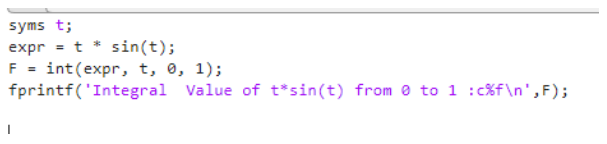
Signal Processing Lab Report -1

Aasrith Reddy and Ritama Sanyal

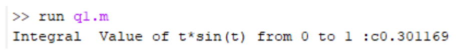
August 12, 2024

1 Definite Integration of t*sint



```
syms t;
expr = t * sin(t);
F = int(expr, t, 0, 1);
fprintf('Integral  Value of t*sin(t) from 0 to 1 :c%f\n',F);
```

Figure 1.1: Code

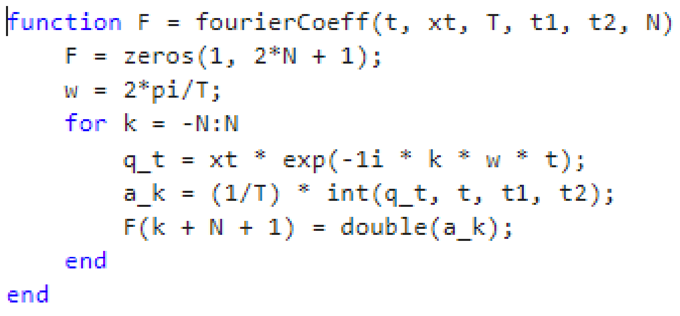


```
>> run q1.m
Integral  Value of t*sin(t) from 0 to 1 :c0.301169
```

Figure 1.2: Value

2 Finding Fourier Series Coefficients

2.1 MATLAB Function : fourierCoeff

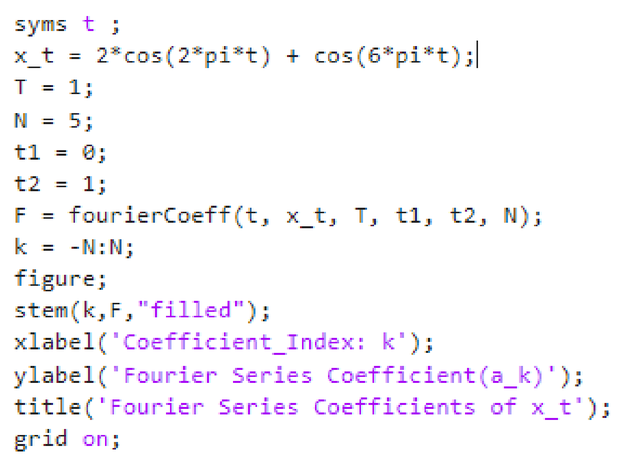


```
function F = fourierCoeff(t, xt, T, t1, t2, N)
    F = zeros(1, 2*N + 1);
    w = 2*pi/T;
    for k = -N:N
        q_t = xt * exp(-1i * k * w * t);
        a_k = (1/T) * int(q_t, t, t1, t2);
        F(k + N + 1) = double(a_k);
    end
end
```

Figure 2.1: Code

The Function fourierCoeff takes inputs t,N,T,t_1,t_2 and the signal.
T is time period of the signal. t_1,t_2 are the left and right limits where the expression x(t) is valid (else zero otherwise) for the purpose of integration.
We need to find 2*N+1 coefficients from indexes -N to N including 0. First a zero vector of size 2*N+1 is defined. Then individual coefficient is calculated from -N to N storing with indexes 0 to 2*N+1. Note float values are noted.

2.2 MATLAB Script of x(t) = 2cos(2*pi*t)+cos(6*pi*t)



```
syms t ;
x_t = 2*cos(2*pi*t) + cos(6*pi*t);
T = 1;
N = 5;
t1 = 0;
t2 = 1;
F = fourierCoeff(t, x_t, T, t1, t2, N);
k = -N:N;
figure;
stem(k,F,"filled");
xlabel('Coefficient_Index: k');
ylabel('Fourier Series Coefficient(a_k)');
title('Fourier Series Coefficients of x_t');
grid on;
```

Figure 2.2: Script

By passing given constraint in the Function **fourierCoeff** the coefficients are calculated for indexes -5 to 5.

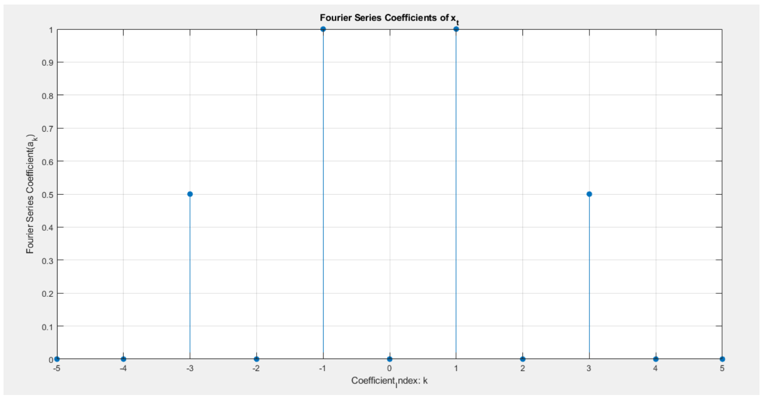


Figure 2.3: graph

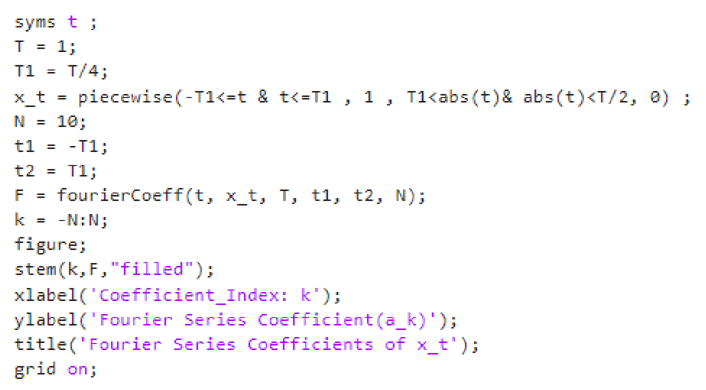
2.2.1 Values of a_k from graph

1 for k = 1 and K = -1
0.5 for k = 3 and K = -3
0 for others

2.2.2 Values of a_k from analysis

We know that a cos(t) can be express as sum of two complex signals e^{-j*t} and e^{j*t}. Considering $w_o = 2*pi$
So, $2cos(2*pi*t)+cos(6*pi*t) = e^{wo*t}+e^{-wo*t} + (e^{3*wo*t}+e^{-3*wo*t})/2$
It is observed that the plot of script resembles with calculated coefficients.

2.3 MATLAB Script of x(t) = Square pulse



```
syms t ;
T = 1;
T1 = T/4;
x_t = piecewise(-T1<=t & t<=T1 , 1 , T1<abs(t)& abs(t)<T/2, 0) ;
N = 10;
t1 = -T1;
t2 = T1;
F = fourierCoeff(t, x_t, T, t1, t2, N);
k = -N:N;
figure;
stem(k,F,"filled");
xlabel('Coefficient_Index: k');
ylabel('Fourier Series Coefficient(a_k)');
title('Fourier Series Coefficients of x_t');
grid on;
```

Figure 2.4: Script

By passing given constraint in the Function **fourierCoeff** the coefficients are calculated for indexes -10 to 10.

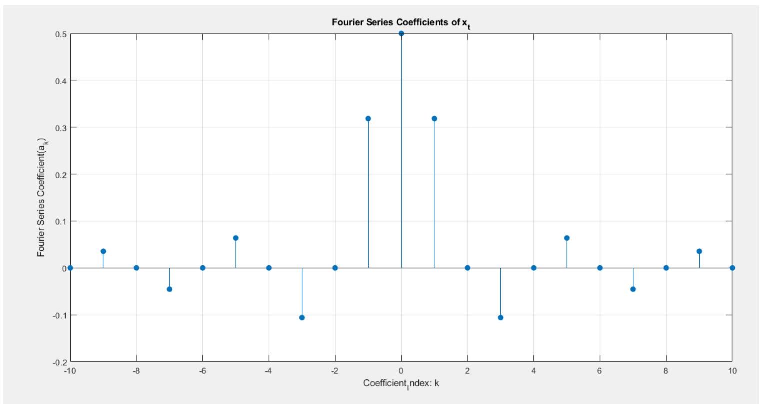


Figure 2.5: Graph

2.3.1 Values of a$_\mathrm{k}$ from graph

0.5 for k = 0
0.3183 for k = 1 and K = -1
-0.1603 for k = 3 and K = -3
0.0632 for k = 5 and K = -5
-0.045 for k = 7 and K = -7
0.0353 for k = 9 and K = -9
0 for others

2.3.2 Values of a$_\mathrm{k}$ from analysis

The Fourier Series coefficients $\{a_k\}$ for a real, periodic square wave can be calculated using the following formula:

$$a_k = \frac{2}{T}\int_{-T_1}^{T_1} x(t)e^{-jk\omega_0 t}dt$$

where $x(t)$ is the square wave, T is the period, T_1 is half the pulse width, and $\omega_0 = \frac{2\pi}{T}$.

Since the square wave has amplitude 1 in $[-T_1, T_1]$, the integral becomes:

$$a_k = \frac{2}{T}\int_{-T_1}^{T_1} 1e^{-jk\omega_0 t}dt$$

Evaluating the integral, we get:

$$a_k = \frac{4}{T_1}\frac{\sin(k\pi T_1/T)}{k\pi}$$

$$a_k = \frac{1}{n\pi}\sin\frac{(n\pi)}{2}$$

The Fourier coefficient a$_\mathrm{k}$ from -k to k is given by

a$_0$ = 1/2

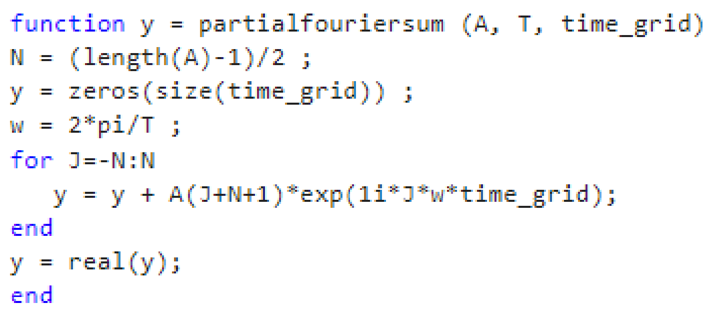
a$_\mathrm{k}$ = 0 for K is even

a$_\mathrm{k}$ = $\sin\frac{(n\pi)}{2}/(n\pi) for K is odd$

It is observed that the plot of script resembles with calculated coefficients.

3 Fourier series Reconstruction and finite Fourier Series approximation error

3.1 MATLAB Function : partialfouriersum



```
function y = partialfouriersum (A, T, time_grid)
N = (length(A)-1)/2 ;
y = zeros(size(time_grid)) ;
w = 2*pi/T ;
for J=-N:N
   y = y + A(J+N+1)*exp(1i*J*w*time_grid);
end
y = real(y);
end
```

Figure 3.1: Code

The Function partialfouriersum takes inputs A,T,time-grid.
T is time period of the signal. A is vector containing Fourier coefficients of signal from -N to N.
We need to find 2*N+1 coefficients from indexes -N to N including 0. So to Calculate N we use following expression

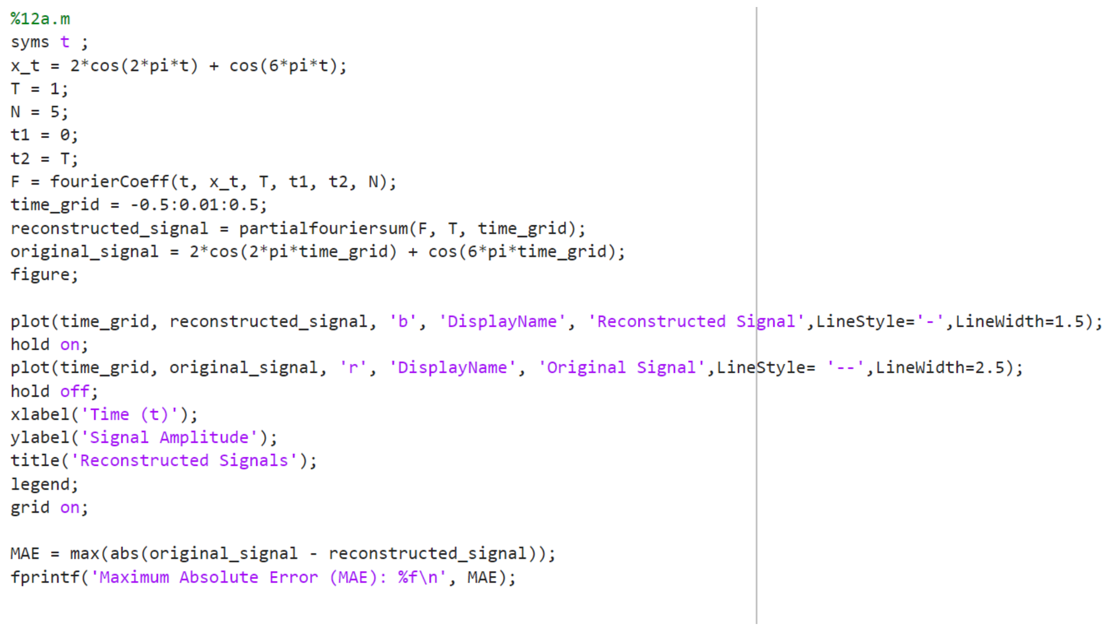
$$N = \frac{(length(A) - 1}{2}$$

First a zero vector y of size time-grid is defined. Then individual coefficient is multiplied with

$$e^{-jk\omega_0 t*time-grid}$$

and add with y from -N to N.

3.2 MATLAB Script: Fourier series Reconstruction



```
%12a.m
syms t ;
x_t = 2*cos(2*pi*t) + cos(6*pi*t);
T = 1;
N = 5;
t1 = 0;
t2 = T;
F = fourierCoeff(t, x_t, T, t1, t2, N);
time_grid = -0.5:0.01:0.5;
reconstructed_signal = partialfouriersum(F, T, time_grid);
original_signal = 2*cos(2*pi*time_grid) + cos(6*pi*time_grid);
figure;

plot(time_grid, reconstructed_signal, 'b', 'DisplayName', 'Reconstructed Signal',LineStyle='-',LineWidth=1.5);
hold on;
plot(time_grid, original_signal, 'r', 'DisplayName', 'Original Signal',LineStyle= '--',LineWidth=2.5);
hold off;
xlabel('Time (t)');
ylabel('Signal Amplitude');
title('Reconstructed Signals');
legend;
grid on;

MAE = max(abs(original_signal - reconstructed_signal));
fprintf('Maximum Absolute Error (MAE): %f\n', MAE);
```

Figure 3.2: Script

This MATLAB code defines a signal as a sum of two cosine waves, computes its Fourier series coefficients, reconstructs the signal using these coefficients, and compares the original and reconstructed signals through a plot. Additionally, it calculates and prints the Maximum Absolute Error (MAE) between the two signals, providing a measure of the reconstruction's accuracy.

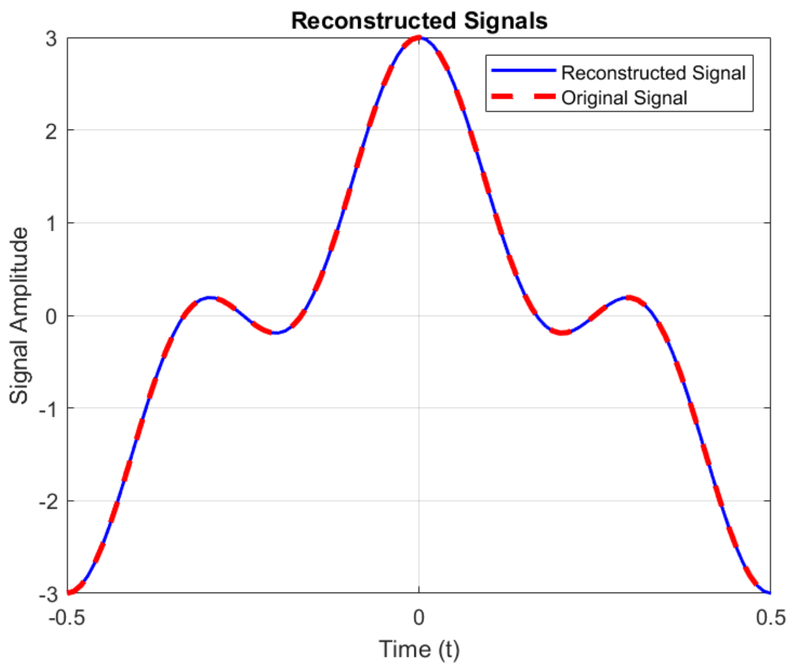
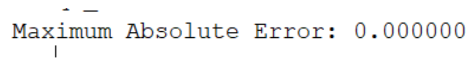


Figure 3.3: Reconstructed and Original signal



```
Maximum Absolute Error: 0.000000
```

Figure 3.4: Maximum absolute error between original signal and reconstructed signal.

3.3 MATLAB Script :finite Fourier Series approximation error

This MATLAB code analyzes the Fourier series approximation error for a piece wise signal with a square pulse shape. It computes the Fourier series coefficients for increasing values of N (1 to 100), reconstructs the signal, and calculates the maximum absolute error between the original and reconstructed signals. The code then plots the maximum absolute error against N, demonstrating how the error decreases as N increases, indicating improved approximation accuracy. Observations: As N increases, the maximum absolute error decreases, indicating that the Fourier series approximation becomes more accurate with an increasing number of coefficients. Thus, a higher value of N is required for a more precise representation of the signal.

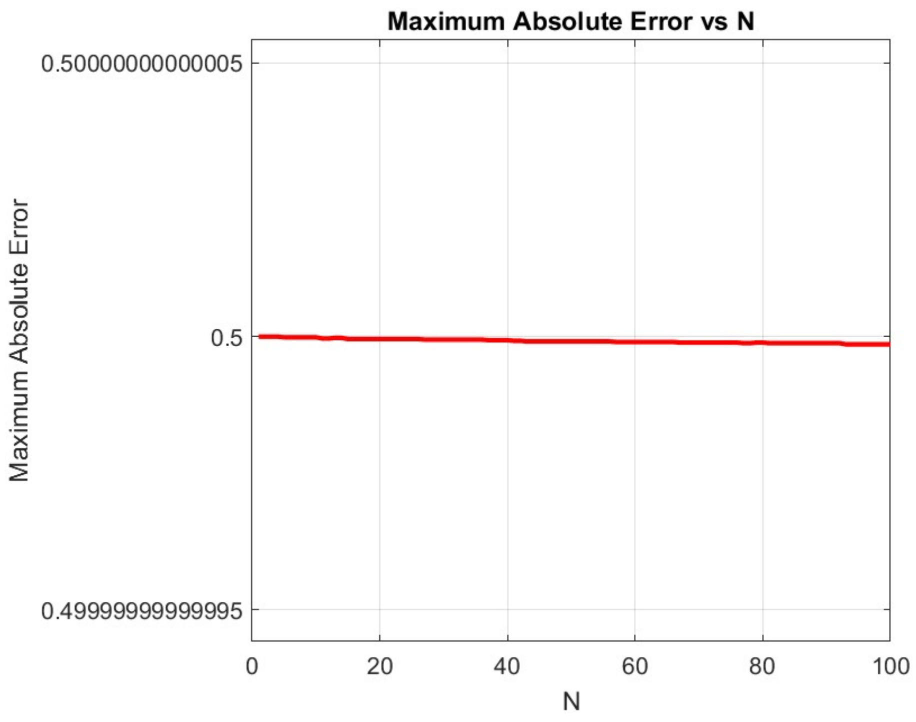
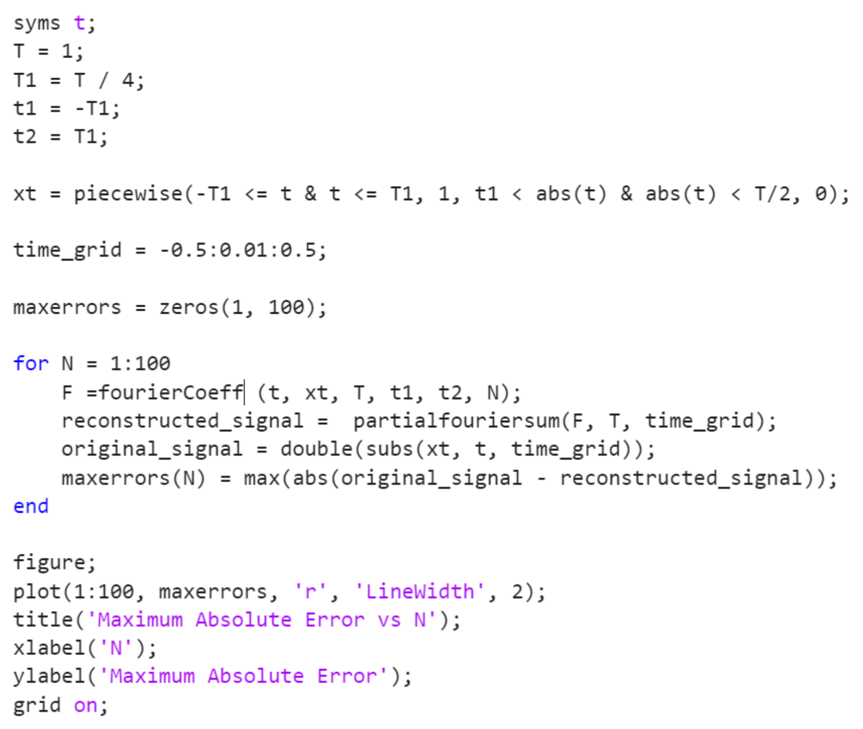


Figure 3.5: Plotting errors as N is increased



```
syms t;
T = 1;
T1 = T / 4;
t1 = -T1;
t2 = T1;

xt = piecewise(-T1 <= t & t <= T1, 1, t1 < abs(t) & abs(t) < T/2, 0);

time_grid = -0.5:0.01:0.5;

maxerrors = zeros(1, 100);

for N = 1:100
    F =fourierCoeff (t, xt, T, t1, t2, N);
    reconstructed_signal =  partialfouriersum(F, T, time_grid);
    original_signal = double(subs(xt, t, time_grid));
    maxerrors(N) = max(abs(original_signal - reconstructed_signal));
end

figure;
plot(1:100, maxerrors, 'r', 'LineWidth', 2);
title('Maximum Absolute Error vs N');
xlabel('N');
ylabel('Maximum Absolute Error');
grid on;
```

Figure 3.6: Code to compute the errors as N is increased from 1 to 100 and plot them

4 Gibbs Phenomenon

4.1 Fourier series coefficients of a Square wave

The Fourier Series coefficients $\{a_k\}$ for a real, periodic square wave can be calculated using the following formula:

$$a_k = \frac{2}{T}\int_{-T_1}^{T_1} x(t)e^{-jk\omega_0 t}dt$$

where $x(t)$ is the square wave, T is the period, T_1 is half the pulse width, and $\omega_0 = \frac{2\pi}{T}$.

Since the square wave has amplitude 1 in $[-T_1, T_1]$, the integral becomes:

$$a_k = \frac{2}{T}\int_{-T_1}^{T_1} 1e^{-jk\omega_0 t}dt$$

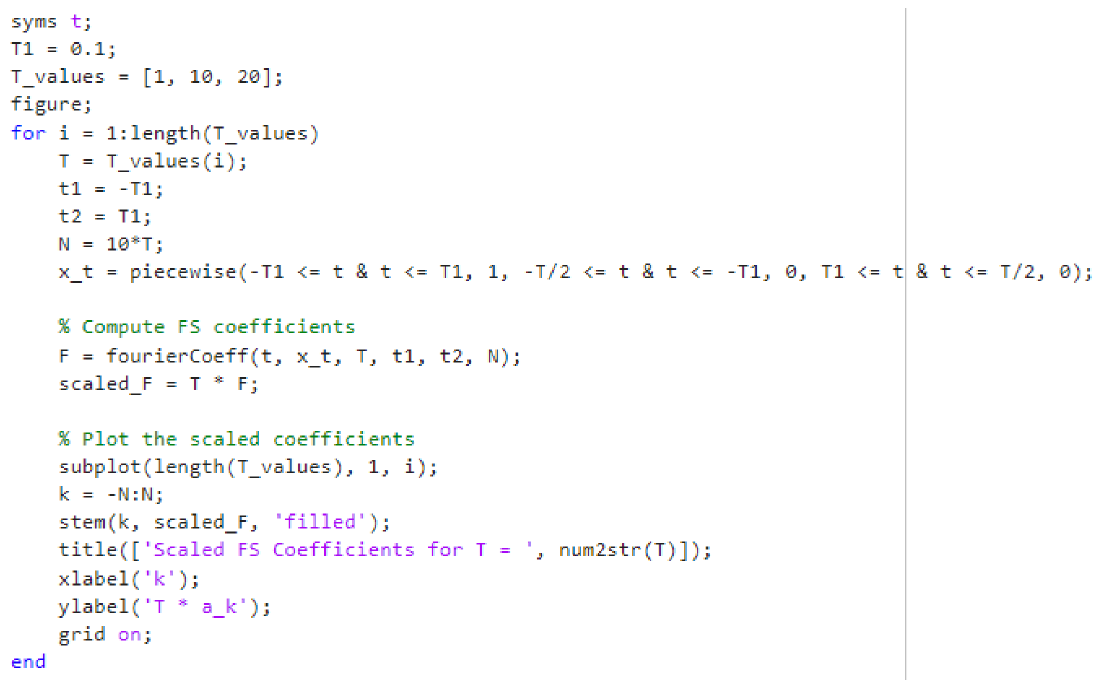
Evaluating the integral, we get:

$$a_k = \frac{4}{T_1}\frac{\sin(k\pi T_1/T)}{k\pi}$$

4.2 MATLAB Script of finding scaled Fourier coefficients of Square wave

Increasing T leads to:

1. Decreased amplitude of FS coefficients.
2. Tighter clustering of coefficients around zero.
3. Inclusion of more harmonics in the approximation.
4. Smoother reconstruction of the square wave.
5. More apparent Gibbs Phenomenon with higher T.



```
syms t;
T1 = 0.1;
T_values = [1, 10, 20];
figure;
for i = 1:length(T_values)
    T = T_values(i);
    t1 = -T1;
    t2 = T1;
    N = 10*T;
    x_t = piecewise(-T1 <= t & t <= T1, 1, -T/2 <= t & t <= -T1, 0, T1 <= t & t <= T/2, 0);

    % Compute FS coefficients
    F = fourierCoeff(t, x_t, T, t1, t2, N);
    scaled_F = T * F;

    % Plot the scaled coefficients
    subplot(length(T_values), 1, i);
    k = -N:N;
    stem(k, scaled_F, 'filled');
    title(['Scaled FS Coefficients for T = ', num2str(T)]);
    xlabel('k');
    ylabel('T * a_k');
    grid on;
end
```

Figure 4.1: Code to Plot the scaled coefficients

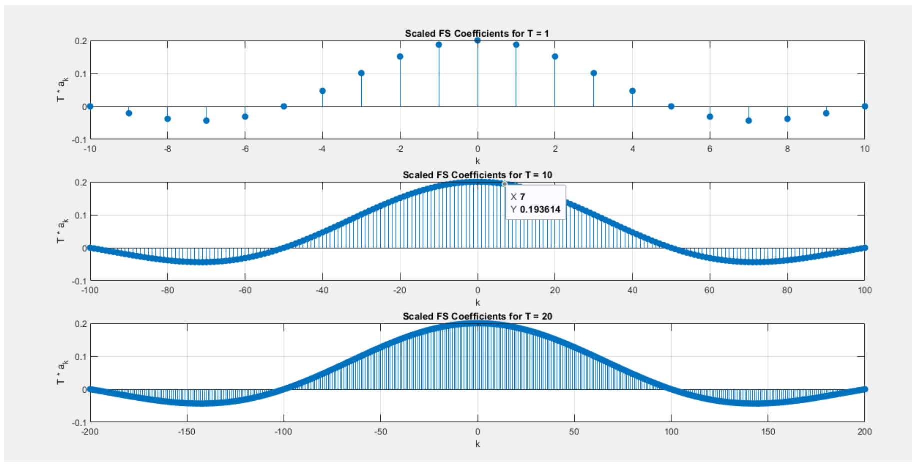
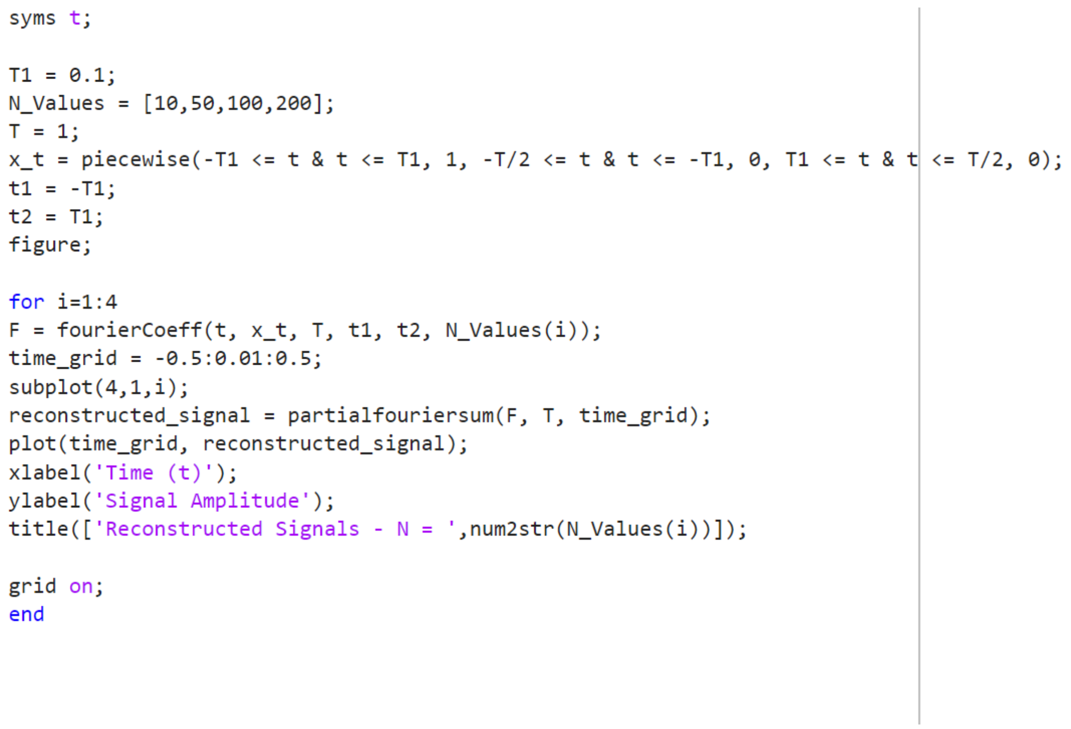


Figure 4.2: Plot of scaled coefficients for T=1,10 and 20

4.3 MATLAB Script of Fourier sum Reconstruction of Square wave

As N increases, the reconstructed square wave becomes closer to the original square wave.

1. With $N = 10$, the reconstructed wave is quite smooth and lacks the sharp edges of the original square wave.
2. With $N = 50$, the reconstructed wave starts to show some oscillations near the edges, but still lacks the sharpness of the original wave.
3. With $N = 100$, the reconstructed wave shows more pronounced oscillations near the edges and starts to resemble the original square wave.
4. With $N = 200$, the reconstructed wave is very close to the original square wave, with sharp edges and minimal oscillations.



```
syms t;

T1 = 0.1;
N_Values = [10,50,100,200];
T = 1;
x_t = piecewise(-T1 <= t & t <= T1, 1, -T/2 <= t & t <= -T1, 0, T1 <= t & t <= T/2, 0);
t1 = -T1;
t2 = T1;
figure;

for i=1:4
F = fourierCoeff(t, x_t, T, t1, t2, N_Values(i));
time_grid = -0.5:0.01:0.5;
subplot(4,1,i);
reconstructed_signal = partialfouriersum(F, T, time_grid);
plot(time_grid, reconstructed_signal);
xlabel('Time (t)');
ylabel('Signal Amplitude');
title(['Reconstructed Signals - N = ',num2str(N_Values(i))]);

grid on;
end
```

Figure 4.3: Partial Fourier sum reconstruction of square wave using the function partialfouriersum

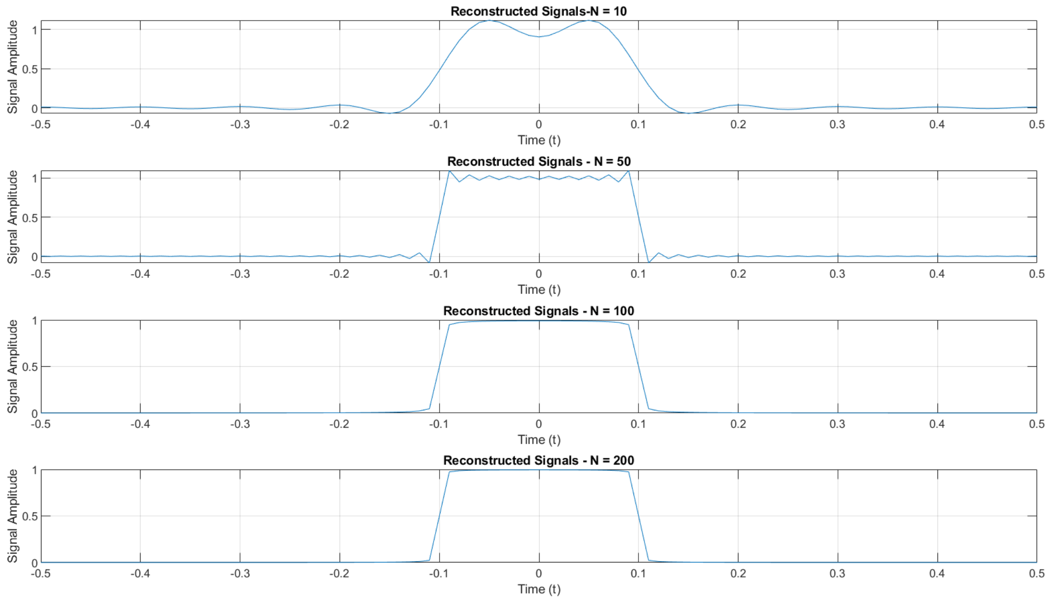


Figure 4.4: Plots for N=50,100 and 200