Fourier Series and MATLAB

PART 1: Solve A.1 – B.4 (Lab 12)

PART 2: Solve B.5 – C (Lab 13)

A Fourier series is an expansion of a periodic function f(x) in terms of an infinite weighted sum of sines and cosines. Since computers cannot handle an infinite number of terms, the implementation of the Fourier series on computers is an approximation of f(x). Mathematically, the Fourier series can be written as shown in Equation (1), where a_0, a_n and b_n are the Fourier series coefficients.

$$f(x) = \frac{1}{2} a_0 + \sum_{n=1}^{\infty} a_n \cos(n x) + \sum_{n=1}^{\infty} b_n \sin(n x), \tag{1}$$

where,

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\, dx$$
$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(n x)\, dx$$
$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(n x)\, dx$$

Part A: Introductory example

The following shows how a sum of odd harmonics can be used to generate a square wave signal.

Run each section (use *Run section*. Do not use *Run*) of the code below in MATLAB and answer the questions.

```
%% Plot the fundamental frequency and hear associated sound
t = 0:.1:100;
y = sin(t);
plot(t,y);
sound(y)

%% Next add the third harmonic to the fundamental, and plot/hear it.
y = sin(t) + sin(3*t)/3;
plot(t,y);
sound(y)
```

```
%% Now use the first, third, fifth, seventh, and ninth harmonics.
y = sin(t) + sin(3*t)/3 + sin(5*t)/5 + sin(7*t)/7 + sin(9*t)/9;
plot(t,y);
sound(y)
```

A.1 What happens to the overall signal when harmonics are added?
Sounds higher in pitch and there are more oscillations at the peaks and valleys

A.2 Observe each plot, listen to the associated sound and explain the relationship you notice between plot and corresponding sound.

Same. More oscillations at valleys and peaks and sounds higher pitched.

A.3 Change `t` to `t = 0:.1:1000;` for you to be able to hear each sound for a longer period. Run each section again and compare the different sounds.

Part B: Approximation of a square wave using the Fourier series

The following MATLAB code can be used to plot a square wave:

```
% CODE 1
% Ideal square wave
t=-10:0.0001:10;
x=0.5*(square(t)+1);
plot(t,x)
```

By means of a weighted sum of sines and cosines, let us try to approximate the square wave plot produced by the code above. To do so, the following MATLAB code can be used.

```
% CODE 2
clc, syms n x
L = pi; N_TERMS = input('Number of terms: '); n = 1:N_TERMS;

% Calculating the coefficients of the Fourier Series
% tic
a0 = (1/L)*int(1, x, 0, pi);
an = (1/L)*int(1*cos(n*x), x, 0, pi);
bn = (1/L)*int(1*sin(n*x), x, 0, pi);

% Plugging the coefficient values in the Fourier Series
f = 0;
for n = 1:N_TERMS
f = f+ (an(n)*cos(n*x) + bn(n)*sin(n*x));
end
% toc

fprintf('Approximation using %d terms\n', n)
f_approx = (a0/2)+ f
ezplot(f_approx, [-10,10])
title('Fourier Series Approximation')
```

```
xlabel('x')
ylabel('F(x) approximated')
```

B.1 Write a brief description of how `CODE 2` implements the Fourier series described by Equation (1).
Take N input, calculate each *f* for N separately, and *f(1) -> f(N)* for the final output

B.2 Using the command `hold on`, plot the graph produced by `CODE 1` and `CODE 2` on the same picture. Consider the number of terms `N_TERMS = 7`. Insert a legend on your plot describing the two signals: name the ideal square wave f(x) and the approximation f'(x).

B.3 Run the code that results from B.2 for `N_TERMS = 10,20 and 30` without closing the Figure each time you run. This way you will have multiple plots on Figure 1. Include the plot on your report and answer the following question: considering the ideal square wave f(x) as a reference, where do the approximations of f(x) have the greatest error/oscillation?
Beginning and end of peaks and valleys

B.4 The Gibbs phenomenon is an *overshoot* (or "ringing") of the Fourier series. Defining the overshoot as the difference in amplitude between the highest point of the approximation and the reference function, record the overshoot values in Table 1 associated with each of the following number of terms. What is the relationship between the overshoot and the number of terms in the series?

Table 1 – Overshoot for different number of terms of the FS.

Number of terms	Overshoot
7	0.09211
20	0.08919
30	0.08345
50	0.08238

B.5 Run `CODE 2` multiple times (for `N_TERMS = 7,20,30,50 and 100.`) and, using the command tic toc, record in Table 2 the time MATLAB takes to perform the approximation of f(x). Notice that you can *uncomment* `% tic` and `% toc` to perform the required task.

Table 2 – Computational time for different number of terms of the FS.

Number of terms	Computational time
7	0.04635
20	0.102084
30	0.153692
50	0.222204

100	36.175.064

B.6 From the previous problems it could be noticed that, the higher number of terms of the series is, more precise the approximation will be; however, the computational time cost increases. In practice, we must find a *balance* between precision and computational cost when using Fourier series. What determines this balance? Accuracy / comp.time

Part C: Approximation of a triangular wave using the Fourier series

`CODE 2` implements the approximation of a square wave using Fourier series. Modify this code so that it will produce the approximation of a triangular wave (more specifically, a sawtooth wave) similar to the one shown in Figure 1. In your report, include, on the same figure, plots of the approximated sawtooth function considering `N_TERMS = 10,20 and 30.`

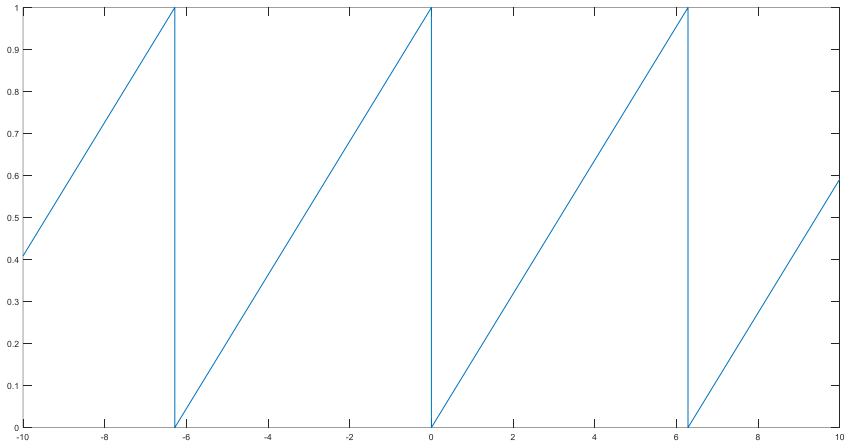

Figure 1 – Sawtooth wave.

Hint: Compare the Fourier series coefficients in `CODE 2` with the coefficient formulas after Equation (1) and figure out which function f(x) you should use. Studying the MATLAB command *int* might also be helpful.