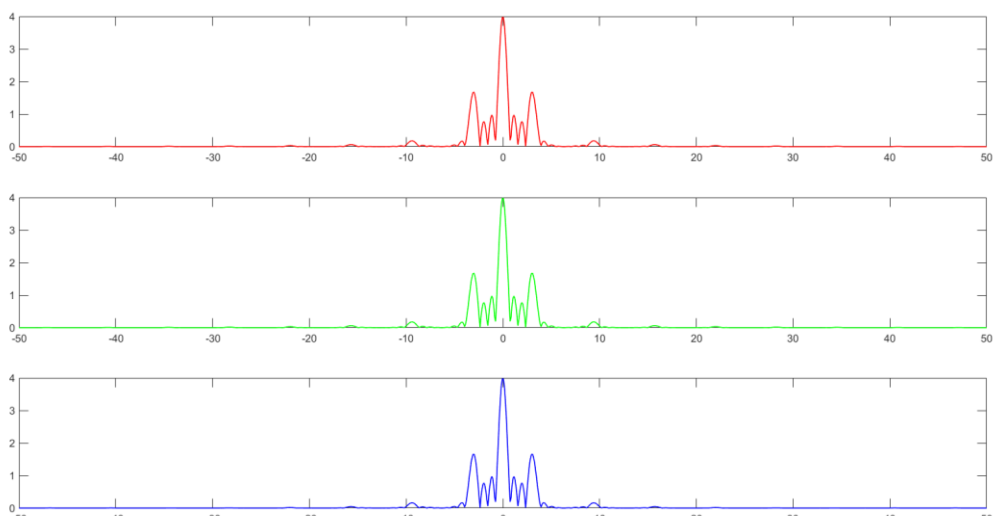
$$F\left\{1-\cos\left(\frac{2\pi t}{5}\right)\right\} = 2\pi\,\delta(\omega) - \pi\,\delta\left(\omega-\frac{2\pi}{5}\right) - \pi\,\delta\left(\omega+\frac{2\pi}{5}\right)$$

$$\hookrightarrow \; F\{f_1(t)\} = \left(5\,\mathrm{Sinc}(5\omega)\right) * \left(2\pi\,\delta(\omega) - \pi\,\delta\left(\omega-\frac{2\pi}{5}\right) - \pi\,\delta\left(\omega+\frac{2\pi}{5}\right)\right)$$

$$\hookrightarrow \; F\{f_1(t)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} 5\,\mathrm{Sinc}(5\omega')\left[2\pi\,\delta(\omega-\omega') - \pi\,\delta\left(\omega-\omega'-\frac{2\pi}{5}\right) - \pi\,\delta\left(\omega-\omega'+\frac{2\pi}{5}\right)\right]d\omega'$$

$$\Longrightarrow F\{f_1(t)\} = 5\left[\mathrm{Sinc}(5\omega) - \frac{1}{2}\,\mathrm{Sinc}\left(5\left(\omega-\frac{2\pi}{5}\right)\right) - \frac{1}{2}\,\mathrm{Sinc}\left(5\left(\omega+\frac{2\pi}{5}\right)\right)\right]$$

```
Fg = @(omega) sinc(omega / 2 / (pi)) .^2 .* (1 + exp(-1j * omega * 2) +
exp(-1j * omega * 4) + exp(-1j * omega * 6));

g = @(t)
triangular_function(t,1,0)+triangular_function(t,1,2)+triangular_function(
t,1,4)+triangular_function(t,1,6);
t = linspace(-10,10,200);
FgTheory = abs(Fg(omega));
FgGeneral=abs(general_fourier_transform(g , omega));
FgApproximate=abs(approximate_fourier_transform(g,t,omega));

figure;
subplot(3,1,1)
plot(omega,abs(FgTheory), Color = 'Red', LineWidth = 1);

subplot(3,1,2)
plot(omega,abs(FgGeneral), Color = 'Green', LineWidth = 1);

subplot(3,1,3)
plot(omega,abs(FgApproximate), Color = 'Blue', LineWidth = 1);
```



در اینجا ۳ نمودار که به ترتیب تبدیل فوریه به روش تئوری، با استفاده از تابع general_fourier_transform و تابع approximate_fourier_transform برای تابع f2 هست را داریم. که بخش دوم و سوم نتایج یکسانی دارند.

محاسبات برای بخش تئوری :

$$f_2(t) = \Lambda(t) + \Lambda(t-2) + \Lambda(t-4) + \Lambda(t-6)$$

$$\hookrightarrow F\{\Lambda(t)\} = \frac{8a}{\tau}\left(\frac{\sin\left(\frac{\omega\tau}{4}\right)}{\omega}\right)^2$$

$$\xrightarrow[\tau = 2T = 2]{a=1} \quad \frac{4}{w^2}\sin^2\left(\frac{w}{2}\right) = \operatorname{sinc}^2\left(\frac{w}{2}\right)$$

$$\hookrightarrow \quad F\{f_2(t)\} = F(\Lambda(t))\left[1 + e^{-2iw} + e^{-4iw} + e^{-6iw}\right]$$

$$= \operatorname{sinc}^2\left(\frac{\omega}{2}\right)\left[1 + e^{-2i\omega} + e^{-4i\omega} + e^{-6i\omega}\right]$$

نکته‌ای که وجود دارد این است که محاسبات روی کاغذ با نتیجه‌ای که به عنوان تئوری در کد وارد شده، متفاوت است. به این دلیل است که در متلب تابع sinc به صورت زیر است :

$$\sin c(\omega) = \frac{\sin(2\pi\omega)}{2\pi\omega}$$

بنابراین در هر تابع یک /2π قرار می‌دهیم تا به تابع درست برسیم.