BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF ELECTRIC AND ELECTRICAL ENGINEERING

EE 391

SIGNALS AND SYSTEMS

Section 2

Mini Project 1

Ömer Oktay Gültekin

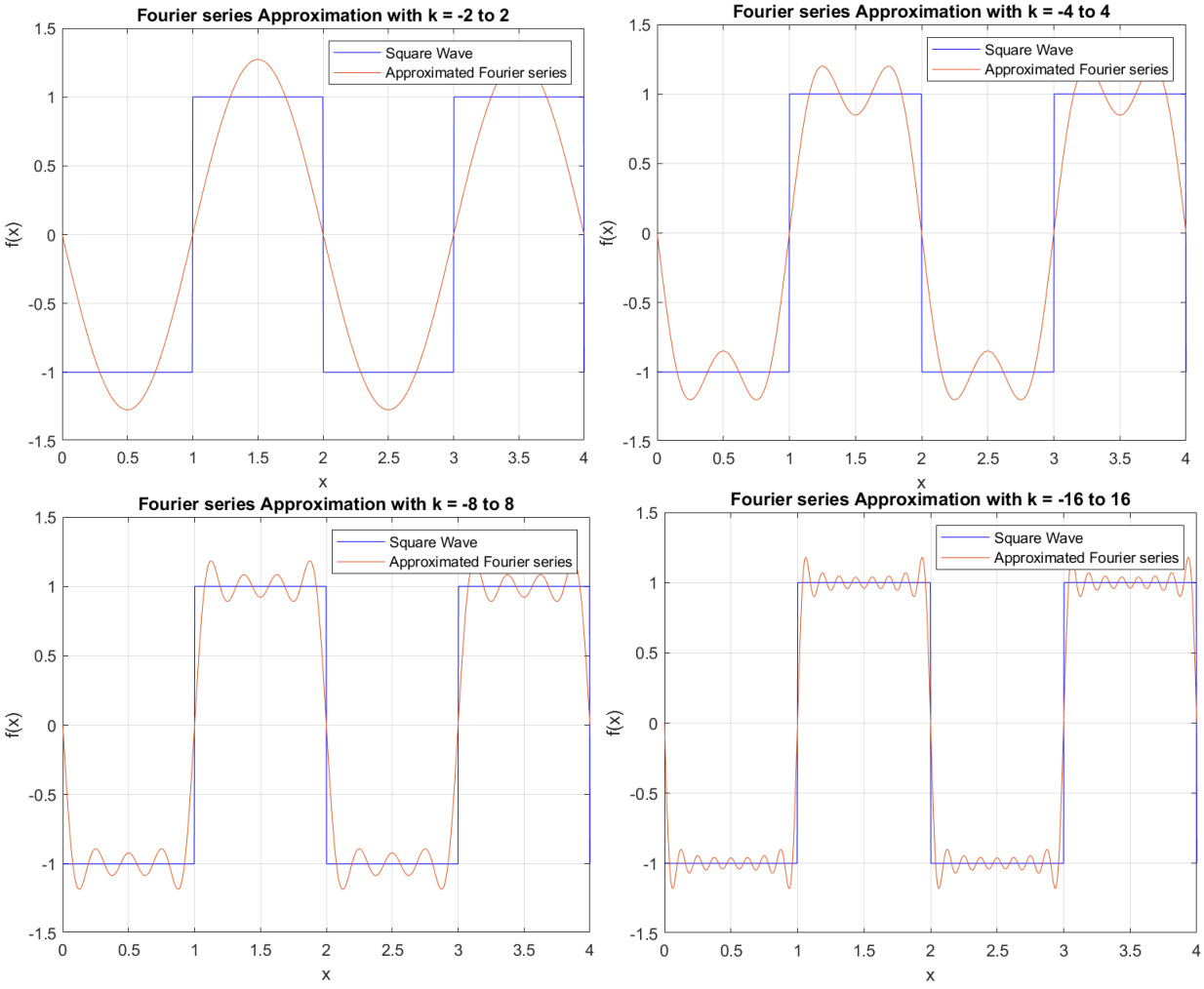
21901413

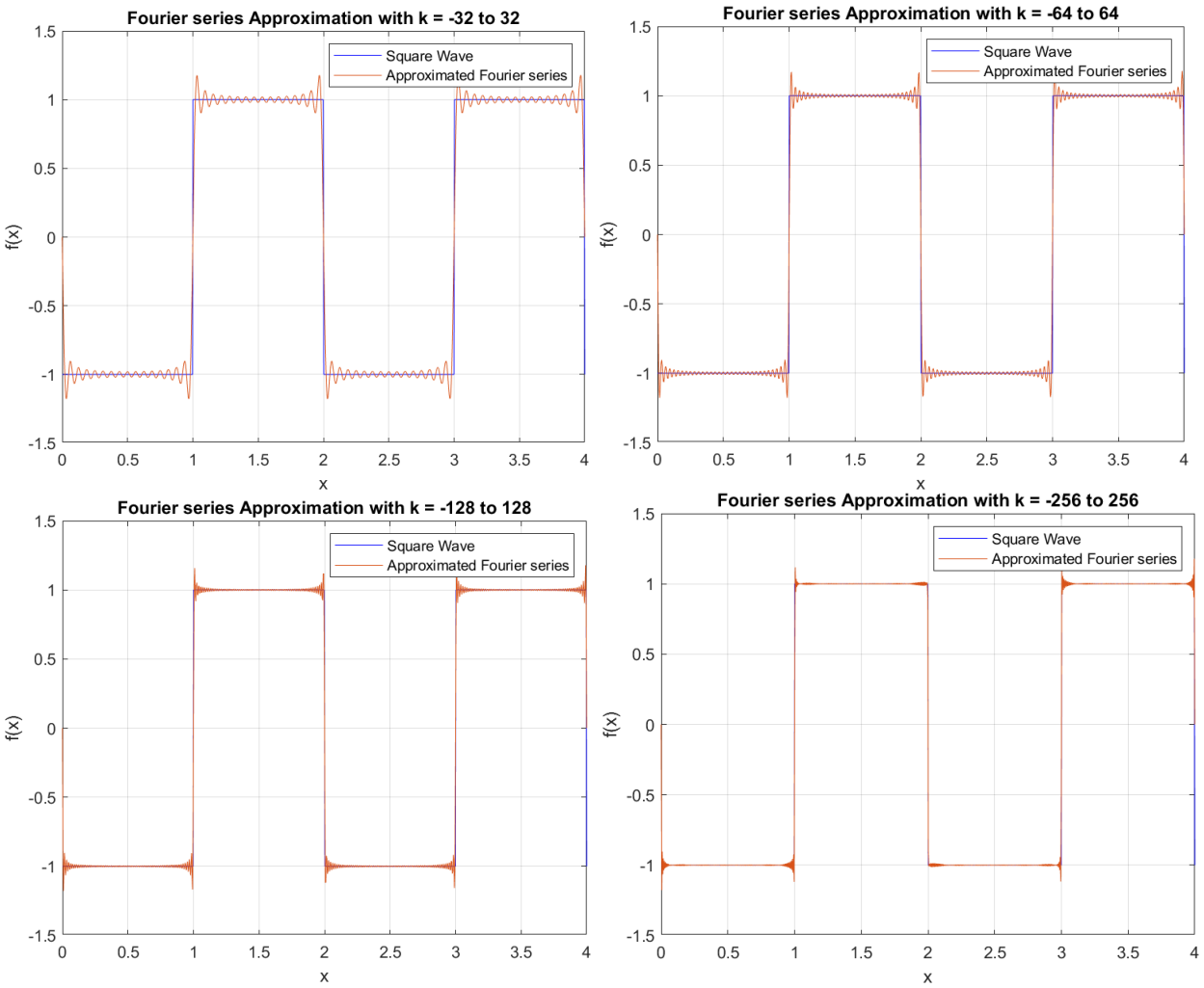
14.11.2023

1) Fourier Series Representation of Periodic Signals

PART A:

Plots:





As we can see on the figures, Fourier series approximation of the signal get closer to the signal while we are using more fourier coefficients to calculate. That is, the more k value increasing, the more approximated signal looks like the square wave signal.

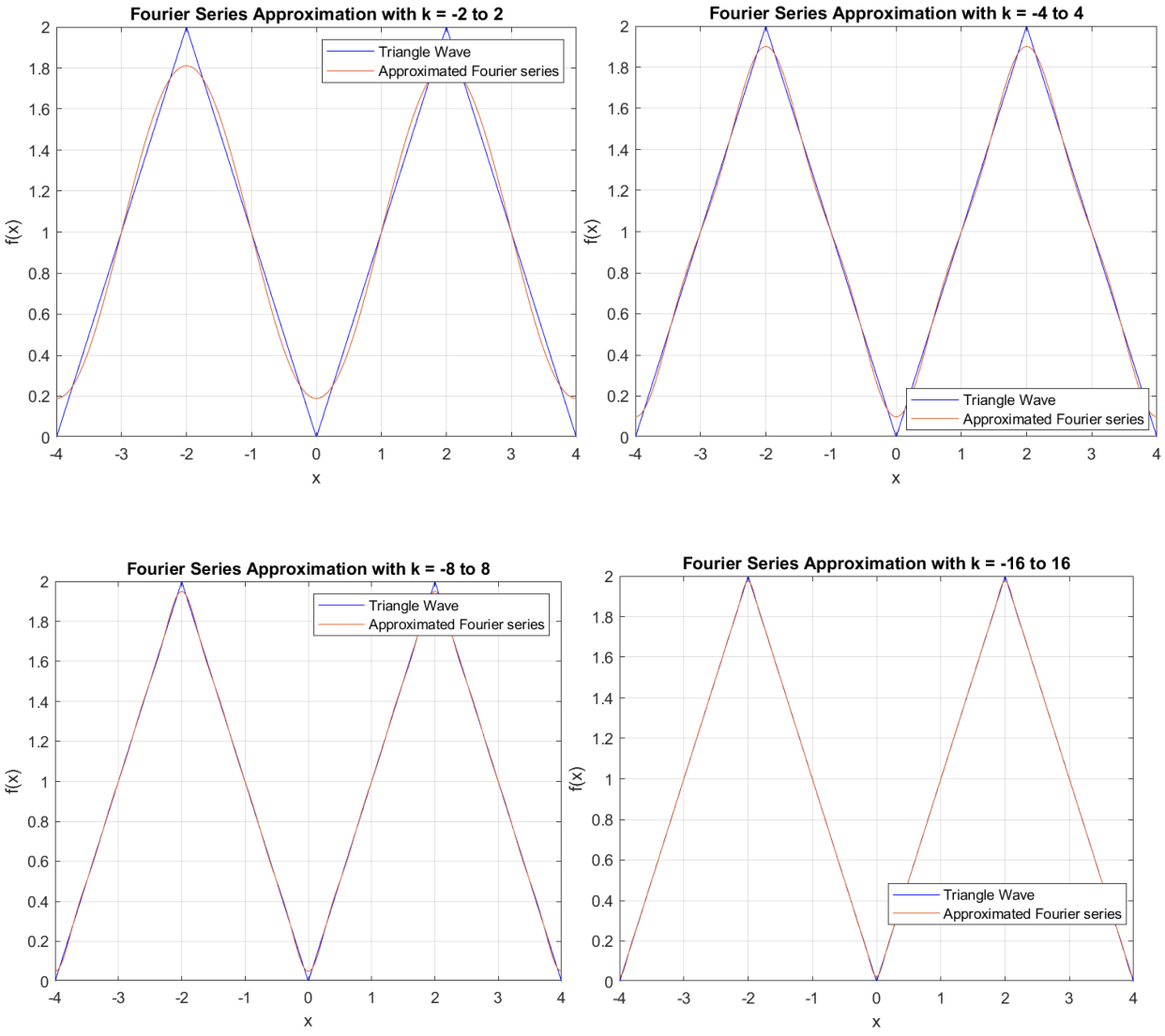
However, we should note that square wave alternates sharply between its high and low states, therefore, there is sharp corners. So the smoothness of the function is not that high, and we need to use more coefficients (higher k values) to approximate the signal.

Code:

```
T = 2;
t = linspace(0, 4, 1000);

shift = 0.50 * T;
x_t = zeros(size(t));
for i = 1:length(t)
    if mod(t(i) + shift, T) < T/2
        x_t(i) = 1;
    else
        x_t(i) = -1;
    end
end

coefficients_to_compute = [2, 4, 8, 16, 32, 64, 128, 256];

for i = 1:length(coefficients_to_compute)
    k_values = -coefficients_to_compute(i):coefficients_to_compute(i);

    % Initialize the Fourier series approximation
    appr_sig = zeros(size(t));

    ak = zeros(size(k_values));

    for j = 1:length(k_values)
        k = k_values(j);

        % Calculate Fourier coefficients using the integral formula
        a_k = (1/T) * (integral(@(t) 1 .* exp(-1j*2*pi*k*t/T), -T/2, 0) + ...
                       integral(@(t) -1 .* exp(-1j*2*pi*k*t/T), 0, T/2));

        fprintf('a_%d = %f + %fi\n', k, real(a_k), imag(a_k));

        % Sum the terms to reconstruct the Fourier series
        appr_sig = appr_sig + (a_k * exp(1j*2*pi*k*t/T));
    end

    figure;
    % Plot original one
    plot(t, x_t, 'b');
    hold on;
    % Plot the approximated one
    plot(t, appr_sig);
    title(['Fourier series Approximation with k = ', ...
        num2str(-coefficients_to_compute(i)), ' to ', ...
        num2str(coefficients_to_compute(i))]);
    xlabel('x');
    ylabel('f(x)');
    grid on;
    legend('Square Wave', 'Approximated Fourier series');
    hold off;
end
```

PART B:

Plots:



Again, the more a_k values we are using, the more the approximated signal looks like the triangular wave signal. However, since the triangle wave has a smoother, more gradual transition between its low and high states, less a_k value is needed to approximate the signal.

Note that, in the figures, k=[-16, 16] is enough to approximate the triangle signal but we couldn't get that much of closeness with even k=[-256, 256] in square wave approximation.

Code:

```
T = 4;
t = linspace(-4, 4, 1000);

x_t = zeros(size(t));
for i = 1:length(t)
    x_val = mod(t(i), 4);
    if x_val < 2
        x_t(i) = 2 - abs(x_val - 2);
    else
        x_t(i) = 2 - abs(2 - x_val);
    end
end

coefficients_to_compute = [2, 4, 8, 16];
for i = 1:length(coefficients_to_compute)
    k_values = -coefficients_to_compute(i):coefficients_to_compute(i);

    % Initialize the Fourier series approximation
    appr_sig = zeros(size(t));

    % Calculate the Fourier series coefficients
    for j = 1:length(k_values)
        k = k_values(j);

        % Calculate Fourier coefficients using the integral formula
        a_k = (1/T) * (integral(@(t) -t .* exp(-1j*2*pi*k*t/T), -T/2, 0) + ...
                       integral(@(t) t .* exp(-1j*2*pi*k*t/T), 0, T/2));

        fprintf('a_%d = %f + %fi\n', k, real(a_k), imag(a_k));

        % Sum the terms to reconstruct the Fourier series
        appr_sig = appr_sig + (a_k * exp(1j*2*pi*k*t/T));
    end

    figure;
    % Plot original one
    plot(t, x_t, 'b');
    hold on;

    % Plot the approximated one
    plot(t, appr_sig);
    title(['Fourier Series Approximation with k = ', ...
        num2str(-coefficients_to_compute(i)), ' to ', ...
        num2str(coefficients_to_compute(i))]);
    xlabel('x');
    ylabel('f(x)');
    grid on;
    legend('Triangle Wave', 'Approximated Fourier series');
    hold off;
end
```

2) A Real Life Example

a)

Closed form of the signals are:

signal1 = cos(2 x pi x 220 * 2^(10/12) * t8)
signal2 = cos(2 x pi x 220 * 2^(6/12) * t2)
signal3 = cos(2 x pi x 220 * 2^(8/12) * t8)
signal4 = cos(2 x pi x 220 * 2^(5/12) * t2)

b, c, d, e)

Code:

```
% Part (c): Define parameters
fs = 8000;
n1 = 2;
t8 = 1/fs:1/fs:n1/8;
t2 = 1/fs:1/fs:n1/2;
sd = zeros(1, round(length(t8)/10));
rest = zeros(1, length(t8));

% Part (b): Define the cosine signals (using the frequencies from part(b))

freq1 = 220 * 2^(10/12);
signal1 = cos(2 * pi * freq1 * t8);

freq2 = 220 * 2^(6/12);
signal2 = cos(2 * pi * freq2 * t2);

freq3 = 220 * 2^(8/12);
signal3 = cos(2 * pi * freq3 * t8);

freq4 = 220 * 2^(5/12);
signal4 = cos(2 * pi * freq4 * t2);

% Part (d): Create the array according to the specified order
combined_signal = [
    signal1, sd, signal1, sd, signal1, sd, ...
    signal2, sd, rest, sd, ...
    signal3, sd, signal3, sd, signal3, sd, signal4
];

% Part (e): Listen to the combination of notes
sound(combined_signal, fs);
```