تمرین دوم کامپیوتری سیگنال‌ها و سیستم‌ها

محدثه حکیمی

۴۰۱۲۴۳۰۴۵

سری فوریه

الف) برای این بخش لازمه که ضرایب سری فوریه رو با فرمول‌های داده شده به دست بیاریم. نکته‌ای که باید بهش توجه کنیم اینه که توی فرمول‌ها انتگرال داریم و برای انتگرال از np.trapz استفاده می‌کنیم.

تابع محاسبه ضرایب:

```python
import numpy as np


def fourier_series(x_t, t, n):
    a_n = np.zeros(n)
    b_n = np.zeros(n)
    T = abs(t[-1] - t[0])
    for i in range(n):
        a_n[i] = 2 / T * np.trapz(x_t * np.cos(2 * np.pi * i * t / T), t)
        b_n[i] = 2 / T * np.trapz(x_t * np.sin(2 * np.pi * i * t / T), t)
    a_0 = 2 / T * np.trapz(x_t, t)
    return a_0, a_n, b_n
```

ب) این بخش هم دوباره پیاده کردن فرموله که باید راه برعکس رو بریم و با ضرایب به دست اومده در بخش الف، تابع رو دوباره بسازیم.

Q1-B

```python
def inverse_fourier_series(x_t, t, n):
    a_0, a_n, b_n = fourier_series(x_t, t, n)
    T = abs(t[-1] - t[0])
    x_t_hat = a_0 / 2
    for i in range(n):
        x_t_hat += a_n[i] * np.cos(2 * np.pi * i * t / T) + b_n[i] * np.sin(2 * np.pi * i * t / T)
    return x_t_hat
```

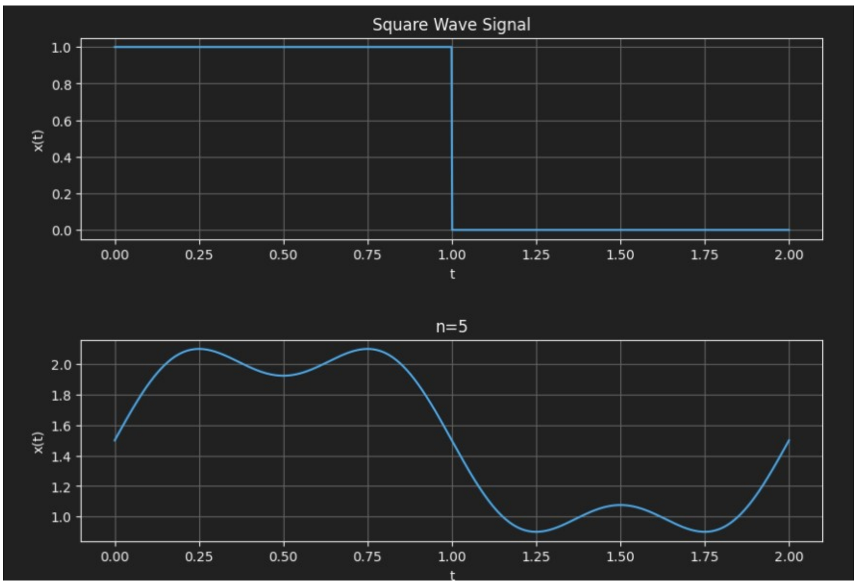
ج) برای این بخش هم باید سیگنال مثلثی و پالس مربعی رو برای N های مختلف تقریب بزنیم.

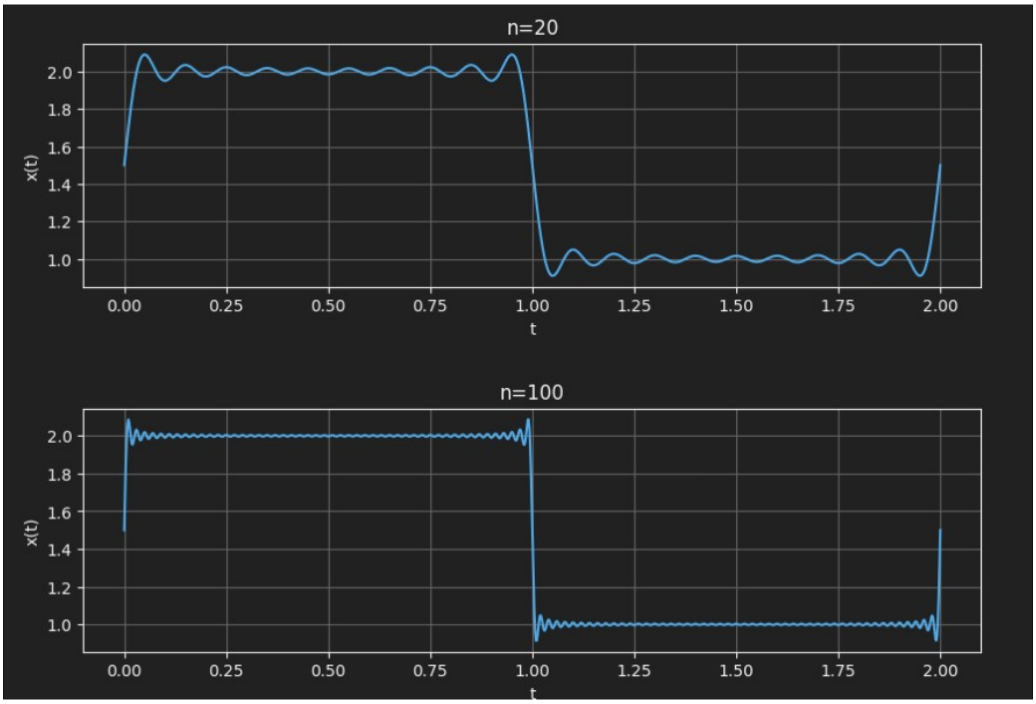
اول پالس مربعی:

```python
import matplotlib.pyplot as plt

# square wave signal
x = np.linspace(0, 2, 1000)
x_t = np.zeros(1000)
x_t[0:500] = 1
x_t[500:] = 0

# plot the square wave signal
plt.figure(figsize=(10, 15))
plt.subplot(4, 1, 1)
plt.plot(x, x_t)
plt.title('Square Wave Signal')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.grid()

# plot the square wave signal with n=5
x_t_hat = inverse_fourier_series(x_t, x, 5)
plt.subplot(4, 1, 2)
```

بقیه‌ی کد هم تکرار این بخش برای nهای مختلفه که اینجا نمیارم تا الکی شلوغ نشه.

خروجی:



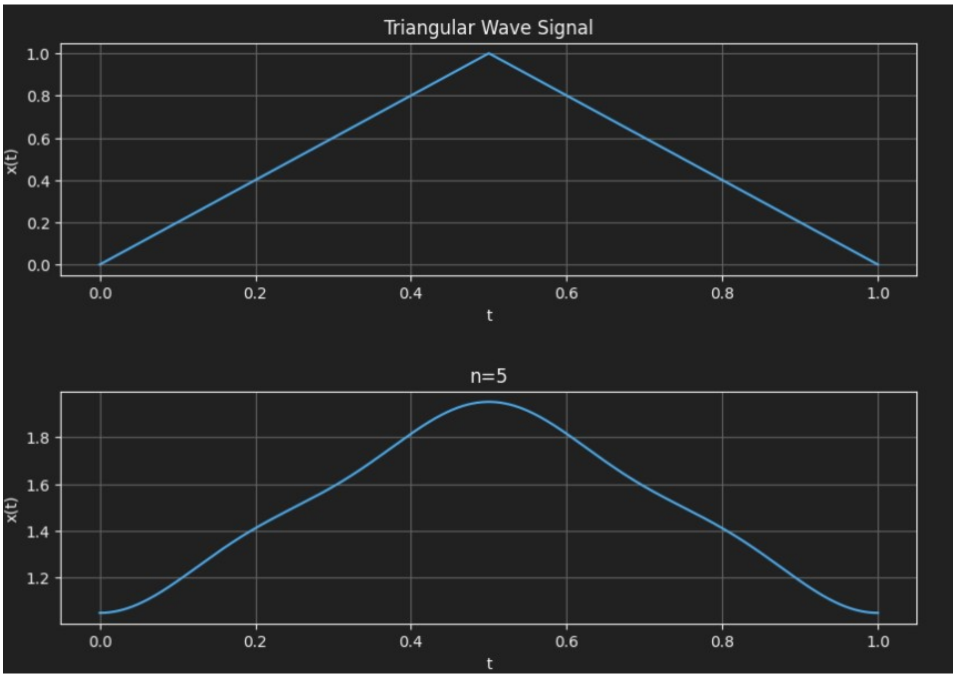


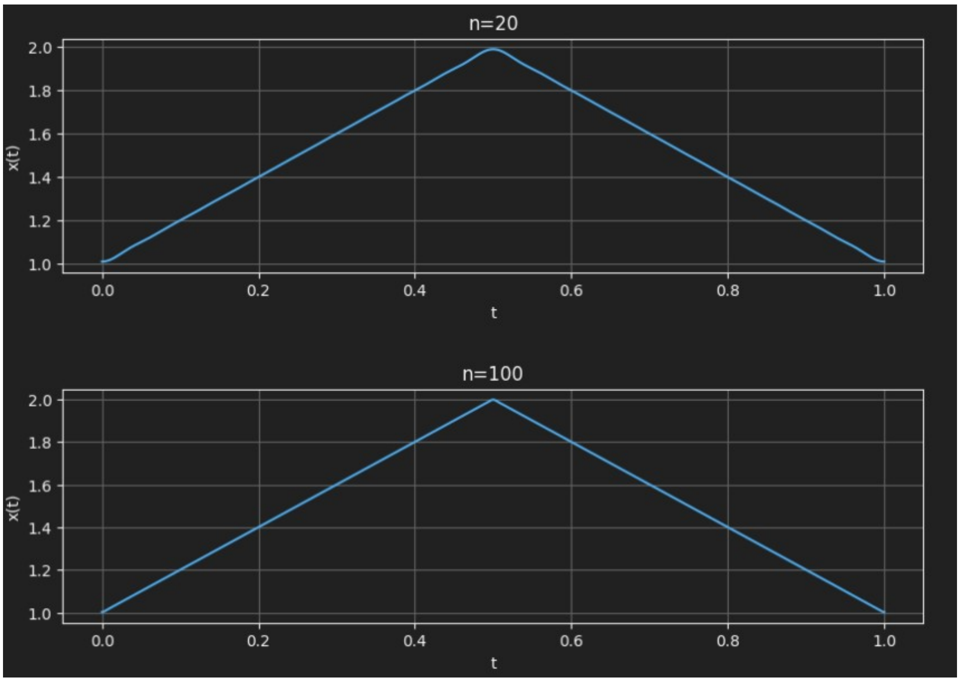
که همونطور که مشخصه با افزایش n، تقریب ما دقیق‌تر می‌شه چون جملات بیشتری رو جمع می‌زنیم.

حالا سیگنال مثلثی (که به بخشی از کد بسنده می‌کنیم):

```python
# triangular wave signal
x = np.linspace(0, 1, 1000)
x_t = np.zeros(1000)
x_t[0:500] = x[0:500] * 2
x_t[500:] = 2 - x[500:] * 2

# plot the triangular wave signal
plt.figure(figsize=(10, 15))
plt.subplot(4, 1, 1)
plt.plot(x, x_t)
plt.title('Triangular Wave Signal')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.grid()

# plot the triangular wave signal with n=5
x_t_hat = inverse_fourier_series(x_t, x, 5)
plt.subplot(4, 1, 2)
plt.plot(x, x_t_hat)
```

خروجی:





که باز هم سیگنال $n = 100$ از باقی حالت‌ها به سیگنال اصلی نزدیک‌تره.

د) پدیده‌ی گیبس به رفتاری گفته می‌شه که برای سری فوریه‌ی یک سیگنال با ناپیوستگی‌های پرش‌دار (تغییر ناگهانی مقدار سیگنال) به وجود میاد. درواقع در نزدیکی ناپیوستگی‌ها نوساناتی در مقدار پیش میاد که هیچ وقت نمی‌تونن به طور کامل برطرف بشن، هر چقدر هم که جملات بیشتری از سری رو حساب کنیم. پدیده‌ی گیبس به این دلیل اتفاق می‌افته که سری فوریه می‌خواد سیگنال رو به شکل جمعی از سینوسی‌ها بنویسه که برای نقاط ناپیوستگی نمی‌تونه به صورت یکنواخت همگرا بشه و نوسانات شکل می‌گیرن.

در بخش ج، برای تابع پالس مربعی این اتفاق افتاد که به دلیل وجود همون تغییرات ناگهانی در مقدار سیگناله و در شکل رسم شده هم دیده می‌شه.
اما این پدیده برای سیگنال مثلثی رخ نداد چرا که مقادیر سیگنال به شکل هموار تغییر می‌کنن و تغییرات ناگهانی نداریم.

تبدیل فوریه

۲ – الف) این بخش هم دوباره پیاده‌سازی فرموله:

$$X(\omega) = \int_{-\infty}^{+\infty} x(t).e^{-j\omega t}dt$$

```
def fourier_transform(x_t, t, w):
    X_w = np.zeros(len(w), dtype=complex)
    for i in range(len(w)):
        X_w[i] = np.trapz(x_t * np.exp(-1j * w[i] * t), t)
    return X_w
```

ب) اینجا هم استفاده از رابطه‌ی سنتز:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\omega).e^{j\omega t}d\omega$$

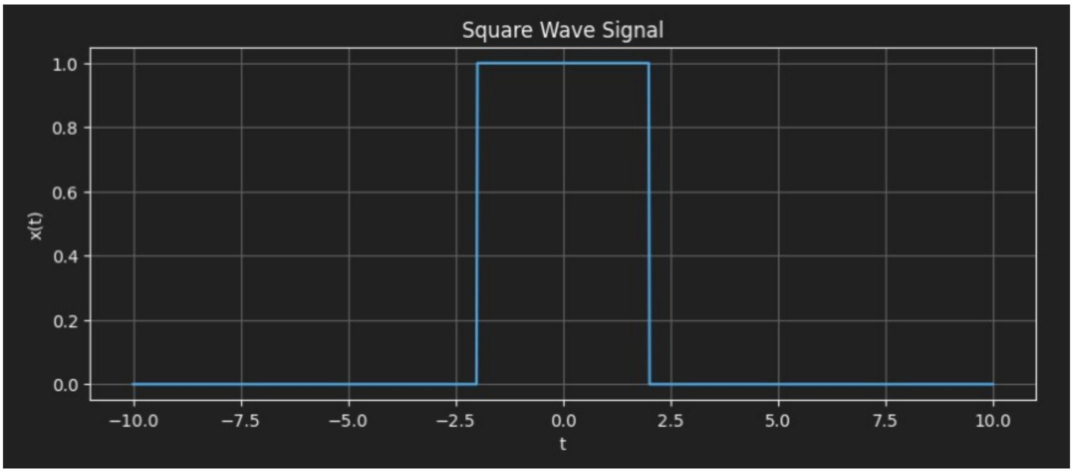
```
def inverse_fourier_transform(X_w, w, t):
    x_t = np.zeros(len(t), dtype=complex)
    for i in range(len(t)):
        x_t[i] = np.trapz(X_w * np.exp(1j * w * t[i]), w) / (2 * np.pi)
    return x_t
```

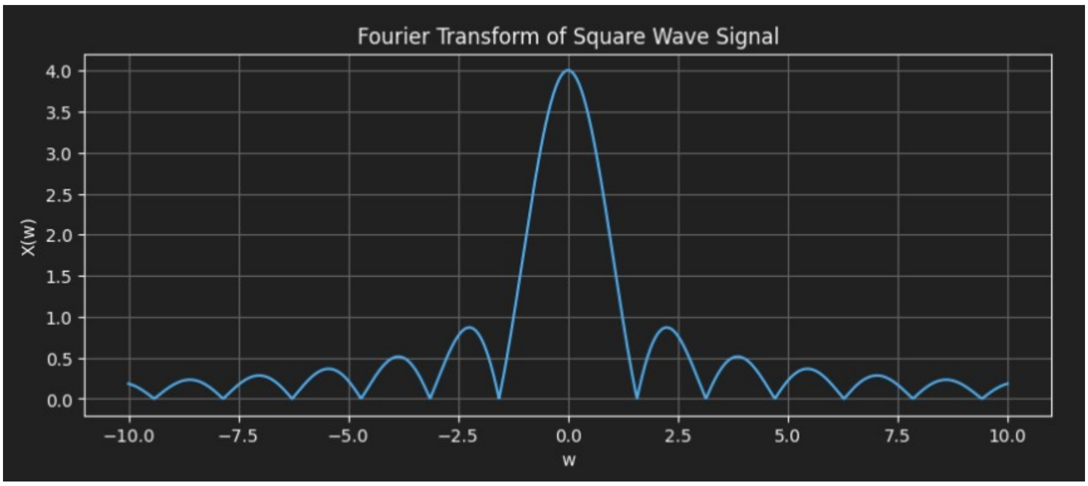
ج) برای این قسمت یک پالس مربعی که از منفی دو تا دو برابر یکه و باقی جاها صفره می‌سازیم و تبدیل فوریه و معکوسش رو حساب می‌کنیم:

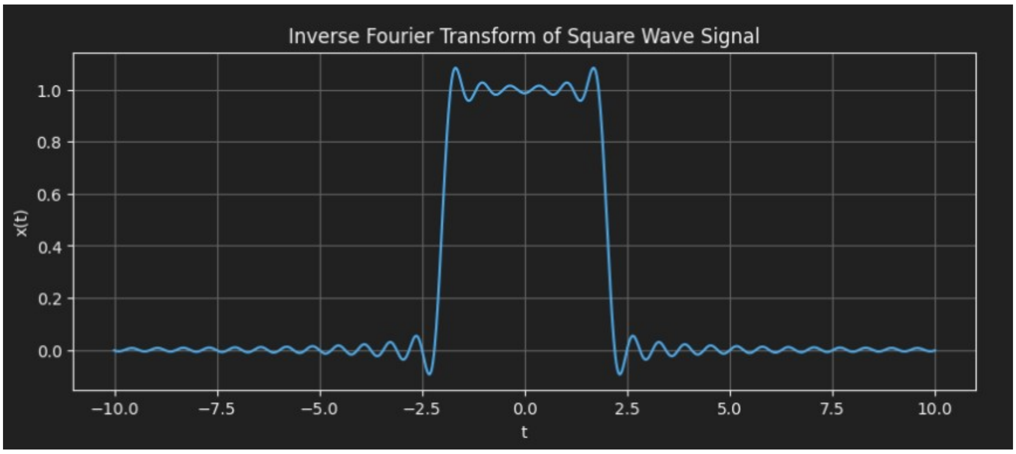
```python
# square wave signal
x = np.linspace(-10, 10, 1000)
x_t = np.zeros(1000)
x_t[400:600] = 1

# plot the square wave signal
plt.figure(figsize=(10, 15))
plt.subplot(3, 1, 1)
plt.plot(x, x_t)
plt.title('Square Wave Signal')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.grid()

plt.subplots_adjust(hspace=0.5)

# fourier transform of the square wave signal
w = np.linspace(-10, 10, 1000)
X_w = fourier_transform(x_t, x, w)
```

```python
# fourier transform of the square wave signal
w = np.linspace(-10, 10, 1000)
X_w = fourier_transform(x_t, x, w)
plt.subplot(3, 1, 2)
plt.plot(w, np.abs(X_w))
plt.title('Fourier Transform of Square Wave Signal')
plt.xlabel('w')
plt.ylabel('X(w)')
plt.grid()

# inverse fourier transform of the square wave signal
x_t_hat = inverse_fourier_transform(X_w, w, x)
plt.subplot(3, 1, 3)
plt.plot(x, x_t_hat)
plt.title('Inverse Fourier Transform of Square Wave Signal')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.grid()
```

خروجی:







به شکل تبدیل فوریه‌ی تابع پالس مربعی تابع sinc گوییم.

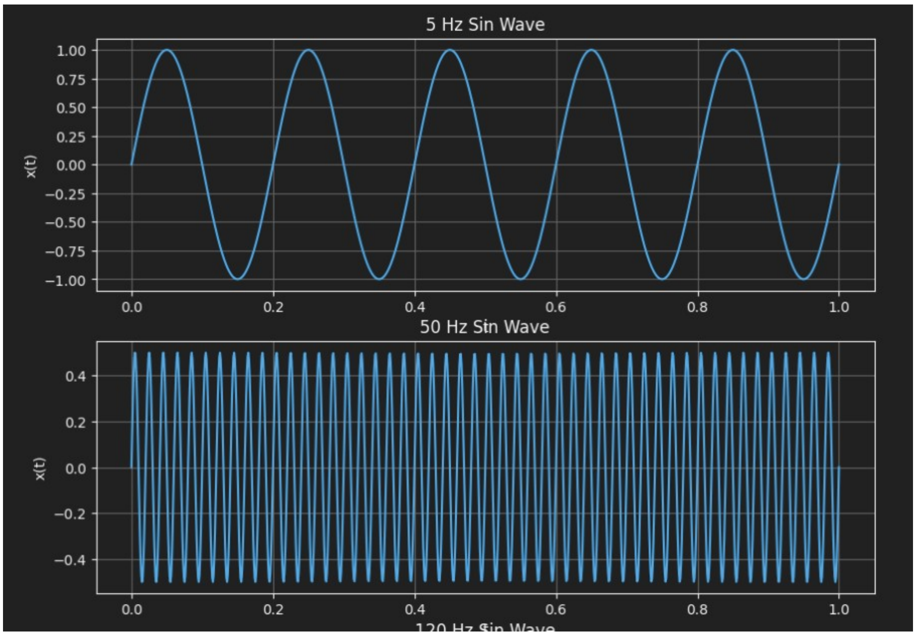
۳ – اول ۳ تابع سینوسی گفته شده و main_signal رو می‌سازیم:

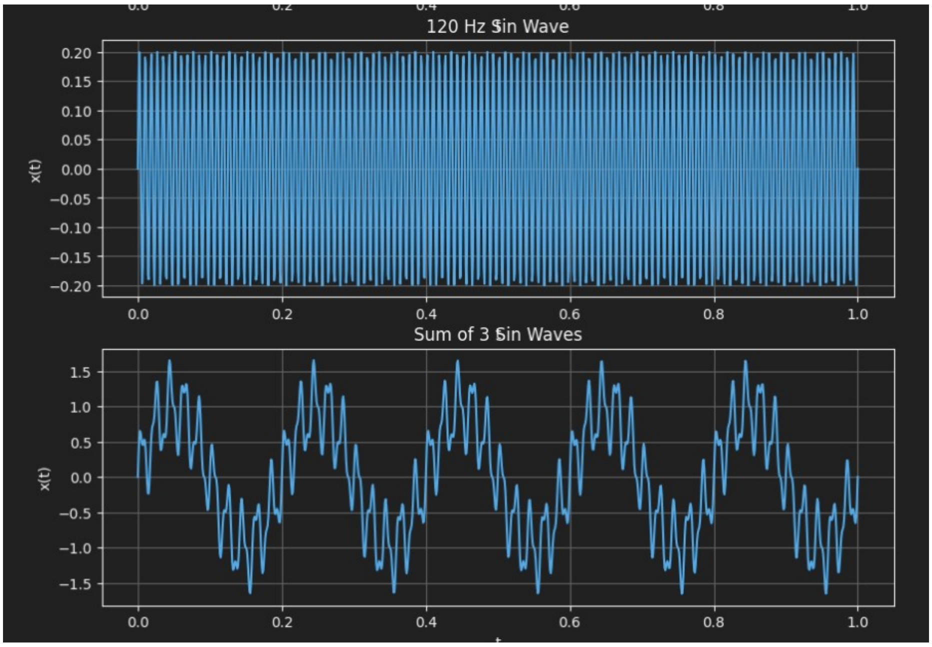
```python
# 3 sin waves with 5, 50, 120 Hz and domains of 1, 0.5, 0.2
T = 1/1000
t = np.linspace(0, 1, 1000)
x_t1 = np.sin(2 * np.pi * 5 * t)
x_t2 = 0.5 * np.sin(2 * np.pi * 50 * t)
x_t3 = 0.2 * np.sin(2 * np.pi * 120 * t)

# sum up the 3 sin waves
main_signal = x_t1 + x_t2 + x_t3
```

رسم هر چهار سیگنال به کمک ساب‌پلات (بخشی از کد):

```python
# plot all 4 signals
plt.figure(figsize=(10, 15))
plt.subplot(4, 1, 1)
plt.plot(t1, x_t1)
plt.title('5 Hz Sin Wave')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.grid()
```

و خروجی:

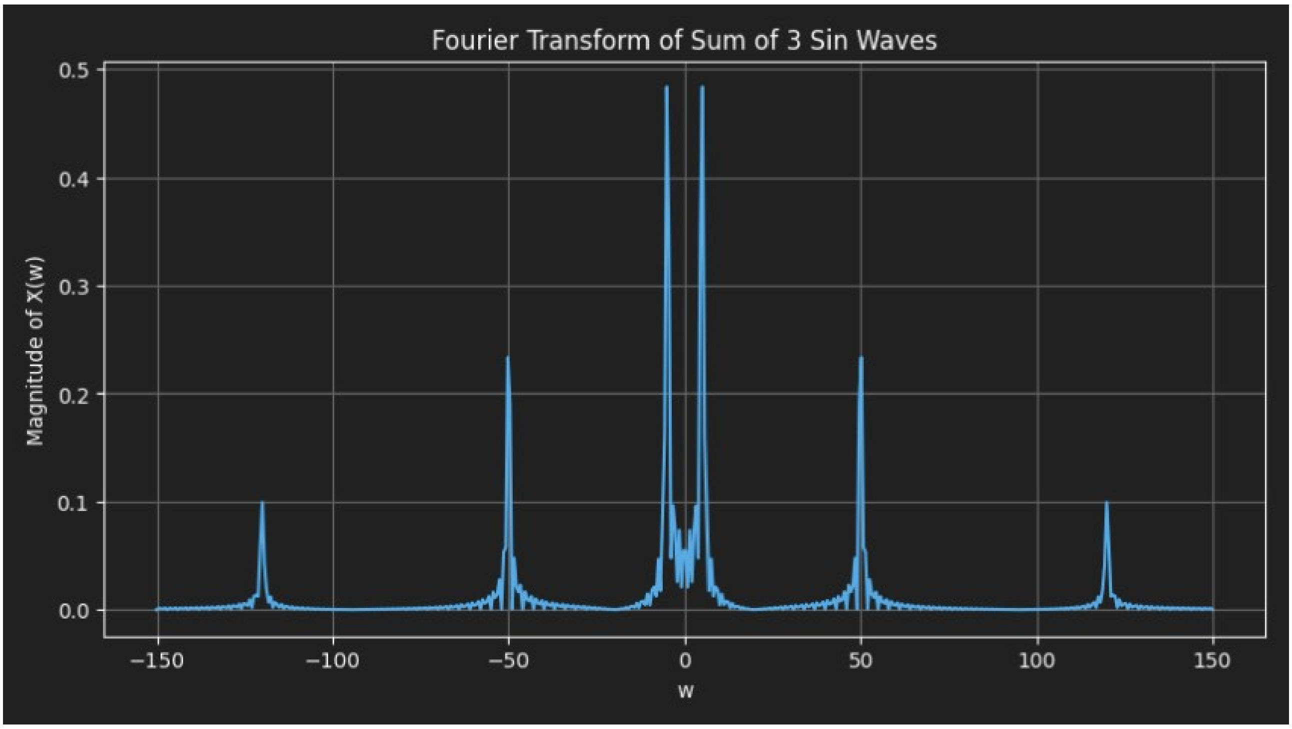




حالا تبدیل فوریه‌ی سیگنال رو محاسبه می‌کنیم و نمایش می‌دیم:

```
# calculate the fourier transform of the main signal
w = 2 * np.pi * np.linspace(-150, 150, 500)
X_w = fourier_transform(main_signal, t, w)

# plot the magnitude of the Fourier Transform
plt.figure(figsize=(10, 5))
plt.plot(w/(2*np.pi), np.abs(X_w))
plt.title('Fourier Transform of Sum of 3 Sin Waves')
plt.xlabel('w')
plt.ylabel('Magnitude of X(w)')
plt.grid()
```

و خروجی:



که می‌بینیم در فرکانس‌های ۵ و ۵۰ و ۱۲۰ و قرینه‌ی این‌ها، نمودار مقدار بسیار بیشتری رو داره که نشون می‌ده سیگنال اصلی هم از این سه فرکانس تشکیل شده. همچنین چگالی هر کدوم از مقادیر هم با مقدار دامنه‌ی اون فرکانس‌ها همخوانی داره. به عبارتی برای فرکانس ۵ که دامنه‌ی یک داره، داخل نمودار هم مقدار بیشتری می‌بینیم و برای فرکانس ۱۲۰ که دامنه‌ی ۰.۲ داره، مقدار کمتری می‌بینیم. فرکانس ۵۰ هم که بین این دو قرار داره.