# ISTANBUL TECHNICAL UNIVERSITY

# COMPUTER ENGINEERING DEPARTMENT

## BLG 354
## Signal&Systems for Comp.Eng.

**Homework1 Report**

Ali Emre Kaya
150210097
kayaemr21@itu.edu.tr

**SPRING 2025**

# Contents

# 1 Problem 1

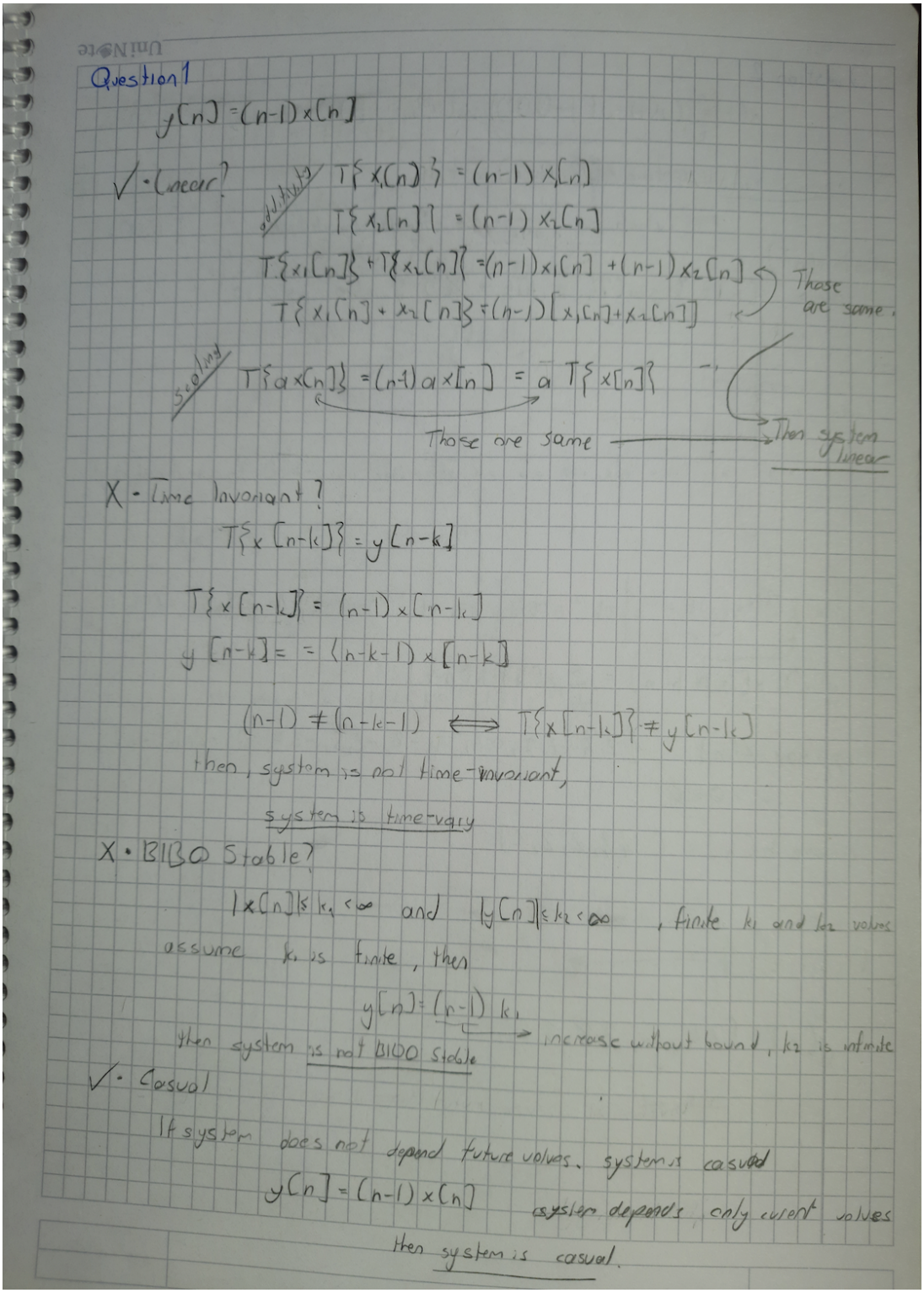


Figure 1: Problem 1

# 2 Problem 2

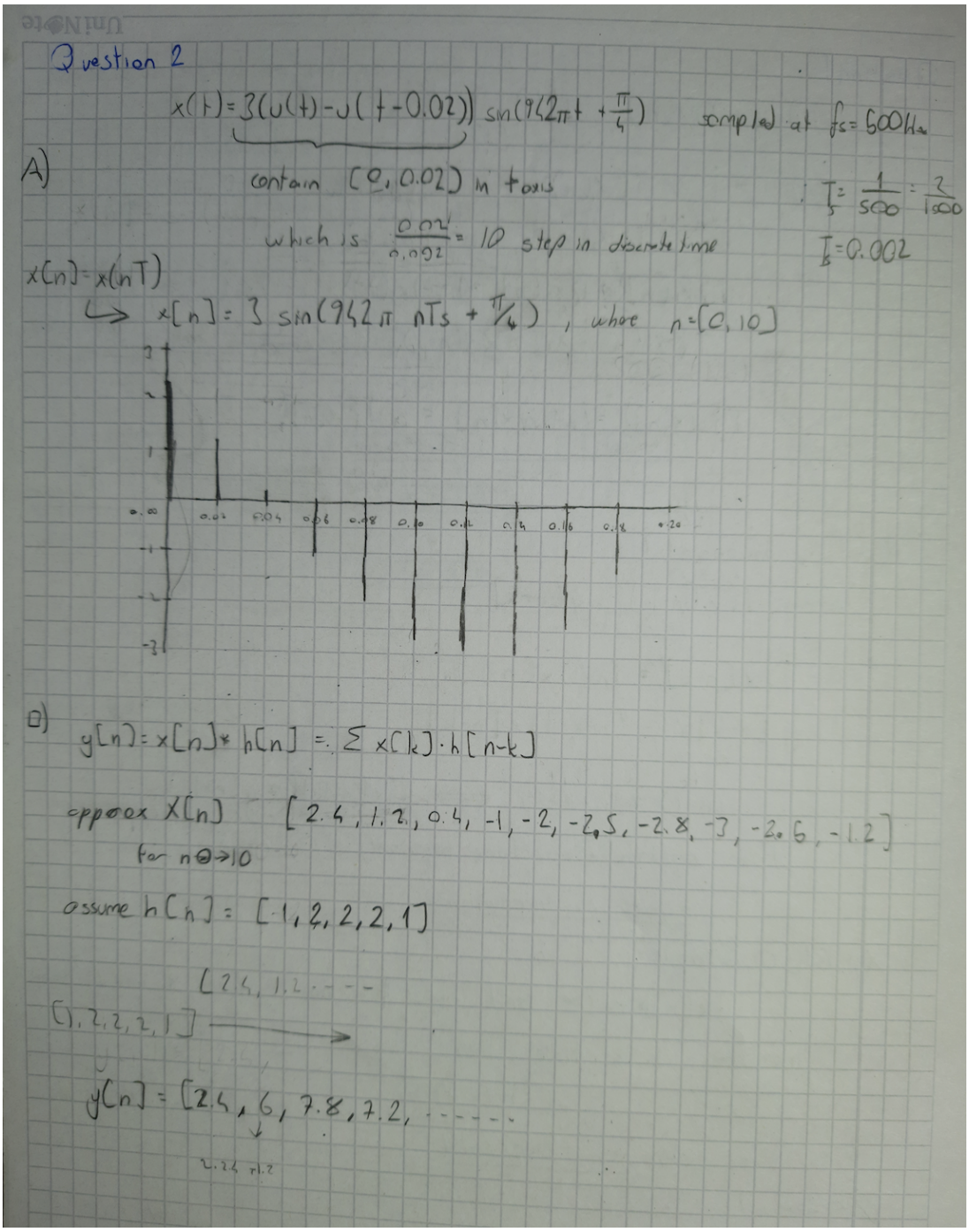


Figure 2: Problem 2

# 3 Problem 3

In problem 3, the mission is to implement a bot for the game **Sound of the Machine** (*https://noobc0re.itch.io/sound-of-the-machine*).

My code follows these steps:

- Press *Enter* to start the game.

- Move to the bottom-left of the game grid.
- Move to the top while recording the amplitude.
- Move to the right while continuing to record the amplitude.
- Navigate to the points with the maximum amplitude.
- To ensure the machine is caught, perform a spiral movement.

I use the numpy, soundcard, pyautogui, time, and threading libraries. The soundcard library is used to calculate the amplitude of sound, while the pyautogui library is used to control the keyboard keys "w", "a", "s", "d", and "Enter". The threading library is used to create asynchronous tasks. For example, when pressing "w", the program should also store the current amplitude. To achieve this, a separate thread is created to listen to the computer's audio.

Listing 1: Part of Example Output

```
----------1----------
Vertical amplitude calculating...
Horizontal amplitude calculating...
Max v amplitude: 0.1314 at time 1.477
Max h amplitude: 0.2697 at time 1.095
Catching...
    .
    .
    .
```

A Selenium driver could also be used to open the game's website directly. However, since the question statement only mentions the use of the pyautogui and soundcard libraries, I did not implement this feature.

There is a known issue where the program listens to the computer's audio through the microphone. As a result, when the environment is noisy, the program's accuracy decreases significantly. I attempted to record audio directly from the speakers, which is easier with some libraries other than soundcard, but I decided to continue using soundcard, meaning the program still captures sound from the microphone. The noise level in the environment should be considered when running this project.

# 4 Problem 4

In the problem 4, the given system should be applied to the Ilber Ortaylı's audio record.

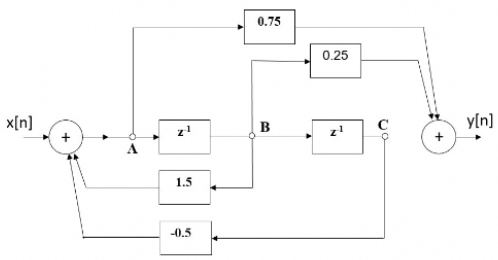


Figure 3: System

I calculate the system like that:

$$
\begin{aligned}
y[n] &= 0.75 \cdot A + 0.25 \cdot B \\
A &= x[n] + 1.5 \cdot B - 0.5 \cdot C \\
B &= z^{-1}A \\
C &= z^{-1}B
\end{aligned}
$$

After calculating this function, I implement an audio filtering code in python. I use scipy to read audio, matplotlib to visualize outputs. I filter the audio 1, 5 and 100 times.

Amplitude visualization of original and filtered audios shown like that:

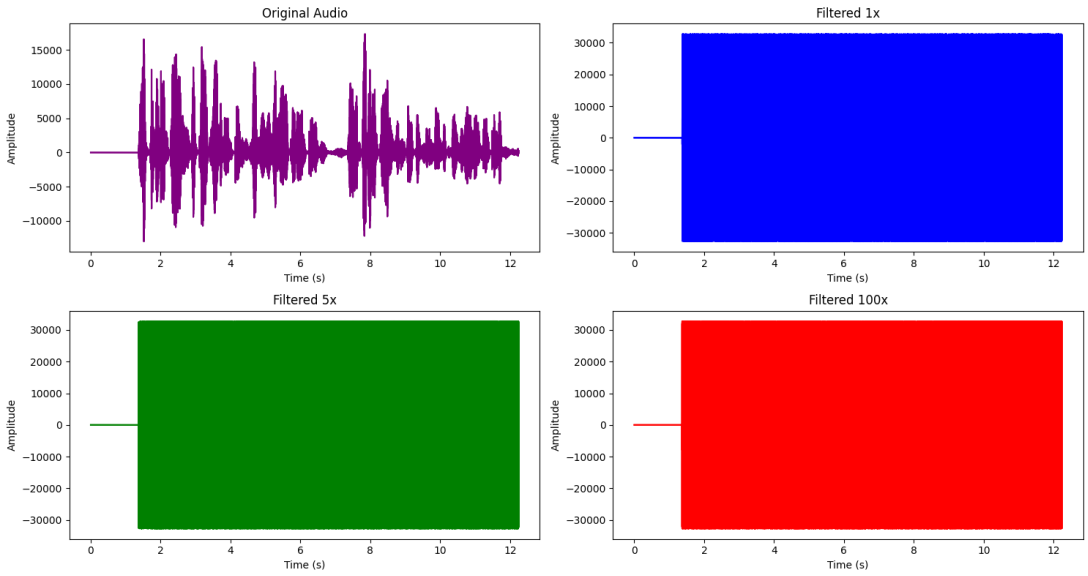


Figure 4: Amplitude Comparison

Even one-time filtering grows exponentially, because of $1.5 \cdot B$. Even $-0.5 \cdot C$ can stop this exponential increasing, because it divide the previous operation of 1.5, in that time $1.5 * 1.5 = 2.25$ and $2.25/2 == 1.125$. That's the cause why audio increase exponentially with **approximately** $1.125^n$.

# REFERENCES