Bilge Kaan Ateş

22102758

28.04.2024

**EEE321 SIGNALS AND SYSTEMS**

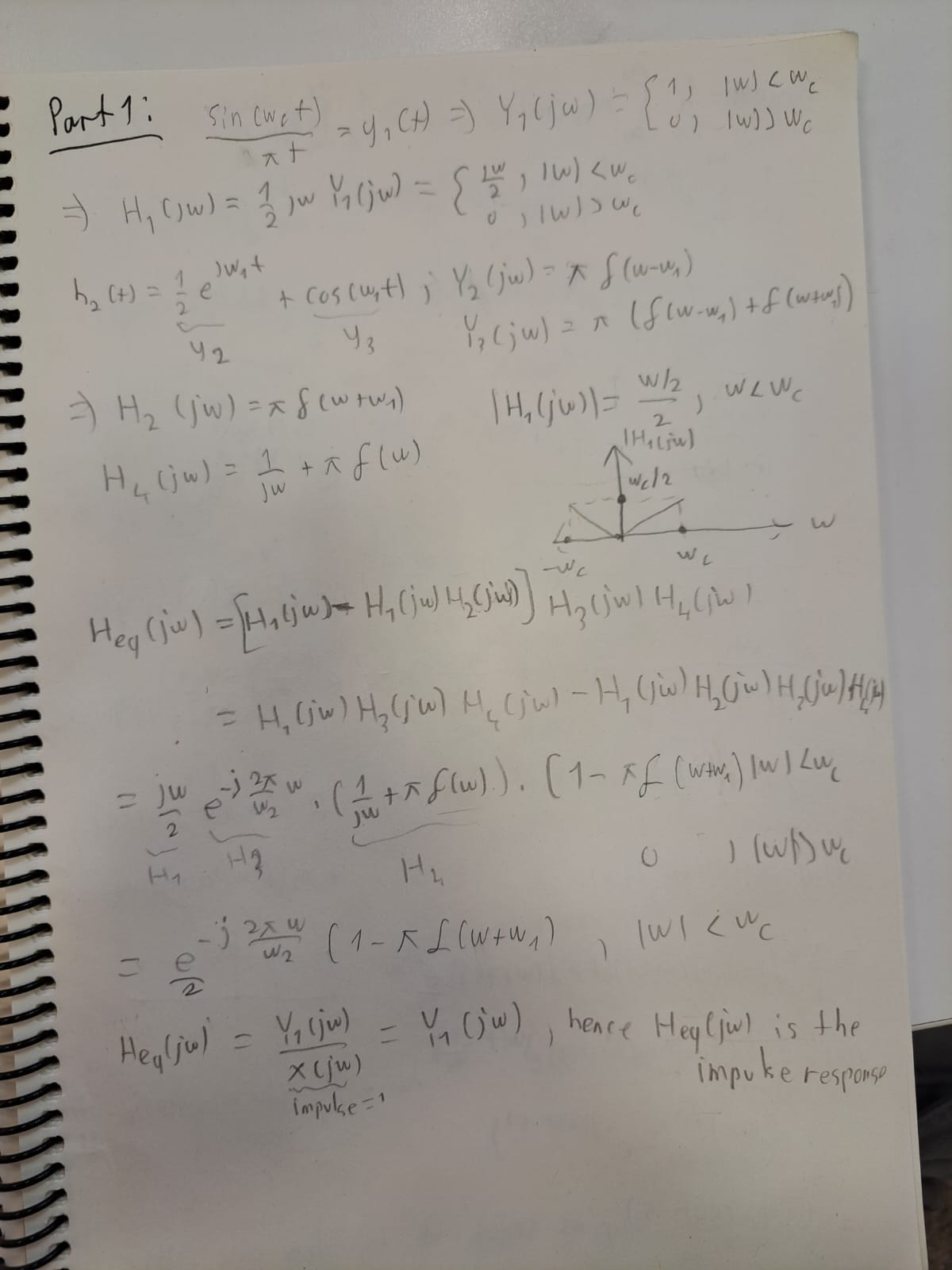
**LAB ASSIGNMENT 4**

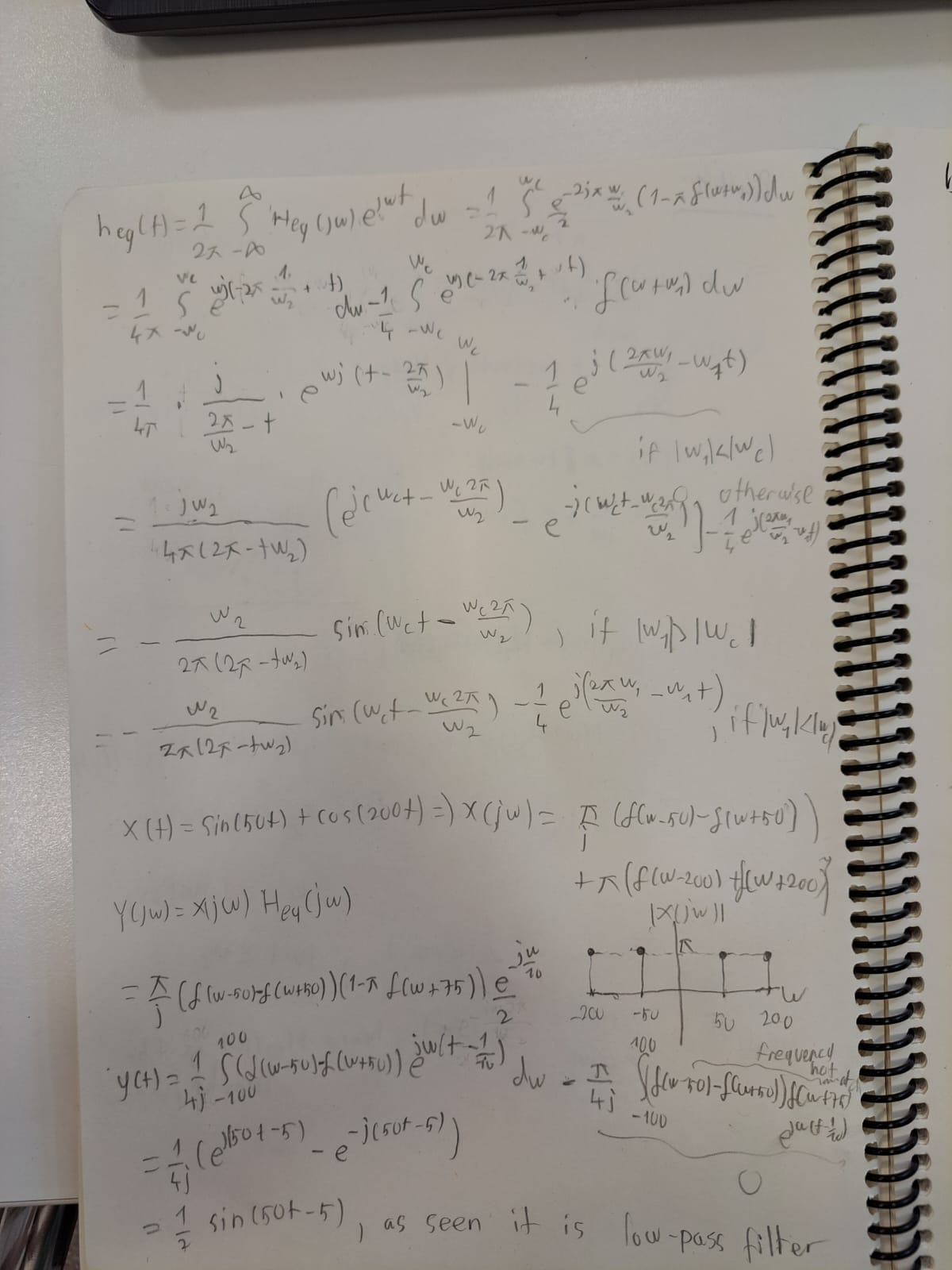
**1) INTRODUCTION:**

In this lab, I work with ideal/imperfect integrator systems and the properties of LTI systems such as linearity, causality, memory etc. Moreover, I analyzed BIBO stability of the systems as well. I studied the impulse and unit step response of the integrator systems. In first part, I manually handled the integrator systems and analyze their LTI properties and test them in MATLAB. In the second part, I write **sumElements()** function and analyzed the systems BIBO stability via Matlab. In the third part, I investigate the differences of ideal and imperfect (exponentially decaying) integrators. In the fourth part, firstly I derived second order difference equation and implement it in Matlab. Then, I found the inverse system of it by manually and implement it in Matlab and observed these systems convolution gives me impulse response.

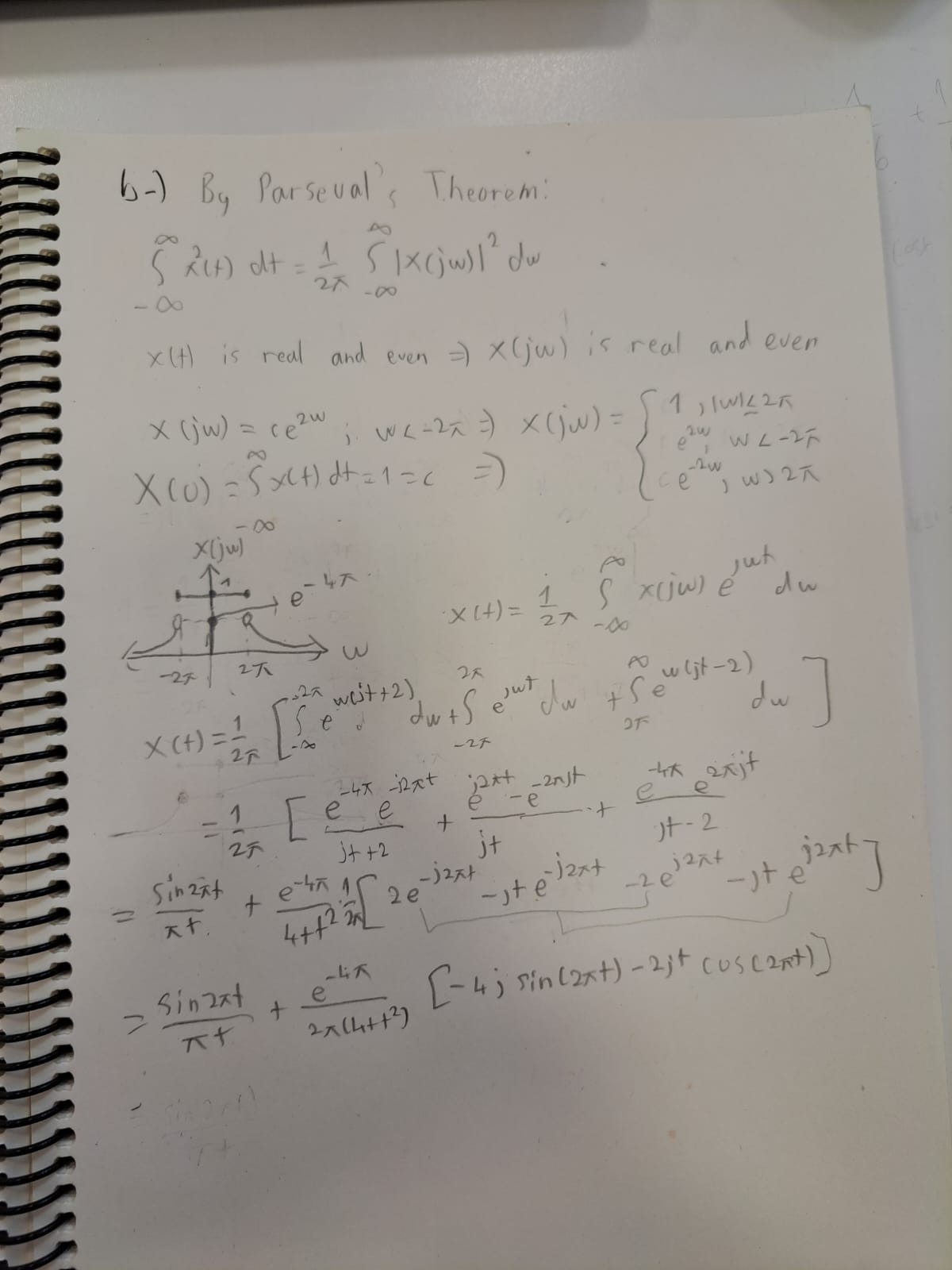
**2) LAB:**

* Part 1:

Part 1.a: 



Part 1.b:



* Part 2:

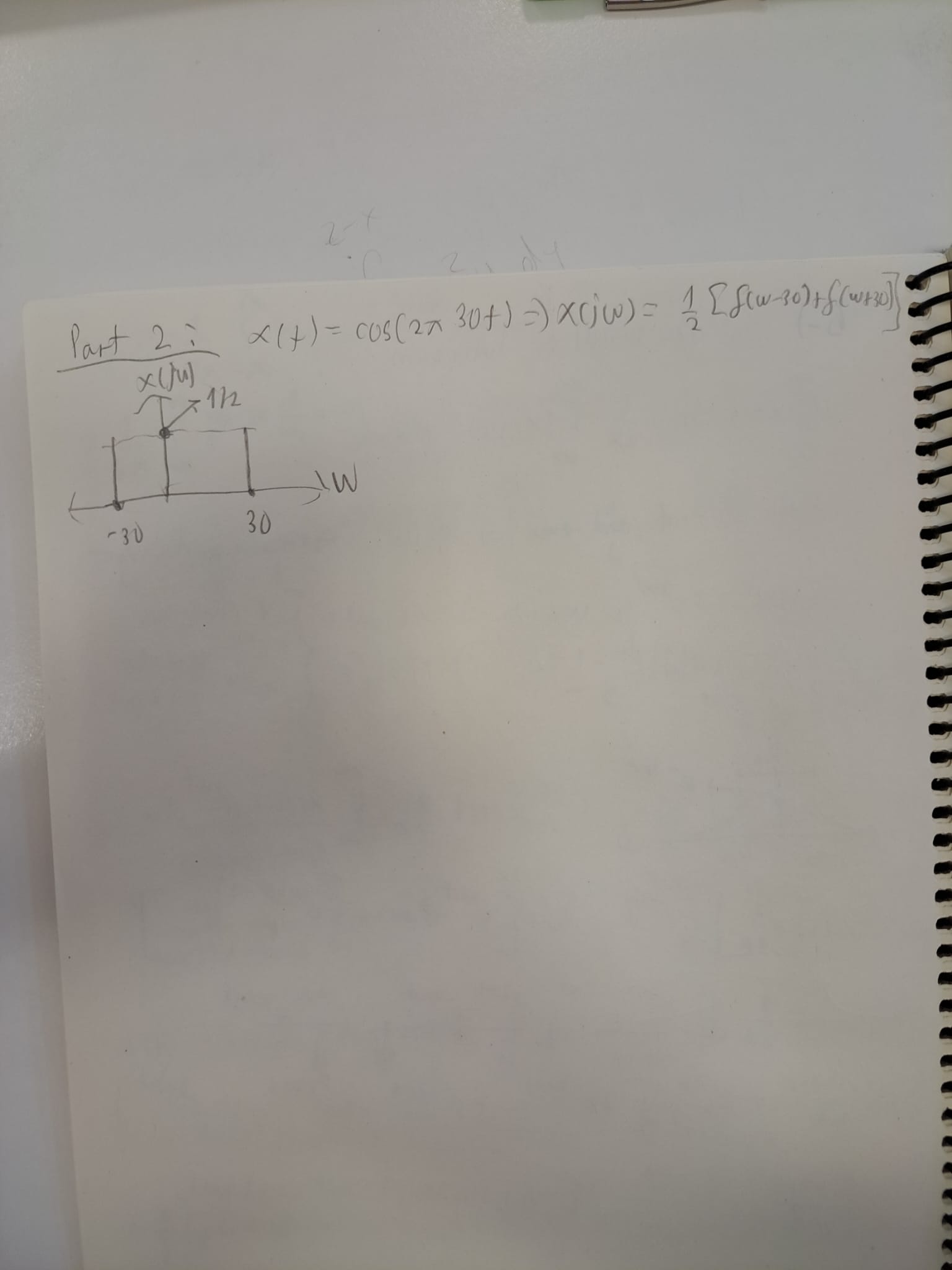


Fig. 1: Hand derivation of Fourier Transform of

In this part, I write **FourierTransform** function which takes the signal array x(t), sampling time array and the sampling period as inputs and gives the corresponding Fourier Transform frequency\_array as output. Firstly, I created time array from -10s to 10s and then discretized the signal with sampling period with this array. For the function, I obtained discrete summation form of the Fourier Transform integration. I stored the length of the signal array in a variable, N. Then, I obtained sampling frequency from and I created the “*frequency*” array for limit the Fourier Transformed signal ranges whose bounds are and with N samples. Finally, I obtained the “*frequency\_array”* by using 2 for loops. First one is for ranging the frequency and the second one is for ranging the time. At the end, I divide the result with sample size to normalize its amplitude.

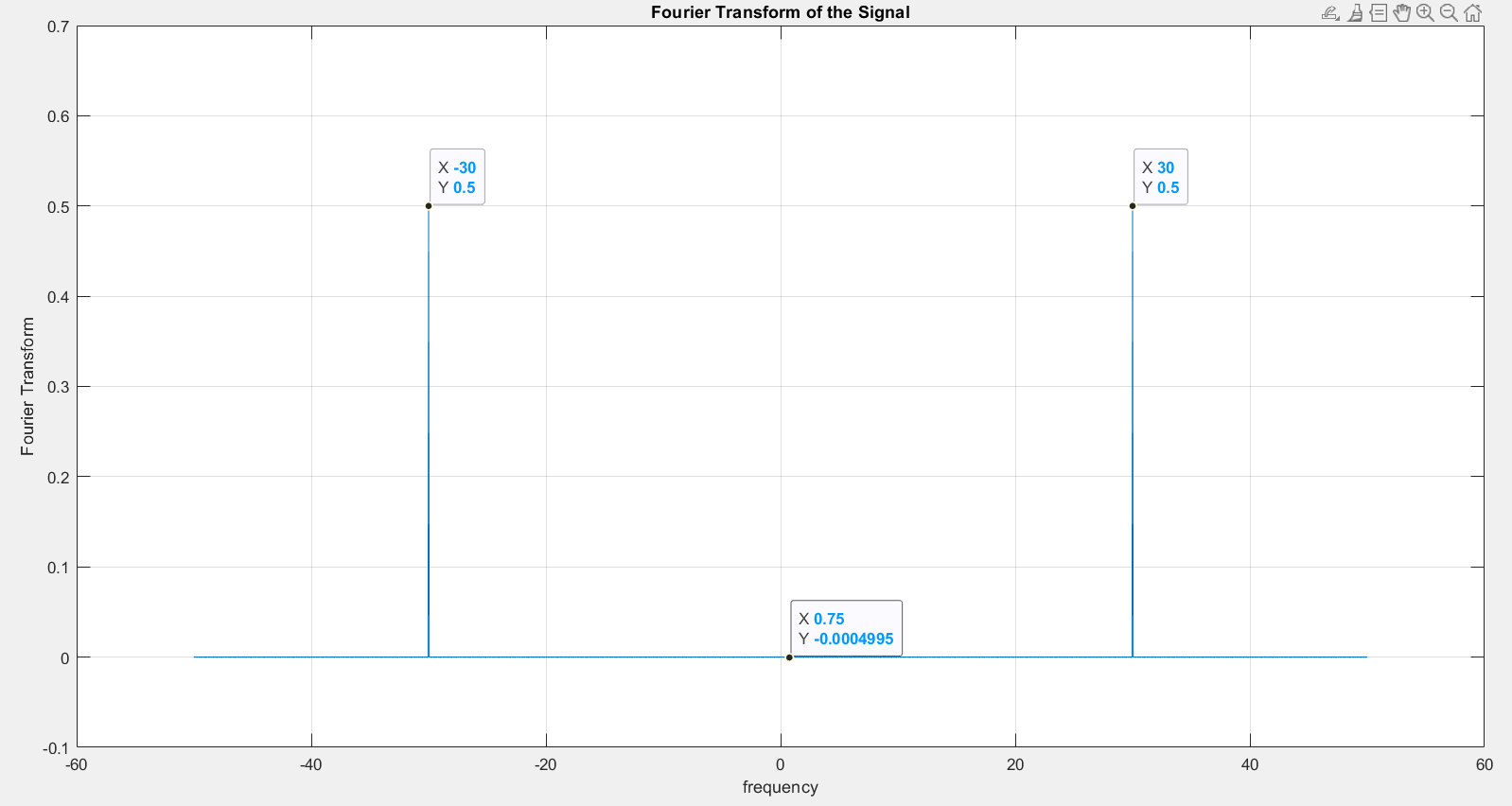


Fig. 2: Testing Results

When I test the function with the given signal , I obtained the same results with my hand derivation as represented in Fig. 1 and 2. There is small error in other frequencies than -30Hz and 30Hz due to numerical calculation. It is same for all frequencies while -30Hz and 30Hz gives the correct results. Hence, we can evaluate this function as successful enough.

You can see the corresponding code below:

function [frequency\_array] = FourierTransform(x, t, Ts)

N = length(x);

Fs = 1 / Ts;

frequency = linspace(-Fs/2, Fs/2, N);

frequency\_array = zeros(1,N);

for k = 1:length(frequency\_array)

sum\_value = 0;

for n = 1:N

sum\_value = sum\_value + x(n) \* exp(-1i \* 2 \* pi \* frequency(k) \* t(n));

end

frequency\_array(k) = sum\_value/(N+1);

end

end

Ts = 0.01;

Fs = 1 / Ts;

t = -10:Ts:10;

frequency = linspace(-Fs/2, Fs/2, length(t));

x = cos(60\*pi\*t);

X = FourierTransform(x,t,Ts);

figure;

plot(frequency, X);

xlabel('frequency');

ylabel('Fourier Transform');

title('Fourier Transform of the Signal');

* Part 3:

Part 3.1:

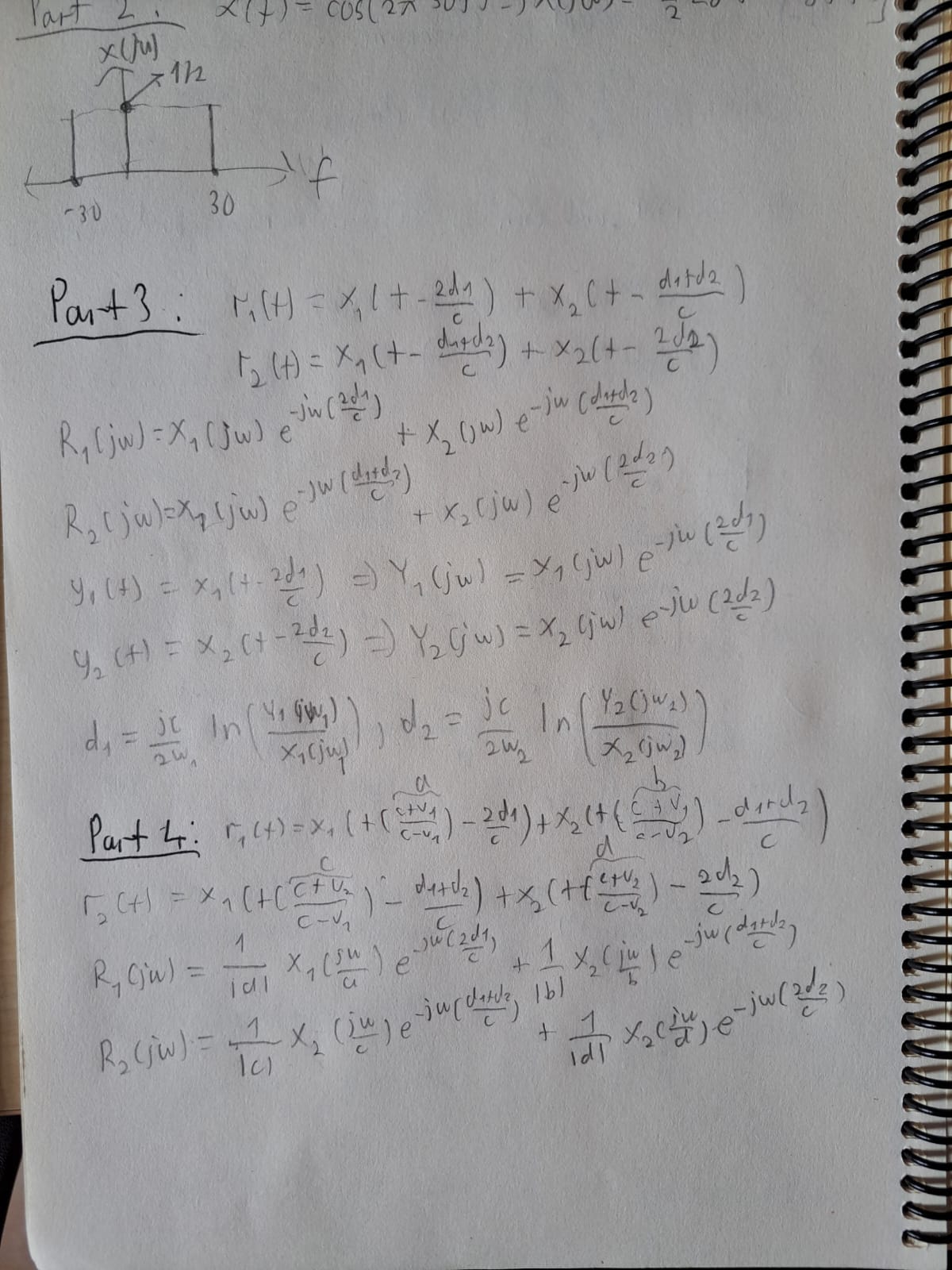


Fig. 3: Hand derivation of radar estimation

In this part, I applied Fourier Transformation in radar systems to obtain distance estimation. In this setup, we have two transmitter-receiver pairs which transmit signals in different frequencies. I obtained the corresponding time-domain and frequency-domain signal expressions by hand as seen in Fig. 3. I used the unchanged signal-amplitude assumption for this application. I used the fact that received signals are time-delayed versions of the original signals.

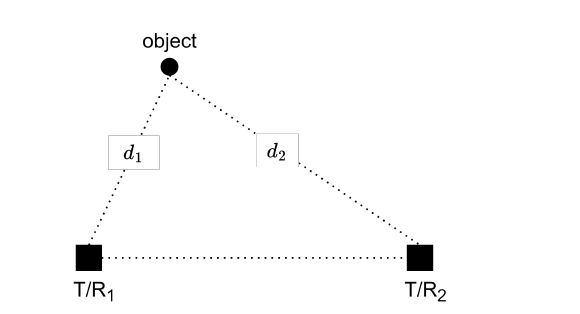


Fig. 4: An example of the setup

In Fig. 4, an example setup can be observed. Here, represents the distance of the object from T/R pairs. I used *c* as the propagation speed of the signals in the medium. Derivations and expressions for can be observed in Fig. 3. In order to differentiate transmitted signals from transmitters, I applied band-pass filter around the transmitted signal frequency in transmitters to simplify the received signal as just the transmitted signal from that transmitter. Finally, I obtained the expression for from these relations.

Part 3.2:

For the Matlab implementation I used the following signals:

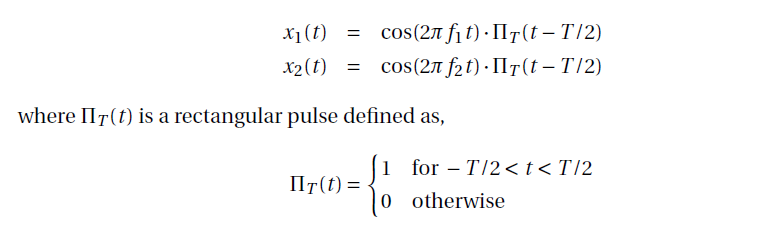


Fig. 5: Test signals

For this setup, I took , , , , , , and . Then, I implement that I derived in part 3.1. Corresponding plots can be seen in Fig. 6 and 7.

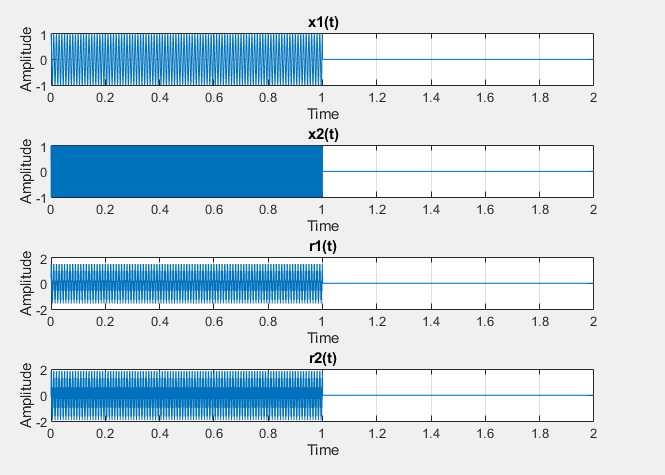


Fig. 6: Signals in time domain

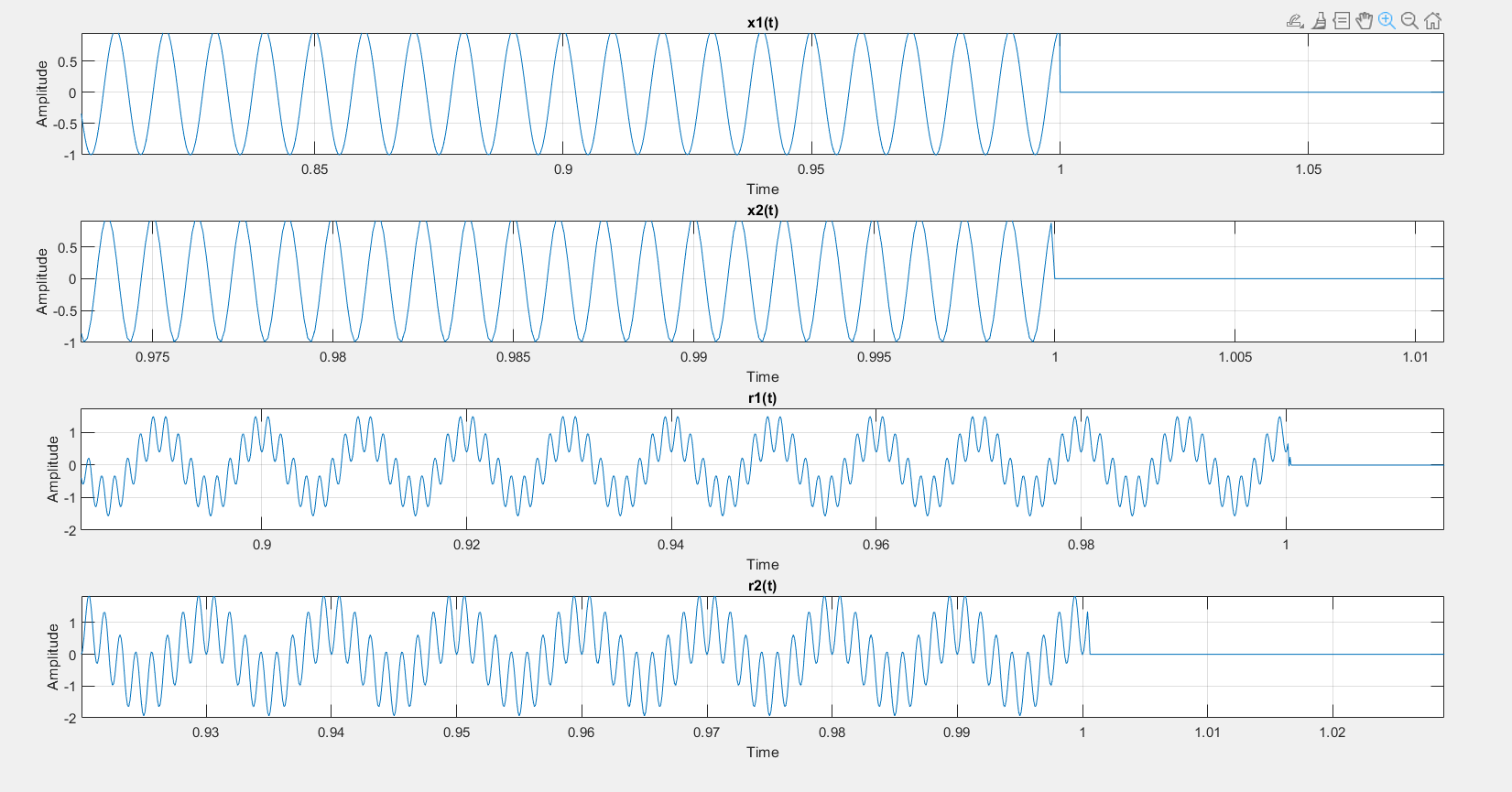


Fig. 7: Signals in time domain with scaling

After this operation, I used **FourierTransform** function that I write in Part 2, and obtained Fourier Transform of those signals. Their plots are presented in Fig. 8. Although and seen as have 2 frequency peak, they contain also blue peaks as well.

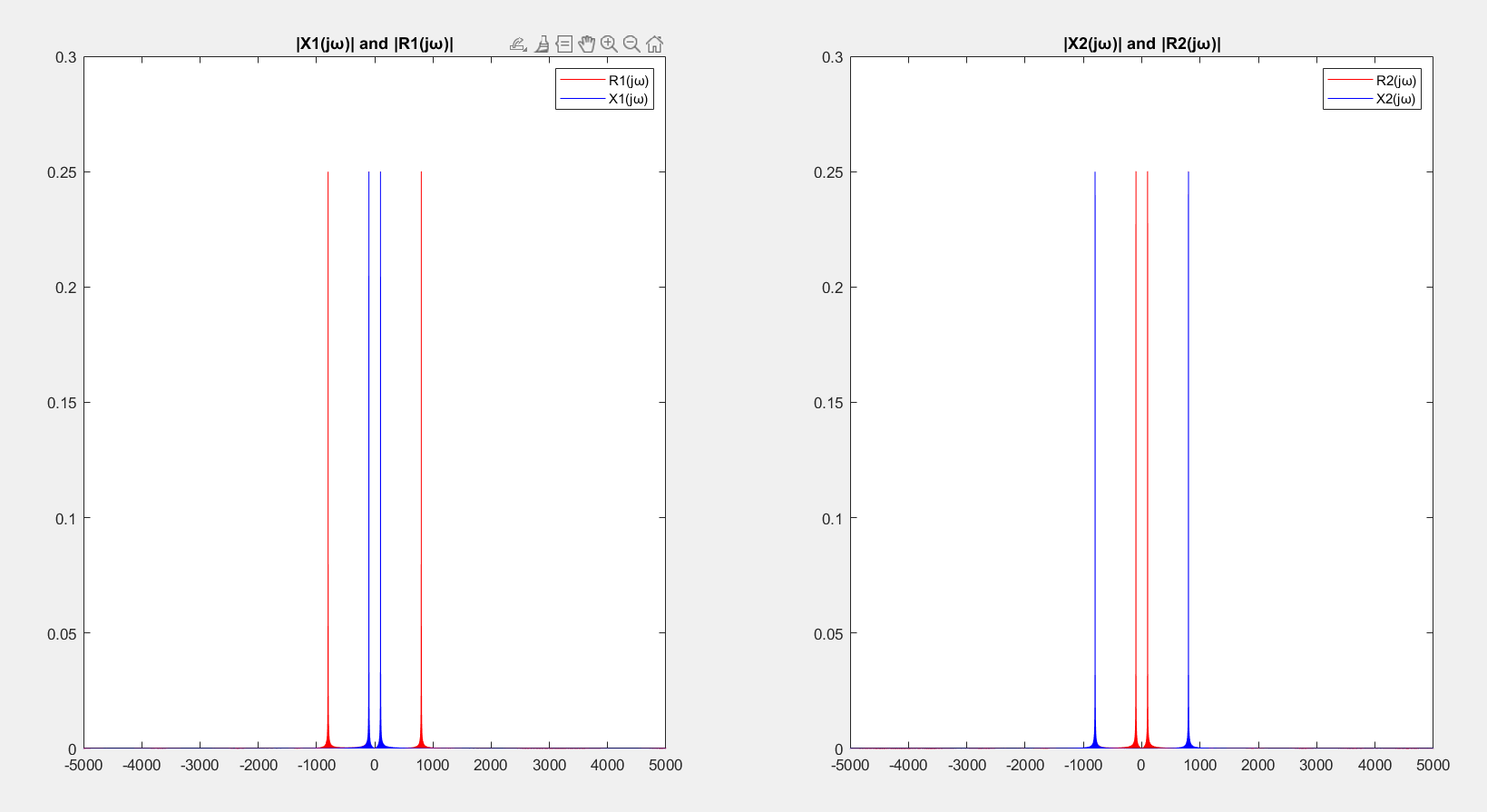


Fig. 8: ,, , signals

In order to obtain filtered signals, I used 50Hz length band-pass filters in operating frequency and obtained as seen in Fig. 9.

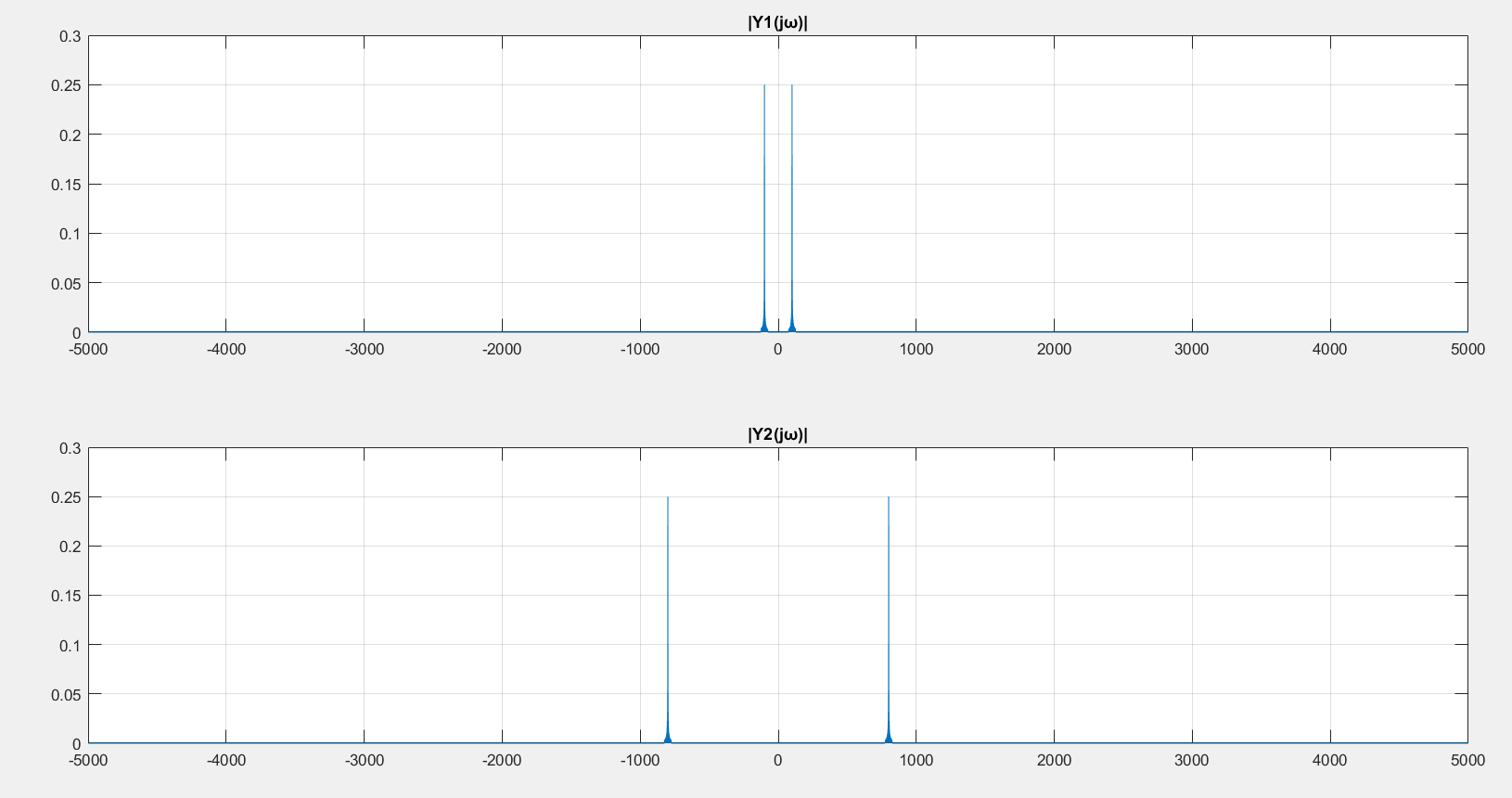


Fig. 9: Filtered signals ,

Finally, I implement the hand-derived expression for distance estimation in Matlab. I obtained the results with less than 0.1% error as seen in Fig. 10. The results are very close to original values. If we use smaller sampling period, we can obtain exactly the actual values.



Fig. 10: Estimated Distances

You can observe the corresponding codes below:

f1 = 100;

f2 = 800;

Ts = 0.0001;

Fs = 1/Ts;

t = 0:Ts:2;

% t1= t + 2\*d1/c;

% t2= t + (d1+d2)/c;

% t3= t + 2\*d2/c;

frequency\_array = linspace(-Fs/2, Fs/2, length(t));

d1 = 0.05;

d2 = 0.1;

c = 343;

T = 2;

omega\_pass = 50;

%

x1 = cos(2\*pi\*f1\*t) .\* rectpuls(t , T);

x2 = cos(2\*pi\*f2\*t) .\* rectpuls(t , T);

%

%Time delay calculation for cosine signals of x1 and x2

read1\_1 = cos(2\*pi\*f1\*t) .\*exp(-1i\*2\*pi\*f1\*(2\*d1/c)).\* rectpuls(t - 2\*d1/c, T);

read1\_2 = cos(2\*pi\*f2\*t) .\*exp(-1i\*2\*pi\*f2\*((d1+d2)/c)).\*rectpuls(t - (d1+d2)/c, T);

read2\_1 = cos(2\*pi\*f1\*t) .\*exp(-1i\*2\*pi\*f1\*((d1+d2)/c)).\*rectpuls(t - (d1+d2)/c, T);

read2\_2 = cos(2\*pi\*f2\*t) .\*exp(-1i\*2\*pi\*f2\*(2\*d2/c)).\*rectpuls(t - 2\*d2/c, T);

r1 = read1\_1 + read1\_2 ;

r2 = read2\_1 + read2\_2;

X1 = FourierTransform(x1, t, Ts);

X2 = FourierTransform(x2, t, Ts);

R1 = FourierTransform(r1, t, Ts);

R2 = FourierTransform(r2, t, Ts);

% Appllication of band-pass filters

Y1 = R1 .\* ((abs(frequency\_array - f1) <= omega\_pass/2)+(abs(frequency\_array + f1) <= omega\_pass/2));

Y2 = R2 .\* ((abs(frequency\_array - f2) <= omega\_pass/2)+(abs(frequency\_array + f2) <= omega\_pass/2));

% %

% figure

% subplot(4,1,1)

% stem(t, x1);

% grid on;

% xlabel('frequency');

% ylabel('Fourier Transform');

% title('Fourier Transform of the Signal');

% subplot(4,1,2)

% stem(t, x2);

% grid on;

% xlabel('frequency');

% ylabel('Fourier Transform');

% title('Fourier Transform of the Signal');

% subplot(4,1,3)

% stem(t, r1);

% grid on;

% xlabel('frequency');

% ylabel('Fourier Transform');

% title('Fourier Transform of the Signal');

% subplot(4,1,4)

% stem(t, r1);

% grid on;

% xlabel('frequency');

% ylabel('Fourier Transform');

% title('Fourier Transform of the Signal');

figure;

subplot(4,1,1);

plot(t, x1);

grid on;

xlabel('Time');

ylabel('Amplitude');

title('x1(t)');

subplot(4,1,2);

plot(t, x2);

grid on;

xlabel('Time');

ylabel('Amplitude');

title('x2(t)');

subplot(4,1,3);

plot(t, r1);

grid on;

xlabel('Time');

ylabel('Amplitude');

title('r1(t)');

subplot(4,1,4);

plot(t, r2);

grid on;

xlabel('Time');

ylabel('Amplitude');

title('r2(t)');

% Plot magnitude of Fourier transforms

figure

subplot(1,2,1);

plot(frequency\_array, abs(R1), 'r');

hold on;

plot(frequency\_array, abs(X1), 'b');

hold off;

title('|X1(jω)| and |R1(jω)|');

legend('R1(jω)', 'X1(jω)');

subplot(1,2,2);

plot(frequency\_array, abs(R2), 'r');

hold on;

plot(frequency\_array, abs(X2), 'b');

hold off;

title('|X2(jω)| and |R2(jω)|');

legend('R2(jω)','X2(jω)');

% Plots of the filtered signals

figure

subplot(2,1,1);

plot(frequency\_array, abs(Y1));

grid on;

title('|Y1(jω)|');

subplot(2,1,2);

plot(frequency\_array, abs(Y2));

grid on;

title('|Y2(jω)|');

% Distance estimation part

[~, index\_f1] = min(abs(frequency\_array - f1));

[~, index\_f2] = min(abs(frequency\_array - f2));

phase\_f1 = angle(Y1(index\_f1));

phase\_f2 = angle(Y2(index\_f2));

d1\_estimate = abs((phase\_f1 \* c) / (2\*pi\*f1))/2;

d2\_estimate = abs((phase\_f2 \* c) / (2\*pi\*f2))/2;

disp(['Estimated d1: ', num2str(d1\_estimate), ' m']);

disp(['Estimated d2: ', num2str(d2\_estimate), ' m']);

* Part 4:

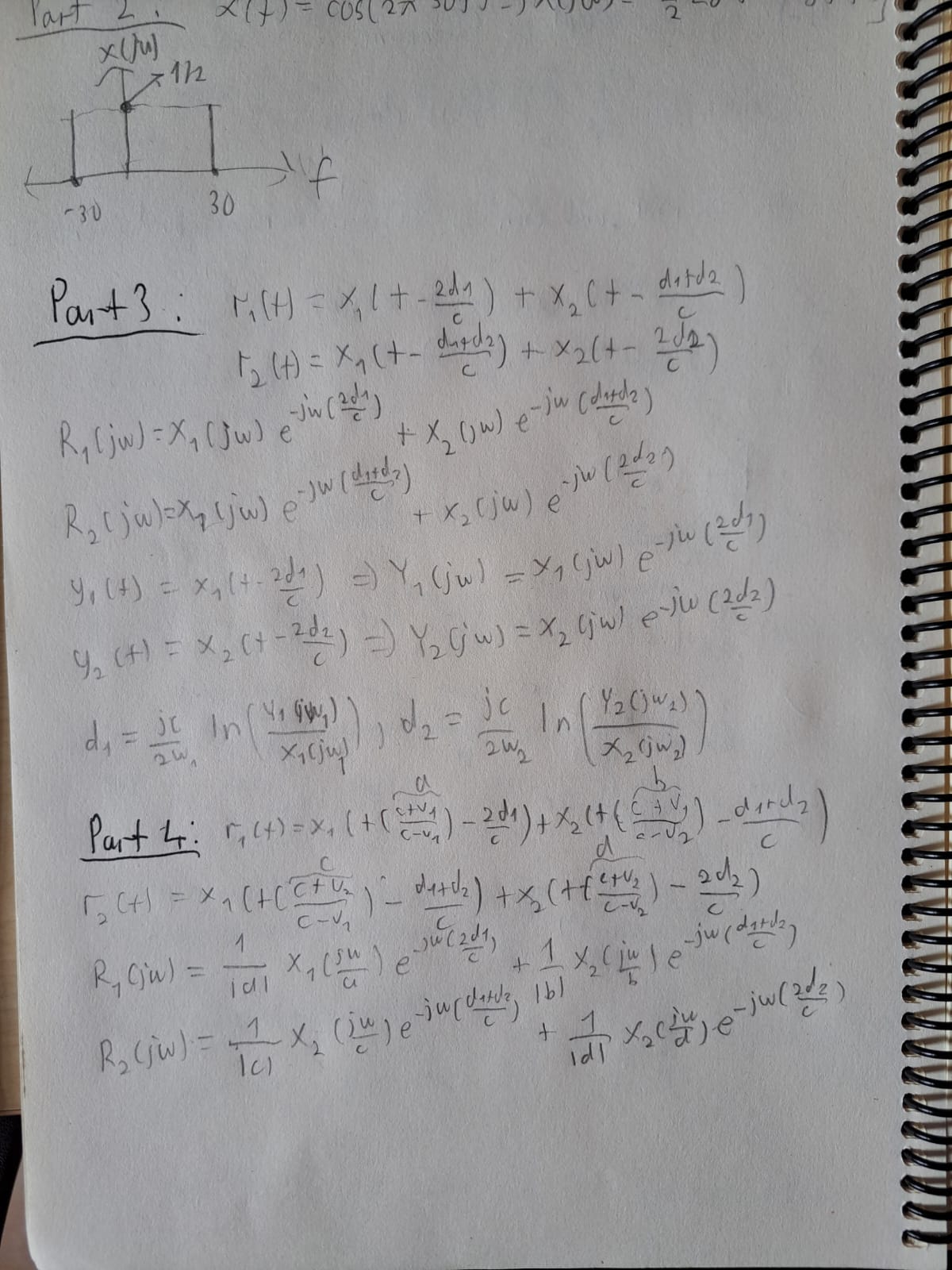
****

Fig. 11:Hand derivation with velocity inclusion

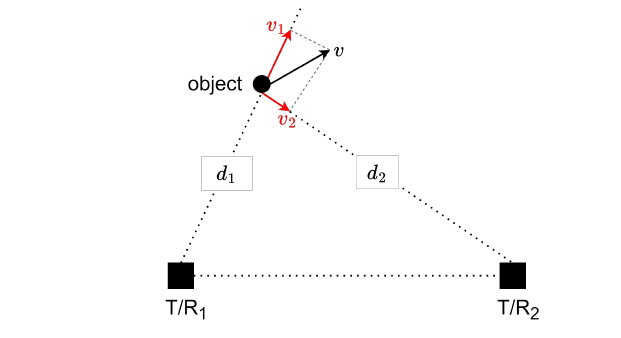


Fig. 12: New setup

In this part, I tried to estimate the velocity of the object with radar systems. Corresponding hand derivations of the received signals are shown in Fig. 11. New setup can be seen in Fig. 12. I used same configuration with Part 3.2 with the time scaling adjustment:

I set and . I obtained by using **FourierTransform** function. You can see the plots of the Fourier Transform amplitudes of the signals in Fig. 13.

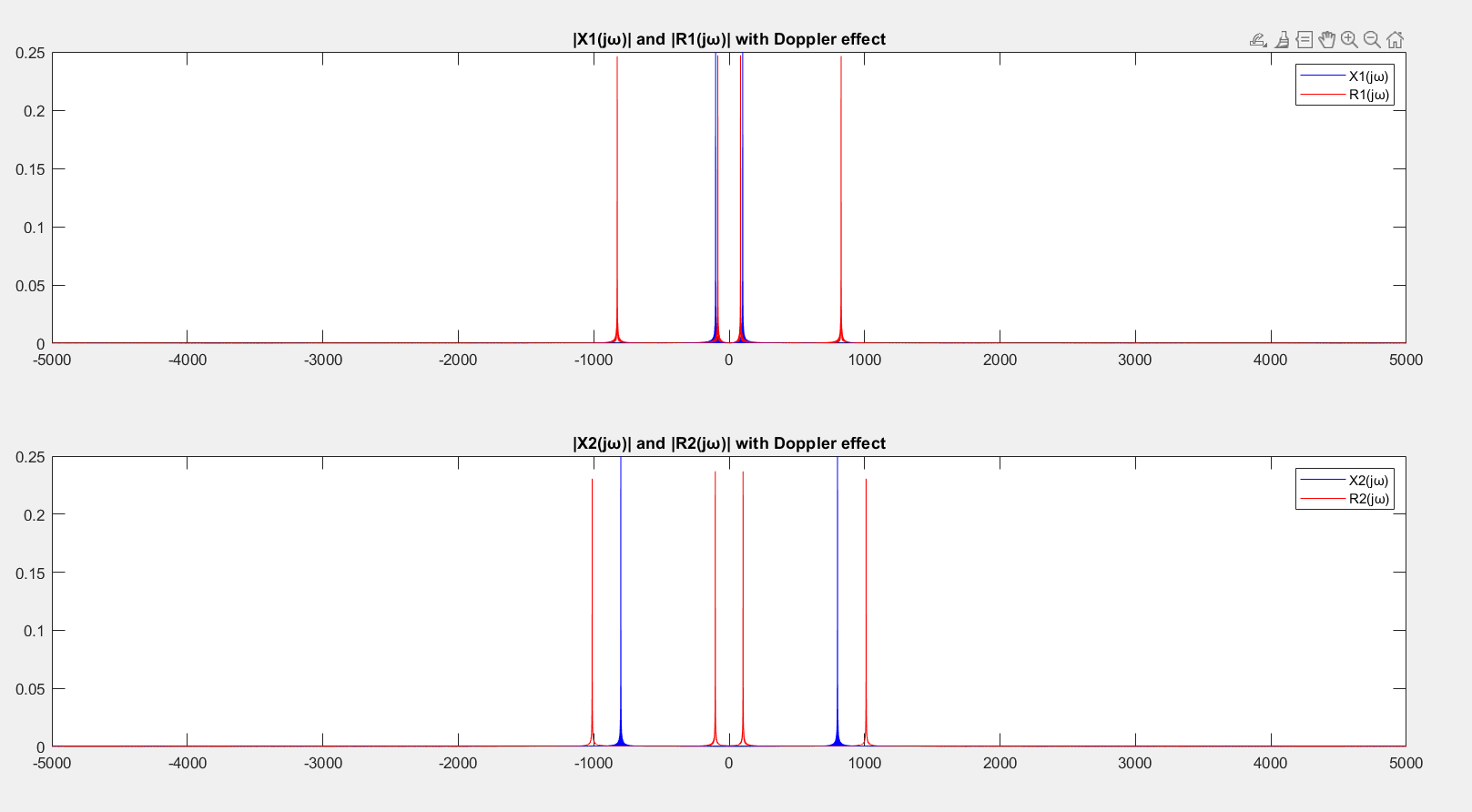


Fig.13: Magnitudes of the signals in frequency domain

Here we see that the peaks doesn’t coincides as the received signals frequency changed due to moving object with Doppler Effect. Closest peak to 100Hz frequency transmitted signal is 84Hz and to 800Hz is 1011 Hz as seen in Fig. 14.

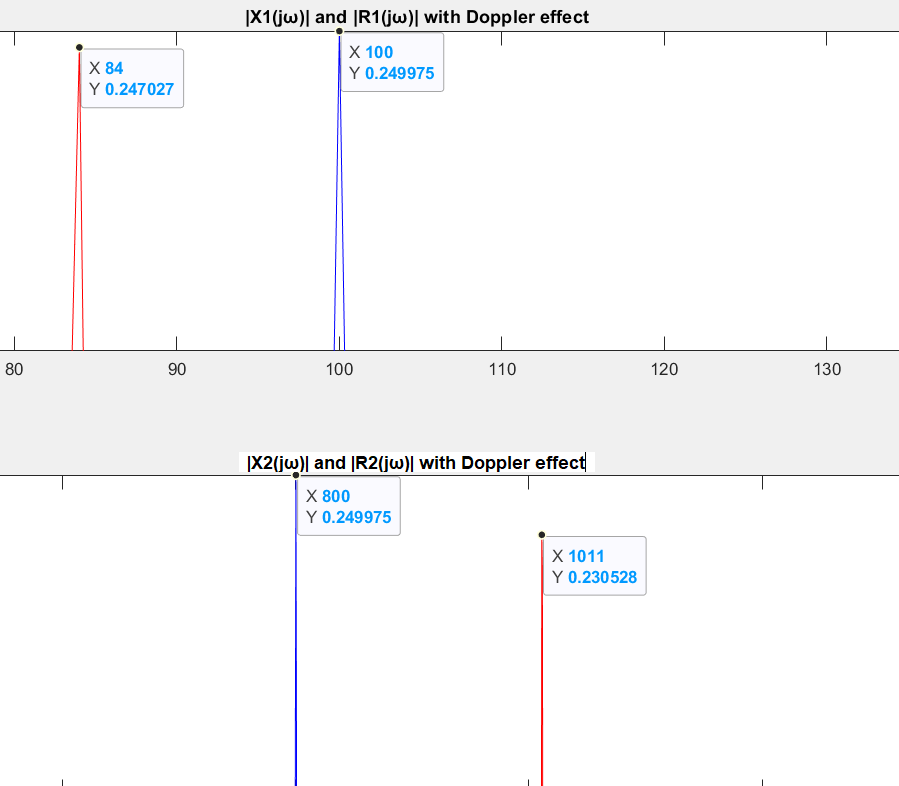


Fig. 14: Peak frequencies of the received signals

With these peak values, I specified their frequency and original and shifted frequencies ratio gives the scaling factor:

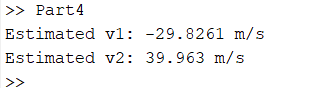


Fig. 15: Estimated velocities

In Fig. 15, estimated velocities presented. Error is less than 1% which means we have highly good estimation. We can also estimate the velocities by using the phase shift in received signal. Phase shift can be calculated by using cross-correlation. Peak of the cross-correlation holds the information of time delay and phase shift of the signal.

Related codes are presented below:

f1 = 100;

f2 = 800;

Ts = 0.0001;

Fs = 1/Ts;

t = 0:Ts:2;

% t1= t + 2\*d1/c;

% t2= t + (d1+d2)/c;

% t3= t + 2\*d2/c;

frequency\_array = linspace(-Fs/2, Fs/2, length(t));

d1 = 0.05;

d2 = 0.1;

c = 343;

T = 2;

v1 = -30;

v2 = 40;

omega\_pass = 50;

%

x1 = cos(2\*pi\*f1\*t) .\* rectpuls(t , T);

x2 = cos(2\*pi\*f2\*t) .\* rectpuls(t , T);

%

scale1 = (c+v1)/(c-v1);

scale2 = (c+v1)/(c-v2);

scale3 = (c+v2)/(c-v1);

scale4 = (c+v2)/(c-v2);

%Implementing the Doppler Effect received signals

read1\_1 = cos(2\*pi\*f1\*t\*scale1) .\*exp(-1i\*2\*pi\*f1\*(2\*d1/c)).\* rectpuls(t - 2\*d1/c, T);

read1\_2 = cos(2\*pi\*f2\*t\*scale2) .\*exp(-1i\*2\*pi\*f2\*((d1+d2)/c)).\*rectpuls(t - (d1+d2)/c, T);

read2\_1 = cos(2\*pi\*f1\*t\*scale3) .\*exp(-1i\*2\*pi\*f1\*((d1+d2)/c)).\*rectpuls(t - (d1+d2)/c, T);

read2\_2 = cos(2\*pi\*f2\*t\*scale4) .\*exp(-1i\*2\*pi\*f2\*(2\*d2/c)).\*rectpuls(t - 2\*d2/c, T);

r1\_doppler = read1\_1 + read1\_2 ;

r2\_doppler = read2\_1 + read2\_2;

% Fourier Transform of received signals

R1\_doppler = FourierTransform(r1\_doppler, t, Ts);

R2\_doppler = FourierTransform(r2\_doppler, t, Ts);

figure;

subplot(2,1,1);

plot(frequency\_array, abs(X1), 'b');

hold on;

plot(frequency\_array, abs(R1\_doppler), 'r');

hold off;

title('|X1(jω)| and |R1(jω)| with Doppler effect');

legend('X1(jω)', 'R1(jω)');

subplot(2,1,2);

plot(frequency\_array, abs(X2), 'b');

hold on;

plot(frequency\_array, abs(R2\_doppler), 'r');

hold off;

title('|X2(jω)| and |R2(jω)| with Doppler effect');

legend('X2(jω)', 'R2(jω)');

% Finding the peaks of the signals

[~, index\_f1\_peak] = max(abs(X1));

[~, index\_R1\_peak] = max(abs(R1\_doppler));

[~, index\_f2\_peak] = max(abs(X2));

[~, index\_R2\_peak] = max(abs(R2\_doppler));

% Scaling factor derivation

scaling\_factor1 = 84 / 100;

scaling\_factor2 = 1011 / 800;

% scaling\_factor1 = 0.84;

% scaling\_factor2 = 1.27;

% Estimation of the velocities

v1\_estimate = c \* (scaling\_factor1 - 1) / (scaling\_factor1 + 1);

v2\_estimate = c \* (scaling\_factor2 - 1) / (scaling\_factor2 + 1);

disp(['Estimated v1: ', num2str(v1\_estimate), ' m/s']);

disp(['Estimated v2: ', num2str(v2\_estimate), ' m/s']);