Digital Signal Processing Lab (ECL333)

Experiment 1: Response of a Discrete-Time System

Department of Electronics and Communication Engineering

Faculty: Jinu Jayachandran, Viji R

Student Name: ____________________ **Roll Number:** ________________

Instructions

1. All students must bring a printed hard copy of the lab sheet and complete the **Pre-lab Work** section before entering the lab.
2. After performing the experiment, students should complete the **Work Done During the Lab**, **Analysis and Inference**, and **Post-lab Questions** sections.
3. Answers for each section must be written **neatly and legibly** within the space provided in the lab sheet. **Additional sheets are not permitted**.
4. The completed lab sheet must be submitted to the instructor before leaving the lab.
5. **Zero marks** will be awarded for late submission of lab sheets.
6. The final, completed code must be printed and attached to the lab sheet and brought for submission in the next lab session.

Objective

- To determine the response of a discrete-time system to basic signals like impulse, step, and sinusoidal inputs using Python and analyze the resulting outputs.
- To implement a first-order low-pass filter using a difference equation and analyze its effectiveness in reducing noise from a signal.

1 Response of Discrete Systems to Basic Signals

1.1 Theoretical Background

In digital signal processing, discrete-time systems process signals that are defined at discrete intervals. A Linear Time-Invariant (LTI) system is one whose output for a given input does not change over time and obeys the principle of superposition. The response of an LTI system is fully characterized by its impulse response.

For a given input $x[n]$ and an LTI system with impulse response $h[n]$, the output $y[n]$ is given by the convolution:

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

Difference equations are used to describe systems in a recursive form, for example:

$$y[n] = a \cdot y[n-1] + x[n]$$

This kind of system is common in filter design and helps in simulating real-time DSP operations.

1.2 Pre-lab Work

Q1. Define the following signals with expressions:

- Unit Impulse

- Unit Step

- Sinusoidal

Q2. What do you expect the system output to be when excited by each of the basic signals?

Q3. Write the algorithm or pseudocode to perform linear convolution of two discrete-time signals. (*Note: Do not write the implementation in Python, C, or any other programming language.*)

1.3 Sample Python Code Snippets:

Unit Impulse Signal:

```
import numpy as np
n = np.arange(-20, 21)
delta = np.where(n == 0, 1, 0)
```

Unit Step Signal:

```
u = np.where(n >= 0, 1, 0)
```

Sinusoidal Signal:

```
fs = 10
f = 2
x = np.sin(0.2 * np.pi * n * f/fs)
```

Sample LTI System (Using Difference Equation):

System: $y[n] = 0.3 \cdot y[n-1] + x[n]$

```
import numpy as np

# Define input: unit impulse
n = np.arange(0, 20)
x = np.zeros_like(n, dtype=float)
x[0] = 1 # impulse at n=0

# Initialize output
y = np.zeros_like(n, dtype=float)

# Difference equation: y[n] = 0.5*y[n-1] + x[n]
for i in range(1, len(n)):
    y[i] = 0.3 * y[i-1] + x[i]
```

1.4 Procedure

1. Generate and plot the input signals: impulse, step, sinusoid. Choose a suitable length for step signal and suitable frequency for sinusoid.
2. Define a discrete-time LTI system using a difference equation, e.g., $y[n] = 0.5y[n-1] + x[n]$.
3. Use Python to compute the system response using linear convolution for each of the input signals (DO NOT use any python function for convolution).
4. Plot and compare the system output for each input (Use the package `matplotlib` for plotting. Also use `stem()` to plot discrete signals).
5. Analyze and comment on output characteristics.

1.5 Analysis and Inference

(Write your conclusions and observations here.)

2 Application - Filter Design for Noise Reduction

In many real-life applications such as sensor signal processing, biomedical signal monitoring, and audio filtering, it's important to smooth out noisy signals. A first-order low-pass filter provides a simple yet powerful way to suppress high-frequency noise and retain the main structure of the signal.

The filter is defined by the following difference equation:

$$y[n] = (1 - \alpha) \cdot y[n - 1] + \alpha \cdot x[n]$$

where:

- $x[n]$: input noisy signal
- $y[n]$: output filtered signal
- $\alpha \in (0, 1)$: smoothing factor (smaller α means more smoothing)

This is a simple IIR filter that approximates exponential smoothing. It is widely used in applications like temperature reading stabilization, ECG signal denoising, and audio tone control.

2.1 Sample Python Code Snippets:

Adding Gaussian noise to signal:

```
# Add Gaussian noise (mean = 0, std = 0.3)
noise = np.random.normal(0, 0.3, size=signal.shape)
noisy_signal = signal + noise
```

2.2 Procedure

1. Generate a sinusoidal signal (e.g., $\sin(0.1\pi n)$) and add Gaussian noise.
2. Implement the first-order low-pass filter for different α values (e.g., 0.1, 0.3, 0.7).
3. Plot and compare the filtered output with the original noisy and clean signals (*Have both input and output figures in the same plot*).
4. Analyze how smoothing affects noise suppression and signal delay.

2.3 Work Done During Lab

(Write down the steps and observations of what you did during the lab session. Include values of α, types of inputs, and your analysis.)

2.4 Analysis and Inference

(Discuss the effect of varying α. What trade-offs are observed? Did the filter suppress the noise effectively?)

3 Post-lab Questions

3.1 Response to DT Systems

1. In Section 1.4, step 1, is the input sinusoid signal distorted if $f = 2$Hz and $f_s = 10$Hz? What happens to the sinusoid when $f_s = 5$Hz?. Explain.

2. Represent the system mentioned in Section 1.4, step 2 in terms of unit step function $u[n]$?

3. What change do you observe in the impulse response of the system mentioned in Section 1.4, step 2, if the value 0.5 is changed to 12? Is the system stable in both cases?. Explain. (*Hint: Check if the system is absolutely summable. Use the unit step representation.*)

3.2 Filter Design for Noise Reduction

1. What does the parameter α control in a low-pass filter?

2. How does this filter affect high-frequency components? Also, do you observe any delay in the output?.

3. Is the system described by the filter causal and stable? Justify.

4. What happens when $\alpha = 1$ or $\alpha = 0$?

4 Code Attachment

Attach your final Python code printout here.

Student Signature: ______________________________ **Date:** ____________________