- iec 234
- Digital Signal Processing — Proakis & Manolakis
- Digital Signal Processing — A computer science perspective Jonathan
- Signalprocessing society dspguide.com.

---

* Signals

- Analog → Digital
  - ↳ Sampling → Quantization

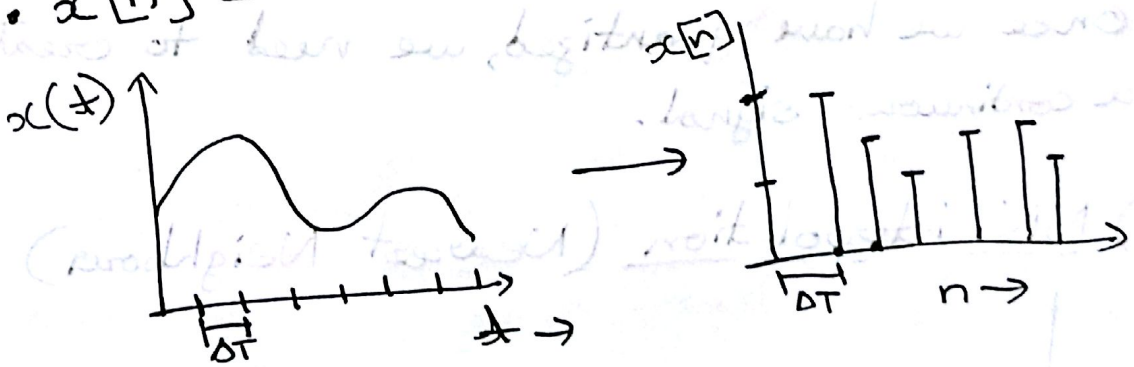
- Why digital?
  - ⇒ Storage
  - ⇒ Reproduce signal } Become easier
    (Analog data can deteriorate)
  - ⇒ Modification of signal is easy.
    (Flexibility & Modularity)

⇒ Disadvantage is that digital signals are slow.

---

* ~~Digital to~~ Analog to Digital; (Sampling)

- $x[n] = x(n\Delta T)$



- $f_s = \frac{1}{\Delta T}$ } Sampling Frequency , $\omega_s = \frac{2\pi}{\Delta T}$

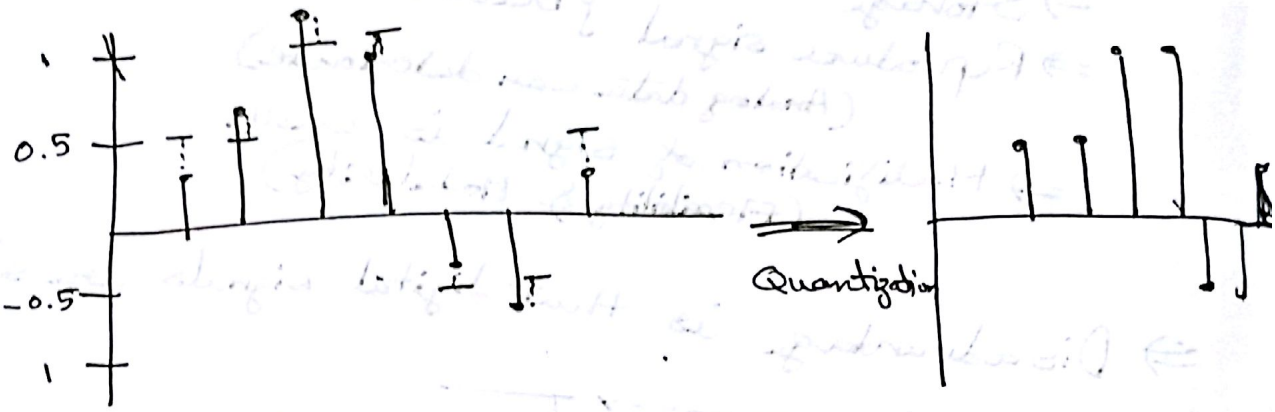
Scanned by CamScanner

Shannon Frequency of Sampling: (Nyquist Frequency)

- If our max frequency of the signal is F_{max}, then we want $F_s \geq 2 \times F_{max}$

Quantization:

- We approximate the value of the signal rounding it off to the values allowed by the precision (levels) we allow.

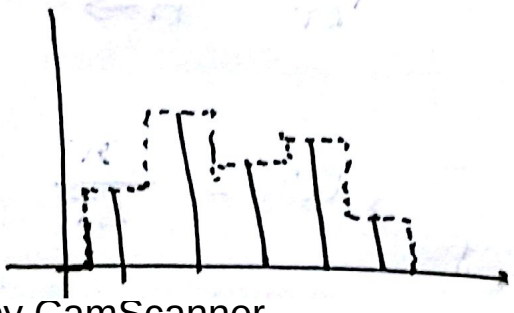


- Dividing y axies into equal intervals.

Interpolation:

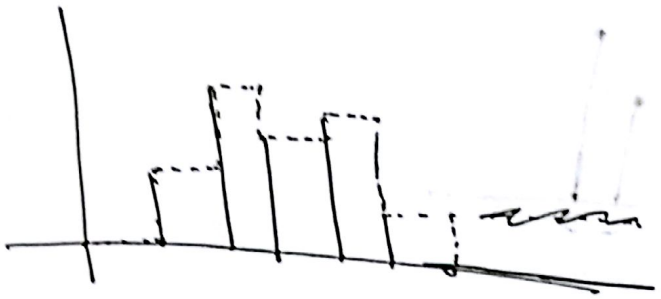
- Once we have quantized, we need to create a continuous signal.

1) NN interpolation (Nearest Neighbour)



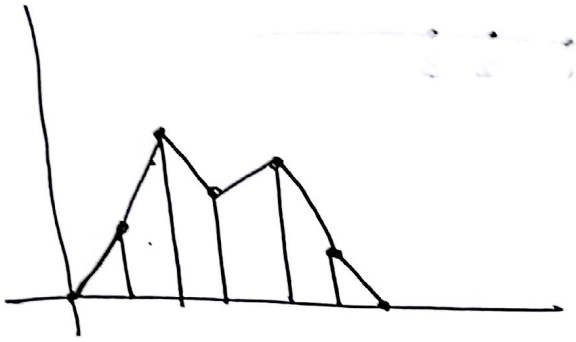
Scanned by CamScanner

2) Zero Order hold;



- Keep using the previous value until you encounter a new value.

3) Linear interpolation



- Linearly join our values (we use convolution functions here to get various linear interpolations).

———x———

*Basic Operations;

1) Flipping: ($f(-x)$ becomes $f(x)$) $x[n] \rightarrow x[-n]$

2) Shifting: ($f(x) = f(x-2)$, $x[n] \rightarrow x[n-2]$)

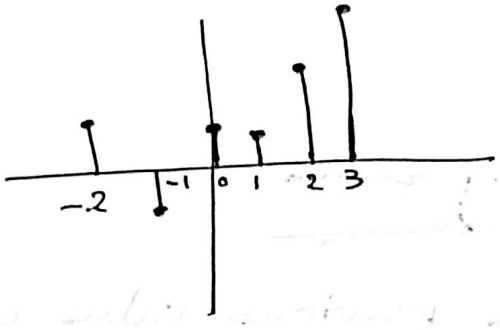
3) Scaling: ($f(x) = f(2n)$, $x[n] \rightarrow x[2n]$)

↳ We will lose data if not continuous functions when we scale.

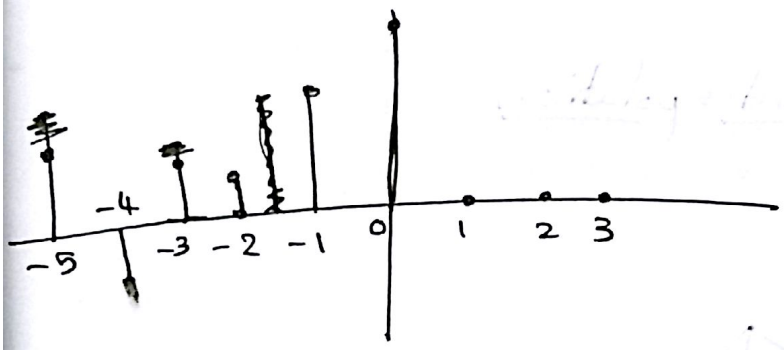
*** Order: Shift → Flip → Scale } Hierarchy of operations (BODMAS)

Scanned by CamScanner

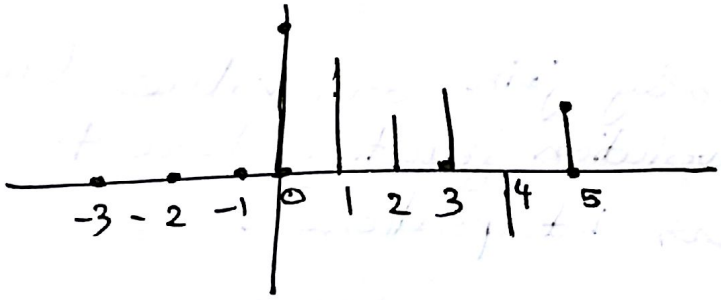
ex: x[-2(n+3)] on given signal, x[-2n+3]



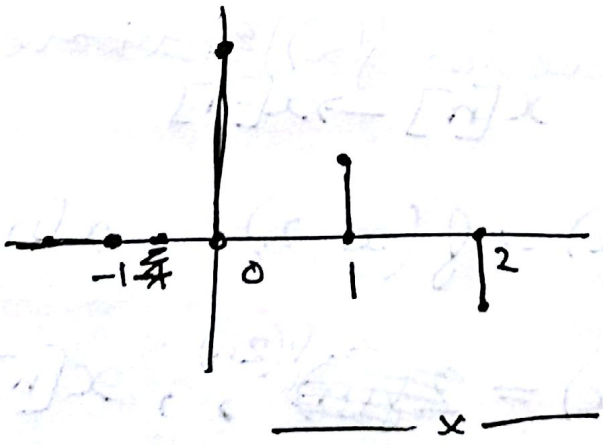
First shift,



Then flip,



Then scale,



Scanned by CamScanner

Characteristics of Signals;

1) Even → $x(n) = x(-n)$
   Odd → $x(n) = -x(-n)$

- Any function can be represented as sum of even & odd functions.

* $x[n] = \underbrace{\frac{1}{2}[x(n)+x(-n)]}_{\text{Even part of } x[n]} + \underbrace{\frac{1}{2}[x(n)-x(-n)]}_{\text{odd part of } x[n]}$

2) Periodic functions;

$x[n] = x[n+N]$

** 3) Energy | Power signal;

$$E_x = \sum_{-\infty}^{\infty} |x[n]|^2$$

$$P_{xc} = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{-N}^{N} |x[n]|^2$$

- If E_x is finite, $P_{xc} = 0$ //

---

ex: $u[n] = \begin{cases} 1 & \text{for } n \geq 0 \\ 0 & \text{otherwise} \end{cases}$

$E_x = \infty$

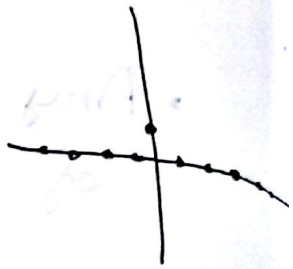
$P_x = \lim_{N\to\infty} \frac{1}{2N+1}(N+1) = \frac{1}{2}$ //

Scanned by CamScanner

*Special Signals:

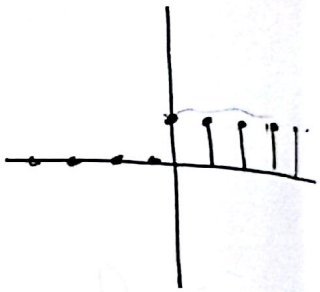
1) Delta:

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



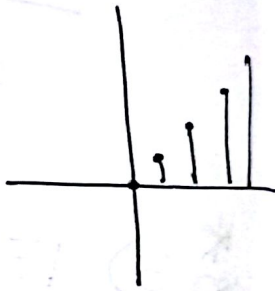
2) Unit Step:

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



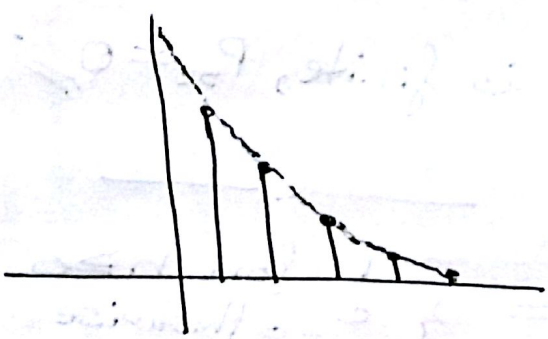
3) Ramp:

$$r[n] = \begin{cases} n & , n \geq 0 \\ 0 & , \text{otherwise} \end{cases}$$



4) Exponential:

$$|a|^n u[n] \quad , 0 < |a| < 1$$



Scanned by CamScanner

Note: $u[n] = \sum_{k=0}^{\infty} \delta[n-k] = \sum_{k=-\infty}^{n} \delta[k]$

$\delta[n] = u[n] - u[n-1]$

$\left(\delta(t) = \frac{d}{dt} u(t) \rightarrow \text{If its continuous}\right)$

— x —

Note: Generalized function,

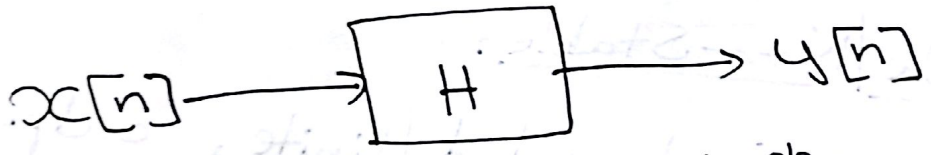
$$x[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot \delta[n-k]$$

$x[n] = \ldots\ldots + a_{-2}\delta[n+2] + a_{-1}\delta[n+1] + a_0\delta[n] + a_1\delta[n-1] + a_2\delta[n-2] + \ldots$

$\Rightarrow a_k = x[k]$ //

— x —

System:



- Filter $\Rightarrow y[n] = \frac{1}{p} \sum_{k=-p/2}^{k=p/2} x[n+k]$

$\Rightarrow$ So for every n, we approximate it to neighbours.

$\Rightarrow$ averaging filter ex: $\frac{1}{5}[\underbrace{1\ 1\ 1\ 1\ 1}_{n-2\ \ n-1\ \ n\ \ n+1\ \ n+2}]$

Window which is used so as to average each point.

This would reduce variations.

— x —

Scanned by CamScanner

* Properties of Systems:

1) Static: (memoryless)

- If we don't need to store previous / later values,

  ex: $y[n] = 7x[n]$

$\Rightarrow y[n] = 3x[n-1]$ is not a static system

2) Causal / Non-causal:

- Output can use values of past but not the future.

  ex: $y[n] = 7x[n-1] + 3x[n-2]$

$\Rightarrow y[n] = x[n+1]$ is not causal.

3) Stable / Non-Stable:

- If input is bounded / finite, output is bounded / finite.

$\Rightarrow$ This is a stable system.

*4) Linear:

- A system is linear if its additive & homogeneous.

  $x_1[n] \xrightarrow{H} y_1[n]$ & $x_2[n] \xrightarrow{H} y_2[n]$

  then

a) Additivity: $x_1[n] + x_2[n] \xrightarrow{H} y_1[n] + y_2[n]$

Scanned by CamScanner

b) Homogenity: $x_1[n] \xrightarrow{H} y_1[n]$

$a \cdot x_1[n] \xrightarrow{H} a\,y_1[n]$

—— x ——

esc: $y[n] = 2x[n] + 7$

$x_1[n] \longrightarrow 2x_1[n] + 7 = y_1[n]$

$x_2[n] \longrightarrow 2x_2[n] + 7 = y_2[n]$

Let $z[n] = x_1[n] + x_2[n]$

$z[n] \longrightarrow \boxed{H} \longrightarrow 2z[n] + 7$

$= 2x_1[n] + 2x_2[n] + 7$

$\neq y_1[n] + y_2[n]$

∴ Non linear.

—— x ——

5) Time invariance: (True only if coefficient of n is 1 & power = 1 for the system) ie: $y[n] = x[an+b]$ where $a=1$.

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

Then, $x[n-n_0] \xrightarrow{H} y[n-n_0]$

⇒ Then the system is time invariant

—— x ——

ex: $y[n] = x[n^2]$ → Not time invariant

$\therefore y[n-n_0] \neq x[(n-n_0)^2]$

Now lets consider $z[n] = x[n-n_0] \rightarrow ①$

Lets check if output $= y[n-n_0]$ or not.

$\Rightarrow z[n] \longrightarrow \boxed{H} \longrightarrow z[n^2]$.

From ①, $z[n^2] = x[n^2 - n_0]$

$\neq y[n-n_0]$

—— x ——

Scanned by CamScanner

esc: $y[n] = x[n] - 3x[n-1]$

*

⇒ This is time invariant since $an+b$, $a=1, b=0$
$a=1, b=-1$
∴ a is 1 in both cases
∴ Its time invariant

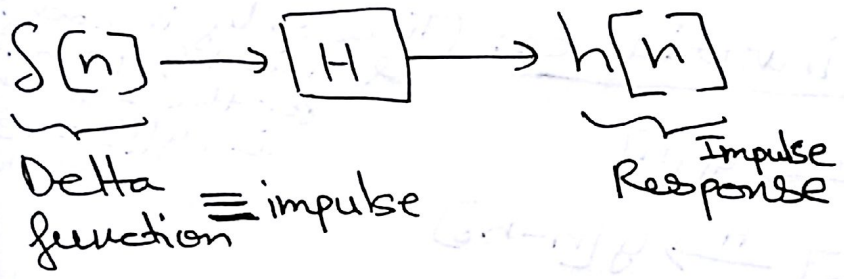
———x———

*LTI System:

- If a system is Linear & Time Invariant, its called an LTI system.

———x———

*Impulse Response:



⇒ An impulse response, if we know it (For an LTI) for all n, then we can find the response of the system for any input.

$\delta[n] \xrightarrow{LTI} h[n]$

$\delta[n-k] \xrightarrow{LTI} h[n-k]$

$\alpha_x \delta[n-k] \xrightarrow{LTI} \alpha_x h[n-k]$

$x[k]\delta[n-k] \xrightarrow{LTI} x[k]h[n-k]$

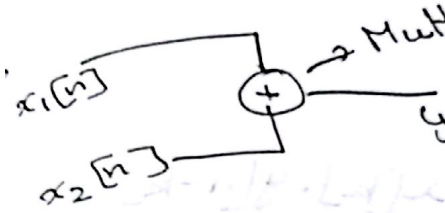
$\Rightarrow \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \xrightarrow{LTI} \sum_{k=-\infty}^{\infty} x[k]h[n-k]$

$\Rightarrow \therefore x[n] \xrightarrow{LTI}$

———x———

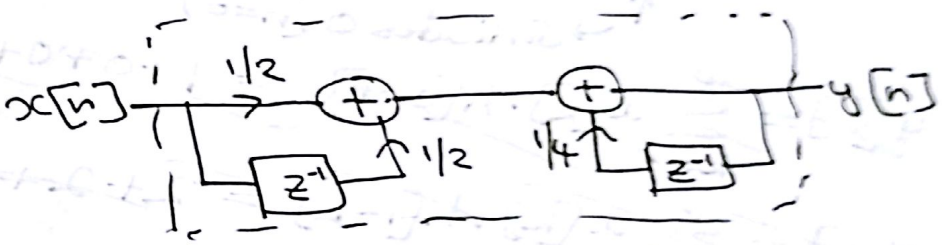
Scanned by CamScanner

Syntax: (Operations on signals representation)



- $x[n] \xrightarrow{a} y[n] = a\,x[n]$
- $x[n] \xrightarrow{z^{-1}} y[n] = x[n-1]$
- $x[n] \xrightarrow{z} y[n] = x[n+1]$

Note:



$$y[n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1] + \frac{1}{4}y[n-1]$$

Continuing impulse response,

$\Rightarrow$ ∴ We can see,

Treat like constants

$$H(x[n]) = H\left(\sum \overbrace{x[k]}\,\delta[n-k]\right)$$

$$= \sum x[k] \cdot H(\delta[n-k])$$

$$= \sum x[k] \cdot h[n-k]$$

$$\Rightarrow y[n] = H(x[n]) = \underbrace{\sum_{-\infty}^{\infty} x[k]h[n-k]}_{\text{Convolution sum}}$$

$$y[n] = \underbrace{x[n] \ast h[n]}_{\text{Convolution operator.}}$$

Scanned by CamScanner

* Convolution:

$\Rightarrow x[n]$ & $y[n]$

$$\Rightarrow conv(x[n], y[n]) = \sum_{k=-\infty}^{\infty} x[k] \cdot y[n-k]$$

- This is the convolution operator.

—x—

**

exc: $h[n] = [1 \underset{\uparrow}{2} 1 \; -1]$ } Impulse response

$x[n] = [\underset{\uparrow}{1} \; 2 \; 3 \; 1]$ } Input signal

↑ → Indicates 0 ($n=0$)

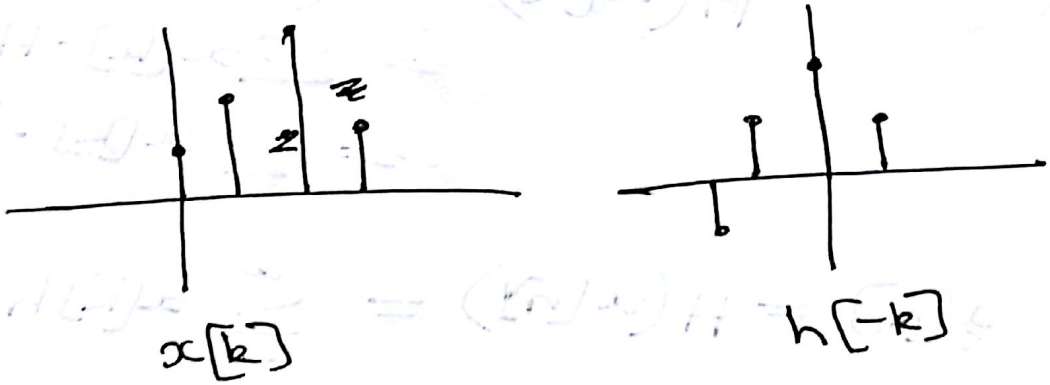
$$y[0] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[0-k] = 1+0+0+0 = 1$$

$$y[1] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[1-k] = 2+2+0+0 = 4$$

$$y[2] = \sum_{k=-\infty}^{\infty}$$

Let $y[n] = x[n] * h[n]$ (Convolution)

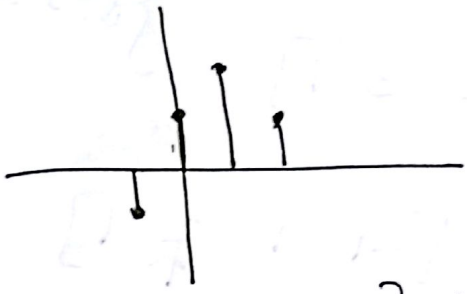
$$\Rightarrow y[0] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[0-k]$$



$$y[0] = 2+2 = 4 //$$

Scanned by CamScanner

$$\Rightarrow y[1] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[1-k]$$



$h[1-k]$ } shift first, then scale/flip

$\therefore y[1] = 1+4+3 = 8$ //

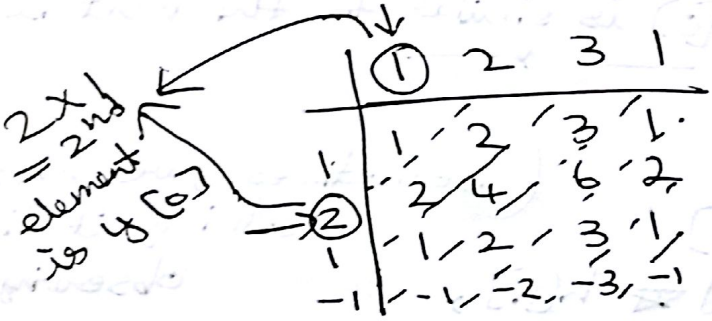
Similarly, find
$\Rightarrow y[2], y[3], \ldots$

—— x ——

* ** * **Shortcut for convolution;**

$h[n] = [1 \underset{\uparrow}{2} 1 \; -1]$

$x[n] = [\underset{\uparrow}{1} \; 2 \; 3 \; 1]$



$\Rightarrow$ Sum of diagonals are the outputs,

$y[n] = [\ldots \; 1 \; \underset{\uparrow}{4} \; 8 \; 8 \; 3 \; -2 \; -1]$

2x1 = 2'nd element

y[0] is this 2nd element

— x —

Scanned by CamScanner

** Shortcut 2 for convolution,

** $h[n] = [1 \;\; \underset{\uparrow}{2} \;\; 1 \;\; -1]$

$x[n] = [\underset{\uparrow}{1} \;\; 2 \;\; 3 \;\; 1]$

} $y[0]$ is 2 X 1 element. = 2nd element

⇒ Flip $h[n] \Rightarrow [-1 \;\; 1 \;\; \underset{\uparrow}{2} \;\; 1]$ } Consider this as a filter

Now consider,

$x[n] \Rightarrow [0 \;\; 0 \;\; 0 \;\; 1 \;\; 2 \;\; 3 \;\; 1 \;\; 0 \;\; 0 \;\; 0]$

$[-1 \;\; 1 \;\; 2 \;\; 1]$

Start with rightmost element of $h[n]$ flip, with the 1st element of $x[n]$.

Start from 1st non zero value (ie atleast one element is nonzero in $x[n]$, total ans may be 0)

$y[n] \Rightarrow [\ldots\ldots\ldots 8 \ldots\ldots]$

Since the filter is position at the 4th element of $x[n]$ ← $y[2]$

So, wherever the arrow points, the value at that $y[n]$ is the result of the filter from that point.

Finding $y[0]$ is similar to the matrix method.

---

eg: $x[n] = a^n$

$h[n] = u[n]$

(Sometimes, we can try solving it directly) observing pattern

Convolution, $y[n] = x[n] * h[n]$,

$y[0] = \sum x[k] \cdot h[-k] = 1$

$y[1] = \sum x[k] \cdot h[1-k] = 1+a$

$\vdots$

$y[2] = 1+a+a^2$

$\vdots$

$y[\infty] = \frac{1}{1-a}$

Scanned by CamScanner

* Properties of Convolution:

- $y[n] = x[n] \star h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$

(1) Commutative:

$y[n] = x[n] \star h[n] = h[n] \star x[n]$

Proof:

Let, $n-k=m$

$k = n-m$,

$$\sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] = \sum_{m=-\infty}^{\infty} \underbrace{x[n-k]}_{x[m]} \cdot \underbrace{h[k]}_{h[n-m]}$$

∴ Its commutative

(2) Distributive:

$$x[n] \star [h_1[n] + h_2[n]] = x[n] \star h_1[n] + x[n] \star h_2[n]$$

(3) Associative:

$$(x[n] \star h_1[n]) \star h_2[n] = x[n] \star (h_1[n] \star h_2[n])$$

---

* Correlation:

- Similar to convolution, but here,

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n+k]$$

- To find the correlation, use 'shortcut 2' for convolution (on prev page), except that you don't flip $h[n]$ before starting to multiply.

Scanned by CamScanner

*Side: Auto-correlation: (correlation with itself)

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot x[n+k]$$

—x—

*2D Convolution: (Image processing)

**

- Flip h $_{w \times w}$ over both x & y axis.
- Then similar to the 1D method, keep going over all the points of $x_{m \times n}$ and generate output $y_{p \times q}$.
- Using some point of the filter h as the center get y similar to normal convolution

—x—

*c: An averaging filter (i.e. all 1's) would give us a blurred image. → In MATLAB, the values of the filter are all divided by n×m (rows & cols of filter)

So a filter of all 1's is an averaging filter.

—x—

c: Gaussian filter: $h[k] = A e^{\frac{-(k-\mu)^2}{2\sigma^2}}$, $A=1$, $\mu=0$, $\sigma=1$

$$h[k] = e^{-k^2/2}$$

$$\Rightarrow \text{In 2D}, \; h[x,y] = A e^{-\left[\frac{(x-\mu_x)^2}{2\sigma_x^2} + \frac{(y-\mu_y)^2}{2\sigma_y^2}\right]}$$

—x—

*Statistical Measures:

- Mean, standard deviation etc.

Scanned by CamScanner

1) Mean:

$$\mu_x = \frac{1}{N}\sum_{i=0}^{N-1} x[i]$$

2) Standard deviation & Variance:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x[i]-\mu)^2$$

Whenever we use a sample out of larger set, we divide by N-1.

$$\Rightarrow \sigma^2 = \frac{1}{N-1}\left[\sum_{i=0}^{N-1}(x[i])^2 + \mu^2 - 2\mu x[i]\right]$$

$$= \frac{1}{N-1}\left[\sum(x[i])^2 + \sum\mu^2 - 2\mu\sum x[i]\right]$$

$$= \frac{1}{N-1}\left[\sum(x[i])^2 + N\mu^2 - 2\mu N\frac{\sum x[i]}{N}\right]$$

$$= \frac{1}{N-1}\left[\sum(x[i])^2 + N\mu^2 - 2\mu^2 N\right]$$

$$\sigma^2 = \frac{1}{N-1}\left[\sum(x[i])^2 - \mu^2 N\right]$$

$$= \frac{1}{N-1}\left[\sum(x[i])^2 - N\left(\frac{\sum x[i]}{N}\right)^2\right]$$

$$= \frac{1}{N-1}\left[\sum(x[i])^2 - \frac{1}{N}\left(\sum x[i]\right)^2\right]$$

$$\sigma^2 = \frac{1}{N-1}\left[\text{sum of squares} - \frac{(\text{sum})^2}{N}\right]$$

* $\Rightarrow$ ∴ We can see that if we add another sample to the system, we can easily get σ^2 → running statistics.

Scanned by CamScanner

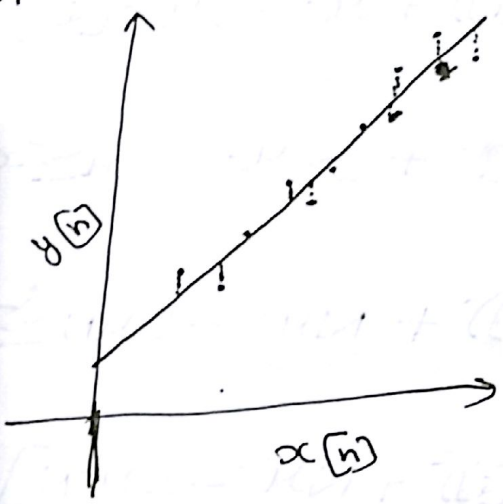
* Histogram: → Does not capture distribution!

- Variable vs Frequency
- In a normalized histogram, we plot against normalized frequency → Probability Mass function (PMF).

⇒ Histograms of images only store values, not location of occurence of intensity of pixels etc, so data is lost.

---

* Linear Regression: (Ordinary least squares)



$$\hat{y}[n] = \underbrace{A}_{\text{Intercept}} + \underbrace{B}_{\text{Slope}} x[n]$$

(approximated)

- $e[k] = \left[y[k] - (A + Bx[k]) \right]^2$

(e[k]: error)

$$\Rightarrow E = \sum_{k=0}^{N-1} e[k] = \sum_{k=0}^{N-1} \left(y[k] - (A + Bx[k]) \right)^2$$

⇒ we try to minimize this error E.

$$\Rightarrow \frac{dE}{dA} = 0 \quad \& \quad \frac{dE}{dB} = 0$$

$$\Rightarrow \frac{dE}{dA} = -2 \sum (y[k] - A - Bx[k]) = 0$$

~~$\frac{dE}{dB}$~~

$$\Rightarrow \sum y[k] - NA - B \sum x[k] = 0$$

$$\Rightarrow A = \frac{\sum y[k]}{N} - \frac{B \sum x[k]}{N} \rightarrow (1)$$

$$\boxed{A = \mu_y - B\mu_x} \rightarrow (2)$$

Scanned by CamScanner

Now, $\frac{dE}{dB} = 0 = -2 \sum \left(y[k] - A - Bx[k]\right) \cdot x[k]$

$\Rightarrow \sum y[k]x[k] - \sum A x[k] - B \sum x^2[k] = 0$

Using A from ①,

$\Rightarrow \sum y[k]x[k] - \sum \left(\frac{\sum y[k]}{N} - \frac{B \sum x[k]}{N} \right) \sum x[k]$

$- B \sum x^2[k] = 0$

$\Rightarrow \sum x[k]y[k] - \frac{\sum y[k]}{N} \sum x[k] - B \sum x^2[k]$

$+ B \frac{\left(\sum x[k]\right)^2}{N} = 0$

$\Rightarrow \boxed{B = \frac{\sum x[k]y[k] - \frac{\sum x[k] \sum y[k]}{N}}{\sum x^2[k] - \frac{\left(\sum x[k]\right)^2}{N}}}$

- So we can see that B can be calculated easily if a new value is added, so running values of A & B can be calculated.

———x———

Note: The line we get from linear regression always pass through the mean x, mean y.

———x———

Scanned by CamScanner

*Sinusoids & Complex Exponentials:

Note: $\cos(\omega n)$ is periodic if we can find an integer $N = 2\pi k/\omega$, where k is an integer. $\cos(\omega n) = \cos(\omega(n+N))$.

*Vector Spaces: (V)

1) • A set is a vector space if $\forall u, v \in V$, $u+v \in V$
2) $\forall u \in V$ & $a \in R$, $av \in V$

Spanning Set: (S)

• A set $S = \{v_1, v_2, \ldots, v_n\}$ is said to span a vector space V if for all $v \in V$,

$$V = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n$$

$\alpha_1 \in R, \alpha_2 \in R, \ldots, \alpha_n \in R$.

Linear Independance:

• A set $S = \{v_1, v_2, \ldots, v_n\}$ is said to be linearly independent if

$$\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = 0 \rightarrow (1)$$

then $\alpha_1 = 0$ and $\alpha_2 = 0$ ..... and $\alpha_n = 0$.

• If equation (1) holds for any one or more non-zero α, then its linearly dependent

Scanned by CamScanner

ex: Is $S = \left\{\begin{bmatrix}1\\-1\\2\end{bmatrix}, \begin{bmatrix}2\\1\\3\end{bmatrix}, \begin{bmatrix}-1\\0\\2\end{bmatrix}\right\}$ a spanning set?

$$\Rightarrow \alpha_1 + 2\alpha_2 - \alpha_3 = a$$
$$-\alpha_1 + \alpha_2 = b$$
$$2\alpha_1 + 3\alpha_2 + 2\alpha_3 = c$$

On solving,

$$\Rightarrow \alpha_2 = \frac{2a+4b+c}{11}$$
$$\alpha_3 = 3\alpha_2 - a - b$$
$$\alpha_1 = \alpha_2 - b$$

} So they are all only dependent on a, b, c

∴ They form a spanning set.

—— x ——

* Inner Product:

- Generalization of dot product.
- $$\langle u, v \rangle = \sum_{k=0}^{N-1} u[k] \cdot \overline{v[k]}$$

Complex conjugate.

—— x ——

* Orthogonality:

- If $\langle u, v \rangle = 0$, they are orthogonal vectors. (u, v)

—— x ——

* Norm:

- $\|u\| = \text{sqrt}\left(\sum_{k=0}^{N-1} u[k]^2\right)$, this is the norm of a vector.
- $\|u\|^2 = \langle u, u \rangle$

—— x ——

Scanned by CamScanner

*Note: If Vectors u, & v, if

$\langle u, v \rangle = 0$ & $\|u\| = 1$ & $\|v\| = 1$

then u & v are orthonormal.

—— x ——

*

*Theorem: If a non-zero set $S = \{v_1, v_2, \ldots, v_n\}$ is orthogonal, it is linearly independent.

$\Rightarrow \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = 0$, For some values of $\alpha_1, \alpha_2, \ldots$

$\Rightarrow$ Take inner product with v_m,

$\langle v_m, \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n \rangle = \langle v_m, 0 \rangle$

$= 0$

$\Rightarrow$ We know $\langle v_i, v_j \rangle = 0$ if $i \neq j$ } orthogonal

$\Rightarrow$ So if LHS = RHS, then

$\alpha_m \langle v_m, v_m \rangle = 0$, then $\alpha_m = 0$

Considering all m from 1 to n,

$\therefore \forall m \in 1, n, \alpha_m = 0$

$\therefore$ If a set is orthogonal, then for $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_n v_n = 0$, it must be linearly independent.

—— x ——

*Basis: (Minimal span)

- A linearly independent set $S = \{v_1, \ldots, v_n\}$ which spans the vector space V is called the basis of V.

Scanned by CamScanner

* Theorem of orthogonal decomposition:

* $V = \alpha_0 v_0 + \alpha_1 v_1 + \dots + \alpha_{n-1} v_{n-1}$, for any $v \in V$ (a vector space)

then, $\alpha_k = \frac{\langle v, v_k \rangle}{\langle v_k, v_k \rangle}$ if $\{v_0, v_1, \dots v_{n-1}\}$ are an orthogonal set

Proof:

$$\langle v, v_k \rangle = \langle (\alpha_0 v_0 + \alpha_1 v_1 + \dots + \alpha_{n-1} v_{n-1}), v_k \rangle$$

$$\langle v, v_k \rangle = \alpha_k \langle v_k, v_k \rangle ,$$

Since all others are 0, from orthogonality.

$$\Rightarrow \alpha_k = \frac{\langle v, v_k \rangle}{\langle v_k, v_k \rangle}$$

---

ex: $v_1 = \begin{bmatrix} \frac{1}{3\sqrt{2}} \\ \frac{1}{3\sqrt{2}} \\ \frac{-4}{3\sqrt{2}} \end{bmatrix}$, $v_2 = \begin{bmatrix} 2/3 \\ 2/3 \\ 1/3 \end{bmatrix}$, $v_3 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \\ 0 \end{bmatrix}$

- This is an orthogonal set of vectors
- $\langle v_1, v_2 \rangle = \langle v_2, v_3 \rangle = \langle v_1, v_3 \rangle = 0$

$\Rightarrow V = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, Let us consider this,

$$\alpha_1 = \frac{-2/3\sqrt{2}}{(1/18 + 1/18 + 16/18)} = -2/3\sqrt{2} = \frac{-\sqrt{2}}{3}$$

$$\alpha_2 = \frac{5/3}{(4/9 + 4/9 + 1/9)} , \quad \alpha_3 = 0$$

---

* Note: Orthonormal set of n vectors always form a basis for an n vector space. But not all basis are orthonormal

** Discrete Fourier Transform:

* Fourier Basis

$$V_k[n] = e^{j \times \frac{2\pi k n}{N}}$$

$n \in [0 \ldots N-1]$

$k \in [0 \ldots N-1]$

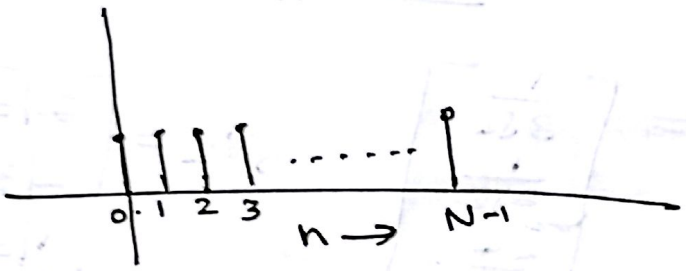
$\Rightarrow$ We have a 2D matrix.

$$V_0 = e^{j \times \frac{2\pi \times 0 \times n}{N}} = [1 \ 1 \ 1 \ 1 \ \ldots\ldots]_N$$

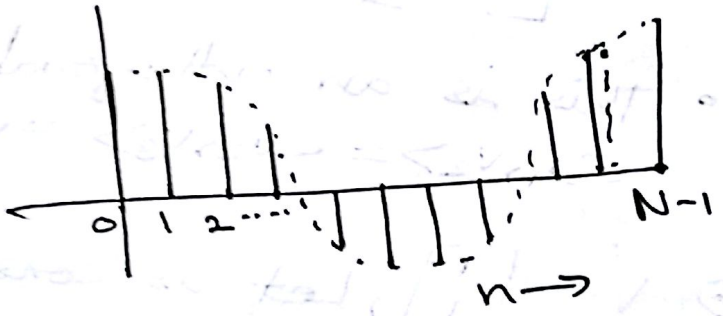
$$V_1 = e^{j \times \frac{2\pi \times 1 \times n}{N}} = \left[e^{j \times \frac{2\pi \times 0}{N}} \quad e^{j\frac{2\pi}{N}} \quad e^{j\frac{4\pi}{N}} \ \ldots\ldots \right]_N$$

* Plotting the real parts,

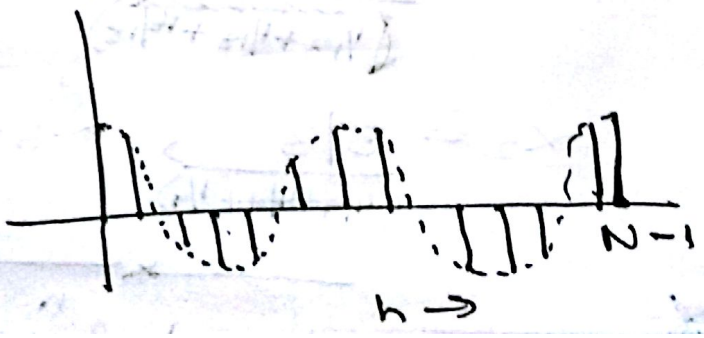
$V_0 \rightarrow$



$V_1 \rightarrow$



$V_2 \rightarrow$ Similar to V_1, but we will form twice as many oscillations as before.



Scanned by CamScanner

Now consider, any V_l, V_k.

$\Rightarrow$ We show $\langle V_k, V_l \rangle = 0$ or $\langle V_k, V_l \rangle = N$

- $\langle V_k, V_l \rangle = \sum_{n=0}^{N-1} V_k[n] \cdot \overline{V_l[n]}$

$$= \sum_{n=0}^{N-1} e^{j \times \frac{2\pi k n}{N}} \cdot e^{-j \times \frac{2\pi k l n}{N}}$$

$$= \sum_{n=0}^{N-1} e^{j \times \frac{2\pi (k-l) n}{N}}$$

Case (1),

$k = l$, $\langle V_k, V_l \rangle = N$

$*$

Case (2),

$k \neq l$

$1 + a + a^2 + \cdots + a^{N-1}$

where $a = e^{j \times \frac{2\pi \times (k-l)}{N}}$

$\therefore$ G.P sum $\Rightarrow \frac{1-a^N}{1-a} = \frac{1 - e^{j \times 2\pi (k-l)}}{1 - e^{j \times \frac{2\pi (k-l)}{N}}}$

$= 0$ //

$\because k-l$ is an integer.

$\therefore$ We have found a basis of N vectors for the vector space of N-dimensional vectors. These vectors may also contain complex values.

$*$ $\Rightarrow$ So any signal (vector of N dimensions) can be expressed as a linear combination of this basis.

Scanned by CamScanner

From the theorem of orthogonal decomposition,

$$x = \sum_{k=0}^{N-1} X_k \cdot V_k \quad , \quad X_k = \frac{\langle x, V_k \rangle}{\langle V_k, V_k \rangle}$$

$$x[n] = \sum_{k=0}^{N-1} X_k \cdot e^{j \times \frac{2\pi \times k \times n}{N}}$$

→ Inverse discrete fourier transform

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \cdot e^{-j \frac{\times 2\pi \times k n}{N}}$$

→ Discrete fourier transform

In some books this normalizations may change.

- Using DFT, we can get the weightages (X_k) of the basis signals / frequencies.

---

* DFT as Matrix Product:

$$\begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 3^0 & 3^0 & 3^0 & \cdots & 3^0 \\ 3^0 & 3^1 & 3^2 & \cdots & 3^{N-1} \\ 3^0 & 3^2 & 3^4 & \cdots & 3^{2(N-1)} \\ \vdots & & & & \\ 3^0 & 3^{(N-1)} & 3^{2(N-1)} & \cdots & 3^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$$

↓ Fourier transform of signal.

$3 \rightarrow W_N$

$$e^{-j \frac{\times 2\pi \times k n}{N}} = (3)^{kn}$$

$$\boxed{3 = e^{-j \frac{\times 2\pi}{N}}}$$

Scanned by CamScanner

Note: Fourier transform: $X = W \cdot x$

—x—

* ex: $x(t) = 5 + 2\cos(2\pi t + \frac{\pi}{2}) + 3\cos 4\pi t$

→ frequency of terms increases as we go to the right

(Now $\omega t = 2\pi t$, $\omega t = 4\pi t$)

$\Rightarrow \omega = 2\pi$, $\omega = 4\pi$

$f = 1$, $f = 2$

(Max frequency ×2)

$\Rightarrow$ So we choose 4 samples in a second

$t = 0, t = 1/4, t = 1/2, t = 3/4$

At $t=0$: $x[0] = 8$

At $t=1/4$: $x[1] = 5+2-3 = 4$

$x[2] = 5+2\times 0+3 = 8$

$x[3] = 0$

} The first 4 entries in our sample.

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}\begin{bmatrix} 8 \\ 4 \\ 8 \\ 0 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 20 \\ -4j \\ 12 \\ 4j \end{bmatrix} = \begin{bmatrix} 5 \\ -j \\ 3 \\ j \end{bmatrix}$$

* $\Rightarrow X_K = X^*_{N-K}$ } For $K = 1, 2, \ldots, N-1$ (∵ If $K=0$, X_N does not exist as of now)

↓ Proof not needed

—x—

* Note: Each of the terms in our given signal (order according to frequency) corresponds to X_K & X_{N-K}.

- So in the above example, 5 is given by $|X_0|$, 2 is given by $|X_1|$ & $|X_3|$, 3 is given by $|X_2|$. (These are the amplitudes)
- Now the phase for each part is given by the first occurrence phase i.e. X_K's phase.

—x—

Scanned by CamScanner

* Fourier Transform:

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-jx2\pi ft} \cdot dt$$

$$X(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} \cdot dt$$

---

* Discrete Time Fourier Transform: (DTFT)

- $X[j\omega] = \sum_{-\infty}^{\infty} x[n] \cdot e^{-j\omega n}$

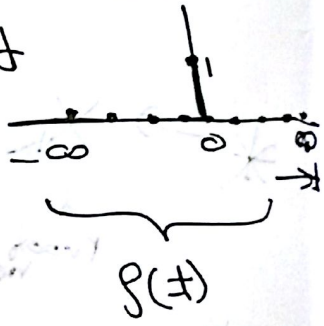
---

* Functions:

1) Delta: ($\delta(t)$)

$$X(f) = \int_{-\infty}^{\infty} \delta(t) \cdot e^{-jx2\pi ft} \cdot dt$$

$$X(f) = 1$$

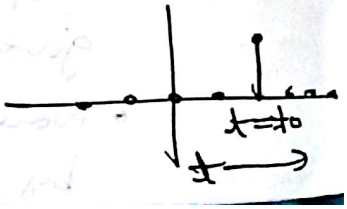


---

2) Delta Shift ($\delta(t - t_0)$)

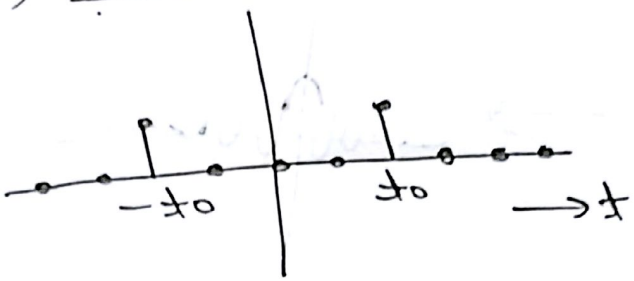
$$X(f) = \int_{-\infty}^{\infty} \delta(t - t_0) \cdot e^{-jx2\pi ft} \cdot dt$$

$$X(f) = e^{-jx2\pi f t_0}$$



Scanned by CamScanner

3) Function with 2 impulses:

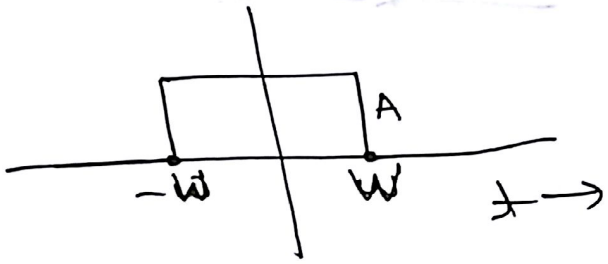


$$X(f) = e^{-j\omega t_0} + e^{j\omega t_0} = 2\cos(\omega t_0)$$

$\omega \rightarrow 2\pi f$

(We can generate a cos function by using fourier transforms on symmetric impulses)

**4)



$$X(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-jx\omega t} \cdot dt$$

$$= \int_{-W}^{W} A e^{-jx\omega t} \cdot dt$$

$$= A \int_{-W}^{W} e^{-j\omega t} \cdot dt$$

$(\omega = 2\pi f)$

$$= -\frac{A}{j\omega}\left[e^{-j\omega W} - e^{+j\omega W}\right]$$

$$= \frac{A}{j\omega}\left[2j\sin(W\times\omega)\right]$$

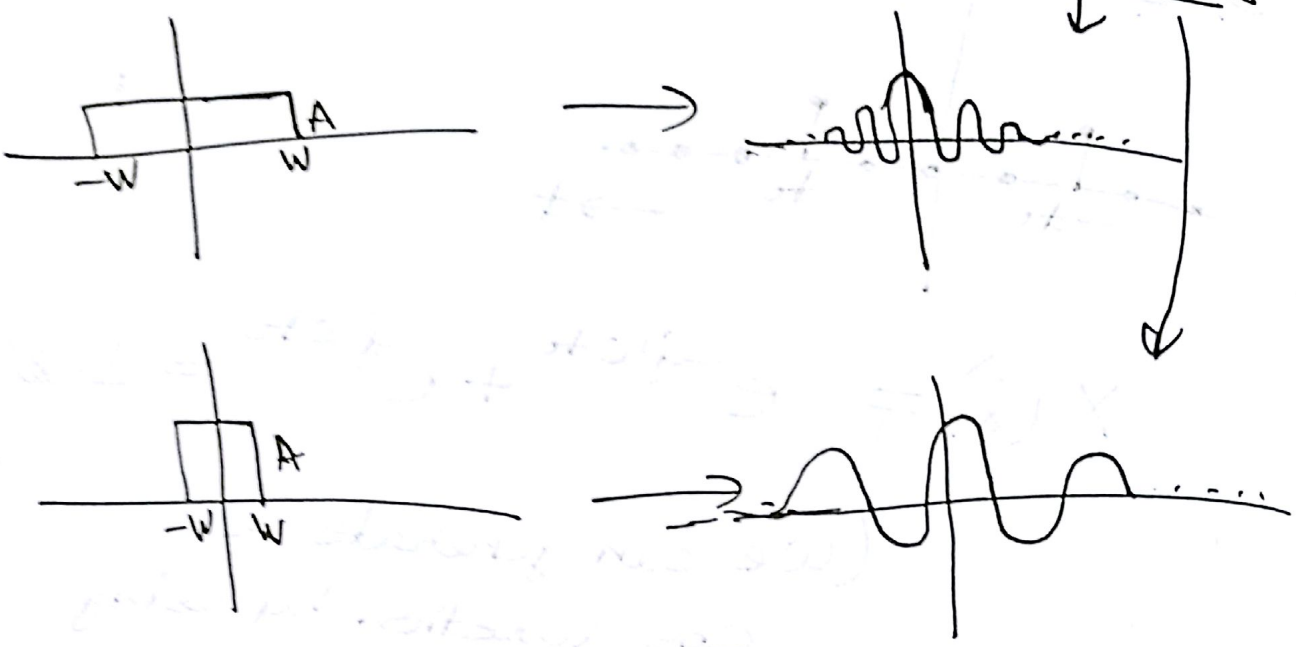
$$\therefore X(f) = \frac{A \times 2j}{2\pi f j}(\sin(W \times 2\pi f)) = \frac{A\sin(W\times 2\pi f)}{\pi f}$$

Scanned by CamScanner

So we can see,

(Amplitude change)

Sinc Functions



(Depending on A & W, we can see how our fourier transform varies)

Note: Signal + Noise $\xrightarrow{FFT}$ Select prominent peaks which could correspond to actual signal (X_k's)

Get a very close approx to signal $\xleftarrow{}$ Inverse FFT (IFFT)

Scanned by CamScanner

Fast Fourier Transform: (FFT)

- DFT takes N^2 operations to calculate values.
- FFT will only take $N \log N$ steps.

$$X_k = \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-2\pi j \times k \times n}{N}},$$

i) Put $k = \frac{N}{2} k_1 + k_0$

$$0 \le k_0 \le \frac{N}{2} - 1,$$

$$k_1 = 0 \text{ or } 1$$

$$\Rightarrow X_k = \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-2\pi j \left(\frac{N}{2} k_1 + k_0\right) n}{N}}$$

$$= \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-2\pi j \times \frac{N}{2} k_1 n}{N}} \cdot e^{\frac{-2\pi j k_0 n}{N}}$$

$$= \sum_{n=0}^{N-1} x[n] \cdot e^{-\pi k_1 n j} \cdot e^{\frac{-2\pi k_0 n j}{N}}$$

$$= \sum_{n=0}^{N-1} x[n] \cdot (-1)^{k_1 n} \cdot e^{\frac{-2\pi k_0 n j}{N}}$$

ii) $\Rightarrow$ Now, split n into even & odd indices

$(2n)$, $(2n+1)$

$$\therefore X_k = \sum_{n=0}^{\frac{N}{2}-1} x[2n] (-1)^{k_1 \times 2n} \cdot e^{\frac{-2\pi j k_0 (2n)}{N}} + \sum_{n=0}^{N/2-1} x[2n+1] (-1)^{k_1 (2n+1)} \cdot e^{\frac{-2\pi j k_0 (2n+1)}{N}}$$

Scanned by CamScanner

*Fast Fourier Transform: (FFT)

* DFT takes N^2 operations to calculate values.
* FFT will only take $N \log N$ steps.

$$X_k = \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-2\pi j k \times n}{N}},$$

i) Put $k = \frac{N}{2} k_1 + k_0$

$$0 \le k_0 \le \frac{N}{2} - 1,$$

$$k_1 = 0 \text{ or } 1$$

$$\Rightarrow X_k = \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-2\pi j \left(\frac{N}{2} k_1 + k_0\right) n}{N}}$$

$$= \sum_{n=0}^{N-1} x[n] \cdot e^{\frac{-2\pi j \times \frac{N}{2} k_1 n}{N}} \cdot e^{\frac{-2\pi j k_0 n}{N}}$$

$$= \sum_{n=0}^{N-1} x[n] \cdot e^{-\pi k_1 n j} \cdot e^{\frac{-2\pi k_0 n j}{N}}$$

$$= \sum_{n=0}^{N-1} x[n] \cdot (-1)^{k_1 n} \cdot e^{\frac{-2\pi k_0 n j}{N}}$$

ii) $\Rightarrow$ Now, split n into even & odd indices

$(2n)$, $(2n+1)$

$$\therefore X_k = \sum_{n=0}^{\frac{N}{2}-1} x[2n] (-1)^{k_1 \times 2n} \cdot e^{\frac{-2\pi j k_0 (2n)}{N}} + \sum_{n=0}^{N/2-1} x[2n+1] (-1)^{k_1 (2n+1)} \cdot e^{\frac{-2\pi j k_0 (2n+1)}{N}}$$

Scanned by CamScanner

$$\therefore X_k = \sum_{n=0}^{N/2-1} x[2n] \cdot e^{\frac{-2\pi j k o n}{(N/2)}} + (-1)^{k_1} \sum_{n=0}^{N/2-1} x[2n+1] \cdot e^{\frac{-2\pi j k o n}{N/2}} \cdot e^{\frac{-2\pi j k o}{N}}$$

$$X_k = \sum_{n=0}^{N/2-1} x[2n] \cdot e^{\frac{-2\pi j k o n}{N/2}} + (-1)^{k_1} \cdot e^{\frac{-2\pi j k o}{N}} \sum_{n=0}^{N/2-1} x[2n+1] \cdot e^{\frac{-2\pi j k o n}{N/2}}$$

~~$X_{[0,N/2]} = F_0 + QF_1$~~

Now, $\therefore X_{(k_0)} = F_0 + QF_1$ (FFT of $N/2$ elements; FFT of $N/2$ elements)

$$** \quad X_{(N/2+k_0)} = F_0 - QF_1$$

where $Q = e^{\frac{-2\pi j k o}{N}}$

(F_0 = DFT of even terms, a total of $N/2$ terms)

(F_1 = DFT of odd terms, a total of $N/2$ terms)

$\therefore$ To calculate DFT of N, we can recursively break it down into parts of size $N/2$ each time.

$$\therefore T_N = 2T_{N/2} + 2N$$

($N/2$ multiplications, $N/2$ additions for $k_1 = 0$ or $k_1 = 1$)

$$\therefore T_N = N \log N //$$

Scanned by CamScanner

2D DFT:

$$\hat{A} = [W_M]_{M\times M} [A]_{M\times N} [W_N]_{N\times N}$$

$$\hat{A}_{k,l} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A[m,n] e^{-j\times 2\pi \left(\frac{kn}{N} + \frac{lm}{M}\right)}$$

Convolution is multiplication of Fourier Transforms:

Let $y[n] = x[n] \star h[n]$, lets prove,

$Y[w] = X[w] \cdot H[w]$ → DTFT's (For ∞ length signals)

Proof:

$$X_w = \sum_{n=-\infty}^{\infty} x[n] \cdot e^{-jwn}$$

$$Y_w = \sum_{n=-\infty}^{\infty} (x[n] \star h[n]) \cdot e^{-jwn}$$

but $x[n] \star h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$

$$\Rightarrow Y_w = \sum_{n=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k] \right) \cdot e^{-jwn}$$

Let $m = n-k$, $n = m+k$

$$\Rightarrow Y_w = \sum_{m=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} x[k] \cdot h[m] \right) \cdot e^{-jw(m+k)}$$

$$= \sum_{m=-\infty}^{\infty} h[m] \cdot e^{-jwm} \cdot \sum_{k=-\infty}^{\infty} x[k] \cdot e^{-jwk}$$

Scanned by CamScanner

$\therefore Y(\omega) = H(\omega) \cdot X(\omega)$

---

* Circular Convolution: (⊛)

- $X[n]$ circular conv $H[n]$

⇒ we don't need to flip $H[n]$ unlike before, we start as it is, then we circularly rotate $H[n]$ then to get $y[n]$

ex: $x_1[n] = [2 \underset{\uparrow}{1} 2 1]$,

$h_1[n] = [\underset{\uparrow}{1} 2 3 4] \Rightarrow h_1'[n] = [4 1 2 3]$

$h_1''[n] = [3 4 1 2]$

$\Downarrow$

....

$y[n] = [14 \ 16 \ 14 \ 16]$

$y[n] = x_1[n] \circledast h[n]$

---

* $\Rightarrow y[n] = x_1[n] \bigstar x_2[n]$ ← From prev proof

$= IDTFT\left(DTFT(x_1[n]) \cdot DTFT(x_2[n])\right)$

↓ Inner product (without sum)

* $\Rightarrow x_1[n] \circledast x_2[n] = IDFT(DFT(x_1[n]) \cdot DFT(x_2[n]))$

(Proof not needed) ↓ Inner product (without sum)

Note: This is because DFT assumes our signal repeats over & over infinitely (from the formulas), so it pretty much acts as a circular shift.

Scanned by CamScanner

Note: If we pad the signals with 0's, then for a certain number of 0's, Circular convolution = normal convolution.

—— x ——

*Non Locality of DFT:

- We won't have information on where the peaks of DFT were present in the original signal.

⇒ We use windowing to solve this.

- Each window has a size M and we move it by n in each iteration. N = signal size.

So, k we iterate from $k = 0, 1, \ldots, \frac{[N-M]}{n}$

⇒ $[x_{kn}, \ldots, x_{kn+M-1}]$ → Data for each window.

—— x ——

Spectrogram: (Audio uses this (Audio matching etc))

- Time vs Frequency. Greyscale is used to show major frequency content. Each block would represent a window.

—— x ——

Scanned by CamScanner

*Filters:

—x—

- Note: When we have DFT of our noised signal, on multiplying it with a rectangular filter & taking IDFT is similar to convolution of signal with a 'sinc' type of similar function, so we will get ripples in our output.

* ⇒ So for a good filter, we use a gaussian filter instead of a rectangular filter. → IDFT using this is like convolution of signal with a pretty much horizontal line function, so no ripples.

—x—

Note: Generally low frequency & high amplitude signals give rise to the overall wave (the actual one without noise). Don't forget the conjugate frequencies which also contribute to the overall wave.

X_k & X_{n-k} ⇒ Contribute to actual wave without noise.

—x—

Note: Salt Noise → Min filter (To ignore white spots)
Pepper Noise → Max filter (To ignore dark spots)

—x—

Note: Low pass filter → Approx Shape of signal captured
High pass filter → Details of signal captured

—x—

Scanned by CamScanner

*Leaky Integrator

* Note: Convolution of a signal with a causal filter, → causal signal (output)

⇒ An averaging filter,

$$y_M[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k] \longrightarrow (1)$$

- $$y_M[n-1] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-1-k]$$

$$= \frac{1}{M}\sum_{k=1}^{M} x[n-k]$$

Similarly,

- $$y_{M-1}[n-1] = \frac{1}{M-1}\sum_{k=1}^{M-1} x[n-k] \longrightarrow (2)$$

$$\Rightarrow \sum_{k=0}^{M-1} x[n-k] = x[n] + \sum_{k=1}^{M-1} x[n-k]$$

⇒ From (1), (2)

$$M \cdot y_M[n] = x[n] + (M-1)\cdot y_{M-1}[n-1]$$

$$\Rightarrow y_M[n] = \frac{x[n]}{M} + \frac{(M-1)}{M} y_{M-1}[n-1]$$

$$= (1-\lambda)x[n] + \lambda y_{M-1}[n-1]$$

Consider $\lambda = \frac{M-1}{M}$

* $\therefore y_M[n] = (1-\lambda)x[n] + \lambda y_{M-1}[n-1]$

Scanned by CamScanner

$\therefore$ $\boxed{y[n] = (1-\lambda)x[n] + \lambda y[n-1]}$, $\lambda = \frac{M-1}{M}$

- This is a leaky integrator, we approximate that m is large, so $y_m[n] \approx y_{m-1}[n]$
  $\therefore$ we get the leaky integrator above.

(In MATLAB this is done by the 'Filter' command)

---

* **Note:** A normal integrator $\rightarrow$ $y[n] = \sum_{k=-\infty}^{n} x[k]$

$\Rightarrow y[n] = x[n] + y[n-1]$

- Here we use $\lambda y[n-1]$, $\lambda < 1$ so we are leaking out some part, so its a leaky integrator.

---

* **Impulse Response:** (Leaky Integrator)

- $h[n] = (1-\lambda)\delta[n] + \lambda h[n-1]$

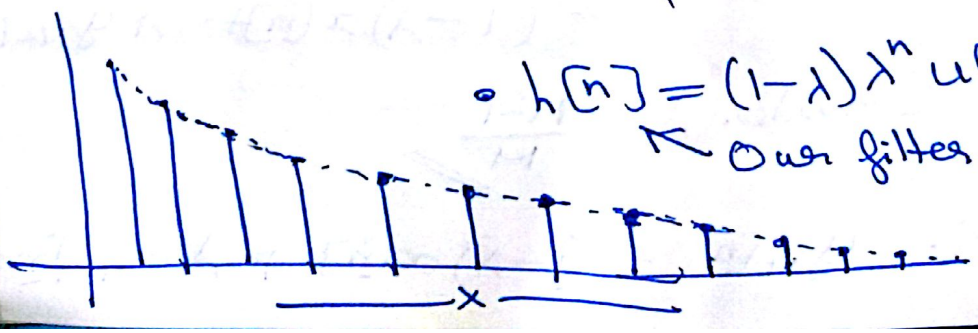
$\Rightarrow$ For $n < 0$, $h[n] = 0$ $\rightarrow$ System is off

$n = 0$, $h[n] = 1-\lambda$

$n = 1$, $h[n] = \lambda(1-\lambda)$

$n = 2$, $h[n] = \lambda(\lambda(1-\lambda)) = \lambda^2(1-\lambda)$

- $h[n] = (1-\lambda)\lambda^n u[n]$ $\leftarrow$ our filter



Scanned by CamScanner

*FIR vs IIR Filters:

—— x ——

*Note: $\sum_{k=0}^{N-1} a_k y[n-k] = \sum_{n=0}^{M-1} b_n x[n-n]$ } (CDE) Coefficient Difference equation

—— x ——

- FIR → Finite Impulse Response filters
  ↓
  Like our moving average filter
- IIR → Infinite Impulse Response filters
  ↓
  Like the leaky integrator

—— x ——

*Compression:

- Lossy & Lossless compression.

*Run Length Encoding: (RLE) (Lossless compression)

*

⇒ 0001100000111101

1) ↳ (0,3) (1,2) (0,4) (1,5) (0,1) (1,1) → 3245111 (If we say it starts with 0)

- So the representation is a lot simpler.

—— x ——

- If lots of single values occur in our data,

2) (00,3) (11,2) (00,4) (11,5) (0) (1)

We save on space here.

—— x ——

3) Suppose we are encoding,

WWWWBBBWWBBWBB

[W, B, W, B, W, B]

We can sometimes limit this → [4, 3, 2, 2, 1, 2] } Now we can encode these strings again as well

—— x ——

Scanned by CamScanner

*Huffman Coding: (Variable length encoding)

- Suppose a string has

A → 8 times

! → 3 times

B → 2 times

R → 2

— → 2

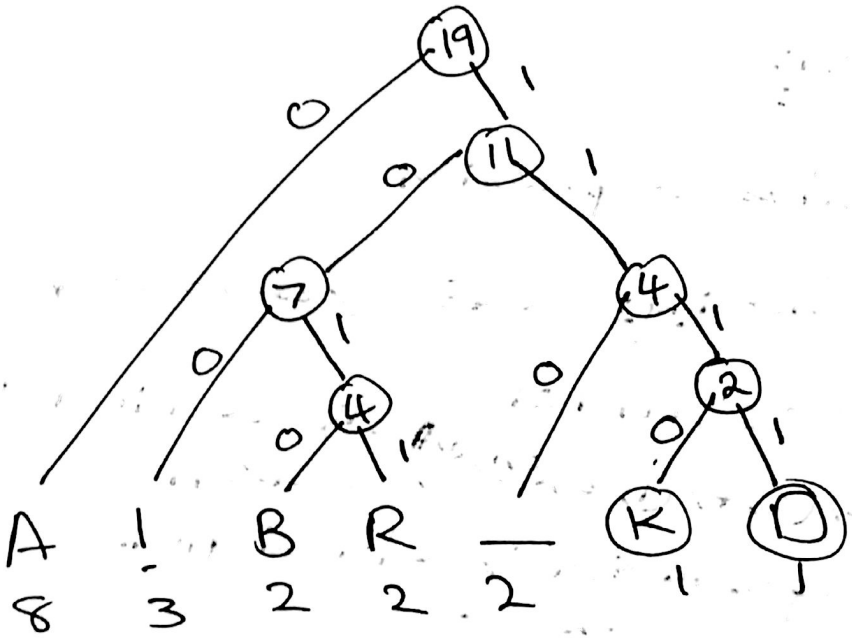
K → 1

D → 1

*• Prefix free
* encoding for each character is generated.

• For any encoding
*we can define compression ratio as (Size actual / compressed size)

• Huffman tree with variable length encoding is formed iteratively by collecting the 2 smallest numbers & merging them at each step.



→ A = 0 — = 110

! = 100 K = 1110

B = 1010 D = 1111

R = 1011

} Prefix Code

• Now we can use this to encode.

Scanned by CamScanner

Self Energy: (I)

- $I = \log_2\left(\frac{1}{p}\right) = -\log_2 p$

  (p = Probability of occurrence)

  - As $p \uparrow$, $I \downarrow$

———x———

Entropy: (E)

- $E = -\sum_{i=0}^{L-1} p_i \log_2(p_i)$

———x———

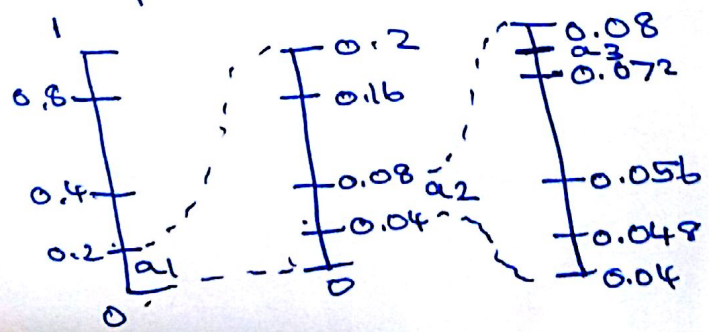
** ⇒ Shannon said that the maximum amount of compression we can achieve = E. (Entropy). → $\frac{\text{No. of Bits}}{\text{No. of chars}}$.

(This can be broken by some carefully crafted strings) ———x———

* Arithmatic Coding:

* • Given a string, we encode it just using a number between 0 & 1.

ex: $S = \{a_1, a_2, a_3, a_4\}$
$P = \{0.2, 0.2, 0.4, 0.2\}$ ∴ String to form is $a_1 a_2 a_3$



⇒ Now just store the midpoint,

⇒ $\frac{0.072 + 0.08}{2}$

$= 0.076$//

Scanned by CamScanner

* ⇒ Now to decode it, we go through the same process, and at each step check which interval our number lies in.

⇒ We get $a_1 a_2 a_3$.

* • To mark the end of the string, we can either add an EOM char & stop when we encounter it or just send the length of the string along as well, and use it to stop.

———x———

* LZW Coding: (Example in slides)

(No need to store any dictionary)

* • We augment another bit, and then instead of just ASCII encoding we use this new encoding where we use the extra 256 possibilities to make our own dictionary. (256 → 511)

• We then send this dictionary so the decoding can be done. But this is expensive, so we follow a similar procedure while decoding where we go ahead building a similar sort of dictionary.

* **Note:** While decoding, if we get current & next as xy where $y > 255$, so that would mean y consists of 2 chars, so we will only consider the first char of our dict[y].

Scanned by CamScanner

∴ We add to our new dictionary,
dict[x] to dict[y][0].

⇒ However if $x > 255$, then we consider the entire x, unlike above.

—— x ——

Note: ~~At times~~, generally 11 to 13 bits are used where the 5 bits are used for creation of our dictionary.

—— x ——
—— x ——

* MP3 Compression:

- Without compression, 44100 Hz → 44,100 samples in a second.

∴ 44100 × 16 × 60 = 42.3 mb } In a minute

16 → 2^{16} datapoints to show value of (frequency) each reading's DFT. quatised

a) ⇒ Since we humans are more sensitive to a certain range of frequencies.

1) Take DFT of signal

2) The sensitive part (2k–5kHz)(band) → use a high level of quantigation (many bits)

* 3) In other parts (bands) → use less bits to quantize.

b) Temporal Masking:

- If we get a small sound after a large one, we can use less bits to quantize the small one since we can't hear it well anyways.

Scanned by CamScanner

* c) Frequency Masking: (It's the loudest in dB as of f_x)

- When a certain frequency (f_x) is playing, then we become less sensitive to nearby frequencies, so we can use fewer bits to quantize those nearby ones.

———— x ————

* Note: All these frequency based compressions are generally done using windows, and DFT's on windows.

————x————

Scanned by CamScanner

* c) Frequency Masking: (It's the loudest in dB as of now)

- When a certain frequency (x) is playing, then we become less sensitive to nearby frequencies, so we can use fewer bits to quantize those nearby ones.

——— x ———

*Note: All these frequency based compressions are generally done using windows and DFT's on windows.

═══ x ═══

* JPEG Compression: (Lossy Compression)

1) Type of coding
2) Spacial redundancy
   (Temporal redundancy → Videos)
3) Perceptual redundancy
   - (We remove colors etc that are harder to perceive)
   - ex: RGB → Y, Cb, Cr
     (Y is like greyscale) (Luminance)
     (Cb, Cr → Color channels)

——— x ———

*I) Y, Cb, Cr → Perceptual redundancy
(Luminance)

- Changes in Y → We can easily find out.

⇒ Changes in Cb, Cr are not that noticable. So we can compress these two channels.

Scanned by CamScanner

- We can convert an RGB image to a YCbCr image.

———x———

*Quality Measurement:

- SNR =
- PSNR =

———x———

Note: A great way to compress is by breaking the image into smaller parts since the redundancy of colours in these small parts is a lot more. So we can quantize better. (Generally each of these parts lie around one or two values of RGB & near them). (shades)

⇒ Parallelization can also be done.

———x———

*JPEG—Method:

- Break image into 8x8 chunks, on each chunk subtract value by 127 (so we have a -ve to +ve range) → This is used by the DCT (Discrete cosine transform) (Implementation detail)
- We then use a DCT & then quantize.

Scanned by CamScanner

Why not DFT? → Experiments.

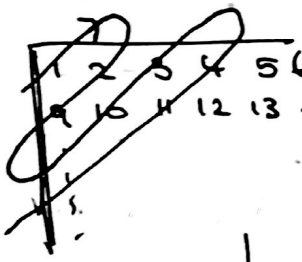
Note: DCT, 8×8 all these things were decided based on experimental results. These led to the best results.
(16×16 actually was best but 8×8 was easier to implement).

2D matrix after DCT → Input

* Quantization → There is a standard 2D matrix of 8×8 cells which we use to quantize.
  - For each chunk
  - → Each of our 8×8 chunks, we divide the values in the chunk with corresponding values in the standard 2D matrix.

* (This was setup by experiments asking users to rate images at different)

* Encoding → Once we have quantized the image chunk, we use a zig-zag reading of the values in our matrix
  - → This is done so that we get lots of repeating adjacent values (Good run length encoding can be done)



→ Once this is done, we will be left with an array. (single)

→ Generally a large chunk of cells are all 0's near the end, so we just represent it with EOB. (approx 75%)

Scanned by CamScanner

* RLE or Huffman coding
  - Send message after coding it
  - ------ Encoding to Decoding

* Decoding
  - Get back vector and reconstruct the matrix.

* De quantize: (We get the data loss here)
  - Multiply each cell with corresponding value in the standard 2D matrix
  - Once this is done we get back a similar matrix to the one we got after DCT. (But we have lost some values) by now.

Inverse DCT:

- Do an inverse DCT & then add back 127.
- Then reassemble the sub parts (8x8 chunks) to get back the original image.

———— x ————

Scanned by CamScanner

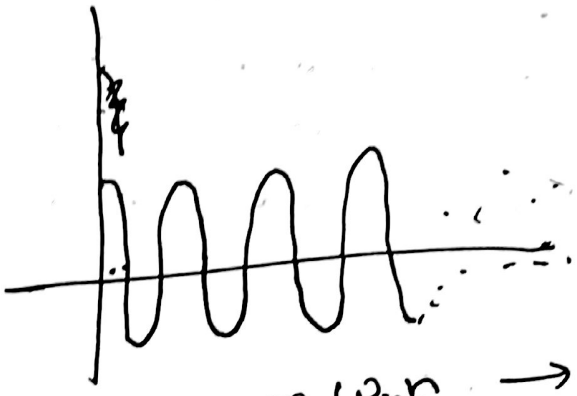
Modulation

(To send signals to large distances) → Radio wave carriers.

*I) Amplitude Modulation:

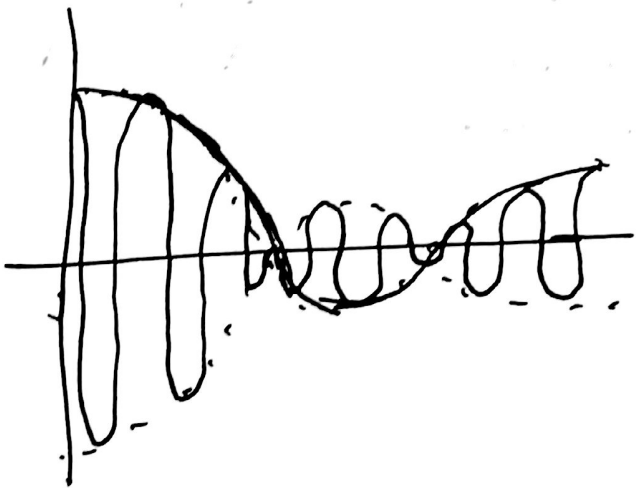
- We use a carrier wave & multiply it with the signal to send it along.

exc:



$\cos \omega_c n$ → Carries

$$S[n] = x[n] \cdot \cos \omega_c n$$



⇒ How do we get back the original signal.

Scanned by CamScanner

*1) At receiver we have,

$$y[n] = x[n]\cos(\omega_c n)$$

* We multiply it with $\cos(\omega_c n)$

$$\rightarrow x[n].\cos^2(\omega_c n)$$

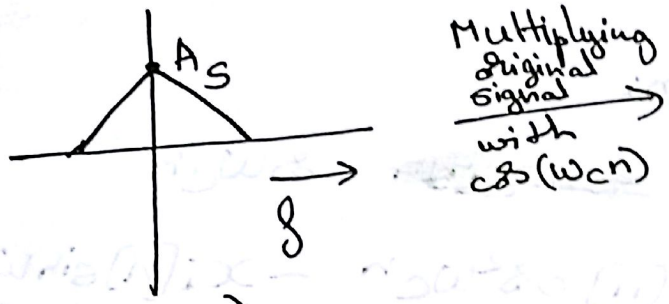
$$= x[n]\left[\frac{1+\cos(2\omega_c n)}{2}\right]$$

$$= \frac{x[n]}{2} + \frac{x[n]}{2}\cos(2\omega_c n)$$

* Then apply a low pass filter to neglect the 2nd term. Then multiply by 2 to get back $x[n]$.

*

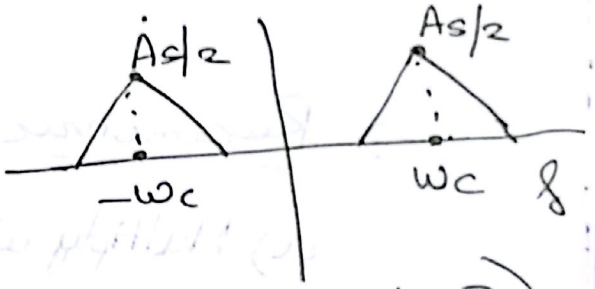
⇒ In frequency domain,



$\text{fft}(x[n])$

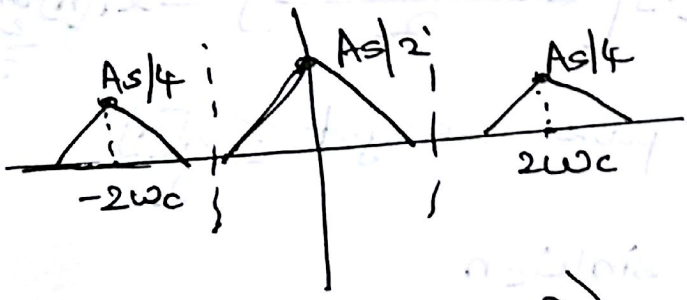
Multiplying original signal with $\cos(\omega_c n)$

(with fft shift)



$\text{fft}(x[n]\cos(\omega_c n))$

On receiving again multiply



$\text{fft}(x[n]\cos^2(\omega_c n))$

⇒ Applying low pass filter & multiplying with 2, we get back the original signals fft.

—— × ——

Scanned by CamScanner

2) If our signal is a complex signal

*

⇒ We send multiplying it with $e^{j\omega_c n}$

⇒ $x[n] = x_r[n] + x_i[n] \cdot j$

∘ We send $Re\{x[n] \cdot e^{j\omega_c n}\}$

⇒ $Re\{(x_r[n] + jx_i[n]) \cdot (\cos\omega_c n + j\sin\omega_c n)\}$

$y[n] = x_r[n]\cos\omega_c n - x_i[n]\sin\omega_c n$

⇒ We send this over

⇒ Reconstruction:-

a) Multiply with $\cos\omega_c n$

⇒ $x_r[n]\cos^2\omega_c n - x_i[n]\sin\omega_c n \cos\omega_c n$

⇒ $x_r[n]\left[\frac{1 + \cos 2\omega_c n}{2}\right] - x_i[n] \cdot \frac{\sin 2\omega_c n}{2}$

→ On low pass we get $\frac{x_r[n]}{2}$

b) Multiply with $\sin\omega_c n$

⇒ $x_r[n]\sin\omega_c n \cos\omega_c n - x_i[n] \cdot \sin^2\omega_c n$

⇒ $x_r[n] \cdot \frac{\sin 2\omega_c n}{2} - x_i[n]\left(\frac{1}{2} - \frac{\cos 2\omega_c n}{2}\right)$

⇒ On low pass we get $\frac{x_i[n]}{2}$

—x—

Scanned by CamScanner

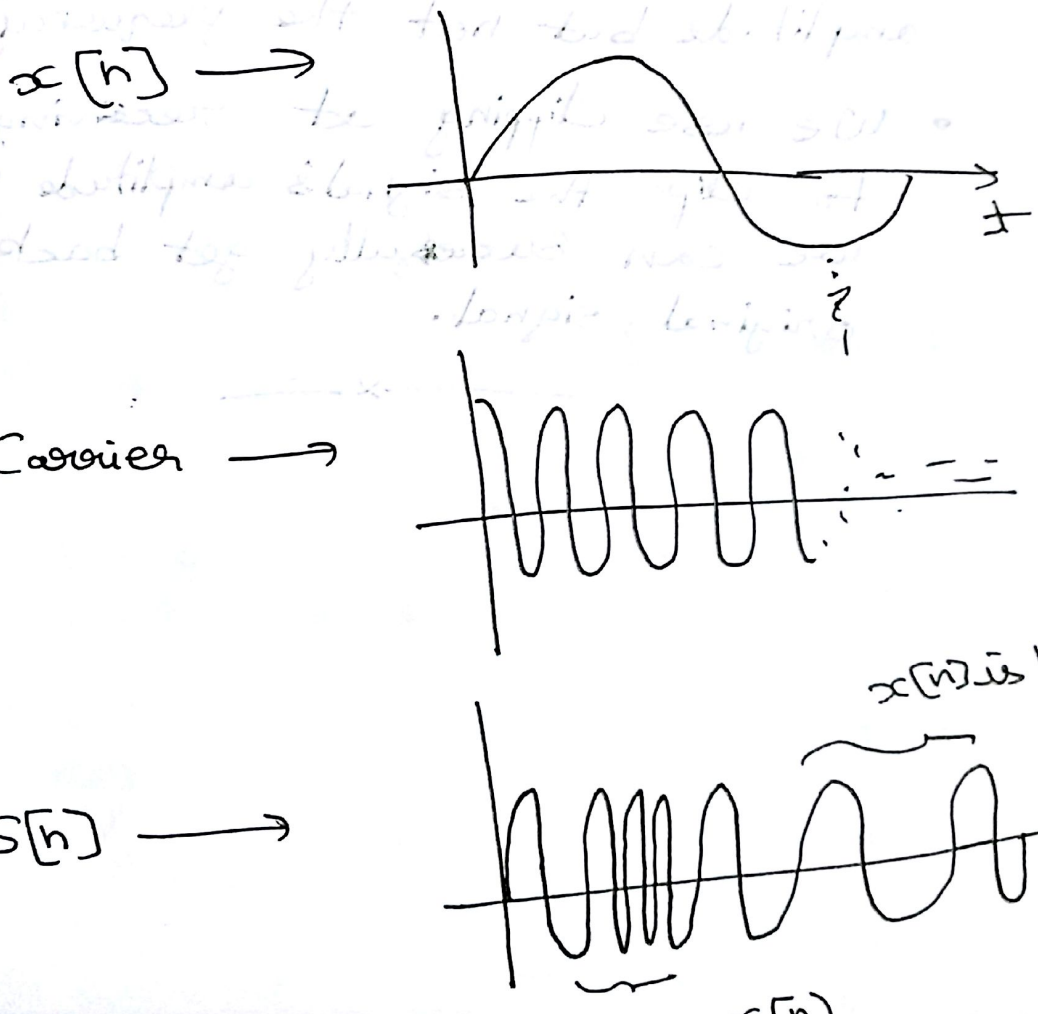
*Note: Different bands (ω_c) are allocated & when we send them, we also do a band pass filtering instead of low pass so that we get our signal at our band (ω_c).

———×———

*Note: In AM signals, noise can modify the signal being sent so we can end up having varied amplitudes. (Noise). So we switch to frequency modulated signals.

———×———

*II) Frequency Modulation:



Scanned by CamScanner

- $x[n] = A_x \sin(\omega_x n)$

$c[n] = A_c \sin(\omega_c n)$

$\Rightarrow s[n] = A_c \sin(2\pi(f_c + A_x \sin \omega_x n)n)$

- We send a signal hidden in the frequency.

—— x ——

* Note: To get back the original signal, there are ways to figure out frequency at a certain given point?

—— x ——

Noise:

- Generally this noise effects only the amplitude but not the frequency mostly.
- We use clipping at receiving end to clip the signal's amplitude & then we can successfully get back our original signal.

—— x ——

Scanned by CamScanner

Dither

- Noise can be helpful at times to help preserve information while quantizing .etc.

⇒ Audio we add noise twice (Like sum of 2 dice, so a lot more cases produce the average →))

(exc: For images instead of colour, we can use intensity to show information (using dithers)

** Stochastic Signal Processing *

- Spectral Power Density ⇒ ? (SPD)

↳ $\sum_{k=1}^{M} \frac{\lim_{N\to\infty} \frac{1}{(2N+1)} \sum_{n=-N}^{N} |x[n]|^2}{M}$ → $x[n]$ →

Practically we consider some large N.

- M experiments, so we average our results.

⇒ Energy doesn't tell us much but power (SPD) does.

* - Filtering in time domain also effects the SPD

Scanned by CamScanner

* Principle Component Axis (PCA)

—— x ——

Note: Variance $= \frac{\Sigma(x - \bar{x})^2}{n-1}$ ($\bar{x}$ → Mean)

—— x ——

* Covariance $(H, M) = \frac{\Sigma (H - \bar{H})(M - \bar{M})}{(N-1)}$

→ Similarity b/w two datasets, if one is increasing & other is decreasing covariance ⇒ -ve.

→ If both are increasing or decreasing, its covariance ⇒ +ve.

• We can use PCA for camera angles etc
* to capture maximum information from a 3D point cloud. → the PCA helps us * find the maximum Variance axis.

* Principle component axis → Points when projected to that axis have maximum variance

—— x ——

* Eigen Vectors / Values : (eqns in slides)

• $\Sigma e = \lambda e \rightarrow (1)$, λ = Eigen values, e = Eigen vector.

$e_M^{-1} = e_M^T$ } The eigen vector matrix) → $e_M = [e_1, e_2]$ (suppose unit)

$\Rightarrow \Sigma(e) - \lambda e = 0$

$\Rightarrow (\Sigma - I\lambda)e = 0$

- This is only possible when $|\Sigma - I\lambda| = 0$
- From this we can get the two eigen values

$\Rightarrow$ On substituting each eigen value in (1),
* we can get the corresponding eigen
* vectors. $\longrightarrow$ Eigen vectors should be unit vectors

———— x ————

* Now if we use Σ = Covariance matrix
*

Cov matrix = $\begin{bmatrix} cov(1,1) & cov(1,2) & \cdots & cov(1,N) \\ cov(2,1) & cov(2,2) & \cdots & \\ \vdots & \vdots & \ddots & \\ & & & cov(N,N) \end{bmatrix}$

If we consider N datasets
↳ Like coordinate axes. x, y, z

- Now if we calculate the eigen vectors, we get the maximum variance axes.

* ~~§~~ ⟹ Use the top ~~k~~ eigen vector corresponding to the max eigen value ~~eigen vectors~~ and ~~approximate~~.
↳ This is the principle component axis.

Other eigen vectors are orthogonal

———— x ————

*Note: Multiplying points with an eigen vector, we get the points ~~on~~ ~~the~~ wrt the new axis. (Assuming the eigen vector to be the new axis)

———— x ————

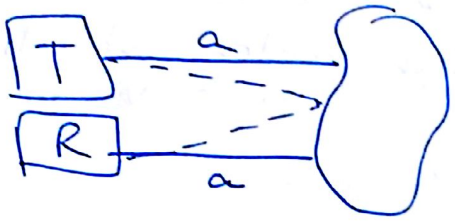
Scanned by CamScanner

RADAR

(Radio Detection & Ranging)

- In radio waves,

$$t = \frac{2a}{c}$$



- Sampling → Best resolution in time is 44.1 KHz
- $\frac{1}{44100}$ s minimum

$$\Rightarrow \frac{1}{44100} = \frac{2a}{3\times10^8}$$

$$\therefore a \approx 3.9\,KM$$

- So by the time we can take another sample, the object must have a dist of about 4 KM from the Transmitter & Receiver.
- So we cannot detect objects nearer than 4 KM.

——— × ———

⇒ Doppler's Effect:

- $F = \left(\frac{c}{c+v_s}\right) \times F_0$ } Source moving towards receiver

$$F = \left(\frac{c+v_r}{c+v_s}\right) \times F_0$$ } Both velocities in the same axis.

→ (1)

Scanned by CamScanner

$$\Rightarrow F = \frac{\left(1 + \frac{V_R}{c}\right)}{\left(1 + \frac{V_S}{c}\right)} \times F_0 = \left(1 + \frac{V_R}{c}\right)\left(1 - \frac{V_S}{c}\right) \times F_0$$

$$\because \frac{1}{1-x} \approx 1 + x \quad \left.\right\} \text{From Gr.P}$$

$$\therefore F = \left(1 - \frac{V_S}{c} + \frac{V_R}{c} - \frac{V_S V_R}{c^2}\right) \times F_0$$

$$\Rightarrow F - F_0 \approx \frac{(V_R - V_S)}{c} \times F_0$$

$$\therefore \Delta f = \frac{\Delta V}{c} \times f_0 \longrightarrow (2)$$

- Now when we send out a signal
* at some f, say $f = 2.4\,GHz$

$\Rightarrow$ On coming back, we observe

$$f' = f \times \left(\frac{c - V}{c + V}\right), \quad V = \text{velocity of moving object}$$

$$\approx f \times \left(1 - \frac{V}{c}\right)^2$$

$$f' \approx f \times \left(1 - \frac{2V}{c}\right)$$

$\therefore$ We have Δf & f_0, so we can get ΔV.

$$\therefore \Delta V = V_R - V_S$$

since V_S is 0.

$$\Delta V = V_R$$

$\Rightarrow$ The problem here is its hard to differentiate f' from f when we plot the DFT.

Scanned by CamScanner

⇒ Suppose our waves are cosines,

- Source $= \cos \omega_0 t = \cos 2\pi F_0 t$
  Recv $= \cos \omega t = \cos 2\pi F t$

* • On multiplying the two signals,
  * $\cos \omega_0 t \cdot \cos \omega t = \cos(\omega + \omega_0)t + \cos$

2

⇒ So on plotting the FFT, the peak at $\omega + \omega_0$ is huge so we can ignore it & consider

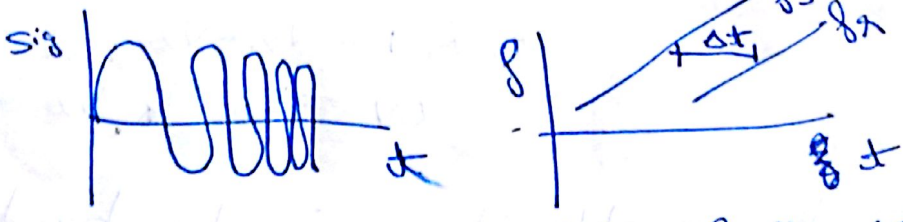
$\Delta F = \frac{\omega_0 - \omega}{2\pi}$ → A peak is observed in DFT. which is easy to read.

- From here we can get Δv.

— x —

* Now we face another problem, we can get the velocities of objects, but how do we get distances of objects in the digital domain?

→ We send a 'Chirp' signal,



- Using the FFTs of the sent & received signals we get Δt.

— x —

Scanned by CamScanner