CSE220 Signals and Linear Systems
Online on Signal Basics

14 September 2024

Time: 35 min
Read the entire instruction carefully before starting to code.

1 Introduction

In this online, your task is to implement the following functions for **discrete** signals

1. `time_scale_signal`
2. `time_scale_signal_interpolate`

2 Representing Discrete Signal

We will represent a discrete signal as a numpy array. We will assume that all signals extend from $-\infty$ to ∞, but the numpy array will only contain the signal values from -8 to 8. Signal values outside this range is considered to be 0.
For example, the array $x = [0, 0, 0, 0, 0, 0, 0.5, 2, 1, 0.5, 1, 0, 0, 0, 0, 0, 0]$ represents the signal $x[n]$ in figure 1.

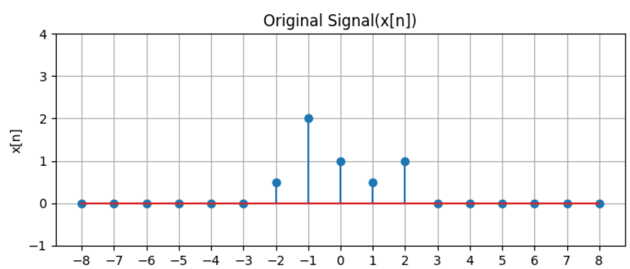

Figure 1: Signal corresponding to $[0, 0, 0, 0, 0, 0, 0.5, 2, 1, 0.5, 1, 0, 0, 0, 0, 0, 0]$

3 Tasks

You are **provided with a python file**. In this file, you have to implement 2 functions.

3.1 Task 1

Function to be implemented: `time_scale_signal(x,k)`
Input Parameters:

- x: A numpy array representing a **discrete** signal.
- k: A positive integer.

The function should return a numpy array representing the time scaled signal `x[n/k]`. **Set intermidate samples to 0**. Example: see figure 2.

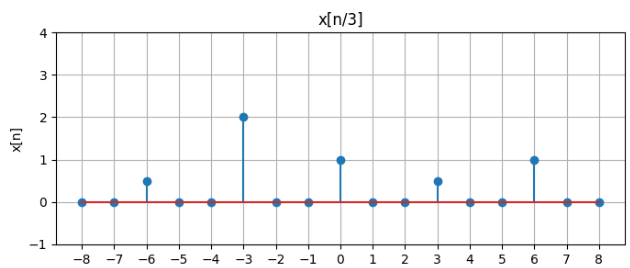

Figure 2: $x[n/3]$

3.2 Task 2

Function to be implemented: `time_scale_signal_interpolate(x,k)`
The task is same as Task 1, except for the value of the intermediate samples. For each new intermediate sample (between -8 and 8), set its value to the **average of the two original signal samples** between which it lies.
For example, the intermediate sample at -1 in figure 2 will be average of $x[0]$ and $x[-1]$ from figure 1. Refer to figure 3.

3.3 Bonus Task

You will get bonus marks if you can complete the tasks using numpy functions rather than using explicit python loops.

4 Mark Distribution

See table 1

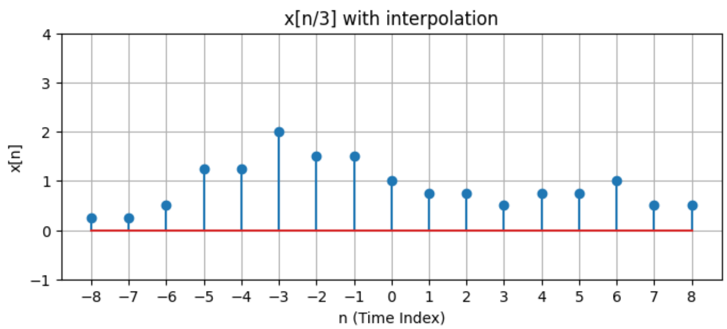

Figure 3: $x[n/3]$ with interpolation

Task 1	6
Task 2	4
Bonus	2

Table 1: Mark Distribution