CSE220 Signals and Linear Systems
Offline on Convolution

Introduction

In this offline, you will work with discrete signals and linear time invariant discrete systems. The goal of this offline is to help you understand and visualize how a linear time invariant system is completely characterized by its impulse response.

Part I: Implementation

In this part, you will implement two core classes: `Signal` and `LTI_System`.

1. Class `Signal`

The `Signal` class represents a finite-length discrete-time signal defined over

$$n \in [-\texttt{INF}, \texttt{INF}]$$

Constructor

- `Signal(INF)`
- Initializes a signal with all values set to zero.

Required Methods

- `set_value_at_time(t, value)` Sets the signal value at time index t.
- `shift(k)` Returns a new `Signal` object corresponding to a time-shifted version:

$$x(n-k)$$

- `add(other)` Returns the sum of two signals:

$$y(n) = x_1(n) + x_2(n)$$

- `multiply(scalar)` Returns a scaled signal:

$$y(n) = a \cdot x(n)$$

- `plot(title)` Produces a stem plot of the discrete-time signal.

2. Class `LTI_System`

The `LTI_System` class models a discrete-time LTI system using its impulse response $h(n)$.

Constructor

- `LTI_System(impulse_response)`

Required Methods

- `linear_combination_of_impulses(input_signal)`

  Decomposes the input signal into impulses ($\delta(n-k)$) and their coefficients ($x(k)$):

  $$x(n) = \sum_k x(k)\delta(n-k)$$

  It returns an array of impulses and an array of corresponding coefficients.

- `output(input_signal)`

  Computes the output using the LTI property:

  $$y(n) = \sum_k x(k)h(n-k)$$

  It uses the `linear_combination_of_impulses` method.

Part II: Signal Smoothing Using an LTI System

In this part, you will read a discrete-time signal from a file, construct the signal, and smooth it using your LTI system implementation.

1. Input Signal File Format

The input signal is stored in a text file with the following format:

- Line 1: Two integers representing start and end indices

  $$n_{\text{start}},\ n_{\text{end}}$$

- Line 2: Signal values for each integer n from n_{start} to n_{end}

2. Tasks

1. Read the signal file and construct a `Signal` object.
2. Plot the noisy input signal.
3. Define an impulse response corresponding to a 5-point moving average filter:

$$h(n) = \begin{cases} \frac{1}{5}, & n = -2, -1, 0, 1, 2 \\ 0, & \text{otherwise} \end{cases}$$

4. Create an `LTI_System` using this impulse response.
5. Compute the output signal using your implementation.
6. Plot the smoothed output signal.

3. Output

Your program should produce:

- A stem plot of the noisy input signal
- A stem plot of the smoothed output signal

4. Important Notes

- You must not use built-in convolution functions.
- All signal operations must use your own class methods.

Mark Distribution

Component	**Marks**
Signal Class Implementation	40
LTI System Implementation	40
File Reading, Smoothing, and Plots (Part II)	20
Total	**100**

Submission

Create separate python files for the two parts. Rename them as {id}_first.py and {id}_second.py. Put the python files in a folder named by your student id. Zip the folder and submit the zip file. **Do not include any images in the folder.**
Deadline: Monday, 19 January, 11:59 PM.