DTLTI Systems

Pieter P

Definition

As the name implies, Discrete-Time Linear Time-Invariant Systems, or DTLTI systems for short, are systems that perform a linear transformation on discrete functions. The fact that they are time-invariant means that the transformation doesn't change over time: it doesn't matter if you apply it to a certain signal now or in ten minutes, the resulting signals are the same.
We'll define these properties of DTLTI systems mathematically:

> T is the transformation performed by a Discrete-Time Linear Time-Invariant (DTLTI) system if and only if
>
> 1. The transformation is linear:
> $T\left(a \cdot h[n] + b \cdot g[n]\right) = a \cdot T\left(h[n]\right) + b \cdot T\left(g[n]\right)$
> 2. The transformation is time-invariant:
> $y[n] \triangleq T\left(h[n]\right) \Rightarrow \forall k \in \mathbb{N} : T\left(h[n-k]\right) = y[n-k]$

Signals

We use square brackets to indicate that signals are discrete functions:

$$x : \mathbb{Z} \rightarrow \mathbb{R} : n \mapsto x[n]$$

$$\vec{x} : \mathbb{Z} \rightarrow \mathbb{R}^m : n \mapsto \vec{x}[n]$$

In the simplest case, x will just map to a scalar ($\mathbb{R}$), but in general, it can also map to an m-dimensional vector ($\mathbb{R}^m$). This will be useful later, when we'll introduce systems with multiple inputs and outputs, or systems with multiple internal states.

Example

We'll define and plot a simple example signal, and then we'll apply a simple transformation to it.

$$x : \mathbb{Z} \rightarrow \mathbb{R} : n \mapsto \cos(\pi n) + 2$$

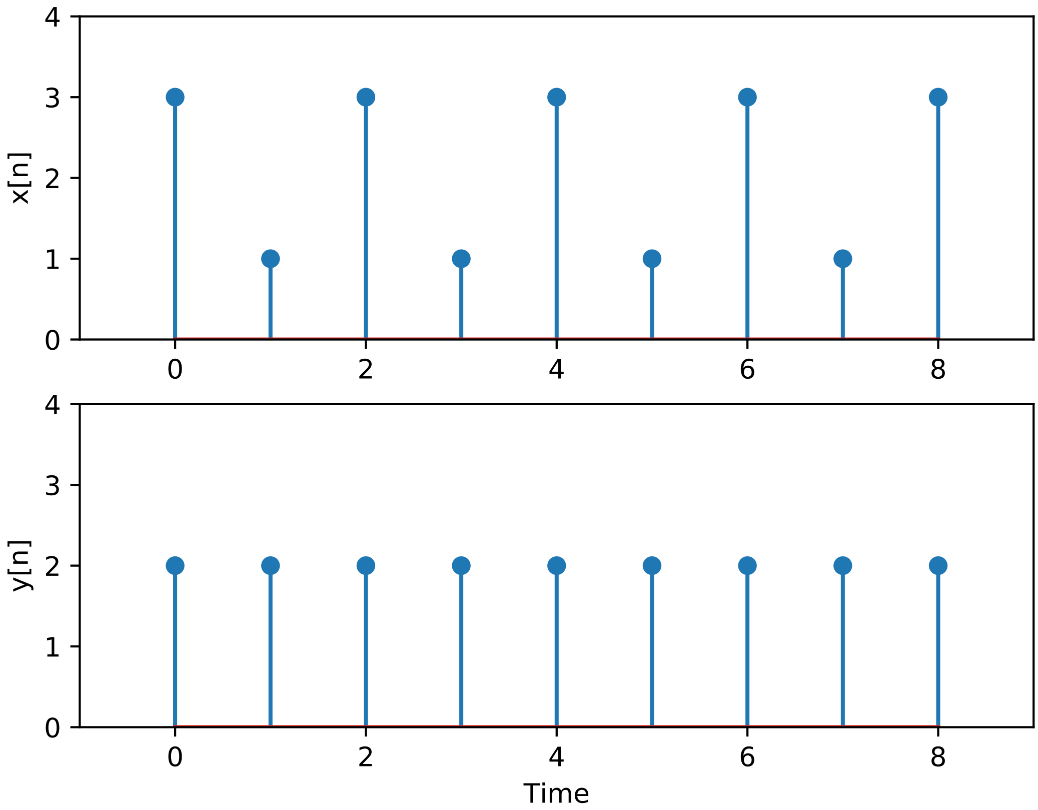
This is just a signal that oscillates between 3 and 1.

$$T : x[n] \mapsto \frac{x[n] + x[n-1]}{2}$$

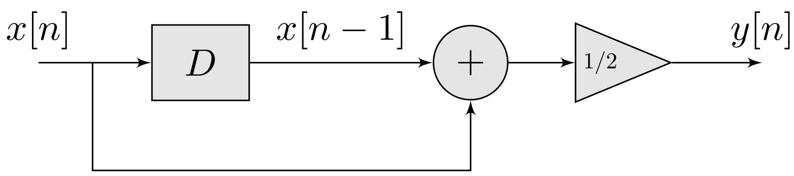
$$y[n] \triangleq T\left(x[n]\right)$$

This transformation has a very simple interpretation: it maps each point of the signal to the average of the current value and the previous value.

The input signal $x[n]$ and the output signal of the system $y[n]$ are plotted in the figure below.



The system can be easily implemented as follows.



The arrows indicate the direction of the data flow. The rectangular D block is called a delay or memory element, and it just delays the incoming signal with one time step. Sometimes, the Greek capital delta (Δ) is used instead, or in some contexts, it is indicated using z^{-1}, for reasons that will become apparent in the page on the Z-transform. The circle with the $+$ is a summator, it just adds all incoming signals together. Finally, the triangle containing a number is a scalar, and it just multiplies the signal with a constant factor.

A possible implementation of this system in Python is given in the code snippet below.
We just have to save the input to the delay element on each time step, because we need it to calculate the next output.

```
from numpy import array, linspace, cos, pi

class ExampleDTLTISystem:
    def __init__(self, initial_state: float = 0.0):
        self.state = initial_state

    def __call__(self, x_n: float) -> float:
        # y[n] = (x[n] + x[n-1]) / 2
        y_n = (x_n + self.state) / 2.0
        # x[n] will be x[n-1] on the next time step,
        # so save it in the system's state
        self.state = x_n
        return y_n

n = linspace(0, 8, 9)         # Create the time variable [0,1,2,…,7,8]
x = cos(pi * n) + 2           # Generate the input signal x[n]

T = ExampleDTLTISystem(1.0)  # Instantiate the system
y = map(lambda x: T(x), x)   # Apply the transformation y[n] = T(x[n])
```

If you're unfamiliar with the `map` function, it is roughly equivalent to the following snippet.

```
y = zeros(x.shape)
for n in range(len(x)):
```

```
    y[n] = T(x[n])
```

It is a good exercise to try to understand how these three representations of the same system are related (the mathematical definition of T, the block diagram, and the Python implementation).