CA1 - mathematical engineering

Mostafa Kermaninia

810101575

1.1

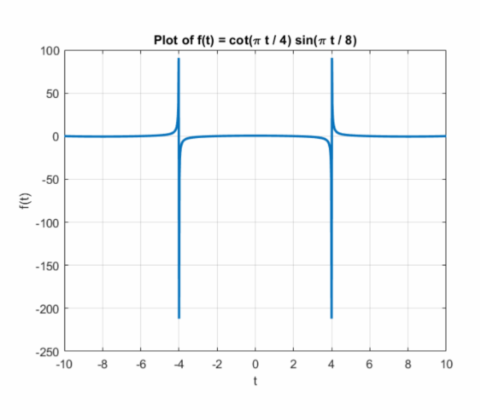
<u>1–تابع اول:</u>

$$\cot\left(\frac{\pi t}{4}\right)\sin\left(\frac{\pi t}{8}\right)$$

<u>❋ کد تابع به همراه توضیحات خط به خط در کامنت ها (Q11.mlx) :</u>

```
% Define the range of t from 0 to 10 with 1000 points for smooth plotting
t = linspace(-10, 10, 1000);

% Define the function f(t) = cot(pi * t / 4) * sin(pi * t / 8)
% Note: Using element-wise multiplication (.*) to apply the function to each element of t
f = cot(pi * t / 4) .* sin(pi * t / 8);

% Create a new figure window for the plot
figure;

% Plot the function f(t) versus t with a line width of 2 for better visibility
plot(t, f, 'LineWidth', 2);

% Label the x-axis as 't'
xlabel('t');

% Label the y-axis as 'f(t)'
ylabel('f(t)');

% Add a title to the plot
title('Plot of f(t) = cot(\pi t / 4) sin(\pi t / 8)');

% Turn on the grid for easier visualization of the plot
grid on;
```

❋ نمودار نهایی:



2–تابع دوم:

$$sgn\left(\frac{1}{t^2}\right)$$

❋ نکات:

تابع sgn(x) اینگونه است:

- 1 if $x > 0$
- 0 if $x = 0$
- -1 if $x < 0$

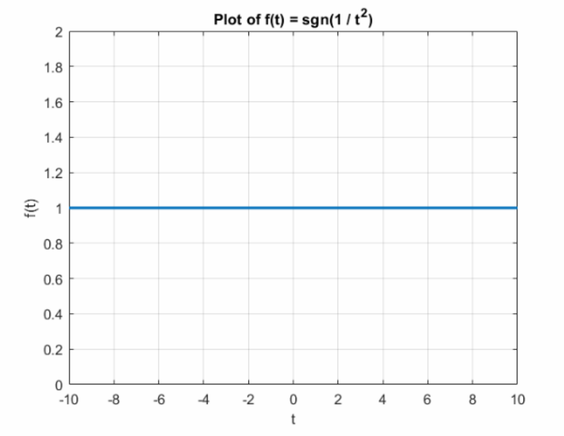
پس تابع sgn(1/t²) بخاطر اینکه به ازای تمام t های ناصفر، مقدار آرگومانش مثبت است، همواره مقدار 1 را برمیگرداند(به ازای تمام t های ناصفر)

پس داخل کد هم همانطور که در توضیحاتش نوشتم، مقدار t=0 را حذف کردم تا از بروز مشکل احتمالی جلوگیری شود.

* کد تابع به همراه توضیحات خط به خط در کامنت ها (Q12.mlx) :

```
% Define the range of t, avoiding t = 0
t = linspace(-10, 10, 1000);  % Generates 1000 points between -10 and 10
t(t == 0) = []; % Remove zero to avoid division by zero

% Define the function f(t) = sgn(1 / t^2)
f = sign(1 ./ t.^2);

% Create a new figure window for the plot
figure;

% Plot the function f(t) versus t with a line width of 2 for better visibility
plot(t, f, 'LineWidth', 2);
xlabel('t');
ylabel('f(t)');
title('Plot of f(t) = sgn(1 / t^2)');
grid on;
```

* نمودار نهایی:



3–تابع سوم:

$$\begin{cases} -1, \quad t < -3 \\ 3ramp(t), \quad -3 < t < 3 \\ e^{-2.5t}, \quad t > 3 \end{cases}$$

❋ نکات:

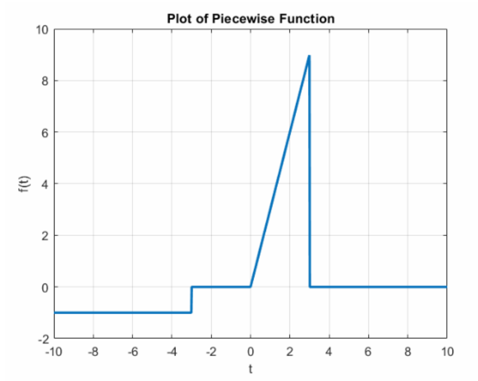
تابع رمپ را میتوان به این صورت براحتی در متلب نمایش داد(زیرا در **t**های منفی، صفر است و در **t**های نامنفی، خود مقدار **t** را میدهد)

$$\mathrm{ramp}(t) = \max(0, t)$$

❋ کد تابع به همراه توضیحات خط به خط در کامنت ها (Q13.mlx) :

```
% Define the range of t from -10 to 10 with 1000 points for smooth plotting
t = linspace(-10, 10, 1000);

% Initialize the function f(t) with zeros
f = zeros(size(t));

% Define the function using piecewise conditions
for i = 1:length(t)
    if t(i) < -3
        f(i) = -1;  % f(t) = -1 for t < -3
    elseif t(i) >= -3 && t(i) <= 3
        f(i) = 3 * max(0, t(i));  % f(t) = 3 * max(0, t) for -3 <= t <= 3
    else
        f(i) = exp(-2.5 * t(i));  % f(t) = exp(-2.5 * t) for t > 3
    end
end

% Create a new figure window for the plot
figure;

% Plot the function f(t) versus t with a line width of 2 for better visibility
plot(t, f, 'LineWidth', 2);
xlabel('t');  % Label the x-axis as 't'
ylabel('f(t)');  % Label the y-axis as 'f(t)'
ylim([-2, 10]);  % start y from -2 to see the plot better
title('Plot of Piecewise Function');  % Add a title to the plot
grid on;  % Turn on the grid for easier visualization
```

❋ نمودار نهایی:



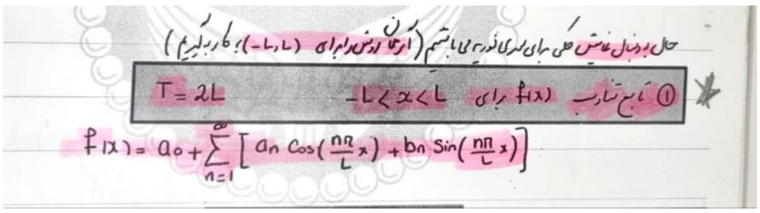
2.1

❋ نکات

توضیحاتی که در صورت سوال راجع به آرگومان ها داده شده بود، بنظرم ناواضح بودند، پس من فرض هایی که خودم در نوشتن کد کردم را مینویسم:

Num*: تعداد جملاتی از سری فوریه که محاسبه می شوند در حین اجرای کد.

P*: دوره تناوب تابع، همچنین فرض میکنم حالت تناوب تابع، از منفی L تا L است که یعنی این فرم از سر ی فوریه را پیاده سازی کردم:



حال به دنبال نمایش کلی برای سری فوریه می باشیم (اگر بخواهیم روش را برای (L,-L) به کار بگیریم)

① تابع متناوب f(x) برای $-L<x<L$ $T=2L$

$$f(x)=a_0+\sum_{n=1}^{\infty}\left[a_n\cos\left(\frac{n\pi}{L}x\right)+b_n\sin\left(\frac{n\pi}{L}x\right)\right]$$

پس یعنی p همان 2L است

❋a: توان چندجمله ای

Nshow*: تعداد جملاتی از سری فوریه که در ترمینال، چاپ خواهند شد و برای رسم نمودار هم استفاده می شوند(پس بدیهتا باید از Num کمتر باشد این عدد، چون نهایتا در طول کد به تعداد Num حساب کرده ایم)

* کد تابع به همراه توضیحات خط به خط در کامنت ها (Q21.m) :

```matlab
% I define the function `fourier` which computes and plots the Fourier series approximation
% of a given function up to a specified number of terms.
function fourier(Num, P, a, Nshow) % I explain arguments in report

    % I define the original function f(x) = x^a
    f = @(x) x.^a;
    % Then calculate the half period L
    L = P / 2;
coefficient
    % Now Compute the a0  (the average value of the function over one period)
    a0 = (1/P) * integral(@(x) f(x), -L, L);

    % These are arrays to store Fourier coefficients a_n and b_n
    a_n = zeros(1, Num);
    b_n = zeros(1, Num);

    % Now Calculate the Fourier coefficients a_n and b_n for n=1 to Num
    for n = 1:Num
        a_n(n) = (1/L) * integral(@(x) f(x) .* cos(pi * n * x / L), -L, L);
        b_n(n) = (1/L) * integral(@(x) f(x) .* sin(pi * n * x / L), -L, L);
    end

    % Create a range of x values for plotting
    x = linspace(-L, L, 1000);
    % It's the main part: At first I initialize the Fourier series approximation with the a0 term
    F_s = a0 * ones(size(x));
    % Then add the first Nshow terms of the Fourier series to the approximation
    for n = 1:Nshow
        F_s = F_s + a_n(n) * cos(pi * n * x / L) + b_n(n) * sin(pi * n * x / L);
    end
```

```matlab

    % Print the coefficients and the terms of the Fourier series
    fprintf('a0 is %f\n', a0);
    for n = 1:Nshow
        fprintf('%dth term of series is\n',n);
        fprintf('%f * cos(%d * pi * x / %d) + %f * sin(%d * pi * x / %d)\n', a_n(n), n, L, b_n(n), n, L);
    end


    % Plot the original function and the Fourier series approximation
    figure;
    fplot(@(x) f(x), [-L L], 'LineWidth', 1.5);
    hold on;
    plot(x, F_s, 'r', 'LineWidth', 1.5);
    legend('Actual Function', 'Fourier Series');
    xlabel('x');
    ylabel('Vertical Axis');
    title(['Fourier Series Approximation of x^', num2str(a), ' for ', num2str(Nshow), ' terms']);
    grid on;
    hold off;

end
```

* تست کردن کد:

با این مثال، درستی کد را بررسی میکنیم:

```
fourier(10, 8, 6, 7);
```

طبق توضیحاتم راجع به آرگومان ها، الان انتظار داریم که:

- تابع x^6 رسم شود (هم نمودار اصلی و هم نمودار تقریب زده با سری فوریه)

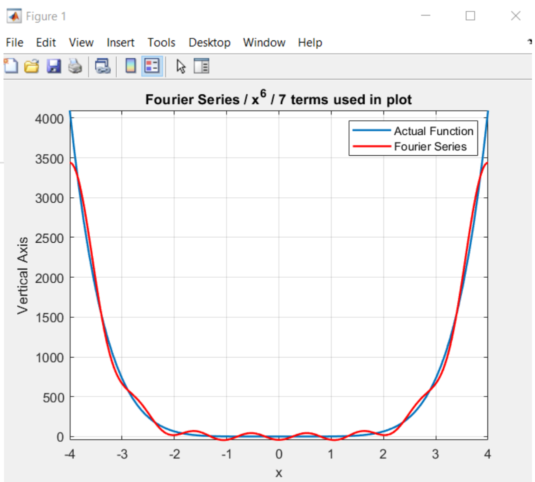
-در داخل کد، **10** جمله ی اول محاسبه شود

-دوره تناوب مساوی **8** باشد که یعنی تابع ما و معادل آن با سری فوریه بین **-4** و **+4** نمایش داده می شود

-از **7** جمله ی اول سری فوریه جهت کشیدن نمودار تقریبی استفاده شده و همان **7** جمله پرینت هم بشوند(دقت کنید چون تابع x^6 زوج است، مقدار تمام ضرایب سینوسی هم صفر است)

نمودار و ترمینال بعد از ران کردن تست:

```
>> Q21
a0 is 585.142857
1th term of series is
-1023.389631 * cos(1 * pi * x / 4) + -0.000000 * sin(1 * pi * x / 4)
2th term of series is
710.154036 * cos(2 * pi * x / 4) + -0.000000 * sin(2 * pi * x / 4)
3th term of series is
-437.173560 * cos(3 * pi * x / 4) + 0.000000 * sin(3 * pi * x / 4)
4th term of series is
273.335137 * cos(4 * pi * x / 4) + -0.000000 * sin(4 * pi * x / 4)
5th term of series is
-183.451207 * cos(5 * pi * x / 4) + 0.000000 * sin(5 * pi * x / 4)
6th term of series is
130.681749 * cos(6 * pi * x / 4) + 0.000000 * sin(6 * pi * x / 4)
7th term of series is
-97.484439 * cos(7 * pi * x / 4) + -0.000000 * sin(7 * pi * x / 4)
```

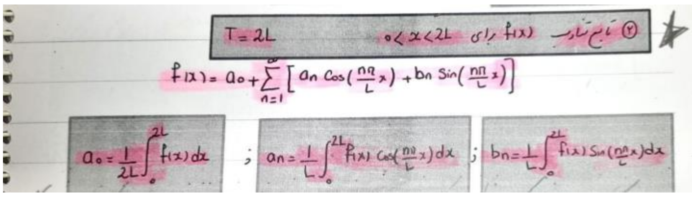


همانطور که مشخص است، همه چیز با پیشبینی ما مطابق بود. و تابع تخمین زده شده با سری فوریه نیز تا حد خوبی نزدیک تابع اصلی کشیده شده است.

2.2

✽ نکات

تغییراتی در تابع بخش قبل بوجود آوردم که شرح میدهم: چون تابع ln دامنه اش فقط شامل اعداد مثبت است، دیگر از فرم فوریه ی L- تا L+ نمیتوان استفاده کرد، پس فرم 0 تا 2L را پیاده سازی می کنیم:



پس کافیست جاهایی از کد که بازه ی انتگرال را L-تا L+ گذاشته بودیم، به 0 تا 2L تغییر بدهیم.

نهایتا هم آرگومان های تابع جدید اینها خواهد بود:

Num, P, Nshow به همان معنی قبلی

Beta, Alph : ثوابت تابعی که قرار است آنرا تقریب بزنیم با سری فوریه:

$$f(x) = x^{\beta}\ln(\alpha x)$$

* کد تابع به همراه توضیحات خط به خط در کامنت ها (Q22.m) :

```matlab

% I change the function `fourier` and build function `fourier_ln`
function fourier_ln(Num, P, alpha, Beta, Nshow) % I explain arguments in report

    % I define the original function f(x) = x^(bate)*ln(alpha*x)
    f = @(x) (x.^Beta).* log(alpha .* x);

    % Then calculate the half period L
    L = P / 2;

    % Now Compute the a0 (the average value of the function over one period)
    a0 = (1/P) * integral(@(x) f(x), 0, P);

    % These are arrays to store Fourier coefficients a_n and b_n
    a_n = zeros(1, Num);
    b_n = zeros(1, Num);

    % Now Calculate the Fourier coefficients a_n and b_n for n=1 to Num
    for n = 1:Num
        a_n(n) = (1/L) * integral(@(x) f(x) .* cos(pi * n * x / L), 0, P);
        b_n(n) = (1/L) * integral(@(x) f(x) .* sin(pi * n * x / L), 0, P);
    end

    % Create a range of x values for plotting
    x = linspace(0, P, 1000);
    % It's the main part: At first I initialize the Fourier series approximation with the a0 term
    F_s = a0 * ones(size(x));
    % Then add the first Nshow terms of the Fourier series to the approximation
    for n = 1:Nshow
        F_s = F_s + a_n(n) * cos(pi * n * x / L) + b_n(n) * sin(pi * n * x / L);
    end

    % Print the coefficients and the terms of the Fourier series
    fprintf('a0 is %f\n', a0);
    for n = 1:Nshow
        fprintf('%dth term of series is\n',n);
        fprintf('%f * cos(%d * pi * x / %d) + %f * sin(%d * pi * x / %d)\n', a_n(n), n, L, b_n(n), n, L);
    end


    % Plot the original function and the Fourier series approximation
    figure;
    fplot(@(x) f(x), [0 P], 'LineWidth', 1.5);
    hold on;
    plot(x, F_s, 'r', 'LineWidth', 1.5);
    legend('Actual Function', 'Fourier Series');
    xlabel('x');
    ylabel('Vertical Axis');
    title(['Fourier Series / x^', num2str(Beta),'ln(', num2str(alpha) ,'x) / ', num2str(Nshow), ' terms used in plot']);
    grid on;
    hold off;

end

```

* تست کردن کد:

طبق صورت سوال، با Num = 10 و Nshow = 5 تست میکنیم تابع را، مقدار الفا و بتا را هم به ترتیب مساوی 2 و 4 میگذاریم:

```
fourier_ln(10, 8, 2, 4, 5);
```

پس انتظار داریم که:

- تابع $x^4\ln(2x)$ رسم شود(هم نمودار اصلی و هم نمودار تقریب زده با سری فوریه)

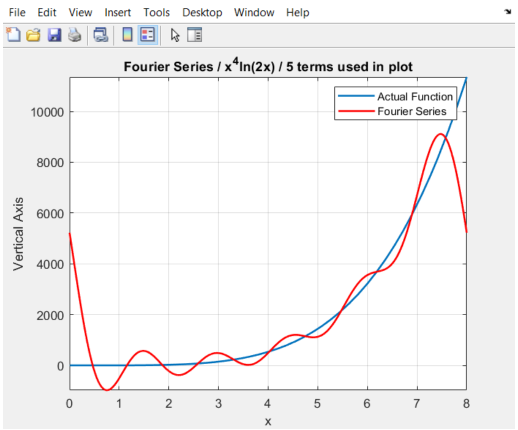
- در داخل کد، ضرایب سری فوریه تا 10 جمله ی اول محاسبه شوند

-دوره تناوب مساوی 8 باشد که یعنی تابع ما و معادل آن با سری فوریه بین 0 و +8 نمایش داده می شود.

-از 5 جمله ی اول سری فوریه جهت کشیدن نمودار تقریبی استفاده شده و همان 5 جمله پرینت هم بشوند

* نمودار و ترمینال بعد از ران کردن تست:

```
>> Q22
a0 is 2107.464681
1th term of series is
1993.927785 * cos(1 * pi * x / 4) + -2333.859865 * sin(1 * pi * x / 4)
2th term of series is
595.867298 * cos(2 * pi * x / 4) + -1643.153952 * sin(2 * pi * x / 4)
3th term of series is
272.627031 * cos(3 * pi * x / 4) + -1155.996765 * sin(3 * pi * x / 4)
4th term of series is
154.871069 * cos(4 * pi * x / 4) + -883.017123 * sin(4 * pi * x / 4)
5th term of series is
99.564586 * cos(5 * pi * x / 4) + -712.364798 * sin(5 * pi * x / 4)
fx >>
```



همانطور که مشخص است، همه چیز با پیشبینی ما مطابق بود. و تابع تخمین زده شده با سری فوریه نیز تا حد خوبی نزدیک تابع اصلی است.

2.3

* نکات

دقیقا از همان کد Q21 استفاده میکنیم، تنها چیزیکه لازم است دقت کنیم است که حین تست کردن، a=2 بگذاریم تا توان x مساوی با 2 باشد.(بخش پرینت کردن در ترمینال را هم میشد حذف کنیم چون خیلی نیازی به آن نداریم)

* کد تابع به همراه توضیحات خط به خط در کامنت ها (Q23.m) :

```
% I define the function `fourier` which computes and plots the Fourier series approximation
% of a given function up to a specified number of terms.
function fourier(Num, P, a, Nshow) % I explain arguments in report

    % I define the original function f(x) = x^a
    f = @(x) x.^a;
    % Then calculate the half period L
    L = P / 2;

    % Now Compute the a0 (the average value of the function over one period)
    a0 = (1/P) * integral(@(x) f(x), -L, L);

    % These are arrays to store Fourier coefficients a_n and b_n
    a_n = zeros(1, Num);
    b_n = zeros(1, Num);

    % Now Calculate the Fourier coefficients a_n and b_n for n=1 to Num
    for n = 1:Num
        a_n(n) = (1/L) * integral(@(x) f(x) .* cos(pi * n * x / L), -L, L);
        b_n(n) = (1/L) * integral(@(x) f(x) .* sin(pi * n * x / L), -L, L);
    end

    % Create a range of x values for plotting
    x = linspace(-L, L, 1000);
    % It's the main part: At first I initialize the Fourier series approximation with the a0 term
    F_s = a0 * ones(size(x));
    % Then add the first Nshow terms of the Fourier series to the approximation
    for n = 1:Nshow
        F_s = F_s + a_n(n) * cos(pi * n * x / L) + b_n(n) * sin(pi * n * x / L);
    end
```

```

    % Print the coefficients and the terms of the Fourier series
    fprintf('a0 is %f\n', a0);
    for n = 1:Nshow
        fprintf('%dth term of series is\n',n);
        fprintf('%f * cos(%d * pi * x / %d) + %f * sin(%d * pi * x / %d)\n', a_n(n), n, L, b_n(n), n, L);
    end


    % Plot the original function and the Fourier series approximation
    figure;
    fplot(@(x) f(x), [-L L], 'LineWidth', 3.5);
    hold on;
    plot(x, F_s, 'r', 'LineWidth', 1.5);
    legend('Actual Function', 'Fourier Series');
    xlabel('x');
    ylabel('Vertical Axis');
    title(['Fourier Series / x^', num2str(a), ' / ', num2str(Nshow), ' terms used in plot']);
    grid on;
    hold off;

end
```

❋ تست کردن کد:

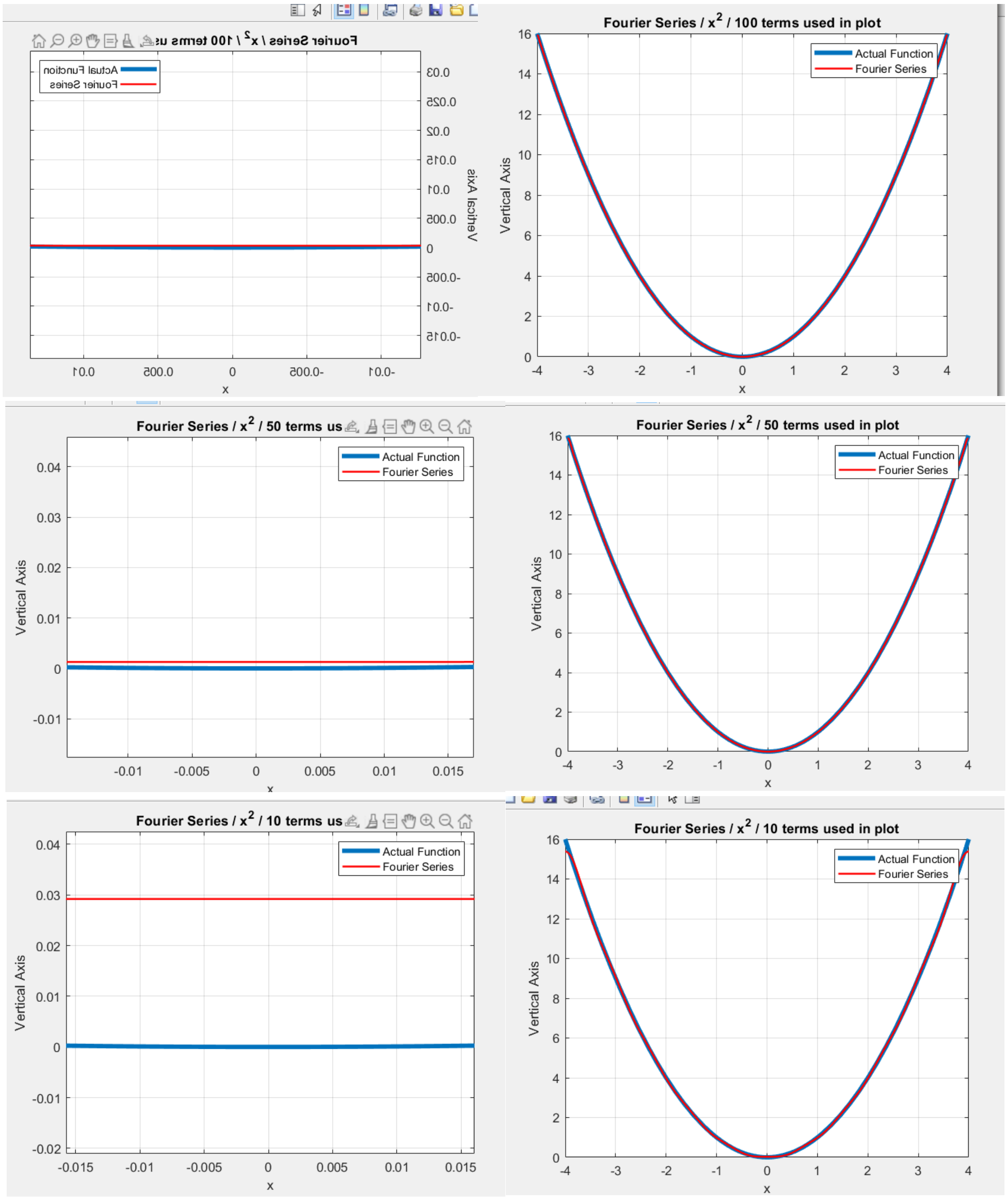
طبق خواسته ی سوال، الفا را **2** گذاشته و به ازای **num** های مختلف امتحان میکنیم کشیدن تابع را(البته چون نموداری که رسم میشود، در اصل براساس **Nshow** رسم میشود، من مقدار **Nshow** را هم مساوی همان مقدار **num** گذاشتم که به هدف سوال که بررسی نمودار تقریبی این تابع با تعداد جملات متفاوت بود، برسیم). مقدار دوره تناوب یا **P** را هم بطور دلخواه همان **8** گذاشتم که نمودار ها از **-4** تا **+4** رسم شوند:

Q23.m Q22.m Q21.m

```
fourier(10, 8, 2, 10);
fourier(50, 8, 2, 50);
fourier(100, 8, 2, 100);
```

❋ نمودار ها بعد از ران کردن تست:

از آنجایی که به ازای **Nshow=50**و **Nshow=100** نمودار تقریبی بشدت نزدیک به نمودار حقیقی میشود، ضخامت نمودار اصلی را بیشتر کردم و بعد از عکس کلی ای که برای هر بخش گذاشتم، یک عکس با زوم زیاد که توسط آن اختلاف دو نمودار، مشخص است هم گذاشتم، تا ببینیم با زوم بیشتر دقیقا کدام نمودار بهتر عمل میکند.



همانطور که واضح است، هر چه تعداد جملات سری فوریه بیشتر میشود، دقت نمودار بیشتر شده و نمودار تقریبی به نمودار واقعی و دقیق میل میکند.

2.4

* حل تئوری

$$f_{(x)} = x^2 \qquad -\pi < x < \pi \longrightarrow L = \pi \longrightarrow$$ دوره تناوب $-L$ تا L است

$$\Rightarrow f^*_{(x)} = a_0 + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)$$

$$a_0 = \frac{1}{2L}\int_{-L}^{L} f_{(x)}\,dx \qquad a_n = \frac{1}{L}\int_{-L}^{L} f_{(x)} \cos\left(\frac{n\pi x}{L}\right) dx \qquad b_n = \frac{1}{L}\int_{-L}^{L} f_{(x)} \sin\left(\frac{n\pi x}{L}\right) dx$$

$$\longrightarrow a_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} x^2\,dx = \frac{1}{2\pi}\,\frac{x^3}{3}\Big]_{-\pi}^{\pi} = \frac{\pi^3}{3\pi} = \frac{\pi^2}{3}$$

$$\longrightarrow a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} \underset{u}{x^2}\,\underbrace{\cos(nx)\,dx}_{dv \rightarrow v = \frac{1}{n}\sin(nx)} = \frac{1}{\pi}\left[\frac{x^2}{n}\sin(nx)\Big]_{-\pi}^{\pi} - \int_{-\pi}^{\pi} \frac{2}{n}\underset{u}{x}\,\underbrace{\sin(nx)\,dx}_{dv}\right]$$

$$u \rightarrow du = 2x\,dx \qquad du = dx \qquad v = -\frac{1}{n}\cos(nx)$$

$$= \frac{1}{\pi}\left[\frac{2\pi^2}{n}\sin(n\pi) - \frac{2}{n}\left(-\frac{x}{n}\cos nx\Big]_{-\pi}^{\pi} + \frac{1}{n}\int_{-\pi}^{\pi}\cos(nx)\,dx\right)\right] \qquad \frac{1}{n}\sin(nx)\Big]_{-\pi}^{\pi}$$

$$= \frac{1}{\pi}\left[\frac{2\pi^2}{n}\cancel{\sin(n\pi)} - \frac{2}{n}\left(-\frac{2\pi}{n}\cos n\pi + \frac{2}{n^2}\cancel{\sin(n\pi)}\right)\right] \qquad n \in \mathbb{N} \Rightarrow \sin(n\pi) = 0$$

$$= \frac{1}{\pi}\left(-\frac{2}{n}\right)\left(-\frac{2\pi}{n}\right)\cos n\pi = \frac{4}{n^2}\cos(n\pi) \overset{n\in\mathbb{N}}{=} \frac{4}{n^2}(-1)^n$$

$$\longrightarrow b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} \underbrace{\overbrace{x^2}^{\text{زوج}}\,\overbrace{\sin(nx)}^{\text{فرد}}}_{\text{فرد}}\,dx = \frac{1}{\pi}\int_{-\pi}^{\pi} (\text{تابع فرد})\,dx = 0$$

$$\Rightarrow \boxed{f^*_{(x)} = \frac{\pi^2}{3} + \sum_{n=1}^{\infty} \frac{4(-1)^n}{n^2}\cos(nx)}$$

$$-\pi < x < \pi$$

حالا برای پاسخ $\sum \frac{1}{n^2}$ که خواسته ی سوال است، باید دو $(-1)^n$ را حذف کرده و به 1 تبدیل کنیم، پس $x=\pi$

$$f^*_{(x)} = \frac{\pi^2}{3} + \sum_{n=1}^{\infty} \frac{4(-1)^n}{n^2}(-1)^n = \frac{\pi^2}{3} + \sum_{1}^{\infty} \frac{4}{n^2} = x^2\Big]_{x=\pi} = \pi^2$$

$$\rightarrow \frac{\pi^2 - \frac{\pi^2}{3}}{4} = \sum_{1}^{\infty} \frac{1}{n^2} \rightsquigarrow \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{2\pi^2}{3 \times 4} = \boxed{\frac{\pi^2}{6}}$$

*نکات کد:

تابع بخش اول را مقداری تغییر دادم و اکنون به ترتیب این کارها را میکند:

- اولا که فقط سه آرگومان میگیرد: p, a, num که قبلا توضیحشان داده ام.

-در وهله ی اول، مقدار واقعی و دقیق پاسخ سری را که همان $\pi^2/6$ است را پرینت میکنم که بتوانیم دقت نتایج بعدی را با آن بسنجیم

- سپس توسط یک حلقه، مقدار سری عددی ای که قرار است حساب کنیم را تا جمله ی *Num*ام آن حساب کرده و آنرا هم پرینت میکنم(اگر عملیات ما روی سری فوریه درست باشد، بعد از محاسبات روی حاصل سریس فوریه دقیقا باید به همین عدد برسیم)

-سپس بقیه ی کد همانن بخش اول است و ضرایب سری فوریه ی مرتبط با تابع X^a محاسبه می شود اما در حلقه ای که برای محاسبه ی جملات سری فوریه داشتیم، بجای *X* ، مقدار π گذاشتم چون طبق محاسباتمان، نهایتا توسط سری فوریه در این نقطه است که به سری عددی مدنظرمان میرسیم.

-نهایتا حاصل سری فوریه در نقطه ی π را پرینت کرده و بعد از آن هم با انجام عملیات زیر، به پاسخ تقریبی ای که برای سیگما میخواستیم، میرسیم و آنرا پرینت میکنیم:

$$f^*_{(\pi)} = \text{سری فوریه در نقطه‌ی } \pi \simeq x^{۲}\Big]_{x=\pi} = \pi^{۲}$$

$$\Rightarrow \sum_{n=1}^{\infty} \frac{1}{n^{۲}} = \frac{\pi^{۲} - \frac{\pi^{۲}}{۳}}{۴} \simeq \frac{f^*_{(\pi)} - \frac{\pi^{۲}}{۳}}{۴}$$

❋ کد تابع به همراه توضیحات خط به خط در کامنت ها (Q24.m) :

```matlab

function fourier_sigma(Num, P, a) % I explain arguments in report
    % Print the real value of sigma result
    fprintf('Actual value of sigma 1/n^2 (pi^2/6) is : %f\n', pi^2/6);

    % Calculating the value of sigma 1/n^2
    result = 0;
    for n = 1:Num
        result = result + 1/n^2;
    end
    % I print the real result and sigma result
    fprintf('Value of sigma 1/n^2 with first %d terms is : %f\n',Num, result);

    % I define the original function f(x) = x^a
    f = @(x) x.^a;
    % Then calculate the half period L
    L = P / 2;

    % Now Compute the a0 (the average value of the function over one period)
    a0 = (1/P) * integral(@(x) f(x), -L, L);

    % These are arrays to store Fourier coefficients a_n and b_n
    a_n = zeros(1, Num);
    b_n = zeros(1, Num);

    % Now Calculate the Fourier coefficients a_n and b_n for n=1 to Num
    for n = 1:Num
        a_n(n) = (1/L) * integral(@(x) f(x) .* cos(pi * n * x / L), -L, L);
        b_n(n) = (1/L) * integral(@(x) f(x) .* sin(pi * n * x / L), -L, L);
    end
```

```matlab

    % Create a range of x values for plotting
    x = linspace(-L, L, 1000);
    % It's the main part: At first I initialize the Fourier series approximation with the a0 term
    F_sigma = a0;
    % Then add the first Nshow terms of the Fourier series to the
    % approximation (now we want to calculate f in x=pi , so we put pi
    % instead of x in Q21 code
    for n = 1:Num
        F_sigma = F_sigma + a_n(n) * cos(pi * n * pi / L) + b_n(n) * sin(pi * n * pi / L);
    end

    % Finaly print result of sourier series
    fprintf('Fourier series in x = pi : %f\n', F_sigma);
    fprintf('So sigma 1/n^2 according to fouries series is : %f\n', (F_sigma - pi^2/3)/4);


end


```

❋ تست کردن کد:

```
1 -    fourier_sigma(30, 2*pi, 2);
```

به این وسیله، بازه ی تناوب را همانند صورت سوال، منفی پی تا پی گذاشتم و همچنین 30 جمله ی اول سری فوریه یا بعبارتی 30 جمله ی اول سیگما را حساب کرده و نتایج را پرینت میکنم.

❋ ترمینال بعد از ران کردن تست:

```
ommand Window
>> Q24
Actual value of sigma 1/n^2 (pi^2/6) is : 1.644934
Value of sigma 1/n^2 with first 30 terms is : 1.612150
Fourier series in x = pi : 9.738469
So sigma 1/n^2 according to fouries series is : 1.612150
>>
```

یعنی مقداری که با میل کردن Num یا تعداد جملات محاسبه شده به بینهایت، باید به آن برسیم، حدود 1.64 است اما عددی که با محاسبه ی 30 جمله ی اول سیگما و همچنین عددی که با استفاده از محاسبه ی 30 جمله ی اول در سری فوریه به آن رسیدیم، حدود 1.61 است که مقدار بسیار نزدیکی به مقدار واقعی است و این یعنی محاسبات کد ما درستند.

(عدد 9.7 هم که پرینت شده، معادل تقریبی ای برای تابع f در نقطه ی π است که آن هم تا حد خوبی دقیق است و مقدار π2 را با تقریب خوبی بیان میکند.چون مقدار واقعی π^2 حدود9.8 است)

2.5

❋ نکات

روند کار این کد اینگونه است :

– اولا سه آرگومان دریافت میکند: لیست مقادیر x، لیست مقادیر f(x) و تعداد هارمونیک هایی که قرار است مدنظر قرار بگیرند .

–در ابتدا، براحتی مقدار A0 بکمک تابع sum و صدا زدن آن روی بردار f(x) حساب میشود.

-سپس توسط دو حلقه ی تو در تو، ابتدا برای هر کدام از ضرایب، سیگمای مربوط به آن ضریب را حساب میکند، یعنی این ها:

$$A_n = \frac{2\sum f(x)\cos(nx)}{n}, \quad B_n = \frac{2\sum f(x)\sin(nx)}{n}$$

-و سپس توسط حلقه ی بیرونی تمام ضرایب محاسبه شده در بردار **An** و **Bn** ذخیره میشوند(به ازای همان تعداد از هارمونیک هایی که تعیین کرده ایم)

-حالا تمام ضرایب حساب شده را لیست کرده و در ترمینال چاپ میکنیم

-در نهایت مقدار تخمین زده شده برای **f** را در **f_approx** ریخته و بکمک **matlabFunction** آنرا تبدیل به تابعی میکنیم که قابل رسم کردن باشد، سپس تابع تخمین زده شده توسط هارمونیک های تعیین شده را رسم کرده و تمام نقاط اولیه ی ورودی را هم رسم میکنیم

✻ کد تابع به همراه توضیحات خط به خط در کامنت ها (Q25.m) :

```matlab
% Example data
x = [0, pi/3, 2*pi/3, pi, 4*pi/3, 5*pi/3, 2*pi];
f_x = [1, 1.4, 1.9, 1.7, 1.5, 1.2, 1];

% Number of harmonics
num_harmonics = 4;

% Call the function
fourier_series_approx(x, f_x, num_harmonics);

function fourier_series_approx(x, f_x, num_harmonics)
    % Number of sample points
    N = length(x);

    % Initialize Fourier coefficients
    A0 = (2/N) * sum(f_x);
    An = zeros(1, num_harmonics);
    Bn = zeros(1, num_harmonics);

    % Calculate An and Bn for n = 1 to num_harmonics
    for n = 1:num_harmonics
        cos_sigma = 0;
        sin_sigma = 0;
        for i = 1: N
            cos_sigma = cos_sigma + (f_x(i) * cos(n * x(i)));
            sin_sigma = sin_sigma + (f_x(i) * sin(n * x(i)));
        end
        An(n) = (2 * cos_sigma) / n;
        Bn(n) = (2 * sin_sigma) / n;
    end
```

```matlab

    % print fourier seri, term by term to selected harmonic
    fprintf('Fourier seri, from first Harmonic to %dth Harmonic: \n', num_harmonics);
    fprintf('f(x) = %f', A0 / 2);
    for n = 1:num_harmonics
        fprintf(' + (%f)cos(%dx) + (%f)sin(%dx) ', An(n), n, Bn(n), n);
    end
    fprintf('\n')

    % Construct the Fourier series approximation
    syms X;
    f_approx = A0 / 2;
    for n = 1:num_harmonics
        f_approx = f_approx + An(n) * cos(n*X) + Bn(n) * sin(n*X);
    end

    % Create a dense grid for plotting the approximation
    x_dense = linspace(x(1) - 2   , x(end) + 2, 1000);
    f_approx_func = matlabFunction(f_approx, 'vars', X);
    f_approx_values = f_approx_func(x_dense);

    % Plot the original points and Fourier series approximation
    figure;
    plot(x, f_x, 'ro', 'MarkerFaceColor', 'r', 'DisplayName', 'Data points');
    hold on;
    plot(x_dense, f_approx_values, 'b-', 'LineWidth', 1.5, 'DisplayName', 'Fourier series approximation');
    legend show;
    xlabel('x');
    ylabel('f(x)');
    title(['Fourier Series Approximation with ', num2str(num_harmonics), ' Harmonics']);
    grid on;
    hold off;
end
```

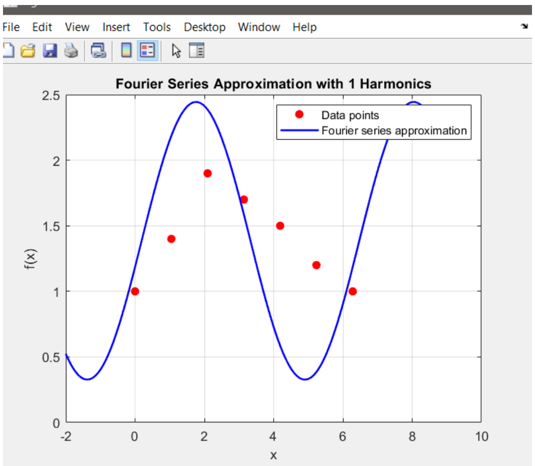
* تست کردن کد:

طبق صورت مسئله، این تابع را برای 1 الی 4 هارمونیک روی تابع داده شده تست میکنیم:

```
% Example data
x = [0, pi/3, 2*pi/3, pi, 4*pi/3, 5*pi/3, 2*pi];
f_x = [1, 1.4, 1.9, 1.7, 1.5, 1.2, 1];

% Call the function
fourier_series_approx(x, f_x, 1);
fourier_series_approx(x, f_x, 2);
fourier_series_approx(x, f_x, 3);
fourier_series_approx(x, f_x, 4);
```

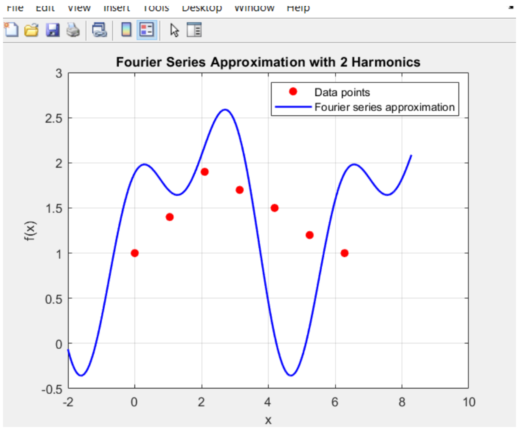
نمودار و ترمینال بعد از ران کردن تست:

هارمونیک اول:



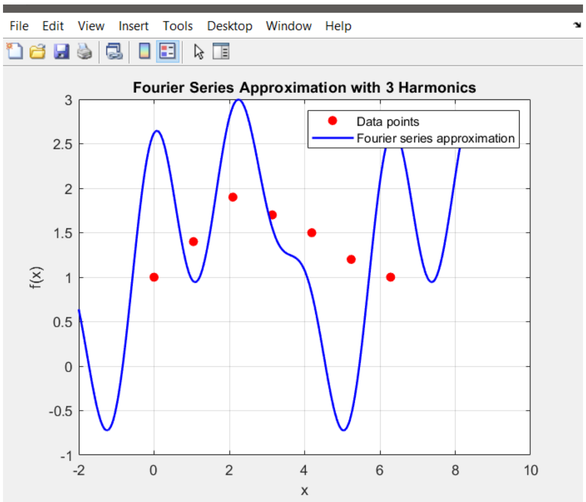

```
>> Q25
Fourier seri, from first Harmonic to 1th Harmonic:
f(x) = 1.385714 + (-0.200000)cos(1x) + (1.039230)sin(1x)
```

هارمونیک اول و دوم:



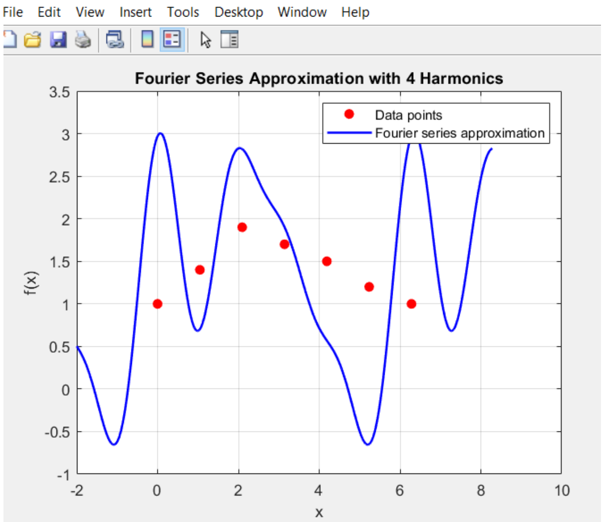

```
Fourier seri, from first Harmonic to 2th Harmonic:
f(x)  = 1.385714 + (-0.200000)cos(1x) + (1.039230)sin(1x)   + (0.700000)cos(2x) + (-0.173205)sin(2x)
```

هارمونیک اول و دوم و سوم:




```
Fourier seri, from first Harmonic to 3th Harmonic:
f(x)  = 1.385714 + (-0.200000)cos(1x) + (1.039230)sin(1x)   + (0.700000)cos(2x) + (-0.173205)sin(2x)   + (0.733333)cos(3x) + (-0.000000)sin(3x)
```

هارمونیک اول و دوم و سوم و چهارم:




```
Fourier seri, from first Harmonic to 4th Harmonic:
f(x) = 1.385714 + (-0.200000)cos(1x) + (1.039230)sin(1x)  + (0.700000)cos(2x) + (-0.173205)sin(2x)  + (0.733333)cos(3x) + (-0.000000)sin(3x)  + (0.350000)cos(4x) + (0.086603)sin(4x)
>>
```

همانطور که مشاهده شد، خروجی ترمینال نیز دقیقا همان چیزی شد که در صورت سوال گفته شده بود(بدلیل کوچک بودن اعداد در عکس آخر، خروجی را کپی پیست هم میکنم)

Fourier seri, from first Harmonic to 4th Harmonic :

f(x) = 1.385714 + (-0.200000)cos(1x) + (1.039230)sin(1x) + (0.700000)cos(2x) + (-0.173205)sin(2x) + (0.733333)cos(3x) + (-0.000000)sin(3x) + (0.350000)cos(4x) + (0.086603)sin(4x)

3.2

<u>* نکات:</u>

fft یا تبدیل فوریه سریع، یک الگوریتم سریع برای محاسبه تبدیل فوریه گسسته (DFT) و معکوس آن است. تبدیل فوریه گسسته سیگنال ورودی را به دامنه فرکانسی تبدیل می‌کند که در آن هر فرکانس دارای یک دامنه و فاز خاص است.

کاربرد fft:

– تحلیل فرکانسی :برای تحلیل فرکانس‌های موجود در یک سیگنال زمانی استفاده می‌شود.

- پردازش سیگنال :در پردازش سیگنال‌های دیجیتال مانند فیلتر کردن، فشرده‌سازی و غیره استفاده می‌شود.

- تحلیل صوت و تصویر :در تحلیل سیگنال‌های صوتی و تصویری کاربرد دارد.

fftshift (Fast Fourier Transform Shift) :

fftshiftبرای جابجایی داده‌های DFT استفاده می‌شود. این دستور داده‌های فرکانس صفر (DC) را به مرکز آرایه منتقل می‌کند. به عبارت دیگر، اگر خروجی fftشامل فرکانس‌های منفی و مثبت باشد، fftshiftاین فرکانس‌ها را به ترتیب درست مرتب می‌کند.

<u>دلیل استفاده از fftshift :</u>

- مرکز قرار دادن فرکانس صفر :در نتیجه محاسبات fft، فرکانس صفر (DC) در ابتدا یا انتهای بردار قرار می‌گیرد. برای نمایش صحیح طیف فرکانسی، لازم است که فرکانس صفر در مرکز باشد.

- نمایش بصری بهتر :برای تحلیل و نمایش بصری بهتر طیف فرکانسی، جابجایی فرکانس‌ها به کمک fftshiftضروری است.

روند اجرای کارها در کد به این صورت است:

-اولا تابع ما 3 آرگومان میگیرد که اولی همان تابعی است که قرار است آنالیزش کنیم، دومی فرکانسی است که با آن فرکانس عملیات نمونه برداری را انجام میدهیم، و سومی یک بردار با دو عنصر است که بازه ی زمانی ای که قرار است تابع را در آن ارزیابی کنیم نشان می دهد.

-در ابتدا یک وکتور میسازیم که در آن بازه ی زمانی ای که داده ایم، به بخش هایی با اندازه ی 1/f تقسیم می شود که f هم فرکانسی است که تعیین کرده بودیم.

- سپس تابع اصلی را میکشیم

-حالا توسط تابع fft مقدار Fast Fourier Transform را برای تابعمان حساب میکنیم و توسط تابع fftshift ، بخش فرکانس صفر را به مرکز طیف می اوریم و وکتور فرکانس را هم می سازیم

-حالا تابع تبدیل فوریه را رسم میکنیم.

* کد تابع به همراه توضیحات خط به خط در کامنت ها (Q32.m) :

```matlab
% Function to analyze and plot the Fourier transform of a given function
function Fourier_Transform(f, freq, time)
    % Generate time vector from the given range with specified sampling frequency
    t = time(1):1/freq:time(2);

    % Evaluate the function handle at the time points
    x = f(t);

    % Plot the original function
    figure;  % Create a new figure
    plot(t, x);  % Plot the function
    title('Original Function');  % Set the title of the plot
    xlabel('Time');  % Label the x-axis
    ylabel('y');  % Label the y-axis

    % Compute the Fourier transform of the function
    X = fft(x);  % Compute the fast Fourier transform
    X_shifted = fftshift(X);  % Shift zero frequency component to the center
    f = (-length(X)/2:length(X)/2-1)*(freq/length(X));  % Frequency vector for plotting

    % Plot the Fourier transform
    figure;  % Create a new figure
    plot(f, abs(X_shifted));  % Plot the magnitude of the Fourier transform
    title('Fourier Transform');  % Set the title of the plot
    xlabel('Frequency (Hz)');  % Label the x-axis
    ylabel('Magnitude');  % Label the y-axis
end
```

* تست کردن کد:

-چون گفته شده تابع کسینوس را در دو دوره تناوب رسم کنیم، اولا میدانیم دوره تناوب تابع $\cos(\pi t)$ مساوی با 2 است که یعنی برای رسم دو دوره تناوب آن، میتوانیم آنرا از 2- تا 2+ وارد کنیم. پس از آنجا که دوتابع دیگری که برای تست کردن معرفی شده اند متناوب نیستند، آنها را هم بین 2- تا 2+ بررسی میکنیم

-از طرفی طبق صورت سوال، مقدار فرکانس را مساوی 1000 میگذاریم

-برای وارد کردن تابع کسینوس که مشکلی نداریم، تابع ثابت 1 را هم باساخت یک وکتور با مقدار 1 در تمام نقاط هندل میکنیم، اما برای پیاده سازی تابع دلتا، مجبوریم از معادل تقریبی آن که دلتای کرونکر است استفاده کنیم که در t=0 مقدار 1 دارد و در بقیه ی نقاط، مقدار 0 دارد.

```matlab
% Define the first test case: cosine function
Fourier_Transform(@(t) cos(pi * t), 1000, [-2 2]);

% Define the second test case: constant function (all ones)
Fourier_Transform(@(t) ones(size(t)), 1000, [-5 5]);

% Define the third test case: delta function (approximated by a Kronecker delta)
Fourier_Transform(@(t) (t==0), 1000, [-5 5]);
```

*محاسبات تئوری برای تست ها:

برای کسینوس:

$$\mathcal{F}(\cos(\pi t)) = \int_{-\infty}^{\infty} \cos(\pi t)\, e^{-i\omega t}\, dt = \int_{-\infty}^{\infty} \frac{e^{i\pi t} + e^{-i\pi t}}{2}\, e^{-i\omega t}\, dt$$

$$= \frac{1}{2}\left[\int_{-\infty}^{\infty} e^{i\pi t} e^{-i\omega t}\, dt + \int_{-\infty}^{\infty} e^{-i\pi t} e^{-i\omega t}\, dt\right] = \frac{1}{2}\left[\int_{-\infty}^{\infty} e^{-i(\omega-\pi)t}\, dt + \int_{-\infty}^{\infty} e^{-i(\omega+\pi)t}\, dt\right]$$

$$\xlongequal{\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-i\omega t}\, dt} \frac{1}{2}\left[2\pi\,\delta(\omega-\pi) + 2\pi\,\delta(\omega+\pi)\right] = \pi\left(\delta(\omega-\pi) + \delta(\omega+\pi)\right)$$

پس انتظار ۲ تابع دلتا در $-\pi$ و π (فرکانس زاویه‌ای) و یا $\frac{-\pi}{2\pi}$ و $\frac{+\pi}{2\pi}$ (فرکانس) داریم $\Leftarrow$ در $f = \pm 0/5$ ، مقدار ∞ و در بقیه جاها مقدار ۰ است.

برای تابع ثابت:

$$\mathcal{F}(1) = \int_{-\infty}^{\infty} e^{-i\omega t}\, dt \xlongequal{\delta(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-i\omega t}\, dt} 2\pi\,\delta(\omega)$$

پس انتظار داریم یک مقدار ∞ در نقطه‌ی $f=0$ داشته باشیم.

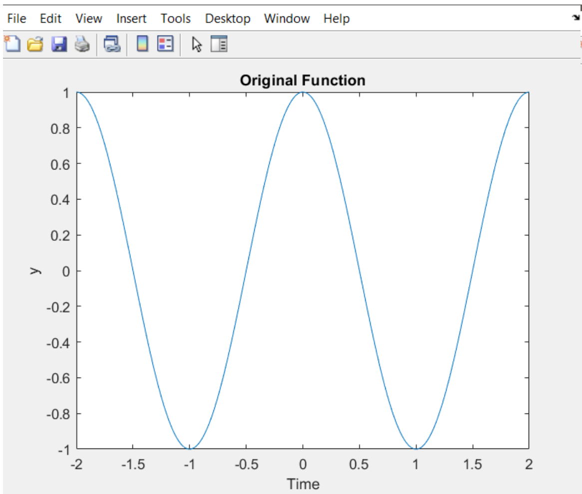
برای تابع دلتا:

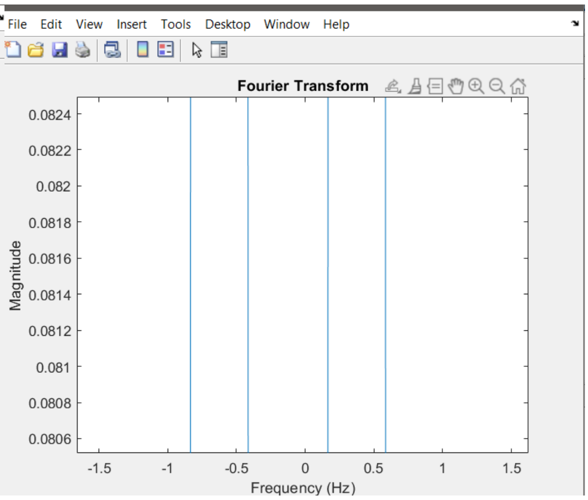
$$\mathcal{F}(\delta(t)) = \int_{-\infty}^{\infty} \delta(t)\, e^{-i\omega t}\, dt \xlongequal{\int_{-\infty}^{\infty} f(t)\,\delta(t)\, dt = f(0)} e^{-i\omega x_0} = 1$$

پس انتظار مقدار 1 در همه جا را داریم.

❊ نمودارهای تست ها:

برای کسینوس:

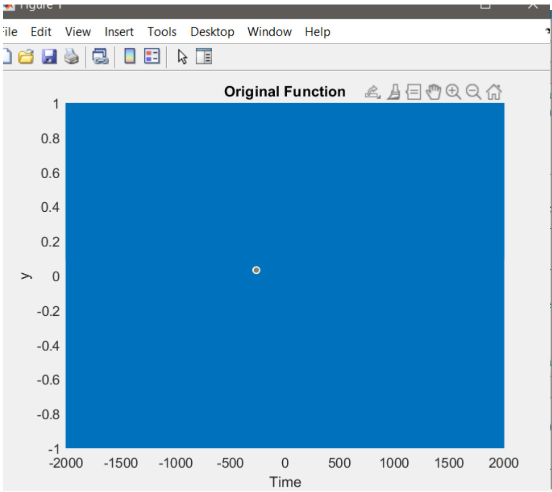


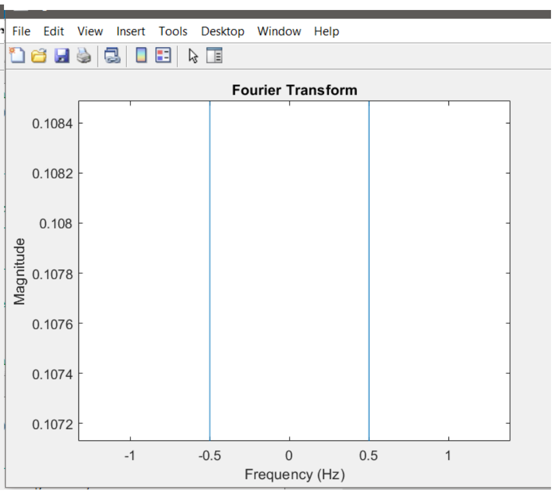


تابع اصلی طبق دستور سوال، در دو دوره تناوب رسم شده اما در نمایش تبدیل فوریه، مقداری خطا داریم و بجای اینکه در نقاط **-0.5** و **+0.5** دارای تابع ضربه باشیم(طبق محاسبه ی تئوری، باید در **-pi** و **+pi** دارای ضربه می بودیم اما در اینجا محور افقی نمایانگر فرکانس است و نه فرکانس زاویه ای، پس باید آنرا ضربدر **2pi** کنیم تا به فرکانس زاویه ای که مدنظرمان هست یعنی **-pi** و **+pi** برسیم.)، گویا چهار تابع ضربه داریم که در نزدیکی **-0.5** و **+0.5** هستند

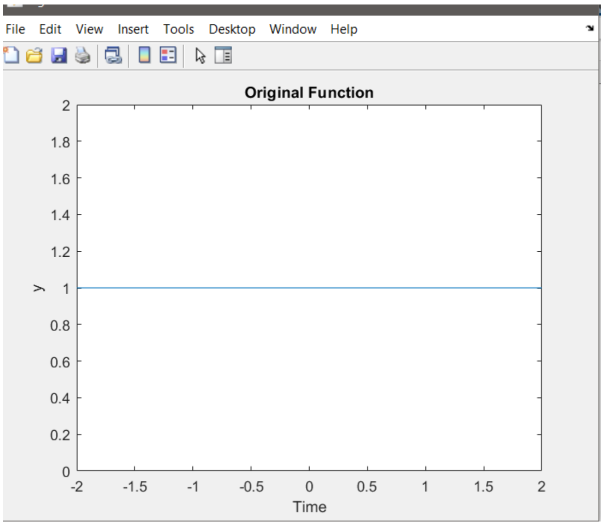
برای رفع مشکل خطا داشتن باید دقت کنیم که دلیل این خطا احتمالا این است که با سرعت خوبی نمونه برداری را انجام نمیدهیم و این منجر به پیک های کاذب شده است و دقت ما را هم پایین آورده است. و همچنین تعداد نمونه ها هم کم است پس می توانیم با بیشتر کردن تعداد تناوب های رسم شده یا همان بازه ی **t_range** مشکل را حل کنیم، مثلا شکل زیر برای این نمونه رسم شده است(البته که شکل اصلی تابع بعلت زیاد بودن نمونه ها و بازه ی زمان، کلا غیرقابل تشخیص است و باید زوم کنیم تا معلوم شود، اما همانطور که از تصویر سمت راست مشخص است، تابع تبدیل فوریه با دقت بسیار خوبی رسم شده است)

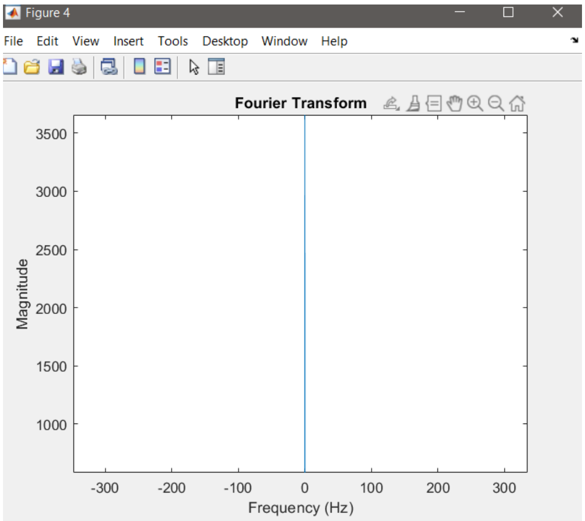
```
2 -    Fourier_Transform(@(t) cos(pi * t), 1000, [-2000 2000]);
```





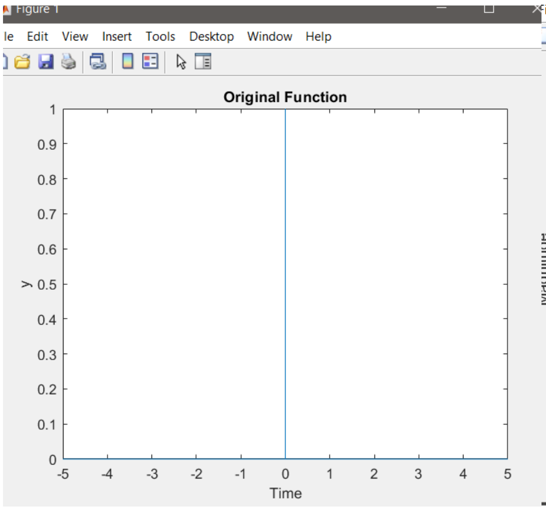
<u>برای تابع ثابت:</u>

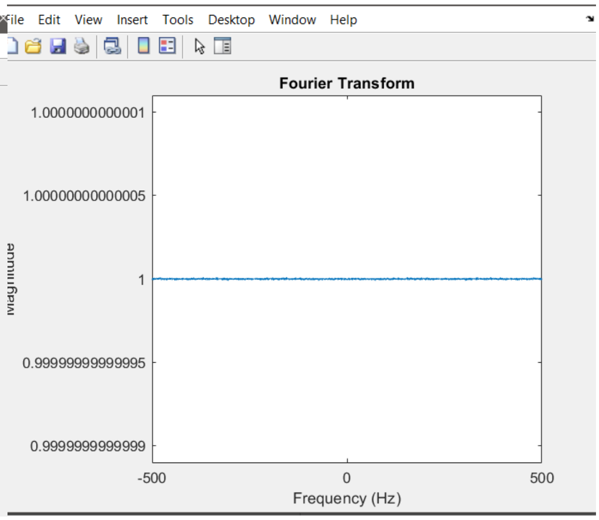




گرچه همچنان اگر در تبدیل فوریه زوم کنیم، مقادیری خطا داریم، اما میتوان آنرا هم با بزرگ کردن بازه ی t_range و روش های دیگر حل کرد، پس مقدار رسم شده نهایتا با مقدار تئوری همخوانی دارد

برای تابع دلتا:





اینجا هم مقدار تبدیل فوریه تا حد خوبی نزدیک 1 است پس با دقت خوبی تبدیل فوریه محاسبه شده است.

3.3

❊ نکات:

مراحل کار این کد به این صورت است:

– در ابتدا فایل ABITW.mp3 توسط تابع audioread خوانده شده و محتوای آن در y و fs ریخته می شود که y شامل دیتای صوتی فایل است و fs نرخ نمونه برداری از فایل صوتی است(تعداد نمونه های برداشته شده در ثانیه)

–سپس نرخ نمونه برداری فایل اصلی را که خوانده شد، پرینت می کنیم

–سپس همان دیتای y را با نرخ نمونه برداری دو برابر(fs*2) در فایل جدیدی بنام double.wav میریزیم. دوبرابر کردن نرخ نمونه برداری باعث میشود که فایل صوتی با سرعتی دو برابر سرعت زمانی که نرخ نمونه برداری معمولی داشتیم، پخش شود. و تند تر شود و صدایی که پخش میشود زیرتر از صدای اصلی است و گویی کیفیت بهتری دارد.

–تابع sound هم که دیتای صوتی ای که به آن داده شده را با نرخ نمونه برداری ای که تعیین شده است برایش، پخش میکند.

- نهایتا هم همان دیتای **y** را با نرخ نمونه برداری نصف برابر(**fs/2**) در فایل جدیدی بنام **half.wav** میریزیم. نصف کردن نرخ نمونه برداری باعث میشود که فایل صوتی با سرعتی نیم برابر سرعت زمانی که نرخ نمونه برداری معمولی داشتیم، پخش شود. و آهسته تر شود و صدایی که پخش می شود بم تر و با کیفیت کمتری است.

❋ کد تابع به همراه توضیحات خط به خط در کامنت ها (Q33.m) :

```
[y, fs] = audioread('ABITW.mp3');
%sound(y, fs);
fprintf('Sampling Rate Frequency: %dHz\n', fs);

audiowrite('D:\uni\semester 4\ریضمو\CAs\CA1\double.wav', y, fs*2);
[y, fs_new] = audioread('double.wav');
%sound(y, fs_new);

audiowrite('D:\uni\semester 4\ریضمو\CAs\CA1\half.wav', y, fs/2);
[y, fs_new] = audioread('half.wav');
sound(y, fs_new);
```

❋فرکانس نمونه برداری این فایل صوتی

خروجی کد ما این است:

```
>> Q33
Sampling Rate Frequency: 44100Hz
```

پس فرکانس (نرخ) نمونه برداری **44100** هرتز است.

قضیه نمونه‌برداری نایکوییست (Nyquist Sampling Theorem)

قضیه نمونه‌برداری نایکوییست بیان می‌کند که برای بازسازی دقیق یک سیگنال پیوسته در حوزه زمان از نمونه‌های گسسته، فرکانس نمونه‌برداری باید حداقل دو برابر حداکثر فرکانس موجود در سیگنال اصلی باشد. این مقدار به عنوان نرخ نایکوییست (Nyquist rate) شناخته می‌شود.

به صورت ریاضی:

$$f_s \geq 2f_{max}$$

که در آن fs فرکانس نمونه برداری است و fmax حداکثر فرکانس موجود در سیگنال اصلی است

اگر فرکانس نمونه‌برداری کمتر از نرخ نایکوییست باشد، پدیده‌ای به نام تداخل (aliasing) رخ می‌دهد که باعث می‌شود فرکانس‌های بالاتر به اشتباه به عنوان فرکانس‌های پایین‌تر شناسایی شوند و در نتیجه بازسازی سیگنال نادرست باشد.

چرا فرکانس نمونه‌برداری فایل صوتی 44100 هرتز است؟

فرکانس نمونه‌برداری 44100 هرتز (44.1 کیلوهرتز) به عنوان استاندارد در فرمت‌های صوتی دیجیتال مانند CD و بسیاری از فایل‌های صوتی استفاده می‌شود. دلایل این انتخاب به شرح زیر است:

1. پاسخگویی به محدوده شنوایی انسان:

   - محدوده شنوایی انسان به طور معمول از حدود 20 هرتز تا 20 کیلوهرتز است. طبق قضیه نایکوییست، برای نمونه‌برداری دقیق از سیگنال‌هایی که تا 20 کیلوهرتز فرکانس دارند، فرکانس نمونه‌برداری باید حداقل 40 کیلوهرتز باشد.
   - انتخاب 44.1 کیلوهرتز به عنوان فرکانس نمونه‌برداری، اطمینان می‌دهد که سیگنال‌های تا 20 کیلوهرتز با حاشیه‌ای ایمن نمونه‌برداری می‌شوند و همچنین فضای کافی برای فیلترهای ضد تداخل (anti-aliasing filters) وجود دارد.

2. سازگاری با تجهیزات و فناوری‌های موجود در زمان معرفی CD :

   - زمانی که CD به عنوان یک رسانه استاندارد برای پخش موسیقی معرفی شد، فناوری‌های موجود و ملاحظات مهندسی به این نتیجه رسیدند که 44.1 کیلوهرتز بهترین تعادل بین کیفیت صوتی و فضای ذخیره‌سازی است.
   - این فرکانس نمونه‌برداری به طور موثری نیازهای کیفیت صوتی بالا را برآورده می‌کند در حالی که نیاز به فضای ذخیره‌سازی بیشتر را افزایش نمی‌دهد.

خلاصه

قضیه نایکوییست تضمین می‌کند که برای بازسازی دقیق یک سیگنال، فرکانس نمونه‌برداری باید حداقل دو برابر حداکثر فرکانس سیگنال باشد. فرکانس نمونه‌برداری 44100 هرتز برای فایل‌های صوتی انتخاب شده است تا محدوده شنوایی انسان را با کیفیت بالا پوشش دهد و در عین حال با تکنولوژی‌ها و نیازهای ذخیره‌سازی سازگار باشد.