Digital Signal Processing

II. Discrete signals and systems

Representation of discrete signals

A discrete signal can represented:

- graphically
- in table form
- as a vector: $x[n] = [..., 0, 0, 1, 3, 4, 5, 0, ...]$, with an **arrow** indicating the origin of time ($n = 0$). If the arrow is missing, the origin of time is at the first element. The dots ... indicate that the value remains the same from that point onwards

Examples: blackboard

$x[4]$ represents the value of the fourth sample in the signal $x[n]$.

Basic signals

Some elementary signals are presented below.

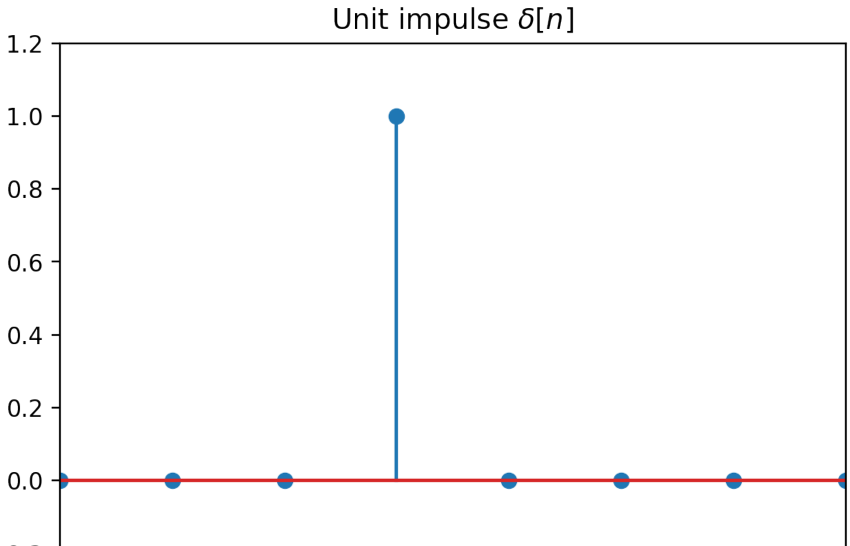
Unit impulse

Contains a single non-zero value of 1 located at time 0. It is denoted with $\delta[n]$.

$$\delta[n] = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{otherwise} \end{cases}.$$

Representation

```
[-3, 4, -0.2, 1.2]
```



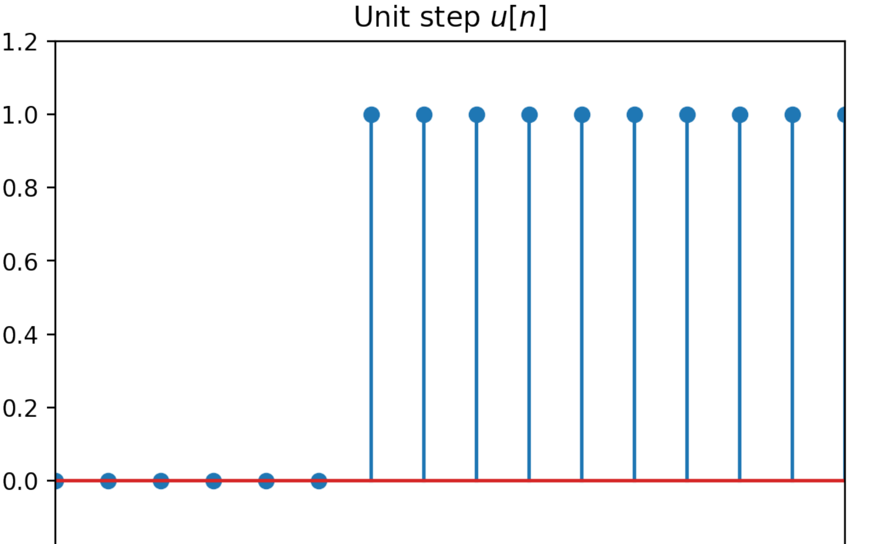
Unit step

- It is denoted with $u[n]$.

$$u[n] = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{otherwise} \end{cases}.$$

Representation

```
[-6, 9, -0.2, 1.2]
```



Unit ramp

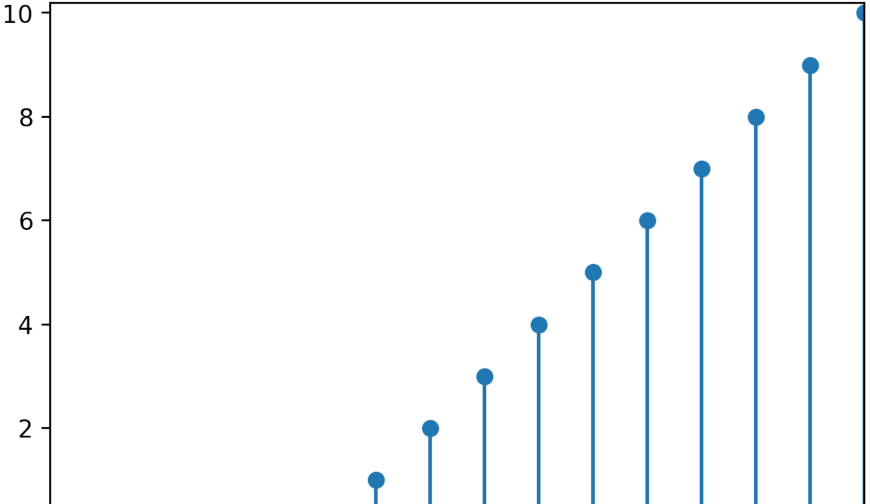
It is denoted with $u_r[n]$.

$$u_r[n] = \begin{cases} n & \text{if } n \geq 0 \\ 0 & \text{otherwise} \end{cases}.$$

Representation

```
[-6, 9, 0, 10.2]
```

Unit ramp $u_r[n]$



Exponential signal

It does not have a special notation. It is defined by:

$$x[n] = a^n.$$

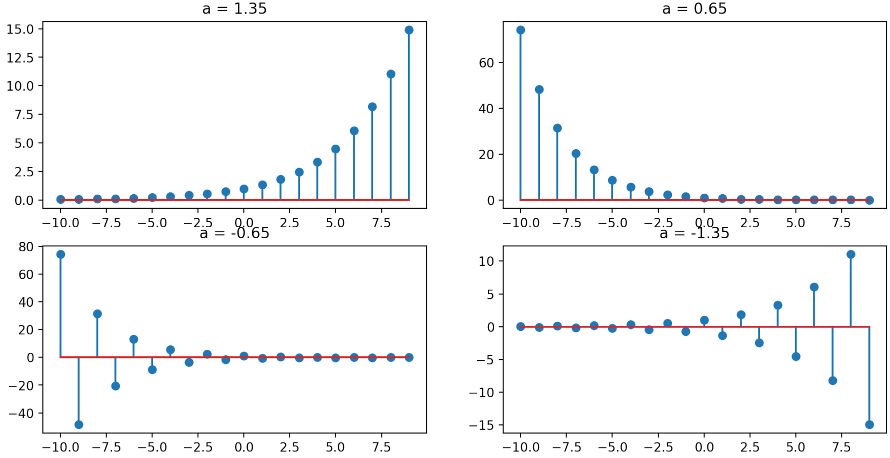
a can be a real or a complex number. Here we consider only the case when a is real.

Depending on the value of a, we have four possible cases:

1. $a \geq 1$
2. $0 \leq a < 1$
3. $-1 < a < 0$
4. $a \leq 1$

Representation

`<matplotlib.text.Text at 0x7f9260b49a58>`



Signals with finite energy

- The **energy of a discrete signal** is defined as

$$E = \sum_{n=-\infty}^{\infty} (x[n])^2.$$

- If E is finite, the signal is said to have finite energy.
- Examples: unit impulse has finite energy; unit step does not.

Signals with finite power

- The **average power of a discrete signal** is defined as

$$P = \lim_{N\to\infty} \frac{\sum_{n=-N}^{N}(x[n])^2}{2N+1}.$$

- In other words, the average power is the average energy per sample.
- If P is finite, the signal is said to have finite power.
- A signal with finite energy has finite power ($P = 0$ if the signal has infinite length). A signal with infinite energy can have finite or infinite power.
- Example: unit step has finite power $P = \frac{1}{2}$ (see proof at blackboard).

Periodic and non-periodic signals

- A signal is called **periodic** if its values repeat themselves after a certain time (known as **period**).

$$x[n] = x[n + N]), \forall t$$

- The **fundamental period** of a signal is the minimum value of N.
- Periodic signals have infinite energy, and finite power equal to the power of a single period.

Even and odd signals

- A signal is **even** if it satisfies the following symmetry:

$$x[n] = x[-n], \forall n.$$

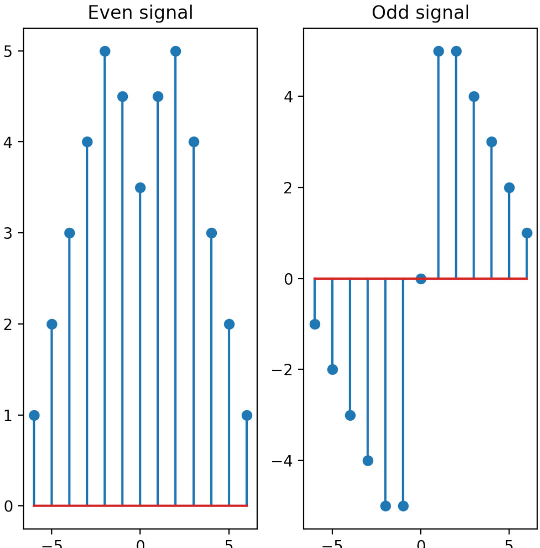
- A signal is **odd** if it satisfies the following anti-symmetry:

$$-x[n] = x[-n], \forall n.$$

- There exist signals which are neither even nor odd.

Even and odd signals: example

```
<matplotlib.text.Text at 0x7f9260ec6550>
```



Even and odd parts of a signal

- Every signal can be written as the sum of an even signal and an odd signal:

$$x[n] = x_e[n] + x_o[n]$$

- The even and the odd parts of the signal can be found as follows:

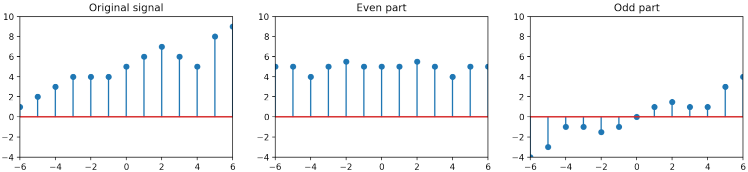
$$x_e[n] = \frac{x[n] + x[-n]}{2}.$$

$$x_o[n] = \frac{x[n] - x[-n]}{2}.$$

- Proof: check that $x_e[n]$ is even, $x_o[n]$ is odd, and their sum is $x[n]$

Even and odd parts: example

```
[-6, 6, -4, 10]
```



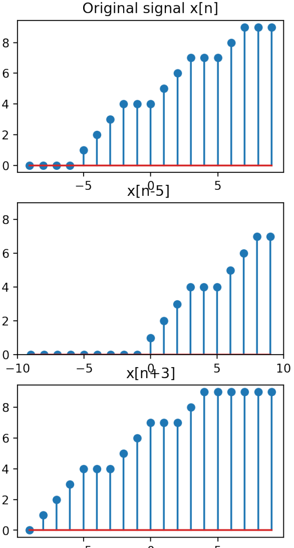
Basic operations - Time shifting

Time shifting:

- Let $x[n]$ be a signal.
- The signal $x[n-k]$ **is** $x[n]$ **delayed with** k **time units**. Graphically, $x[n-k]$ is shifted k units to the **right** compared to the original signal.
- The signal $x[n+k]$ **is** $x[n]$ **anticipated with** k **time units**. Graphically, $x[n+k]$ is shifted k units to the **left** compared to the original signal.

Time shifting: representation

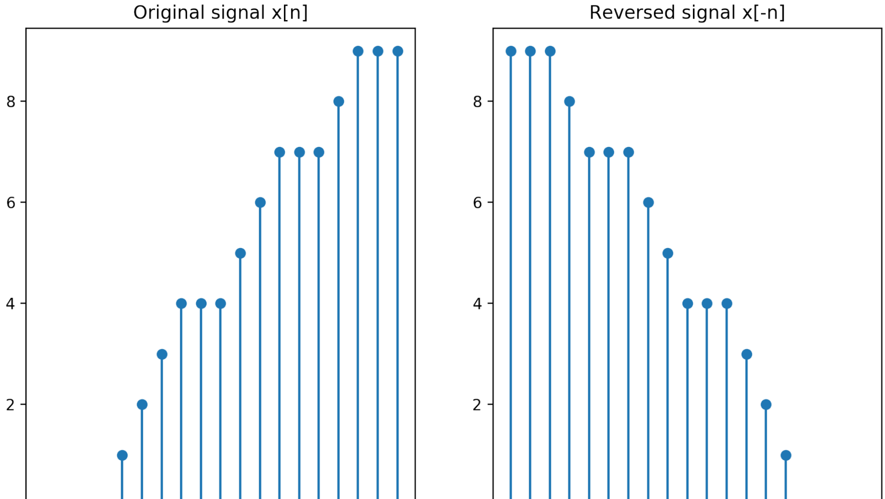
```
<matplotlib.text.Text at 0x7f926036d6d8>
```



Time reversal

Changing the variable n to $-n$ produces a signal $x[-n]$ which mirrors $x[n]$.

```
<matplotlib.text.Text at 0x7f92601b02b0>
```

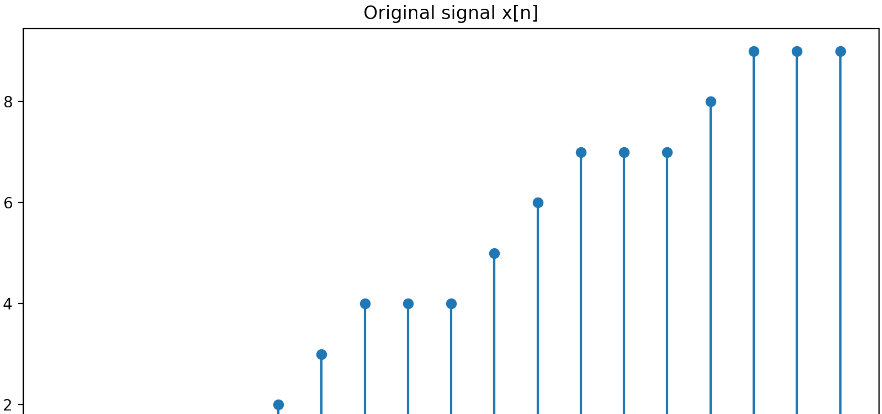


Subsampling

$x_{M\downarrow}[n] = x[Mn]$ is a **subsampled** version of $x[n]$ with a factor of M.

Only 1 sample out of M are kept from the original signal $x[n]$, the rest are discarded.

```
<matplotlib.text.Text at 0x7f9260dafc50>
```



Interpolation

Interpolation by a factor of L adds L of zeros between two samples in the original signal.

$$x_{L\uparrow} = \begin{cases} x[\frac{n}{L}] & \text{if } \frac{n}{L} \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}.$$

```
%matplotlib inline
import matplotlib.pyplot as plt, numpy as np
x1 = [1, 2, 3, 4, 4]
x2 = [1, 0, 0, 2, 0, 0, 3, 0, 0, 4, 0, 0, 4, 0, 0, ]
plt.figure(figsize=(3,4));
plt.stem(x1); plt.title ('Original signal x[n]')
plt.figure(figsize=(9,4));
plt.stem(x2); plt.title ('Interpolated signal by a factor or
```

Mathematical operations

A signal $x[n]$ can be **scaled** by a constant A, i.e. each sample is multipled by A.

$$y[n] = Ax[n].$$

Two signals $x_1[n]$ and $x_2[n]$ can be **summed** by summing the individual samples:

$$y[n] = x_1[n] + x_2[n]$$

Two signals $x_1[n]$ and $x_2[n]$ can be **multiplied** by multiypling the individual samples:

$$y[n] = x_1[n] \cdot x_2[n]$$

Discrete systems

- A **system** is a device or algorithm which produces an **output signal** based on an **input signal**.
- We will only consiuder systems with a single input and a single output.
- Common notation: $x[n]$ is the input, $y[n]$ is the output, H is the system.

Discrete systems - notation

- The relation between the signals can be written as

$$y[n] = H\left[x[n]\right],$$

- Translates as "*the system H applied to the input x[n] produces the output y[n]*".
- It can also be represented as

$$x[n] \xrightarrow{H} y[n],$$

- Translates as "*the input* $x[n]$ *is transformed by the system H into y[n]*".

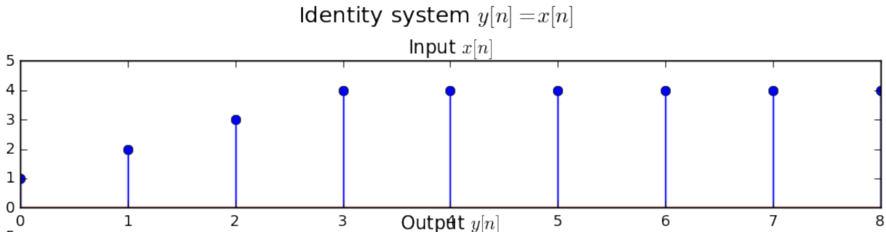
Examples

Examples:

1. $y[n] = x[n]$ (the identity system)
2. $y[n] = x[n-3]$
3. $y[n] = x[n+1]$
4. $y[n] = \frac{1}{3}(x[n+1] + x[n] + x[n-1])$
5. $y[n] = \max(x[n+1], x[n], x[n-1])$
6. $y[n] = (x[n])^2 + \log_{10} x[n-1]$
7. $y[n] = \sum_{k=-\infty}^{n} x[k] = x[n] + x[n-1] + x[n-2] + \ldots$

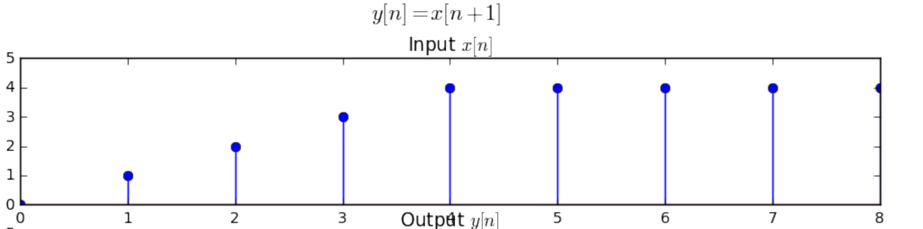
Examples - 1

```
%matplotlib inline
import matplotlib.pyplot as plt, numpy as np
x = [1, 2, 3, 4, 4, 4, 4, 4, 4, 4, 4]
y = [0, 0, 0, 1, 2, 3, 4, 4, 4, 4, 4]
plt.figure(figsize=(10,4));
plt.subplot(2,1,1); plt.stem(x); plt.title ('Input $x[n]$');
plt.subplot(2,1,2); plt.stem(y); plt.title ('Output $y[n]$');
plt.suptitle('Identity system $y[n] = x[n]$', fontsize='x-lar
plt.gcf().subplots_adjust(top=0.85)
```



Examples - 2

```
%matplotlib inline
import matplotlib.pyplot as plt, numpy as np
x = [0, 1, 2, 3, 4, 4, 4, 4, 4]
y = [1, 2, 3, 4, 4, 4, 4, 4, 4]
plt.figure(figsize=(10,4));
plt.subplot(2,1,1); plt.stem(x); plt.title ('Input $x[n]$');
plt.subplot(2,1,2); plt.stem(y); plt.title ('Output $y[n]$');
plt.suptitle('$y[n] = x[n+1]$', fontsize='x-large')
plt.gcf().subplots_adjust(top=0.85)
```



Examples - 3

```
%matplotlib inline
import matplotlib.pyplot as plt, numpy as np
x = [0, 1, 2, 3, 4, 4, 4, 4, 4]
for i in range(len(x)):
    if i==0:
        y[i] = (x[i] + x[i+1])/3.0;    # x[-1] does not exist
    elif i == len(x)-1:
        y[i] = (x[i-1] + x[i])/3.0;    # x[len(x)] does not ex
    else:
        y[i] = (x[i-1] + x[i] + x[i+1])/3.0;
print 'x = ',x
print ('y = ['+', '.join(['%.2f']*len(y))%tuple(y)+']')
plt.figure(figsize=(10,4));
plt.subplot(2,1,1); plt.stem(x); plt.title ('Input $x[n]$');
plt.subplot(2,1,2); plt.stem(y); plt.title ('Output $y[n]$');
plt.suptitle('$y[n] = (x[n+1] + x[n] + x[n-1])/3$', fontsize=
plt.gcf().subplots_adjust(top=0.85)
```

Examples - 4

```
%matplotlib inline
import matplotlib.pyplot as plt, numpy as np
x = [0, 1, 2, 3, 4, 4, 4, 4, 4]
for i in range(len(x)):
    if i==0:
        y[i] = max(x[i], x[i+1])   # x[-1] does not exist in
    elif i == len(x)-1:
        y[i] = max(x[i-1], x[i])   # x[len(x)] does not exist
    else:
        y[i] = max(x[i-1], x[i], x[i+1])
print 'x = ',x
print ('y = ['+', '.join(['%.2f']*len(y))%tuple(y)+']')
plt.figure(figsize=(10,4));
plt.subplot(2,1,1); plt.stem(x); plt.title ('Input $x[n]$');
plt.subplot(2,1,2); plt.stem(y); plt.title ('Output $y[n]$');
plt.suptitle('$y[n] = \max(x[n+1], x[n], x[n-1])$', fontsize=
plt.gcf().subplots_adjust(top=0.85)
```

II.4 Classification of discrete systems

Memoryless / systems with memory

A system is **memoryless (or static)** if the output at some time n depends only on the input **from the same moment** n. Otherwise, the system **has memory (dynamic)**.

Examples:

- memoryless: $y[n] = (x[n])^3 + 5$
- with memory: $y[n] = (x[n])^3 + x[n-1]$

For systems with memory, if the output at time n $y[n]$ depends only the current input and on the last N inputs, $x[n-N], x[n-(N-1)], ...x[n]$, then the system has **memory N**. If N is finite, the system has **finite memory**; if $N = \infty$, the system has infinite memory.
Examples: - finite memory of order 4: $y[n] = x[n] + x[n-2] + x[n-4]$ - infinite memory: $y[n] = \sum_{k=-\infty}^{n} x[k] = x[n] + x[n-1] + x[n-2] + ...$

Time-Invariant and Time-Variant systems

A relaxed system H is **time-invariant** if and only if

Impulse response of Linear Time-Invariant (LTI) systems

Notation: An **LTI** system (**L**inear **T**ime-**I**nvariant) is a system which is simultaneously **linear** and **time-invariant**.

LTI systems can be described via either (or both):

1. the **impulse response** $h[n]$
2. the **difference equation**

$$
\begin{aligned}
y[n] &= -\sum_{k=1}^{N} a_k y[n-k] + -\sum_{k=1}^{M} b_k x[n-k] \\
&= -a_1 y[n-1] - a_2 y[n-2] - \ldots - a_N y[n-N] + b_0 x[n] + b_1 x[n -
\end{aligned}
$$

The impulse response

The **impulse response** of a system is the output (response) of the system when the input signal is the impulse signal $\delta[n]$:

$$h[n] = H(\delta[n]).$$

The impulse response **fully characterizes the system**: based on $h[n]$ we