

# Fourier Series

## Laboratory II

Patrycja Nazim, Adrian Król, Kamil Chaj

### 1 Aim of the exercise

The purpose of exercise was to experimentally familiarize with the Fourier series - an operation that allows to represent any real periodic signal by the sum of sinusoidal signals. We measured the signals generated by the independent function generator with a vector spectrum analyzer.

### 2 What is Fourier Series

A Fourier series is an expansion of a periodic function  $f(x)$  in terms of an infinite sum of sines and cosines. Fourier series make use of the orthogonal relationships of the sine and cosine functions. The computation and study of Fourier series is known as harmonic analysis and is extremely useful to break up an arbitrary periodic function into a set of simple terms that can be plugged in, solved individually, and then recombined to obtain the solution to the original problem or an approximation to it to whatever accuracy is desired or practical

### 3 Course of measurements

During our measurements we skipped first part of exercise and measured all signals in configuration for second part (Fig. 1).

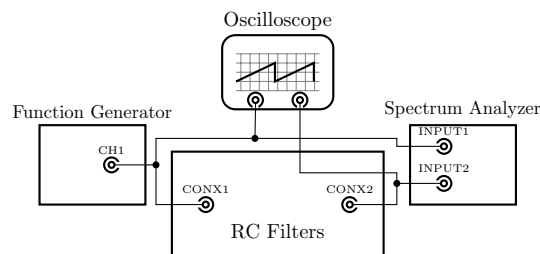


Figure 1: Measurements configuration

After connecting Function Generator, Spectrum Analyzer, Oscilloscope and board with RC filters according to above configuration (Fig. 1) we set Function Generator to Amplitude to  $V_{pp} = 5V$  and Frequency 1 kHz. With everything set up we proceeded with exercise and recording output of Oscilloscope and Spectrum Analyzer for signals:

- Sin wave
- Square wave 50% duty cycle
- Square wave 25% duty cycle
- Triangle wave 50% symmetry ratio
- Triangle wave 40% symmetry ratio

For Square wave 50% duty cycle and Triangle wave 40% symmetry ratio we also recorded Response for both RC Circuits (Fig. 2)



Figure 2: Measured RC Circuits

## 4 Method for calculating Fourier Series coefficients

All calculations are made using Matlab with Signal Processing Toolbox, source code can be found in Appendix A

First step in our calculation is generating signals that we measured during exercise. Pure signals were obtained with built-in functions of Signal Processing Toolbox and Filtered signals were evaluated using NI Multisim SPICE software.

Next step is finding first 10 coefficients of Fourier series for a signal which we calculated by taking inner product of signal with basis vector. In our calculations we used  $\sin nt$ ,  $\cos nt$  and 1 as our basis vectors.

$$a_0 = \langle s(t), 1 \rangle \quad (1)$$

$$a_n = \langle s(t), \cos n\omega t \rangle \quad (2)$$

$$b_n = \langle s(t), \sin n\omega t \rangle \quad (3)$$

During exercise we recorded Fourier series of a signal in Amplitude-Phase form so we need to convert  $a_n$  and  $b_n$  coefficients into amplitude  $A_n$  and phase  $\varphi_n$

$$A_n = \sqrt{a_n^2 + b_n^2} \quad (4)$$

$$\varphi_n = \arg(a_n - jb_n) \quad (5)$$

Knowing first 10 coefficients we can plot approximation of a signal using one of following formulas. Example of approximation of square wave is on the second plot in Figure 3.

$$s(t) \approx \frac{a_0}{2} + \sum_{n=1}^{10} (a_n \cos(n\omega t) + b_n \sin(n\omega t)) \quad (6)$$

$$s(t) \approx \frac{A_0}{2} + \sum_{n=1}^{10} (A_n \cos(n\omega t + \varphi_n)) \quad (7)$$

For third and fourth plot in Figure 3 we are calculating Discrete Fourier transform for 10 oscillations of a signal using Fast Fourier transform algorithm from Signal Processing Toolbox and then plotting magnitude and argument of FT both in decibel scale on vertical axis.

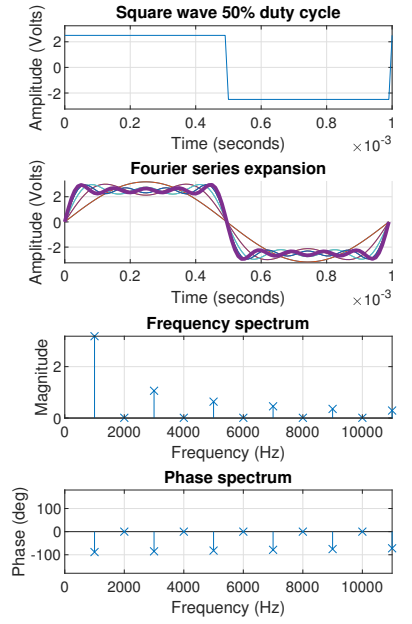


Figure 3: example plot

## 5 Comparison

We are going to be comparing calculated and measured signals, their Fourier series expansion and/or Fourier transform. During comparison we will determine if Harmonic analysis is useful in signal analysis and what benefits it brings.

\* Yellow signal on oscilloscope is pure signal, Blue is filtered signal

## 5.1 Pure signals

Starting off with first signal, Sinus, we can see that  $A_n$  coefficient for 1 kHz is approximately equal to amplitude of a signal and rest of coefficients are very small almost negligible, phase coefficient  $\varphi_n$  at 1 kHz is approximately  $-90^\circ$  therefore according to eq.(7) first element of Fourier series is just  $A_1 \sin \omega t$  and the rest of elements are very small, negligible.

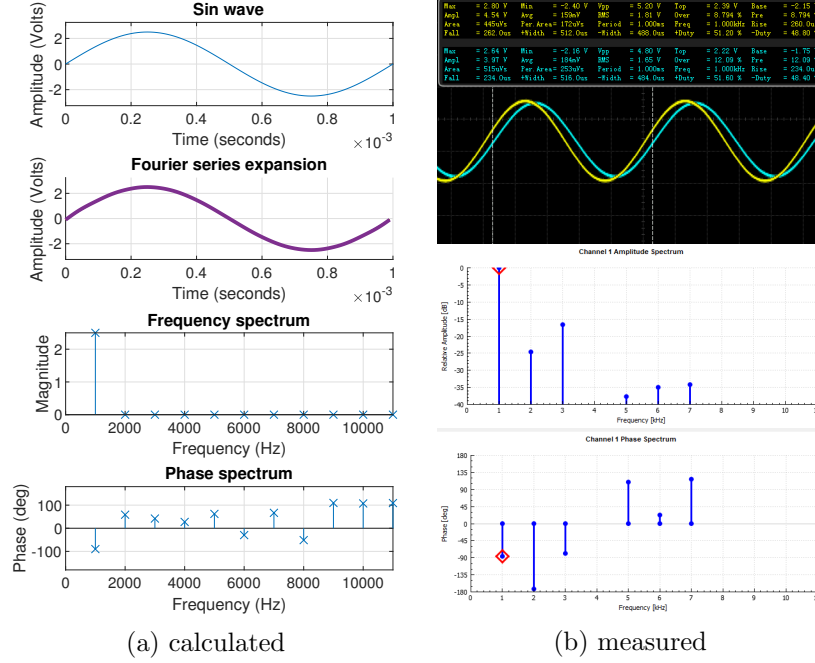


Figure 4: Sin wave

Now moving to more interesting signal than simple sin, square wave we can see many more components of Fourier series that are not negligible. Starting with first element which is exactly the same as in sin signal we can deduce that square signal have similar shape to sinus and is mostly odd function, but looking further on the rest of coefficient we can see that all of them are shifted by  $-90^\circ$ , so square signal is built entirely from sins therefore is completely odd.

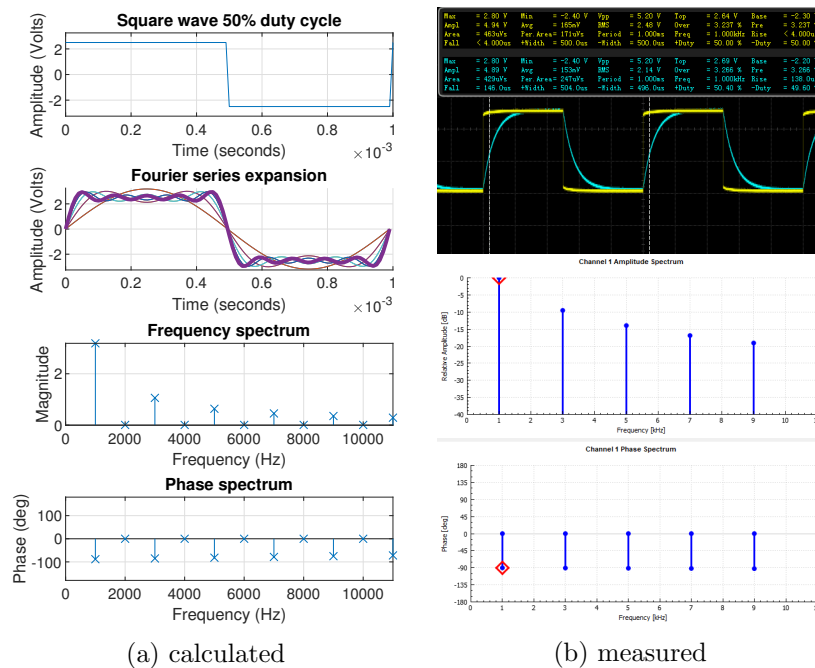


Figure 5: Square wave 50% duty cycle

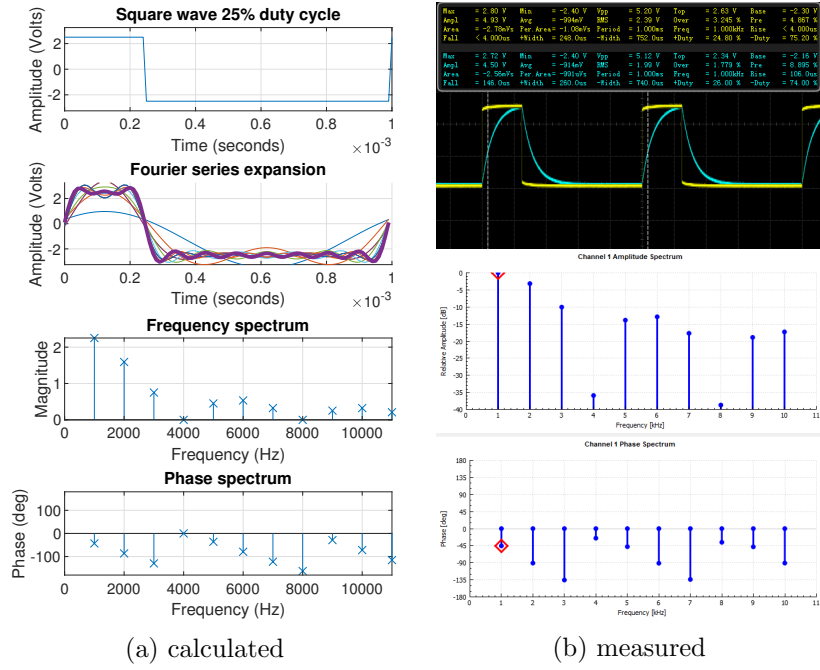


Figure 6: Square wave 25% duty cycle

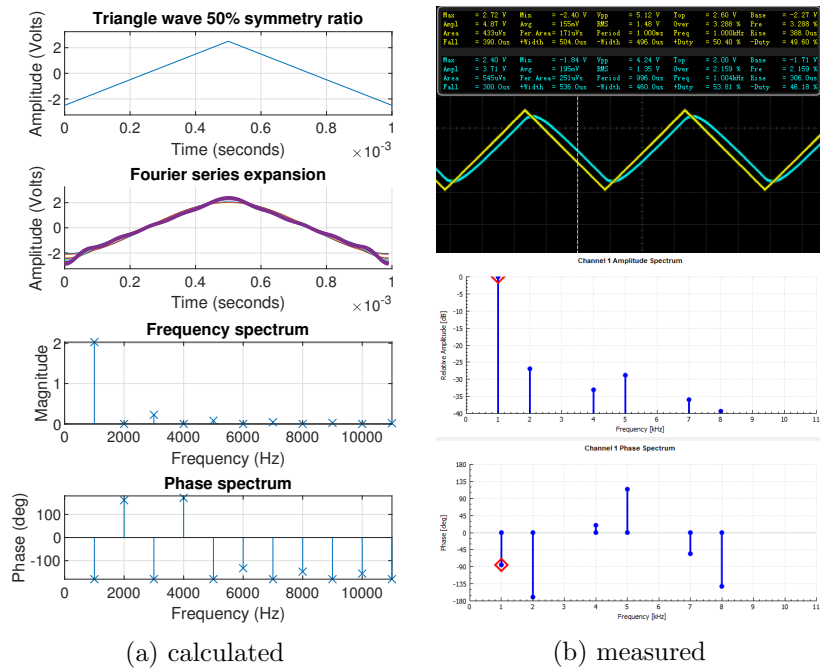


Figure 7: Triangle wave 50% symmetry ratio

## 5.2 RC filters

### 5.2.1 Low-pass filter

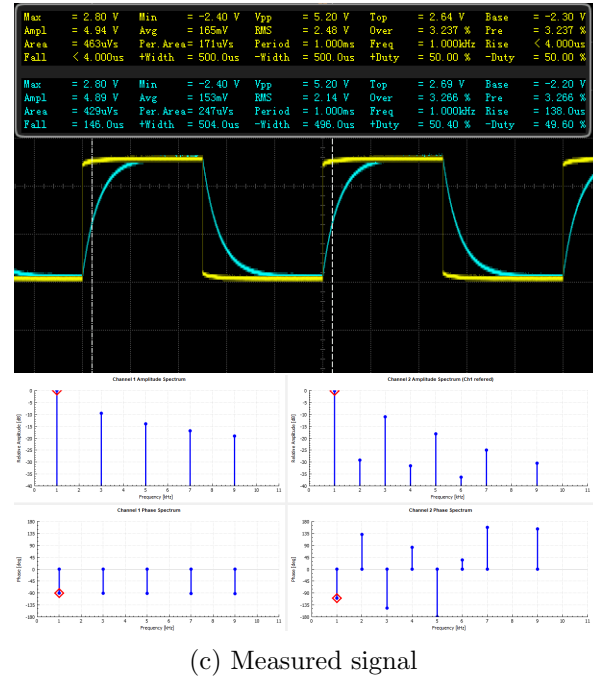
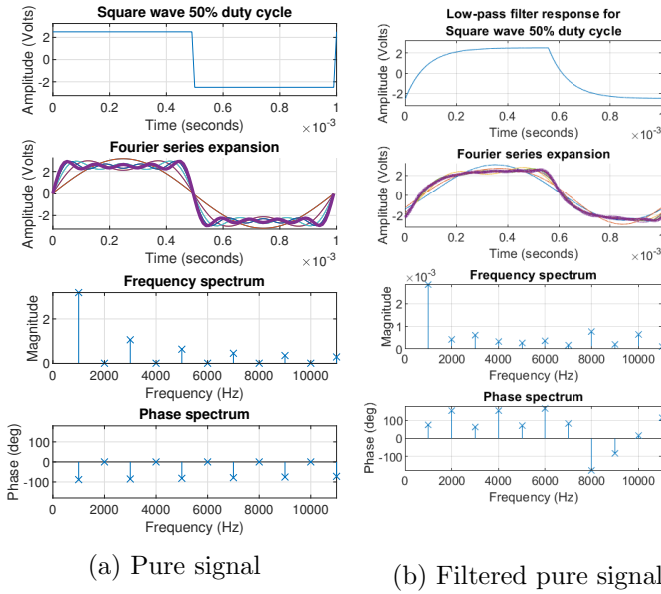


Figure 8: Low-pass filter response for Square wave 50% duty cycle

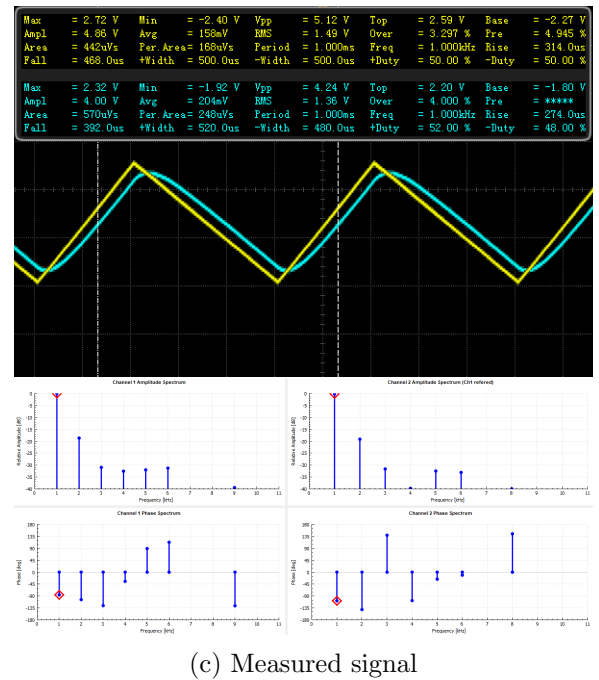
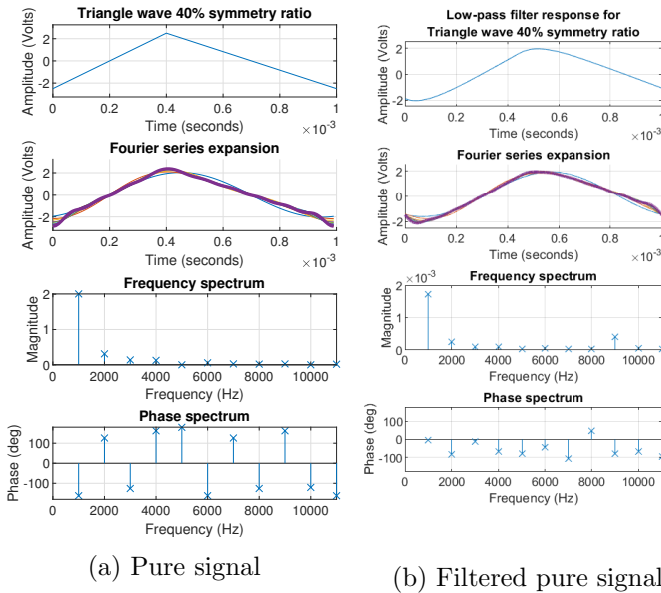


Figure 9: Low-pass filter response for Triangle wave 40% symmetry ratio

## 5.2.2 High-pass filter

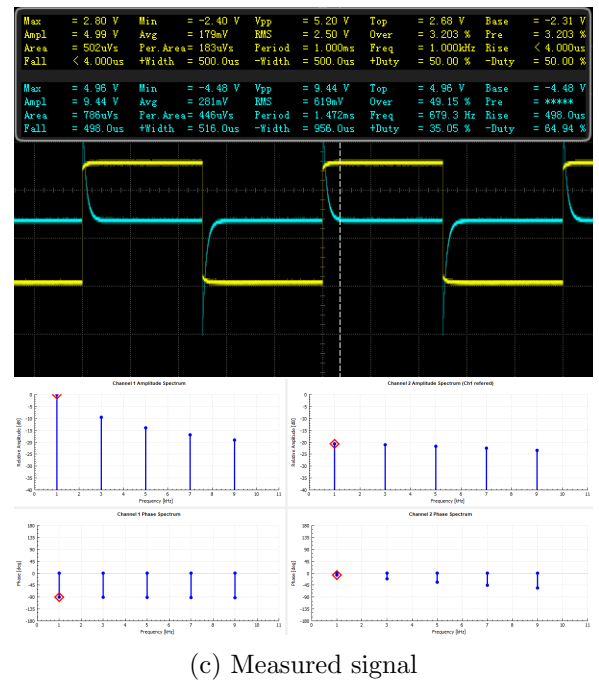
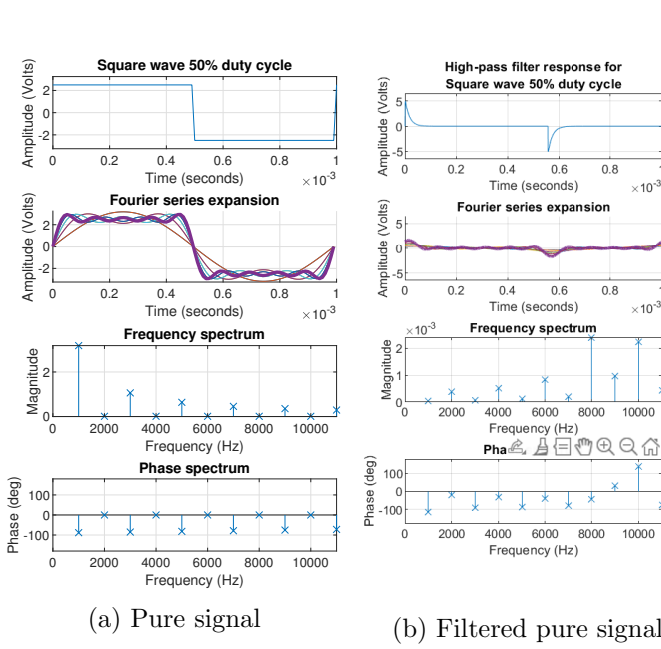


Figure 10: Low-pass filter response for Square wave 50% duty cycle

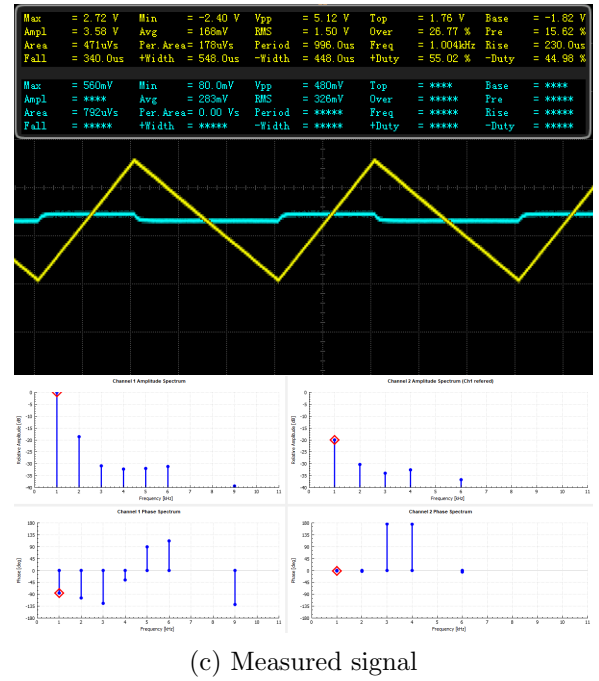
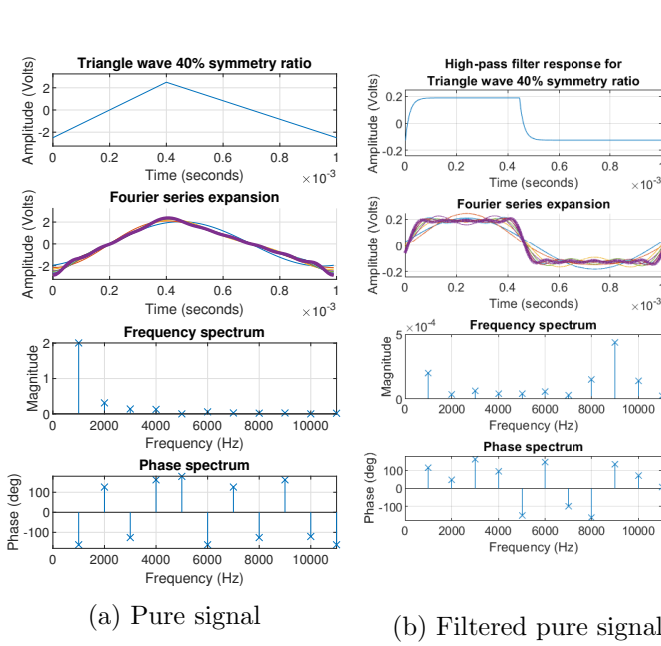


Figure 11: Low-pass filter response for Square wave 50% duty cycle

## 6 Summary

## A Source Code

[GITHUB repository](#)

### lab2.m

```
clear all; close all; clc;
[Square50_integrator, Square50_differentiator, Triangle40_integrator, Triangle40_differentiator] = ...

fs = 100000;
duration = 0.01;
N = fs * duration;
t = 0:1/fs:duration-1/fs;
f = 1000;
a = 2.119;
phi = 0;

a = 2.5;
phi = 0;

s = a * sin(2*pi*t*f + phi);
tr50 = a * sawtooth(2*pi*t*f + phi, 0.5);
tr40 = a * sawtooth(2*pi*t*f + phi, 0.4);
sq50 = a * square(2*pi*t*f + phi, 50);
sq25 = a * square(2*pi*t*f + phi, 25);

Final_Plots(s, 'Sin wave', 'sin');
% Final_Plots(tr50, 'Triangle wave 50% symmetry ratio', 'tri50');
% Final_Plots(tr40, 'Triangle wave 40% symmetry ratio', 'tri40');
% Final_Plots(sq50, 'Square wave 50% duty cycle', 'sqr50');
% Final_Plots(sq25, 'Square wave 25% duty cycle', 'sqr25');
% Final_Plots(Square50_integrator, ['Low-pass filter response for' newline 'Square wave 50% duty cycle'], 'int50');
% Final_Plots(Square50_differentiator, ['High-pass filter response for' newline 'Square wave 50% duty cycle'], 'diff50');
% Final_Plots(Triangle40_integrator, ['Low-pass filter response for' newline 'Triangle wave 40% symmetry ratio'], 'int40');
% Final_Plots(Triangle40_differentiator, ['High-pass filter response for' newline 'Triangle wave 40% symmetry ratio'], 'diff40');

% close all;

%% converted to Final_Plot function
% figure(123);
% %sin
% subplot(4,3,1)
% plot(t,s)
% xlabel('Time (seconds)');
% xlim([0,0.002])
% ylabel('Amplitude (Volts)');
% ylim([-3,3])
% title('Time-domain sinus');
%
% %triangle with 50% duty cycle
% subplot(4,3,2)
% plot(t,tr)
% xlabel('Time (seconds)');
% xlim([0,0.002])
```

```

% ylabel('Amplitude (Volts)');
% ylim([-3,3])
% title('Time-domain triangle');
%
% %square with 50% duty cycle
% subplot(4,3,3)
% plot(t,sq)
% xlabel('Time (seconds)');
% xlim([0,0.002])
% ylabel('Amplitude (Volts)');
% ylim([-3,3])
% title('Time-domain square');
%
% %fft sin
% s_fft = fft(s);
% %fft triangle
% tr_fft = fft(tr);
% %fft square
% sq_fft = fft(sq);
%
% s_oneSide = s_fft(1:N/2);
% frequencies = fs * (0:N/2-1) / N;
% S_magnitude = abs(s_oneSide) / (N/2);
% S_phase = angle(s_oneSide) * 180/pi;
%
% tr_oneSide = tr_fft(1:N/2);
% Tr_magnitude = abs(tr_oneSide) / (N/2);
% Tr_phase = angle(tr_oneSide) * 180/pi;
%
% sq_oneSide = sq_fft(1:N/2);
% Sq_magnitude = abs(sq_oneSide) / (N/2);
% Sq_phase = angle(sq_oneSide) * 180/pi;
%
% freq_of_interest = [1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000];
% indices = zeros(size(freq_of_interest));
% for i = 1:length(freq_of_interest)
%     [~, indices(i)] = min(abs(frequencies - freq_of_interest(i)));
% end
%
% subplot(4,3,4)
% plot(frequencies, db(S_magnitude));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylim([-40, 10]);
% ylabel('Magnitude');
% title('Frequency spectrum');
%
% subplot(4,3,5)
% plot(frequencies, db(Tr_magnitude));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylabel('Magnitude');
% ylim([-40, 10]);

```



```

% title('Frequency spectrum');
%
% subplot(4,3,6)
% plot(frequencies, db(Sq_magnitude));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylabel('Magnitude');
% ylim([-20, 10]);
% title('Frequency spectrum');
%
%
% subplot(4,3,7)
% stem(frequencies(indices), S_magnitude(indices));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylabel('Magnitude');
% title('Frequency spectrum');
%
% subplot(4,3,8)
% stem(frequencies(indices), Tr_magnitude(indices));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylabel('Magnitude');
% title('Frequency spectrum');
%
% subplot(4,3,9)
% stem(frequencies(indices), Sq_magnitude(indices));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylabel('Magnitude');
% title('Frequency spectrum');
%
%
% subplot(4,3,10)
% stem(frequencies(indices), S_phase(indices));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylabel('Phase (deg)');
% ylim([-180, 180]);
% title('Phase spectrum');
%
% subplot(4,3,11)
% stem(frequencies(indices), Tr_phase(indices));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylabel('Phase (deg)');
% ylim([-200, 200]);
% title('Phase spectrum');
%
% subplot(4,3,12)
% stem(frequencies(indices), Sq_phase(indices));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);

```

```

% ylabel('Phase (deg)');
% ylim([-200, 200]);
% title('Phase spectrum');

```

## Final Plots.m

```

function [] = Final_Plots(f, name, filename)
frequency = 1000;
duration = 10/frequency;
N = size(f,2);
sampling_frequency = N/duration;
t = 0:1/sampling_frequency:duration-1/sampling_frequency;
fig = figure('Name', name);
fig.Position(3:4) = [300 500];

%% signal
subplot(4,1,1);
plot(t, f);
xlabel('Time (seconds)');
xlim([0,0.001])
ylabel('Amplitude (Volts)');
ylim([-max(f)-(max(f)*0.3),max(f)+(max(f)*0.3)])
title(join(['', name]));
grid on;

%% fourier series
X = t(1:size(t,2)/10);
% f_short = f(1:size(x,2));
L = 1/frequency;
dx = 2*L/(N/10-1);
x = -L:dx:L;
f_short = f(1:size(x,2));
subplot(4,1,2); hold on;
% plot(x, f_short, 'r*');
A0 = (1/L)*sum(f_short.*ones(size(x)))*dx;
fFS = A0/2;
for k=1:10
    A(k) = (1/L)*sum(f_short.*cos(pi*k*x*frequency))*dx;
    B(k) = (1/L)*sum(f_short.*sin(pi*k*x*frequency))*dx;
    fFS = fFS + A(k)*cos(k*pi*x*frequency) + B(k)*sin(k*pi*x*frequency);
    plot(X, fFS);
    pause(.05);
end
plot(X, fFS,'LineWidth',2);
hold off;

xlabel('Time (seconds)');
xlim([0,0.001])
ylabel('Amplitude (Volts)');
ylim([-max(f)-(max(f)*0.3),max(f)+(max(f)*0.3)])
title('Fourier series expansion');
grid on;

%% fft amplitude

```

```

% plot
f_fft = fft(f);
f_oneSide = f_fft(1:N/2);
frequencies = sampling_frequency * (0:N/2-1) / N;
f_magnitude = abs(f_oneSide) / (N/2);
f_phase = angle(f_oneSide) * 180/pi;

freq_of_interest = [1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000];
indices = zeros(size(freq_of_interest));
for i = 1:length(freq_of_interest)
    [~, indices(i)] = min(abs(frequencies - freq_of_interest(i)));
end

% subplot(4,1,3)
% plot(frequencies, db(f_magnitude));
% xlabel('Frequency (Hz)');
% xlim([0, 11000]);
% ylim([-40, 10]);
% ylabel('Magnitude');
% title('Frequency spectrum');

%stem
subplot(4,1,3)
stem(frequencies(indices), f_magnitude(indices), 'x');
xlabel('Frequency (Hz)');
xlim([0, 11000]);
ylabel('Magnitude');
title('Frequency spectrum');
grid on;

%% fft phase
subplot(4,1,4)
stem(frequencies(indices), f_phase(indices), 'x');
xlabel('Frequency (Hz)');
xlim([0, 11000]);
ylabel('Phase (deg)');
ylim([-180, 180]);
title('Phase spectrum');
grid on;

%% saving img
print(join(['img/',filename]), '-depsc');
end

```

## Import csv.m

```

function [Vout] = RC_circuit(frequency, N)
figure('Name', 'RC circuit');
Vpp = 5;
frequency = 1000;
T = 1/frequency;
R = 1.5e3;
C1 = 47e-4;

```

```

C2 = 10;
dt = T/(N-1);
t = 0:dt:T;
Vout = 0*t;
Vout(N/2+1:N) = Vout(N/2+1:N) + Vpp;
Vout(N/2+1:N) = Vout(N/2+1:N) + (-Vpp)*exp(-(1:N/2)/(R*C1));
Vout(1:N) = Vout(1:N) + (Vpp)*exp(-(1:N)/(R*C1));
Vout(1:N) = Vout(1:N) - Vpp/2;
plot(t, Vout);
pause(.1)
end

```