(b)

clear all;

close all;

function a = FourierCoefficients(k)

    % computes the coefficients 'a' based on the values in vector 'k'.

    % For each element in 'k':

    %     - If the element is zero, the corresponding 'a' value is set to 1/2.

    %     - Otherwise, the value is calculated as sin(k\*pi/2)/(k\*pi).

    a = zeros(size(k));

    for idx = 1:length(k)

        if k(idx) == 0

            a(idx) = 1/2;

        else

            a(idx) = sin(k(idx) \* pi/2) / (k(idx) \* pi);

        end

    end

end

approximations = [1, 3, 7, 19, 43, 79];

maximum\_approx = max(approximations);

k\_values = -maximum\_approx: maximum\_approx;

ak = FourierCoefficients(k\_values);

N = 400; %This will determine the number of points in the time domain

t = linspace(-2\*5, 2\*5, 4\*N\*5); %t stands for the time domain, 4\*N\*5 points in total

%Square Wave Signal

y\_single = [zeros(1, N) ones(1, 2\*N) zeros(1, N)];

y = [y\_single y\_single y\_single y\_single y\_single];

% plotFourier Plots the Fourier series approximation of a signal and prints the overshoot

%   Inputs:

%       j               - Index for the current approximation order.

%       t               - Time vector for plotting.

%       approximations  - Vector containing the number of terms for each approximation.

%       maximum\_approx  - Maximum order of approximation (used for indexing).

%       ak              - Vector of Fourier coefficients.

%       y               - Original signal to be approximated.

%

%   The function creates a new figure, plots the original signal in white,

%   overlays the Fourier approximation in green, and displays the

%   approximation order in the title. It also prints the maximum and

%   minimum values of the approximation to the command window.

function plotFourier (j, t, approximations, maximum\_approx, ak, y)

    figure();

    plot(t, y, 'w');

    hold on;

    axis([-8 8 -0.2 1.2]);

    grid on;

    x = 0.\*t; % Initialize x to zero for the current approximation

    for k = -approximations(j):approximations(j) % Loop through the range of k values for the current approximation

        x = x + real(ak(k + maximum\_approx + 1) \* exp(1i \* k \* pi/2 .\* t));

    end

    plot(t, x,'g-');

    title(['N =', num2str(approximations(j))]);

    hold off;

    fprintf("maximum value for %d: %f\n", approximations(j), max(x));

    %Determine the percentage overshoot of the approximated square pulse for each of the above partial sum approximations

    overshoot = (max(x) - 1) / 1 \* 100; % Calculate the percentage overshoot

    fprintf("Percentage overshoot for N = %d: %.2f%%\n\n", approximations(j), overshoot);

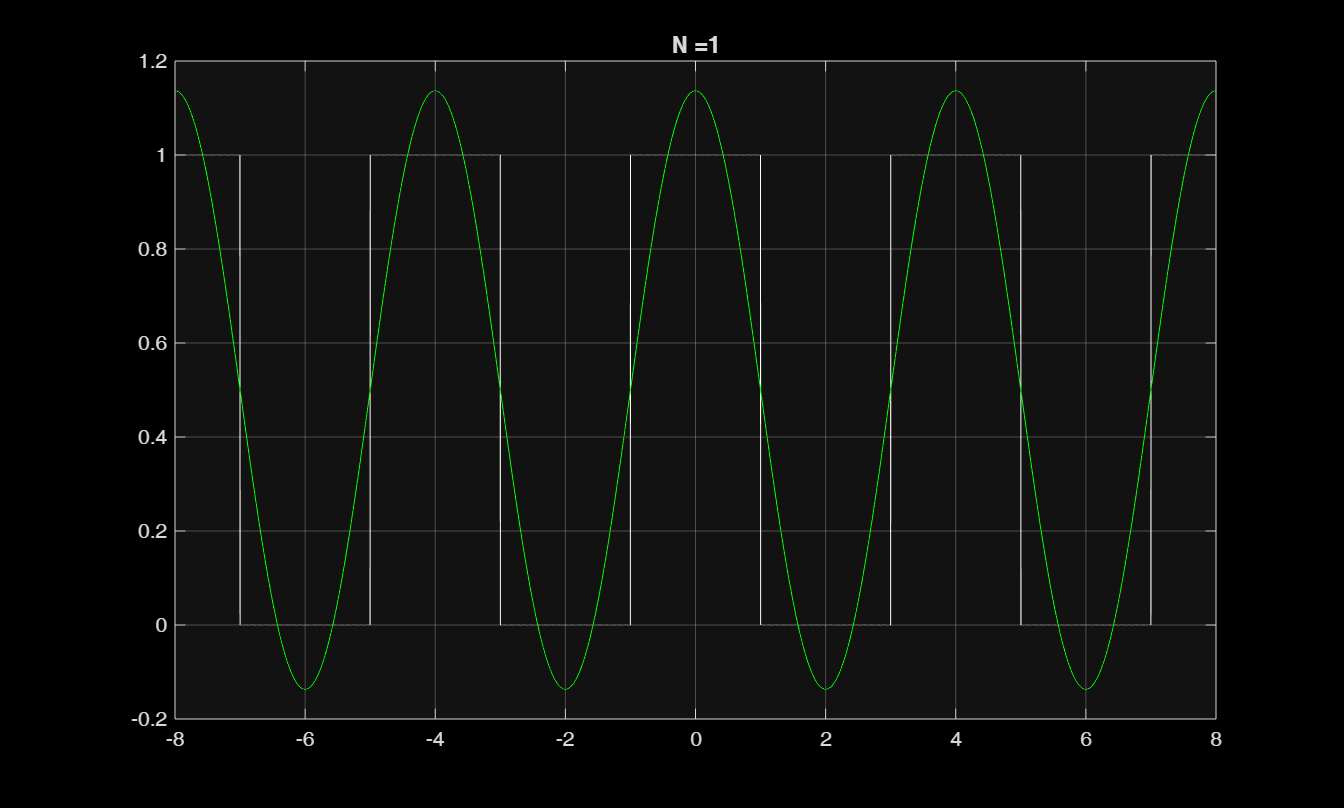
end

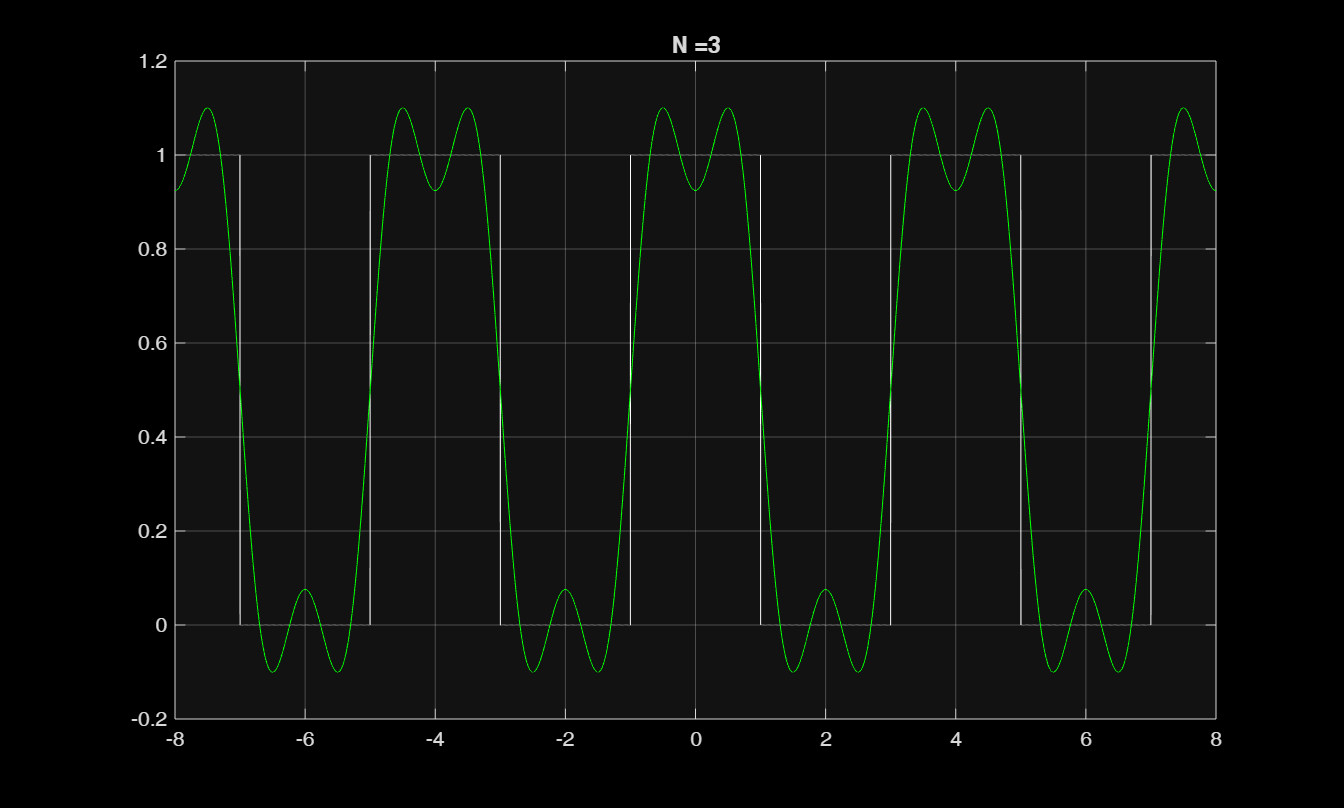
%Creating Graphs for all approximations

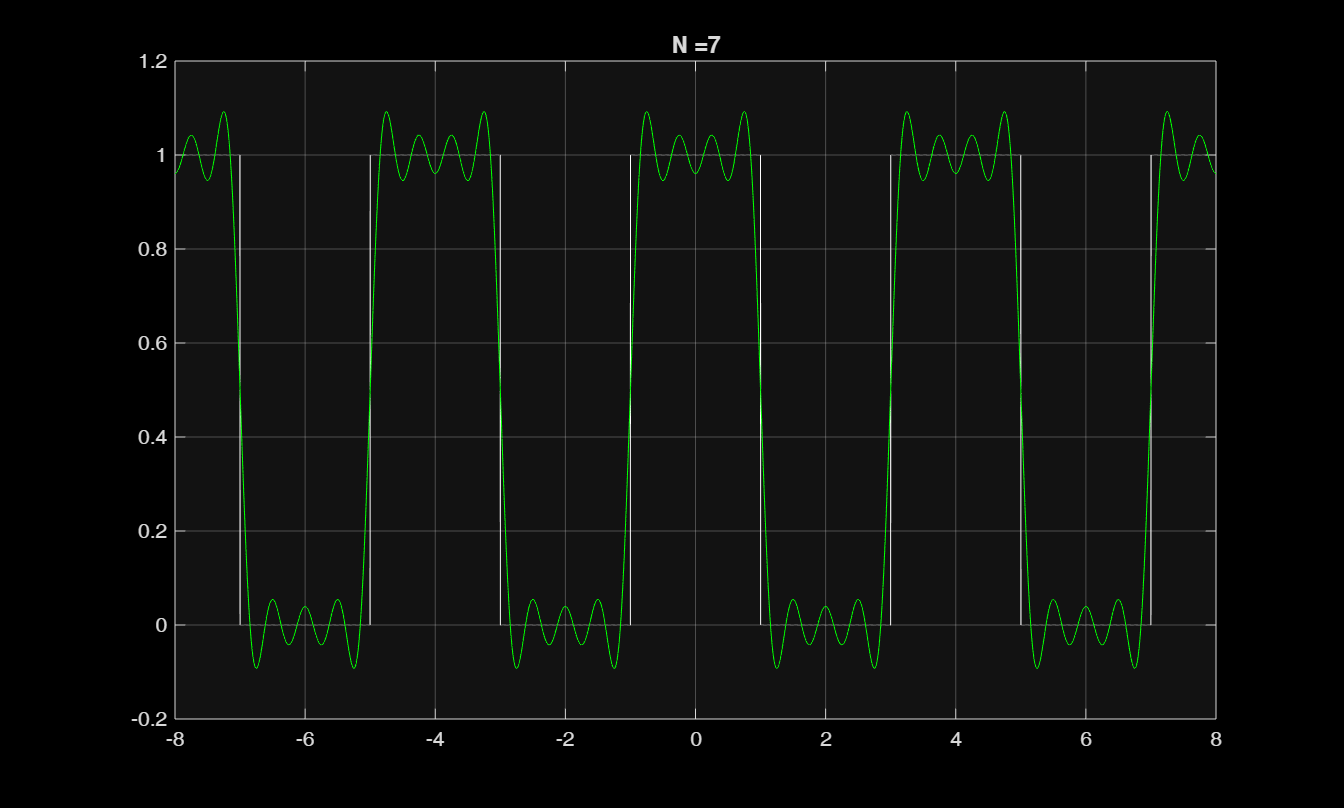
for j = 1: length(approximations)

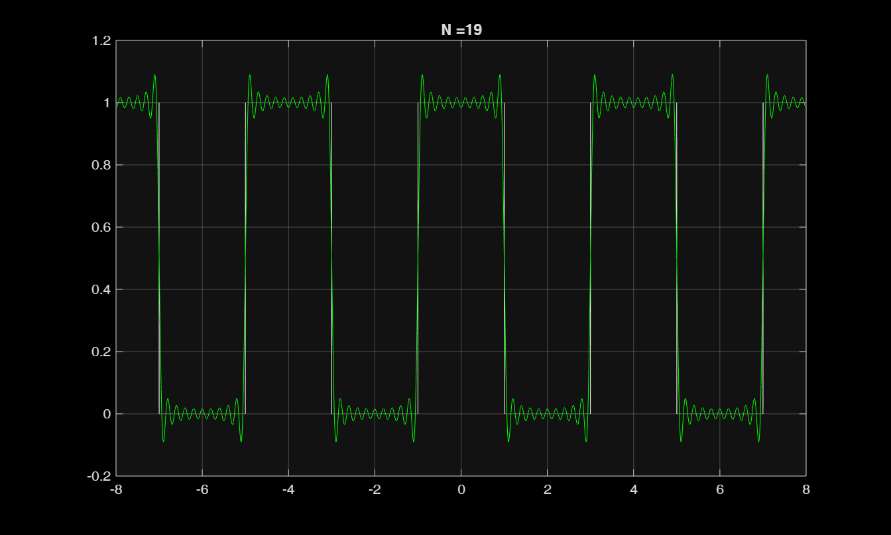
    plotFourier (j , t, approximations, maximum\_approx, ak, y);

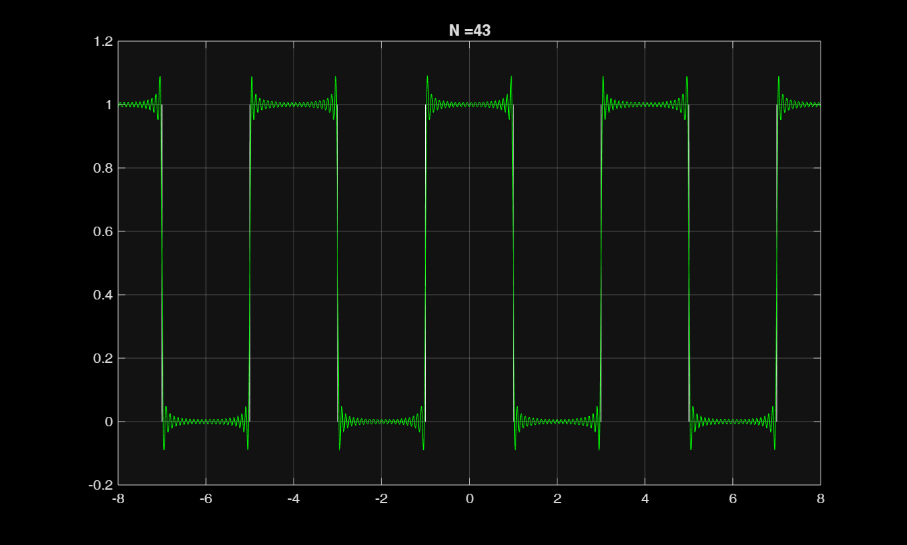
end

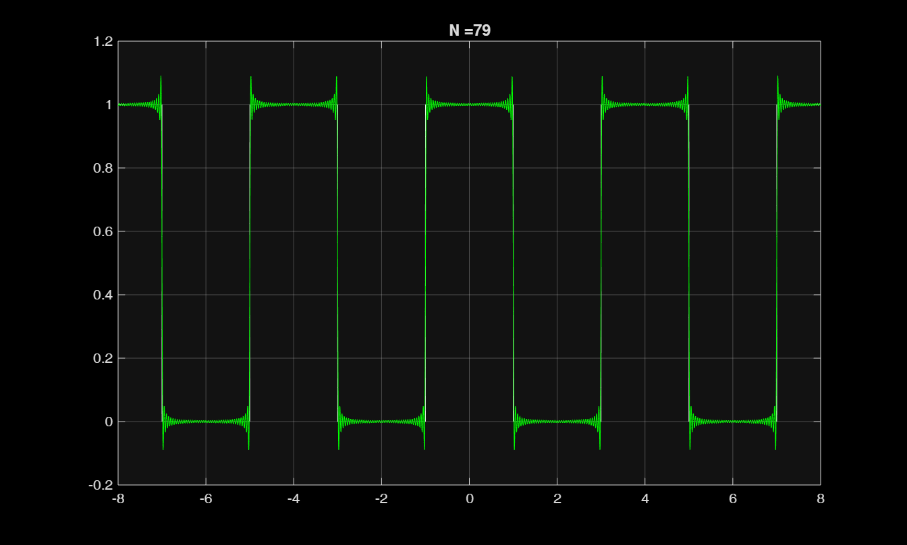












(c)

Formula used :

maximum value for 1: 1.136620

Percentage overshoot for N = 1: 13.66%

maximum value for 3: 1.100211

Percentage overshoot for N = 3: 10.02%

maximum value for 7: 1.092112

Percentage overshoot for N = 7: 9.21%

maximum value for 19: 1.089906

Percentage overshoot for N = 19: 8.99%

maximum value for 43: 1.089568

Percentage overshoot for N = 43: 8.96%

maximum value for 79: 1.089503

Percentage overshoot for N = 79: 8.95%