پروژه شماره یک درس ریاضیات مهندسی

آنالیز فوریه

پریا پاسه ورز

شماره دانشجویی:810101393

1. آشنایی با Matlab

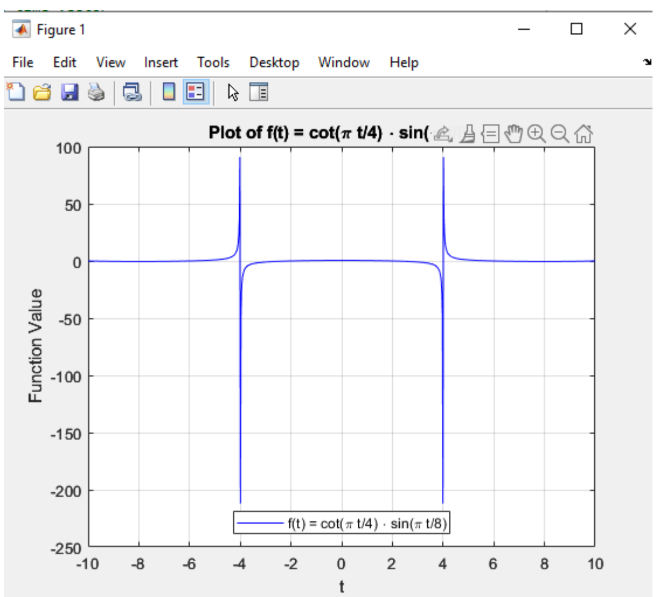
(1.1

$$\cot\frac{\pi t}{4}\,\sin\frac{\pi t}{8}$$

Code:

```
% Define the time vector
t = linspace(-10, 10, 1000); % 1000 points from -π to π

% Compute the combined function
combined_function = cot(pi * t / 4) .* sin(pi * t / 8);

% Plot the combined function
figure;
plot(t, combined_function, 'b', 'DisplayName', 'f(t) = cot(\pi t/4) \cdot sin(\pi t/8)');
xlabel('t');
ylabel('Function Value');
title('Plot of f(t) = cot(\pi t/4) \cdot sin(\pi t/8)');
legend('Location', 'south');
grid on;
```

Result:



همانطور که در کد نیز قابل مشاهده است، بازه حرکت $(-10,10)$ و گام حرکت، 1000 در نظر گرفته شده است.

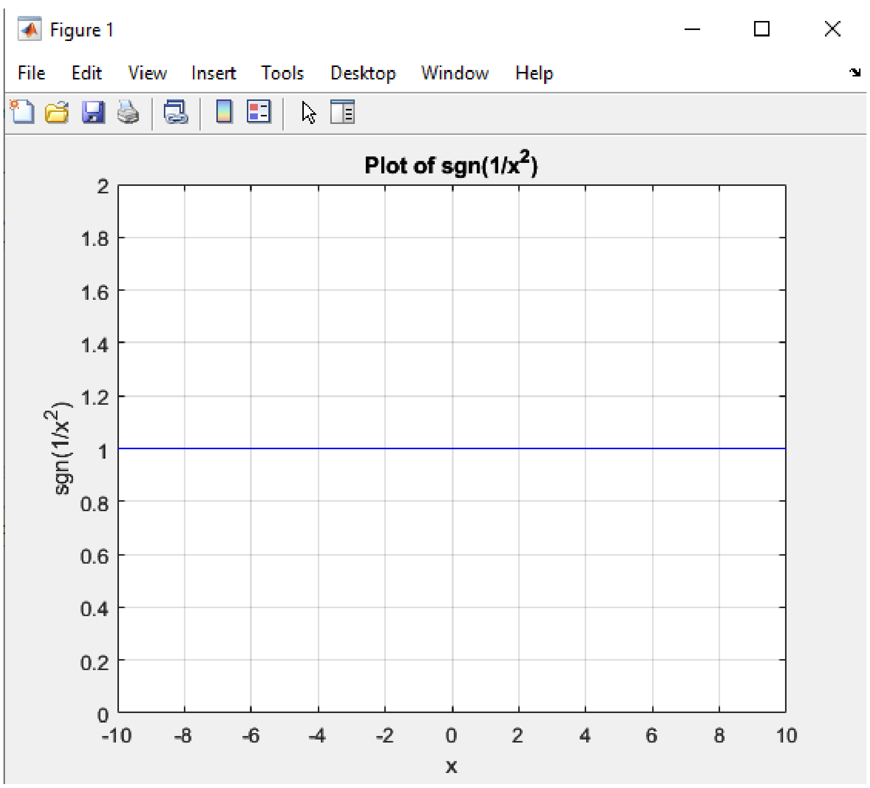
$sgn(\frac{1}{t^2})$

Code:

Introduction_2.m

```
% Define the function f(x) = 1/x^2
f = @(x) 1 ./ (x.^2);

% Define the domain (e.g., from -10 to 10)
x_values = linspace(-10, 10, 1000);

% Compute the sign of the function
sign_function = sign(f(x_values));

% Plot the function
figure;
plot(x_values, sign_function, 'b', 'DisplayName',  'sgn(1/x^2)');
xlabel('x');
ylabel('sgn(1/x^2)');
title('Plot of sgn(1/x^2)');
grid on;
```

Result:



همانطور که در کد نیز قابل مشاهده است، بازه حرکت $(-10, 10)$ و گام حرکت، 1000 در نظر گرفته شده است.

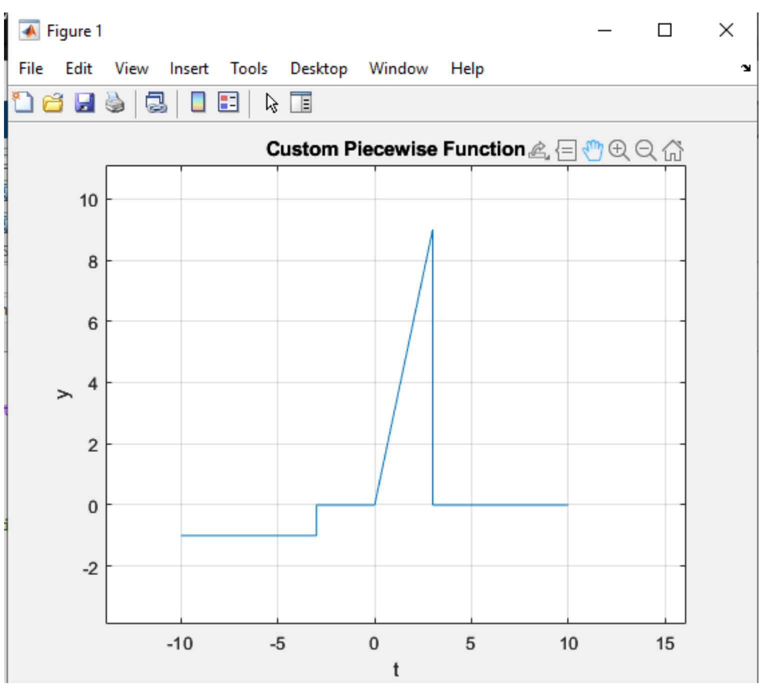
برای محاسبه تابع $sign$ از تابع داخلی متلب استفاده کرده ایم.

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

Code:

```
% Plot the function over the desired interval
fplot(@piecewise, [-10, 10])
title('Custom Piecewise Function')
xlabel('t')
ylabel('y')
grid on

% Define the piecewise function
function y = piecewise(t)
    if t < -3
        y = -1;
    elseif t < 3
        y = 3 * ramp(t);
    else
        y = exp(-2.5 * t);
    end
end

% Define the ramp function
function r = ramp(t)
    r = max(0, t);  % Simple ramp function
end
```

Result:



همانطور که در کد نیز قابل مشاهده است، بازه حرکت $(-10, 10)$ در نظر گرفته شده است.

چون اینجا تابع چند ضابطه ای است، خودمان یک تابع جدید تعریف کردیم تا بتوانیم آن را بسازیم.

همچنین برای ساخت تابع ramp نیز یک تابع جداگانه تعریف کرده ایم.

2. سری فوریه

1.2 محاسبه سری فوریه

```
function [f, t] = exponential_1(Num, P, alpha, Nshow)

t = linspace(-2*P, 2*P, 1000);

a_0 = (1/(P)) * integral(@(x) x.^alpha, -P, P);
disp(a_0)
a_n = zeros(Num, 1);
b_n = zeros(Num, 1);
for n = 1:Num
    a_n(n) = (1/P) * integral(@(x) x.^alpha .* cos((pi*n*x)/P), -P, P);
    b_n(n) = (1/P) * integral(@(x) x.^alpha .* sin((pi*n*x)/P), -P, P);
end
disp('an')
disp(a_n)
disp('bn')

disp(b_n)

f = (a_0/2);
for n = 1:Nshow
    f = f + a_n(n)*cos(pi*n*t/P) + b_n(n)*sin(pi*n*t/P);
end

```

همانطور که در کد نیز قابل مشاهده است، بازه حرکت $(-2p, 2p)$ و گام حرکت، 1000 در نظر گرفته شده است.

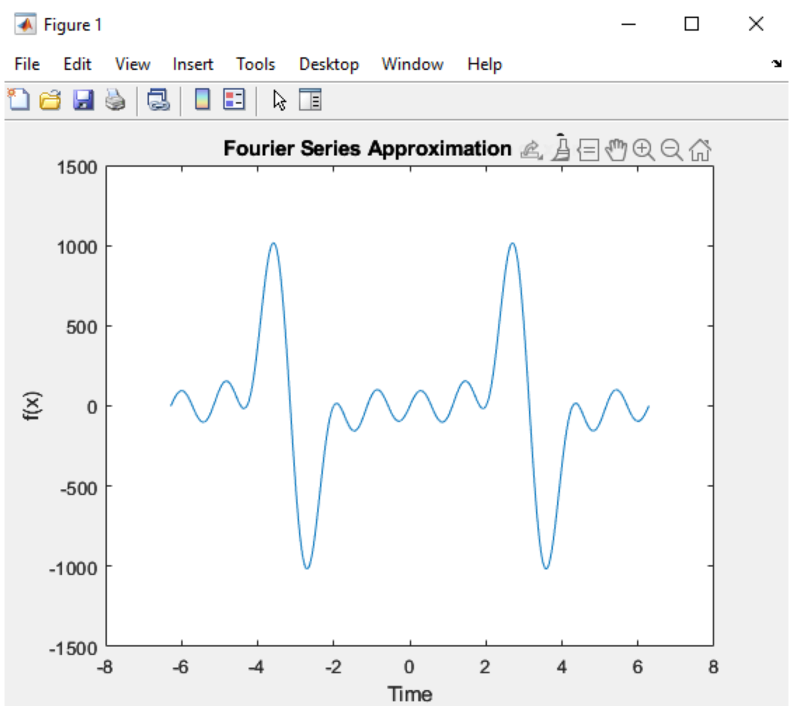
2.2 محاسبه سری فوریه یک تابع خاص

Code:

```
function [f, t] = ln_function(Num, P, a, b , Nshow)

t = linspace(-2*P, 2*P, 1000);


a0 = (1/(P)) * integral(@(x) x.^b .* log(a.*x), -P, P);
disp(a0)
an = zeros(Num, 1);
bn = zeros(Num, 1);
for n = 1:Num
    an(n) = (1/P) * integral(@(x) x.^b .* log(a.*x) .* cos((pi*n*x)/P), -P, P);
    bn(n) = (1/P) * integral(@(x) x.^b .* log(a.*x) .* sin((pi*n*x)/P), -P, P);
end
disp('an')
disp(an)
disp('bn')
disp(bn)

f = (a0/2);
for n = 1:Nshow
    f = f + an(n)*cos(pi*n*t/P) + bn(n)*sin(pi*n*t/P);
end

end

```

```matlab
Num = 10;
P = pi;
a = 100;
b = 5;
Nshow = 5;

% Calculate the Fourier series
[f, t] = ln_function(Num, P, a, b , Nshow);

%Plot the Fourier series approximation
plot(t, f);
xlabel('Time');
ylabel('f(x)');
title('Fourier Series Approximation of x^2lnx');
```

Result:



همانطور که در کد نیز قابل مشاهده است، بازه حرکت $(-2p, 2p)$ و گام حرکت، 1000 در نظر گرفته شده است.

مقدار a نیز 100 و مقدار b، 5 در نظر گرفته شده است.

3.2 رسم سری فوریه و مقایسه با تابع اصلی

Num = 10:

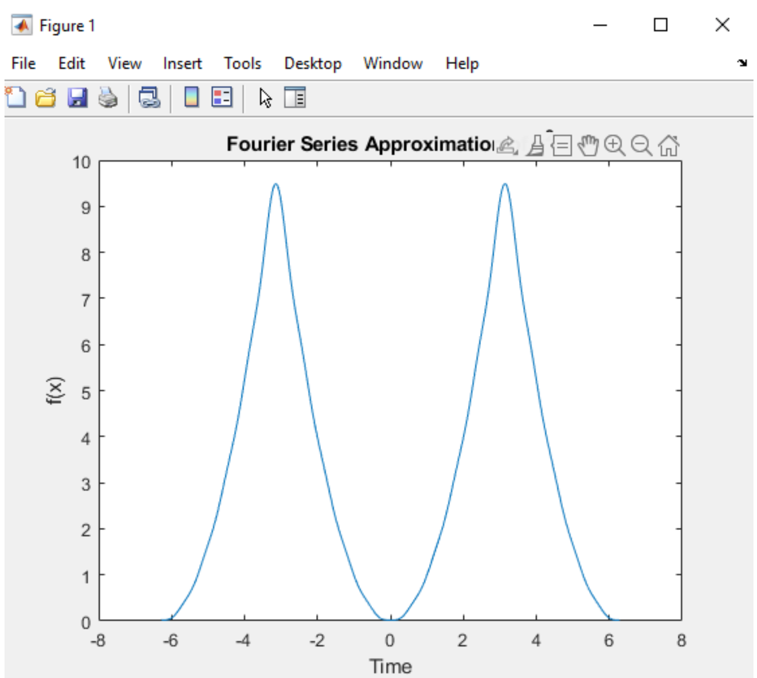
Code:

Editor - E:\Documents\Rizmo\CA\Engineering Mathematics CA1 Fourier Analysis Spring 2024\Codes\2\fourier_series_3.m

exponential_1.m | fourier_series_3.m

```
Num = 10;
P = pi;
alpha = 2;
Nshow = 10;

[f, t] = exponential_1(Num, P, alpha, Nshow);

plot(t, f);
xlabel('Time');
ylabel('f(x)');
title('Fourier Series Approximation of x^2');
```

Result:



Num = 50:

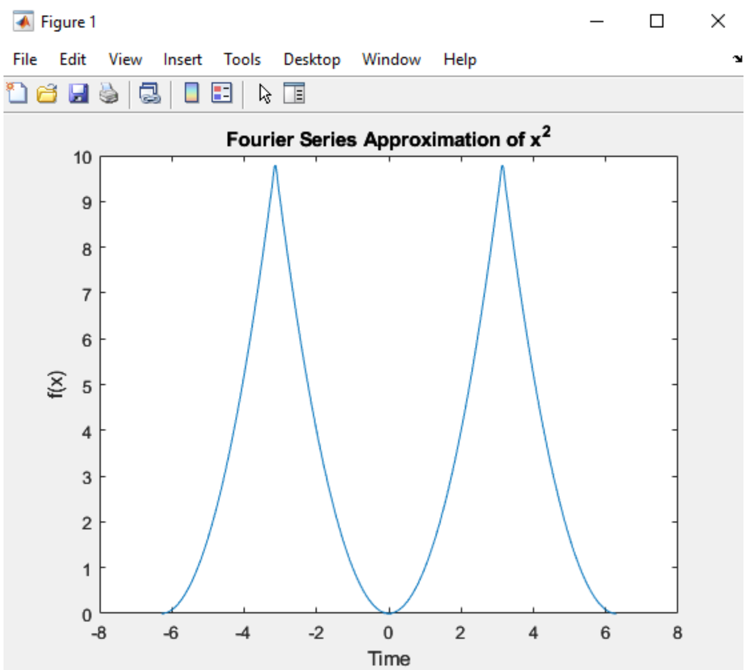
Code:

Editor - E:\Documents\Rizmo\CA\Engineering Mathematics CA1 Fourier Analysis Spring 2024\Codes\2\fourier_series_3.m

exponential_1.m fourier_series_3.m

```
Num = 50;
P = pi;
alpha = 2;
Nshow = 50;

[f, t] = exponential_1(Num, P, alpha, Nshow);

plot(t, f);
xlabel('Time');
ylabel('f(x)');
title('Fourier Series Approximation of x^2');
```

Result:

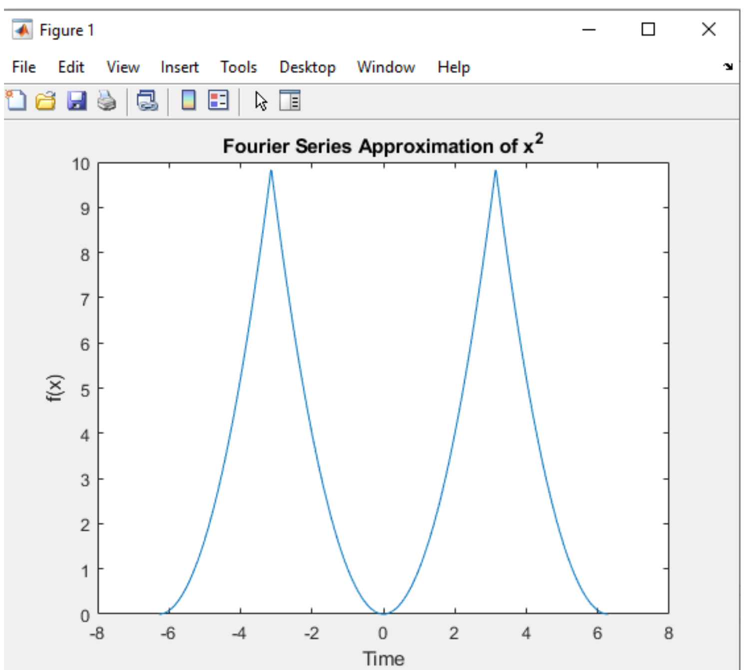


Num = 100:

Code:

```
Num = 100;
P = pi;
alpha = 2;
Nshow = 100;

[f, t] = exponential_1(Num, P, alpha, Nshow);

plot(t, f);
xlabel('Time');
ylabel('f(x)');
title('Fourier Series Approximation of x^2');
```

Result:



می دانیم هر چه جملاتی که سری فوریه را به ازای آن محاسبه می کنیم بیشتر باشد، حاصل نهایی به مقدار حقیقی تابع نزدیک تر خواهد بود.

در اینجا نیز مشاهده می کنیم که با افزایش مقدار Num، شکل نمودار رسم شده به $f(x) = x^2$ نزدیک تر شده و نقاطی که هر دوره تناوب شروع می شود، تیزتر شکسته می شود.

4.2 محاسبه حد مجموع و تطبیق نتایج

تحلیل تئوری:

$$f(x) = a_0 + \sum_{n=1}^{+\infty} a_n \cos(nx) + b_n \sin(nx)$$

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

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} x^2 \cos(nx) \, dx = \frac{1}{\pi}\frac{2(\pi^2 n^2 - 2)\sin(n\pi) + 4\pi n \cos(n\pi)}{n^3} = \frac{4(-1)^n}{n^2}$$

چون تابع $f(x) = x^2$ تابعی زوج است، پس:

$$b_n = 0 \quad \forall n \geq 1$$

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

حال اگر قرار دهیم $x = \pi$، خواهیم داشت:

$$\pi^2 = \frac{\pi^2}{3} + 4\sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} \cos(n\pi)$$

$$2\frac{\pi^2}{3} = 4\sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} (-1)^n$$

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

تحلیل کامپیوتری:

کد را مشابه روبه‌رو تغییر می دهیم و مقدار t را π می گذاریم تا مقدار محاسبه شده برای سری فوریه را در این نقطه بیابیم:

Code:

Editor - E:\Documents\Rizmo\CA\Engineering Mathematics CA1 Fourier Analysis Spring 2024\Codes\2\exponen

sound_1.m | exponential_1.m | fourier_series_3.m | +

```
function [f, t] = exponential_1(Num, P, alpha, Nshow)
t = pi;

a_0 = (1/(2*P)) * integral(@(x) x.^alpha, -P, P);
disp('a0')
disp(a_0)

a_n = zeros(Num, 1);
b_n = zeros(Num, 1);
for n = 1:Num
    a_n(n) = (1/P) * integral(@(x) x.^alpha .* cos((pi*n*x)/P), -P, P);
    b_n(n) = (1/P) * integral(@(x) x.^alpha .* sin((pi*n*x)/P), -P, P);
end
disp('an')
disp(a_n)
disp('bn')

disp(b_n)

f = (a_0);
for n = 1:Nshow
    f = f + a_n(n)*cos(pi*n*t/P) + b_n(n)*sin(pi*n*t/P);
end

disp('f')
disp(f)

end
```

Result:

a0 : 3.289

f : 9.8298

حال اگر از رابطه سری فوریه استفاده کنیم، خواهیم داشت:

$$9.8298 = 3.289 + 4\sum_{n=1}^{\infty}\frac{(-1)^n}{n^2}(-1)^n$$

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{9.8298 - 3.289}{4} = 1.6352$$

در روش تئوری به دست آوردیم:

$$\sum_{n=1}^{\infty}\frac{1}{n^2} = \frac{\pi^2}{6} = 1.6449$$

همین طور که مشاهده می کنیم، اندکی خطا در این دو آنالیز وجود دارد، ولی تا حد خوبی مطابقت دارند.

5.2 آنالیز هارمونیک در سری فوریه

1.5.2 شبیه سازی آنالیز هارمونیک

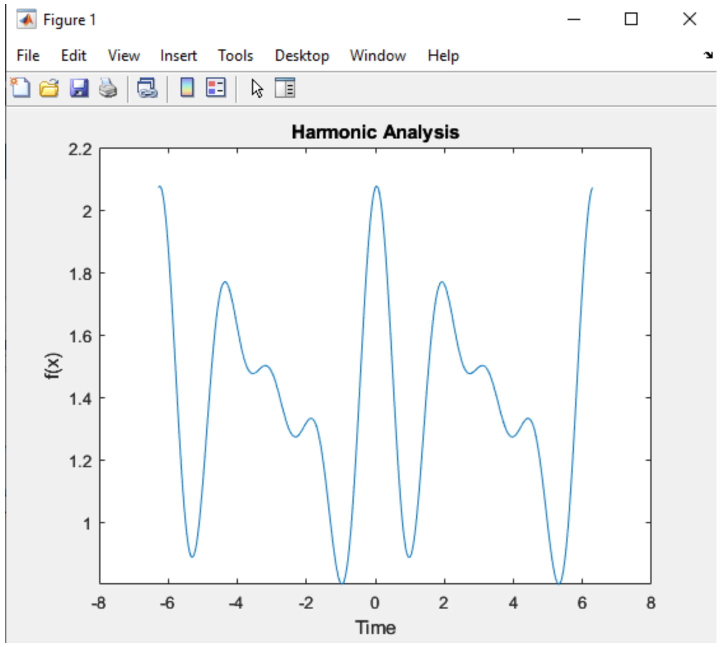
Code:

```
function [f, t] = harmonic_1(x_values, fx_values, Nshow, P)

t = linspace(-2*P, 2*P, 1000);
arr_size = numel(x_values);

a_0 = 2 * mean(fx_values);

disp('a0')
disp(a_0)

a_n = zeros(Nshow, 1);
b_n = zeros(Nshow, 1);

for n = 1:Nshow
    sum_fx_cos_nx = 0;
    sum_fx_sin_nx = 0;
    for i = 1:arr_size
    x = x_values(i);
    sum_fx_cos_nx = sum_fx_cos_nx + fx_values(i) * cos(n * x);
    end
    mean_fx_cos_nx = sum_fx_cos_nx / arr_size;
    a_n(n) = 2 * mean_fx_cos_nx;

    for i = 1:arr_size
    x = x_values(i);
    sum_fx_sin_nx = sum_fx_sin_nx + fx_values(i) * sin(n * x);
    end
    mean_fx_sin_nx = sum_fx_sin_nx / arr_size;
    b_n(n) = 2 * mean_fx_sin_nx;
end

disp('an')
disp(a_n)
disp('bn')

disp(b_n)

f = (a_0/2);
for n = 1:Nshow
    f = f + a_n(n)*cos(n*t) + b_n(n)*sin(n*t);
end
```

```
x_values = [0, pi/3, (2*pi)/3, pi, (4*pi)/3, (5*pi)/3, 2*pi];
fx_values = [1, 1.4, 1.9, 1.7, 1.5, 1.2, 1];
P = pi;
Nshow = 4;

[f, t] = harmonic_1(x_values, fx_values, Nshow, P);

plot(t, f);
xlabel('Time');
ylabel('f(x)');
title('Harmonic Analysis');
```

همانطور که در کد نیز قابل مشاهده است، بازه حرکت $(-2p, 2p)$ و گام حرکت، 1000 در نظر گرفته شده است.

Result:



ضرایب در صورت پروژه به اشتباه محاسبه شده اند، مقادیر صحیح:

a0

2.7714

an

-0.0286

0.2000

0.3143

0.2000

bn

0.1485

-0.0495

0.0000

0.0495

3 تبدیل فوریه

3.2 بررسی حوزه زمان و فرکانس چند تابع

نحوه کار تابع fft و fftshift:

وقتی fft را روی یک تابع اعمال می کنیم، تبدیل فوریه اعمال شده در فرکانس صفر مرکزی نشده است، به همین دلیل نتیجه درستی را نشان نمی دهد. با اعمال کردن fftshift، تابع را شیفت می دهد تا فرکانس صفر مرکز آن قرار گیرد و نمایش صحیحی داشته باشیم.

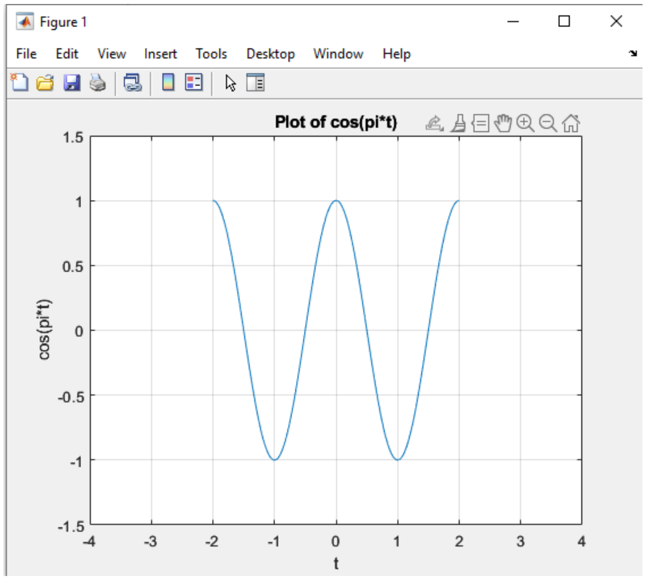
بررسی تابع $f(t) = cos(\pi t)$:

رسم تابع در دو دوره تناوب:

Code:

```
fs = 1000;
y = cos(pi*t);
t = -2:1/fs:2;

%Plot cos(pi*t)
plot(t,y)
xlim([-4 4])
ylim([-1.5 1.5])
xlabel('t')
ylabel('cos(pi*t)')
title('Plot of cos(pi*t)')
grid on

%U fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
%plot(f, abs(y1))
%xlabel('Frequency (Hz)');
%ylabel('F(w)');
%title('Fourier Transform of f(t) = cos(pi*t)');
%grid on

y_shifted = fftshift(y1);

%Plot y_shifted
% plot(f, abs(y_shifted));
% xlabel('Frequency (Hz)');
% ylabel('F(w)');
% title('Fourier Transform of f(t) = cos(pi*t)');
% grid on

```

Result:



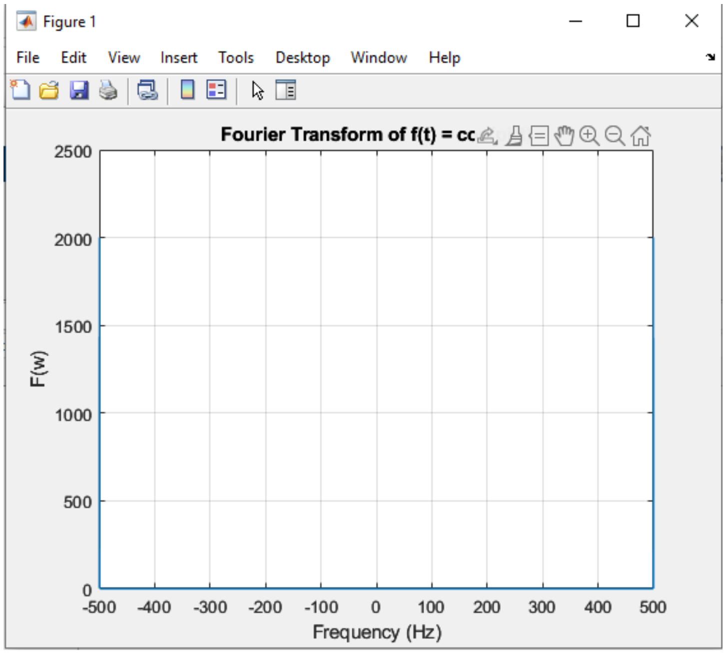
محاسبه تبدیل فوریه با کمک دستور fft:

Code:

fourier_series_3.m | harmonic_1.m | harmonic_test.m | f_tranfrom_1.m

```
fs = 1000;
y = cos(pi*t);
t = -2:1/fs:2;

%Plot cos(pi*t)
% plot(t,y)
% xlim([-4 4])
% ylim([-1.5 1.5])
% xlabel('t')
% ylabel('cos(pi*t)')
% title('Plot of cos(pi*t)')
% grid on

%Use fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
plot(f, abs(y1))
xlabel('Frequency (Hz)');
ylabel('F(w)');
title('Fourier Transform of f(t) = cos(pi*t)');
grid on

y_shifted = fftshift(y1);

%Plot y_shifted
% plot(f, abs(y_shifted));
% xlabel('Frequency (Hz)');
% ylabel('F(w)');
% title('Fourier Transform of f(t) = cos(pi*t)');
% grid on

```

Result:

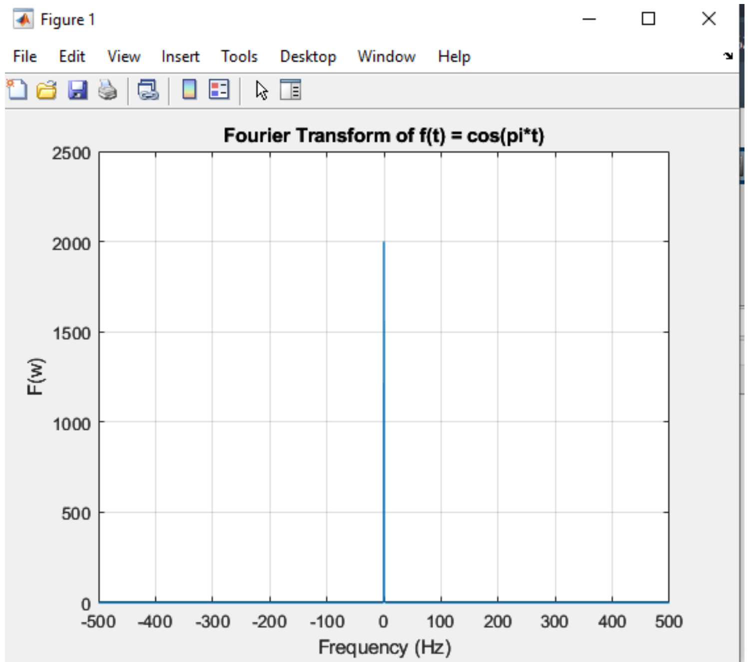


محاسبه تبدیل فوریه با کمک دستور fftshift:

Code:

```
fs = 1000;
y = cos(pi*t);
t = -2:1/fs:2;

%Plot cos(pi*t)
% plot(t,y)
% xlim([-4 4])
% ylim([-1.5 1.5])
% xlabel('t')
% ylabel('cos(pi*t)')
% title('Plot of cos(pi*t)')
% grid on

%Use fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
% plot(f, abs(y1))
% xlabel('Frequency (Hz)');
% ylabel('F(w)');
% title('Fourier Transform of f(t) = cos(pi*t)');
% grid on

y_shifted = fftshift(y1);

%Plot y_shifted
plot(f, abs(y_shifted));
xlabel('Frequency (Hz)');
ylabel('F(w)');
title('Fourier Transform of f(t) = cos(pi*t)');
grid on
```

Result:



محاسبه تبدیل فوریه به صورت دستی:

$$f(t) = Cos(w_0 t)$$

$$F(w) = \int_{-\infty}^{\infty} f(t)e^{-iwt}\,dt = \int_{-\infty}^{\infty} Cos(w_0 t)\, e^{-iwt}\,dt$$

$$Cos(w_0 t) = \frac{e^{-iw_0 t} + e^{iw_0 t}}{2}$$

$$F(w) = \int_{-\infty}^{\infty} (\frac{e^{-iw_0 t} + e^{iw_0 t}}{2})e^{-iwt}\ dt = \frac{1}{2}\int_{-\infty}^{\infty} e^{-i(w-w_0)t}\ dt + \frac{1}{2}\int_{-\infty}^{\infty} e^{-i(w+w_0)t}\ dt$$

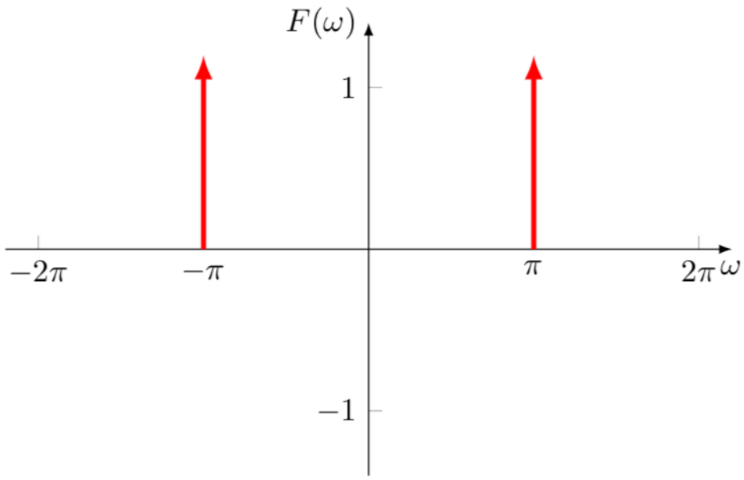
$$\delta(\boldsymbol{t}) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-iwt}\ dt$$

$$\int_{-\infty}^{\infty} e^{-i(w-w_0)t}\ dt = 2\pi\delta(w - w_0)$$

$$\int_{-\infty}^{\infty} e^{-i(w+w_0)t}\ dt = 2\pi\delta(w + w_0)$$

$$F\big(Cos(w_0 t)\big) = \pi\big(\delta(w - w_0) + \delta(w + w_0)\big)$$

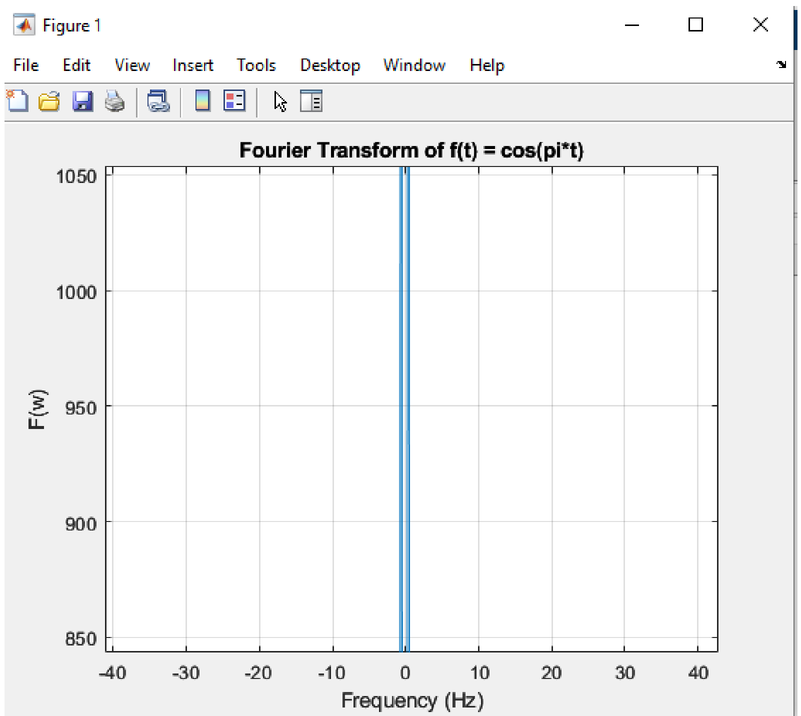
$$w_0 = \pi \Rightarrow F\big(Cos(w_0 t)\big) = \pi\big(\delta(w - \pi) + \delta(w + \pi)\big)$$



همانطور که توضیح دادیم، نمایش تبدیل فوریه حاصل از اعمال تابع fft، صحیح نیست، ولی پس از اعمال تابع fftshift، مطابق با محاسبات عملی روی دو نقطه 0.5 و -0.5 پیک می زند.

دقت کنید دلیل اینکه روی π و $-\pi$ پیک نمی زند این است که ما اینجا فرکانس را اندازه می گیریم، نه فرکانس زاویه ای.

برای بهتر دیدن نقاط پیک، روی نمودار زوم می کنیم:



بررسی تابع $f(t) = 1$:

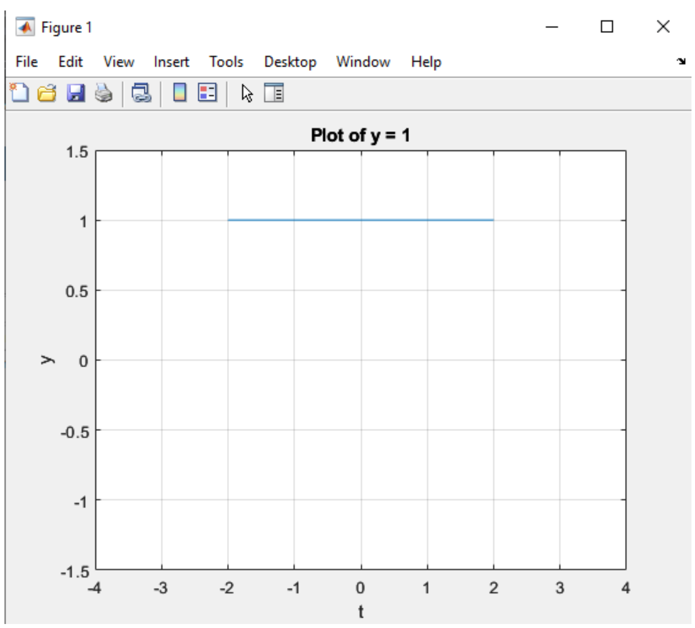
رسم تابع در دو دوره تناوب:

Code:

fourier_series_3.m | harmonic_1.m | harmonic_test.m | f_tranfrom_1.m | f_tranfrom_2.m

```
fs = 1000;
a = 1;
y = a * ones(size(t));
t = -2:1/fs:2;

%Plot y=1
plot(t,y)
xlim([-4 4])
ylim([-1.5 1.5])
xlabel('t')
ylabel('y')
title('Plot of y = 1')
grid on

%Use fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
%plot(f, y1)
%xlabel('Frequency (Hz)');
%ylabel('F(w)');
%title('Fourier Transform of f(t) = 1');
%grid on

y_shifted = fftshift(y1);

%Plot y_shifted
% plot(f, abs(y_shifted));
% xlabel('Frequency (Hz)');
% ylabel('F(w)');
% title('Fourier Transform of f(t) = 1');
% grid on
```

Result:

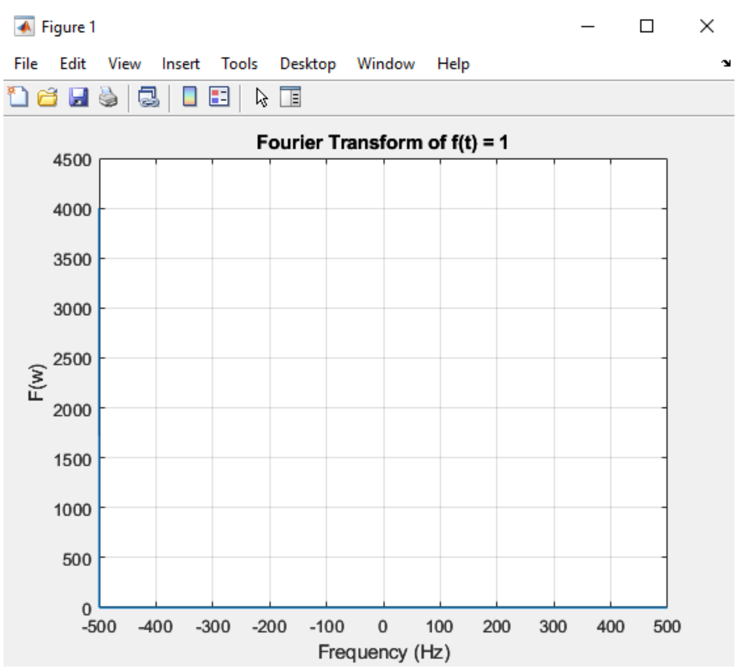


محاسبه تبدیل فوریه با کمک دستور fft:

Code:

```
fs = 1000;
a = 1;
y = a * ones(size(t));
t = -2:1/fs:2;

%Plot y=1
% plot(t,y)
% xlim([-4 4])
% ylim([-1.5 1.5])
% xlabel('t')
% ylabel('y')
% title('Plot of y = 1')
% grid on

%Use fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
plot(f, y1)
xlabel('Frequency (Hz)');
ylabel('F(w)');
title('Fourier Transform of f(t) = 1');
grid on

y_shifted = fftshift(y1);

%Plot y_shifted
% plot(f, abs(y_shifted));
% xlabel('Frequency (Hz)');
% ylabel('F(w)');
% title('Fourier Transform of f(t) = 1');
% grid on
```

Result:

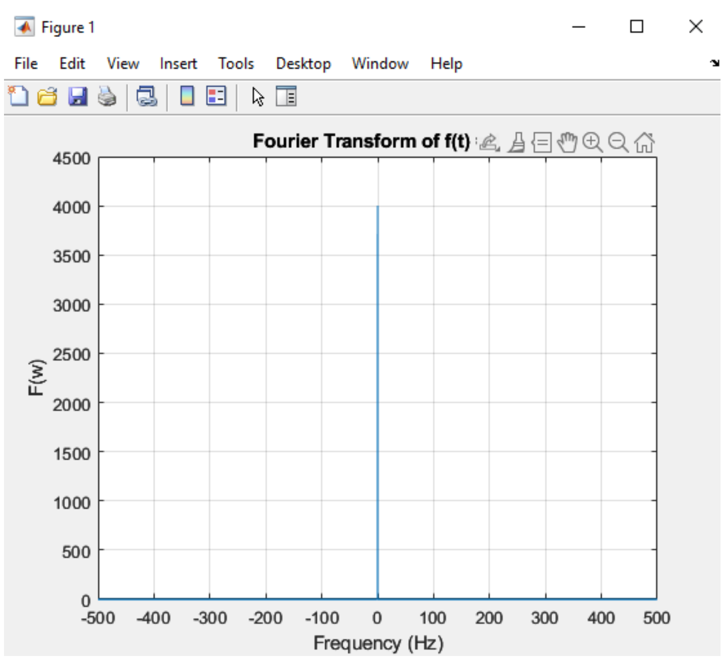


محاسبه تبدیل فوریه با کمک دستور fftshift:

Code:

```
fs = 1000;
a = 1;
y = a * ones(size(t));
t = -2:1/fs:2;

%Plot y=1
% plot(t,y)
% xlim([-4 4])
% ylim([-1.5 1.5])
% xlabel('t')
% ylabel('y')
% title('Plot of y = 1')
% grid on

%Use fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
% plot(f, y1)
% xlabel('Frequency (Hz)');
% ylabel('F(w)');
% title('Fourier Transform of f(t) = 1');
% grid on

y_shifted = fftshift(y1);

%Plot y_shifted
plot(f, abs(y_shifted));
xlabel('Frequency (Hz)');
ylabel('F(w)');
title('Fourier Transform of f(t) = 1');
grid on
```

Result:



محاسبه تبدیل فوریه به صورت دستی:

$$f(t) = A$$

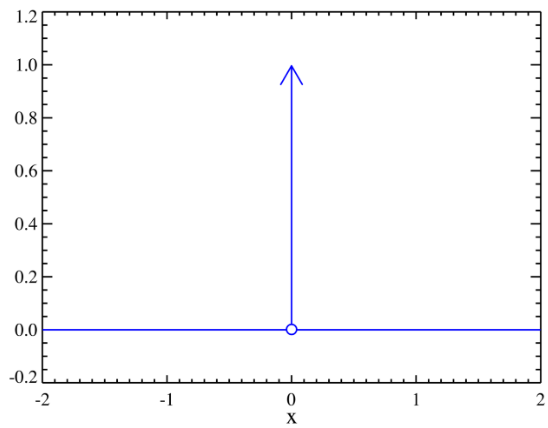
$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-iwt}\, dt = \int_{-\infty}^{\infty} A\, e^{-iwt}\, dt$$

$$\delta(\boldsymbol{t}) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-iwt}\ dt$$

$$\int_{-\infty}^{\infty} A\, e^{-iwt}\, dt\ =\ A\ \int_{-\infty}^{\infty} e^{-iwt}\, dt\ =\ 2\pi A\ \delta(w)$$

$$F(A) = 2\pi A \delta(w)$$

$$A = 1 \Rightarrow\ F(1) = 2\pi \delta(w)$$



همانطور که توضیح دادیم، نمایش تبدیل فوریه حاصل از اعمال تابع fft، صحیح نیست، ولی پس از اعمال تابع fftshift، مطابق با محاسبات عملی روی نقطه صفر پیک می زند.

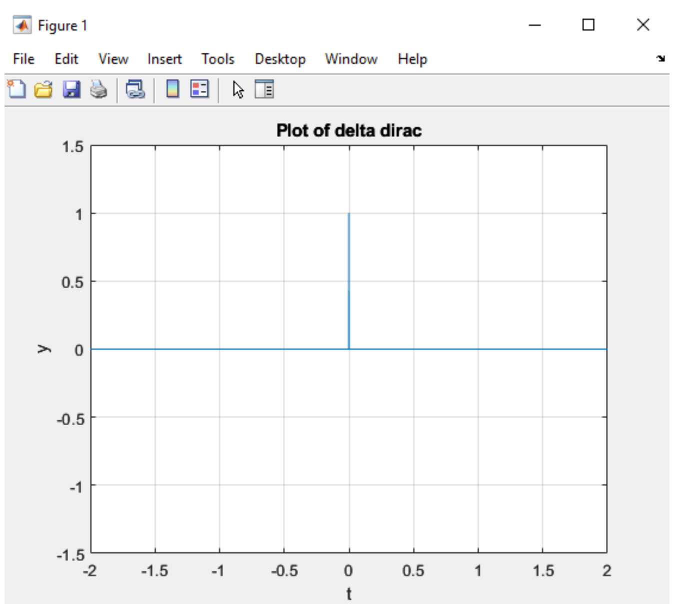
بررسی تابع $f(t) = \delta(t)$:

رسم تابع در دو دوره تناوب:

Code:

```
fs = 1000;
t = -2:1/fs:2;
y = double(t==0);

%Plot y=delta(t)
plot(t,y)
xlim([-2 2])
ylim([-1.5 1.5])
xlabel('t')
ylabel('y')
title('Plot of delta dirac')
grid on

%Use fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
%plot(f, y1)
%xlabel('Frequency (Hz)');
%ylabel('F(w)');
%title('Fourier Transform of Dirac Delta');
%grid on

y_shifted = fftshift(y1);

%Plot y_shifted
% plot(f, abs(y_shifted));
% xlabel('Frequency (Hz)');
% ylabel('F(w)');
% ylim([-5 5])
% title('Fourier Transform of Dirac Delta');
% grid on
```

Result:

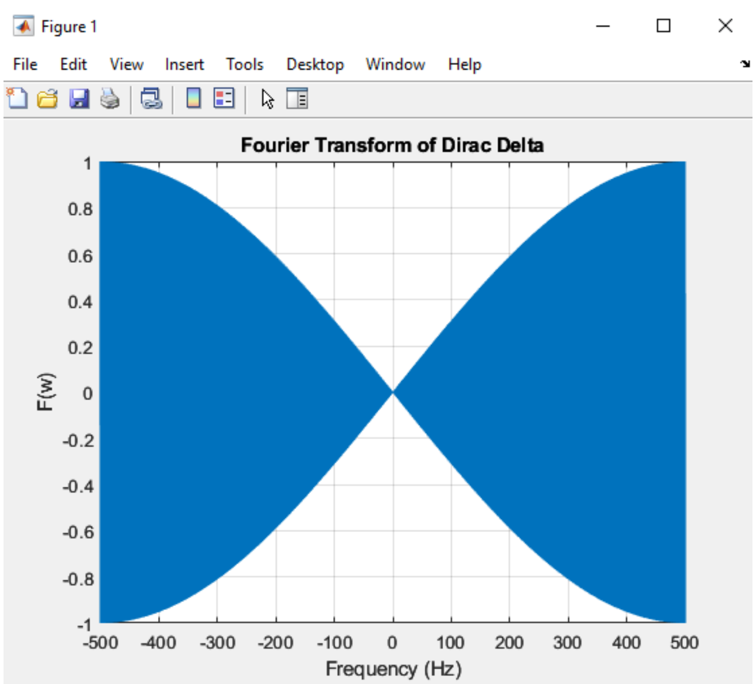


محاسبه تبدیل فوریه با کمک دستور fft:

Code:

```
fs = 1000;
t = -2:1/fs:2;
y = double(t==0);

%Plot y=delta(t)
% plot(t,y)
% xlim([-2 2])
% ylim([-1.5 1.5])
% xlabel('t')
% ylabel('y')
% title('Plot of delta dirac')
% grid on

%Use fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
plot(f, y1)
xlabel('Frequency (Hz)');
ylabel('F(w)');
title('Fourier Transform of Dirac Delta');
grid on

y_shifted = fftshift(y1);

%Plot y_shifted
% plot(f, abs(y_shifted));
% xlabel('Frequency (Hz)');
% ylabel('F(w)');
% ylim([-5 5])
% title('Fourier Transform of Dirac Delta');
% grid on
```

Result:

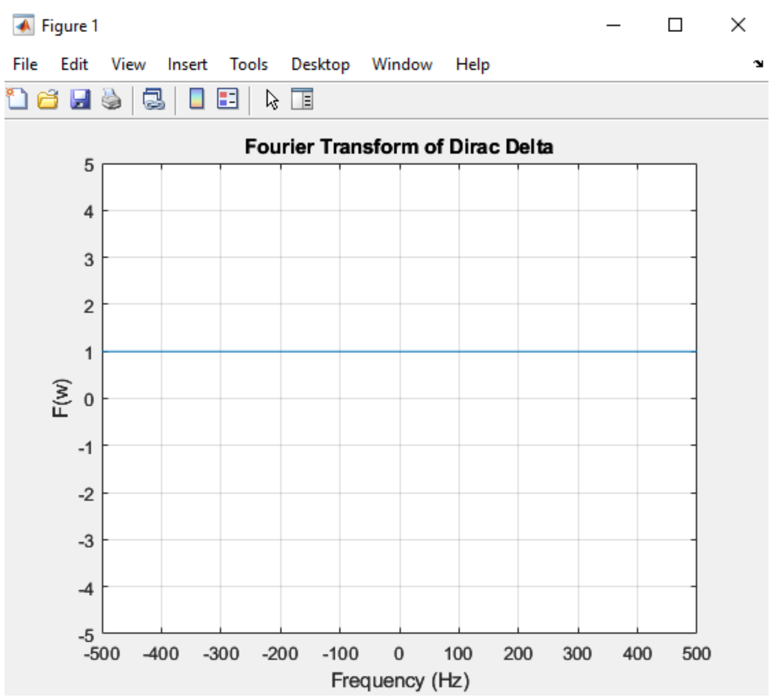


محاسبه تبدیل فوریه با کمک دستور fftshift:

Code:

```
fs = 1000;
t = -2:1/fs:2;
y = double(t==0);

%Plot y=delta(t)
% plot(t,y)
% xlim([-2 2])
% ylim([-1.5 1.5])
% xlabel('t')
% ylabel('y')
% title('Plot of delta dirac')
% grid on

%Use fft and fftshift to calculate the Fourier transform
y1 = fft(y);
f = (-length(y)/2:length(y)/2-1) * fs/length(y);

%Plot y1
%plot(f, y1)
%xlabel('Frequency (Hz)');
%ylabel('F(w)');
%title('Fourier Transform of Dirac Delta');
%grid on

y_shifted = fftshift(y1);

%Plot y_shifted
plot(f, abs(y_shifted));
xlabel('Frequency (Hz)');
ylabel('F(w)');
ylim([-5 5])
title('Fourier Transform of Dirac Delta');
grid on
```

Result:



محاسبه تبدیل فوریه به صورت دستی:

$$f(t) = \delta(\boldsymbol{t})$$

$$F(w) = \int_{-\infty}^{\infty} f(t)e^{-iwt}\,dt = \int_{-\infty}^{\infty} \delta(\boldsymbol{t})\, e^{-iwt}\,dt \;=\; \int_{-\infty}^{\infty} \delta(\boldsymbol{t})\, e^{-iw(0)}\,dt \;=\; 1$$

$$F\big(\delta(\boldsymbol{t})\big) = 1$$

همانطور که توضیح دادیم، نمایش تبدیل فوریه حاصل از اعمال تابع fft، صحیح نیست، ولی پس از اعمال تابع fftshift، مطابق با محاسبات عملی مقدار ثابت یک نمایش داده می شود.

3.3 موسیقی

فایل audio داده شده را با کمک audioread و sound باز می کنیم:

Code:

```
file_path = 'ABITW.mp3';
[y, Fs] = audioread(file_path);
sound(y, Fs);
```

گزارش فرکانس نمونه برداری:

Code:

```
file_path = 'ABITW.mp3';
[y, Fs] = audioread(file_path);
fprintf('Sampling frequency : %d Hz', Fs);
```

Result:

Sampling frequency : 44100 Hz

Nyquist–Shannon sampling theorem

قضیه نمونه‌برداری نایکوئیست-شانون بیان می‌کند که یک سیگنال را می‌توان از روی سیگنال نمونه‌برداری شده به طور دقیق بازسازی کرد، اگر فرکانس نمونه‌برداری بزرگتر از دو برابر بالاترین مولفه فرکانسی سیگنال باشد. در عمل، غالباً فرکانس نمونه‌برداری را بزرگتر از دو برابر پهنای باند لازم در نظر می‌گیرند.

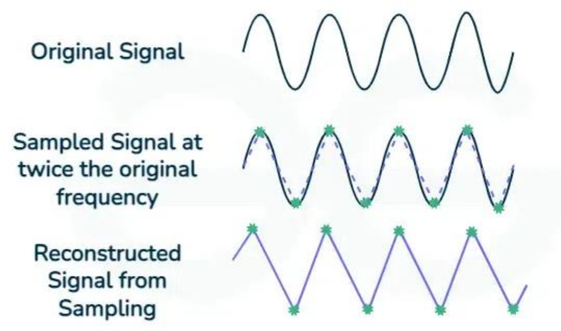
این کار به ما اطمینان می دهد که به ازای هر واحد زمانی، تعداد کافی نمونه برداری انجام شده است تا تمام جزئیات موج اصلی استخراج شود، بدون آنکه aliasing رخ دهد.

Aliasing اثری ناخواسته در سیگنال است که باعث کاهش کیفیت آن می‌شود. منظور از آن، تغییرکردن سیگنال اصلی پس از بازیابی سیگنال نمونه‌برداری‌شده است. نمونه‌ای از aliasing، دندانه‌دارشدن (پیکسلی‌شدن) تصویر دیجیتالی است.

بازه شنوایی انسان بین 20 Hz تا 20000 Hz است. بنابراین با توجه به قضیه نمونه‌برداری نایکوئیست-شانون، برای جلوگیری از از دست رفتن اطلاعات، sample rate باید بزرگ تر از بیشترین مقدار ممکن برای فرکانس باشد. پس sampling frequency باید عددی بزرگتر از 40 KHz باشد.

دلیل اینکه sampling frequency را مقدار 44.1 KHz قرار می دهیم، نه 44 KHz این است که ممکن است خطایی در دستگاه ضبط صدا یا پخش صدا وجود داشته باشد و فرکانس با اندکی خطا اندازه گیری شود. این buffer یا فضای اضافی از بروز مشکلات در این شرایط جلوگیری می کند.

نحوه نمونه برداری و ساخت signal جدید:



فرکانس نمونه برداری را دو برابر حالت اولیه کرده و نتیجه را با کمک دستور audiowrite ذخیره می کنیم:

Code:

```
input_file = 'ABITW.mp3';
output_file = 'ABITW2.wav';
[y, Fs] = audioread(input_file);
new_Fs = 2 * Fs;
y_resample = resample(y, new_Fs, Fs);
audiowrite(output_file, y_resample, new_Fs);
```

فایل audio جدید را با کمک audioread و sound باز می کنیم و sampling frequency را گزارش می دهیم:

Code:

```
file_path = 'ABITW2.wav';
[y, Fs] = audioread(file_path);
sound(y, Fs);
fprintf('Sampling frequency : %d Hz', Fs);
```

Result:

Sampling frequency : 88200 Hz

فرکانس نمونه برداری را نصف حالت اولیه کرده و نتیجه را با کمک دستور audiowrite ذخیره می کنیم:

Code:

```
input_file_path = 'ABITW.mp3';
output_file_path = 'ABITW3.wav';
[y, Fs] = audioread(input_file_path);
new_Fs = 1/2 * Fs;
y_resampled = resample(y, new_Fs, Fs);
audiowrite(output_file_path, y_resampled, new_Fs);
```

فایل audio جدید را با کمک audioread و sound باز می کنیم و sampling frequency را گزارش می دهیم:

Code:

```
file_path = 'ABITW3.wav';
[y, Fs] = audioread(file_path);
sound(y, Fs);
fprintf('Sampling frequency : %d Hz', Fs);
```

Result:

Sampling frequency : 22050 Hz

همانطور که مشاهده می کنیم، حجم فایل ABITW2، تقریبا دو برابر حجم فایل ABITW است و حجم فایل ABITW3، تقریبا نصف حجم فایل ABITW است.

هرچه sampling frequency بیشتر باشد، تعداد نمونه های انتخاب شده در هر واحد زمانی بیشتر است، پس فضای مورد نیاز برای ذخیره سازی آنها نیز بیشتر خواهد بود، و برعکس.

اما توجه کنید که در هر دو حالت، طول فایل یکسان باقی می ماند.