

## Contents

<b>1 Week 1 Module 1: Basics of Digital Signal Processing</b>	<b>1</b>
1.1 Introduction to digital signal processing . . . . .	1
1.1.1 Signal . . . . .	1
1.1.2 Processing . . . . .	1
1.1.3 Digital . . . . .	2
1.1.4 From Analog to Digital Signal Processing . . . . .	2
1.2 Discrete-Time Signals . . . . .	3
1.2.1 Basic Definitions . . . . .	3
1.2.2 Octave Algorithm for some basic Signals . . . . .	3
1.2.3 Classes of Discrete-Time signals . . . . .	5
1.2.4 Energy and Power . . . . .	7
1.3 Basic signal processing . . . . .	7
1.3.1 How a PC plays discrete-time sounds . . . . .	7
1.3.2 The Karplus Strong Algorithm . . . . .	8
1.4 Digital Frequency . . . . .	9
1.5 The Reproduction Formula . . . . .	10

## 1 Week 1 Module 1: Basics of Digital Signal Processing

### 1.1 Introduction to digital signal processing

#### 1.1.1 Signal

- Description of the evolution of a physical phenomenon

phenomenon	signal
weather	temperature
sound	pressure
sound	magnetic deviation
light intensity	gray level on paper

#### 1.1.2 Processing

- **Analysis:** Understanding the information carried by the signal
- **Synthesis:** Creating a signal to contain the given information

### 1.1.3 Digital

- Discrete Time
  - Splice up time into a series of discrete instance without losing information
  - Harry Nyquist and Claude Shannon state with the [Sampling Theorem](#) that continuous time representation and discrete time representation are equivalent.
  - The Sampling Theorem: Under appropriate "slowness" conditions for  $x(t)$  we have

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \operatorname{sinc}\left(\frac{t - nT_s}{T_s}\right) \quad (1)$$

- The condition under which the Sampling Theorem holds was given by Fourier and it's [Fourier Analysis](#).
  - The Fourier transform will give us a quantitative measure how fast a signal moves
- Discrete Amplitude
  - Through discretisation of amplitudes only a set of predefined values are possible.
  - The set of levels is countable i.e. we can always map the level of a sample to an integer. If our data is represented by integer it becomes complete abstract and general which has very important consequences in the following three domains:
    - \* **Storage** special devices for recoding needed
    - \* **Processing** General purpose microprocessor is sufficient
    - \* **Transmission** Reproduction of the original signal and therefore eliminating noise is easy

### 1.1.4 From Analog to Digital Signal Processing

- Analog asks for  $f(t)$  =?
- Digital represents data as a sequence of numbers (scaled with a factor of 1000)

-12 -12 -12 -11 -11 -12 -12 -11 -11 -10

-10 -10 -9 -10 -10 -9 -9 -9 -9 -9  
 -8 -8 -7 -7 -8 -8 -8 -7 -7 -7

## 1.2 Discrete-Time Signals

### 1.2.1 Basic Definitions

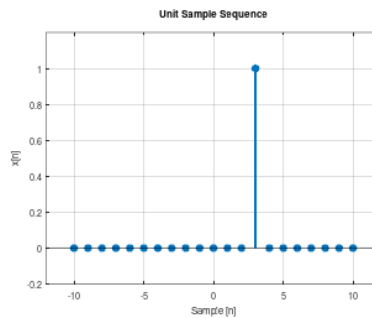
- Sequence: defined as [complex-valued function](#)
- Discrete-Time Signal: a sequence of complex numbers
  - one dimension (for now)
  - notation:  $x[n]$
  - two-sided sequences:  $x: \mathbb{Z} \rightarrow \mathbb{C}$
  - $n$  is *a-dimensional* "time", sets an order on the sequence of samples
  - analysis: periodic measurement
  - synthesis: stream of generated samples, reproduce a physical phenomenon

### 1.2.2 Octave Algorithm for some basic Signals

#### Unit Impulse

```
function [x,n] = impseq(n0,n1,n2)
% Generates  $x(n) = \delta(n-n0)$ ;  $n1 \leq n0 \leq n2$ 
% -----
%  $[x,n] = \text{impseq}(n0,n1,n2)$ 
%
  n = [n1:n2]; x = [(n-n0) == 0];
end
```

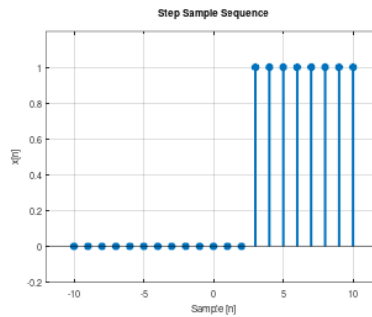
$$x[n] = \delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$



## Unit Step

```
function [x,n] = stepseq(n0,n1,n2)
% Generates x(n) = delta(n-n0); n1 <= n0 <= n2
% -----
% [x,n] = stepseq(n0,n1,n2)
%
n = [n1:n2]; x = [(n-n0) >= 0];
end
```

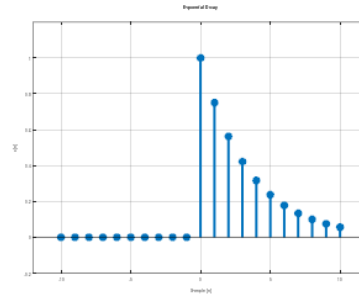
$$x[n] = u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$



## Real-valued exponential Sequence

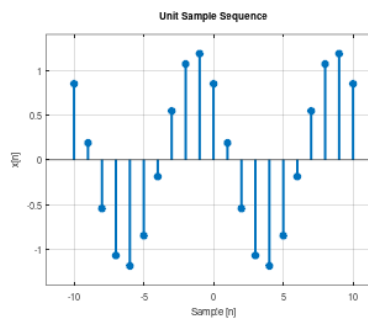
```
function [x,n] = expseq(n1,n2,a)
% Generates x(n) = a^n
% -----
% [x,n] = expseq(n1,n2,A,omega,phi)
%
n = [n1:n2];
for (i = 1 : length(n))
    if (n(i) >= 0)
        x(i) = (a).^n(i);
    else
        x(i) = 0;
    end
end
end
```

$$x[n] = a^n, \forall n \in \mathbb{Z}$$



## Sinusoidal Sequence

```
function [x,n] = cosseq(n1,n2,A, omega, phi)
% Generates x(n) = A*cos(2*pi*omega*n + phi); n1 <= n2
% -----
% [x,n] = cosseq(n1,n2,A,omega,phi)
%
n = [n1:n2]; x = A*cos(2*pi*omega*n + phi);
end
```



$$x[n] = A \cos(\omega_0 n + \Phi)$$

### 1.2.3 Classes of Discrete-Time signals

#### 1. Finite-Length

- indicate notation:  $x[n], n = 0.1.2.....N - 1$
- vector notation:  $x = [x_0, x_1, ...x_{N-1}]^T$
- practical entities, good for numerical packages (e.g. numpy)

#### 2. Infinte-Length

- sequence notation:  $x[n], n \in \mathbb{Z}$
- abstraction, good for theorems

### 3. Periodic

- N-periodic sequence:  $\tilde{x}[n] = \tilde{x}[n + kN]$ ,  $n, k, N \in \mathbb{Z}$
- same information as in finite-length of length N
- [natural bridge](#) between finite and infinite length

### 4. Finite-Support [Finite-support sequence](#)

$$\bar{x}[n] = \begin{cases} x[n] & \text{if } 0 \leq n < N, n \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

- same information as in finite-length of length N
- another bridge between finite and infinite lengths

### 5. Elementary Operations

#### Scaling

$$y[n] = ax[n] \rightarrow \begin{cases} a > 0 & \text{amplification} \\ a < 0 & \text{attenuation} \end{cases} \quad (3)$$

#### Sum

$$y[n] = x[n] + z[n] \quad (4)$$

#### Product

$$y[n] = x[n] * z[n] \quad (5)$$

#### Shift

$$y[n] = x[n - k] \rightarrow \begin{cases} k > 0 & \text{deleay} \\ k < 0 & \text{anticipate} \end{cases} \quad (6)$$

#### Integration

$$y[n] = \sum_{k=-\infty}^n x[k] \quad (7)$$

#### Differentiation

$$y[n] = x[n] - x[n - 1] \quad (8)$$

[logo=,couleur=yellow!10,barre=snake,arrondi=0.1]Relation Operator and Signals

- The [unit step](#) can be obtained by applying the [integration](#) operator to the [discrete time pulse](#).
- The [unit impulse](#) can be obtained by applying the [differentiation](#) operator to the [unit step](#).

### 1.2.4 Energy and Power

**Energy** Many sequences have an infinity amount of energy e.g. the unit step  $u[n]$ ,

$$E_x = \|x\|_2^2 = \sum_{k=-\infty}^{\infty} |x[n]|^2 \quad (9)$$

**Power** To describe the energetic properties of the sequences we use the concept of power

$$P_x = \|x\|_2^2 = \frac{1}{N} \sum_{n=0}^{N-1} |x[n]|^2 \quad (10)$$

Many signals have infi

## 1.3 Basic signal processing

### 1.3.1 How a PC plays discrete-time sounds

1. The discrete-time sinusoid

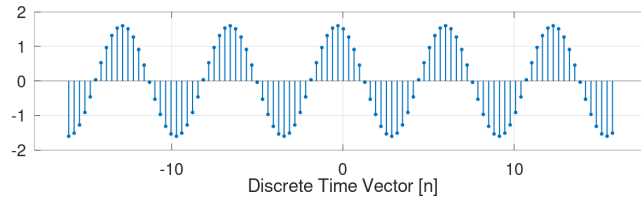
$$x[n] = \sin(\omega_0 t + \Theta)$$

```
N=33                                # Vector lenght
n=-(N-1)/2:pi/10:(N-1)/2; # Discrete Time Vector
omega0 = pi/10;
theta = pi/2

f = 1.6*sin(omega0+n + theta); # The sinusoid

# Do not open the graphic window in org
figure( 1, "visible", "off");

stem(n,f, "filled", "linewidth", 2, "markersize", 6);
axis([- (N-1+4)/2 (N-1+4)/2 -2 2])
set(gca, "fontsize", 24);
grid on ;
xlabel("Discrete Time Vector [n]");
print -dpng "-S1400,350" ./image/sin.png;
# Org-Mode specific output
ans = "./image/sin.png";
```



## 2. Digital vs physical frequency

- Discrete Time:
  - Periodicity: how many samples before the pattern repeats (M)
  - n: no physical dimension
- Physical World:
  - Periodicity: how many seconds before the pattern repeats
  - frequency measured in Hz
- Soundcard  $T_s$  System Clock
  - A sound card takes every  $T_s$  a new sample from the discrete-time sequence.
  - periodicity of M samples  $\rightarrow$  periodicity of  $M T_s$  seconds
  - real world frequency

$$f = \frac{1}{M T_s} \text{ Hz} \quad (11)$$

- Example
  - usually we choose  $F_s$  the number of samples per seconds
  - $T_s = 1/F_s$

$$F_s = 48000 \text{ e.g. a typical value}$$

$$T_s = 20.8 \mu s$$

$$f = 440 \text{ Hz, with } M = 110$$

### 1.3.2 The Karplus Strong Algorithm

#### 1. The Moving Average

- simple average (2 point average)

$$m = \frac{a + b}{2} \quad (12)$$



- moving average: take a "local" average

$$y[n] = \frac{x[n] + x[n-1]}{2} \quad (13)$$

- Average a sinusoid

$$\begin{aligned} x[n] &= \cos(\omega n) \\ y[n] &= \frac{\cos(\omega n) - \cos(\omega (n-1))}{2} \\ y[n] &= \cos(\omega n + \theta) \end{aligned}$$

[couleur=yellow!10, arrondi = 0.1, logo=, ombre=true]Linear Transformation Applying a linear transformation to a sinusoidal input results in a sinusoidal output of the same frequency with a phase shift.

## 2. Reversing the loop

$$y[n] = x[n] + \alpha y[n-1] \rightarrow \text{The Karplus Strong Algorithm} \quad (14)$$

- **Zero Initial Conditions:**

- set a start time (usually  $n_0 = 0$ )
- assume input and output are zero for all time before  $N_0$

## 1.4 Digital Frequency

[logo=,couleur=yellow!10,barre=snake]Digital Frequency

$$\begin{aligned} \sin \left( n(\omega + 2k\pi) \right) &= \sin(n\omega + \phi), k \text{ in } \mathbb{Z} \\ &= e^{i(\phi + n*2\pi\omega)} \end{aligned} \quad (15)$$

[logo=,couleur=yellow!10, barre=snake]Complex Exponential

$$\omega = \frac{M}{N} \times 2 \times \pi \quad (16)$$

## 1.5 The Reproduction Formula

[logo=, couleur=yellow!10, barre=snake, arrondi=0.1]Reproduction Formula

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \quad (17)$$

Any **signal** can be expressed as a linear combination of wighted and shifted pulses.