5/3/19

Process Economics / Finance / A/c

- ROI
- NPV
- IRR
- Depreciation — SLM, NDN
- Cost Index
- Break-even point
- Cost Estimation.

Science / Technology / Engineering

Science → Lab scale activity to demonstrate and explain a phenomena (explicate nature)

Technology → Scale up (More quantity) [In case of Solar cells, the science was there E=hν, but the fields of solar panels has come up recently. This huge scaling up is the Technology]

Engineering → The mass production is made economically viable by engineering. The cost of producing Solar electricity is now comparable to thermal electricity. Another factor is safety, which needs to be ensured by engg. (Bhopal Gas leak) Another is sustainability. The process shouldn't harm the environment

The primary objective of a venture is PROFIT

Profit = Income − Expenditure

Income = Sales Revenue.

Expenditure = Raw material + Manpower + Maintenance + Power Consumption + Post Production Activity (Sale, Marketing, Logistics)
or
Post Manufacturing Activity

Marketing is basically marketing research like segmentation etc. (Intelligence) also is the implementation of the selling plan. They try to push the product into the market.

Accounting is keeping track of all the financial transactions finance is the brain behind how the money will flow through the company. How to save, from where money will come.

Gross profit = EBDIT

Net profit = EBDIT − (Depreciation + Interest + Tax).

EBDIT – Earnings before depreciation, interest, tax.

Net profit → Dividend
Reserve & Surplus (for expansion, etc.)

Interest is the service to debt

Dividend goes to the equity holder.

Haldia Petrochemical → Initial Investment = ₹ 1200 Cr.

Debt : Equity = 3 : 1

Equity = 300 Cr → Out of this, only 100Cr paid by Promoter

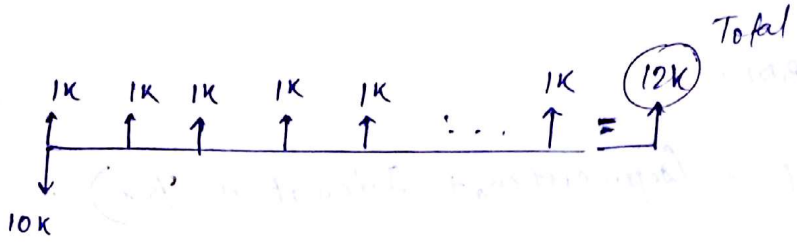
Debt = 900 Cr → Rest arranged from his own network. ← Purnendu Chatterjee.

The profit is used to repay the debt. If the debt can't be repayed then the debt is converted to equity.

$12000 = 10000(1+r)^{12}$

$\Rightarrow (1+r)^{12} = 1.2$

Say today you pay ₹10000 and every month you get ₹1000 for 1 year.



r = Discounted Cash Flow Rate of Return (DCFRR)

$10 \times (1+r)^{12} = 1(1+r)^{12} + 1(1+r)^{11} + \cdots + 1(1+r)^{1} + 1$

$\Rightarrow 10 \times (1+r)^{12} = \frac{1}{1-(1+r)}\left[1 - (1+r)^{12}\right]$

If we get 12K in total after a year, the annual r.r = 20%

But if we get 1K every month for 12 months, the annual r.r > 20%. 1st month, 10K ⇒ gives 1K. Next month, 9K gives 1K. On the 12th month, 1K gives 1K, so very high r.r.

9) Cash flow Table

year (t)	Project A	Project B
0	-210	-50
1	70	20
2	70	20
3	70	20
4	70	20
5	70	20

r.r. = 10% p.a

$$NPV_A = -210 + \left(\frac{70}{1.1} + \frac{70}{1.1^2} + \cdots + \frac{70}{1.1^5} \right)$$

$$= -210 + \frac{70}{1.1}\left(1 + \frac{1}{1.1} + \cdots + \frac{1}{1.1^4} \right)$$

$$= -210 + \frac{70}{1.1} \frac{\left[1 - \left(\frac{1}{1.1}\right)^5 \right]}{1 - \frac{1}{1.1}} = 55.351$$

$$NPV_B = -50 + \frac{20}{1.1} \frac{\left[1 - \left(\frac{1}{1.1}\right)^5 \right]}{1 - \frac{1}{1.1}} = 25.31$$

But NPV is higher in A as its initial investment is higher

So, for IRR.

$$-210 + \frac{70}{(1+r)} \frac{\left[1 - \frac{1}{(1+r)^5} \right]}{1 - \frac{1}{(1+r)}} = 0$$

$r_A = 0.1984 = 19.84\%$

$$-50 + \frac{20}{(1+r)}\left(\frac{1-\frac{1}{(1+r)^5}}{1-\frac{1}{(1+r)}}\right) = 0 \Rightarrow r = 0.4377$$

$$= 43.77\%$$

Cost Index

Sensex has shares of 100 companies in the list. There are certain weights associated with each company (based on several factors). ← Sensitive Index

$$\text{Index} = \sum w_i P_i$$

where P_i is the price of share of i^{th} company
w_i is its weight

If say Index goes from 10000 → 11000, then there is an overall increase in the share price of all the companies by 10% (in Sensex) although not individually.

Chemical Engineering Cost Index

[Cost Index is always based on past cost data]

Year	Cost Index
1975	→ 182
1980	→ 261
1985	→ 325
1990	→ 358
2000	→ 394
2005	→ 416
2010	→ 410
2015	→
2020	→ 475 (by extrapolation)

Economy of Scale for Chem E plants

$$\frac{C_2}{C_1} = \left(\frac{Cap_2}{Cap_1}\right)^n$$

C = Cost , $n \in [0.48, 0.87]$

Cap = Capacity (prod. cap.)

~~no. of years~~

For any Equip or plant = 0.6 (n value)

Q) Cost of a 1250 TPD ammonia plant in 1990 was $140 million. What is the cost estimation of a 2500 TPD ammonia plant today

Peter & Timmerhose Process economics

Extrapolate the data till year 2020

$$\therefore \frac{C_2}{C_1} = \left(\frac{2500}{1250}\right)^{0.6}$$

$$\Rightarrow \frac{C_2}{140\,m} = 2^{0.6}$$

$\Rightarrow C_2 = \$212.2\,m$ in 1990.

$$\therefore C_2 \text{ in } 2020 = 212.2 \times \frac{475}{358} = \$281.55\,m$$

↑ Cost estimate of ammonia plant in 2020. (Today).

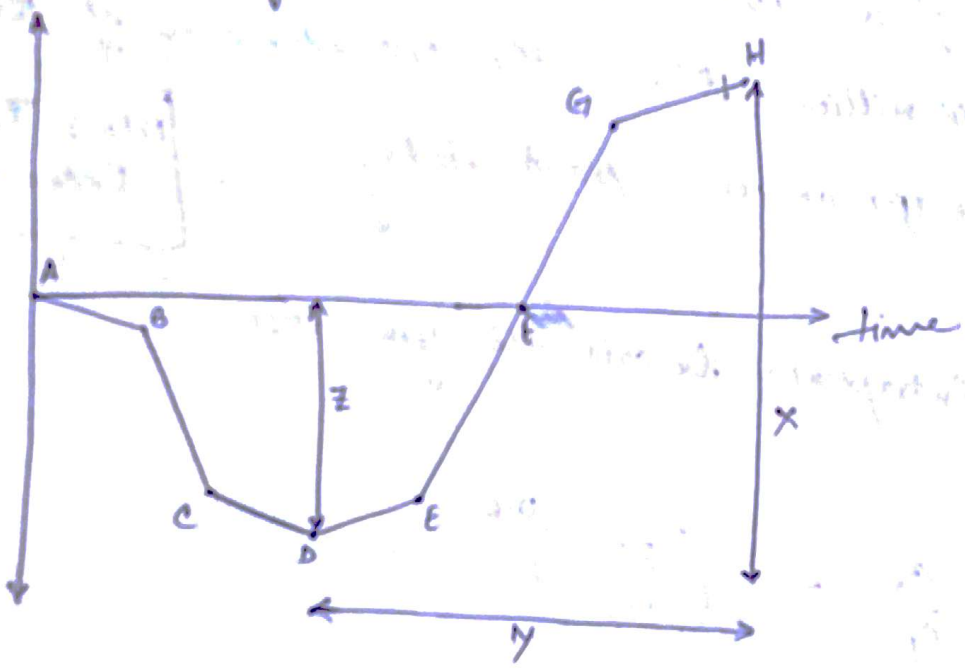
This $ 200 cm is the plant Affin including, land, EPC contractor's margin, regulatory activity. The total project cost is say $ 350 cm.

(engineering, procurement, construction) (like L&T)

Say, we assign $ 100 cm → equity
$ 250 cm → debt

After the procurement (where is there is technical bid and commercial bid, where different companies are selected for material and service procurement), trial run of the plant is done.

Com. Cost(₹) Life Cycle of Plant



AB - Design, development → Preliminary work

BC - Procurement & Construction (Civil, Mech., Electrical) Phase

CD - Commissioning Phase (where working Capital is invested)

DE - Production starts with some trial run

EG - Full Commercial Production

GH - Plant becomes aged, Technology becomes aged.

At G H, high maintenance.
So at H, retire plant, or modernize plant.

BC → Max. money invested, as building of plant

CD → Working capital is ~ 5-8% of Project cost

∴ Total Investment is upto D, i.e. from A to D.

DE → Product is sold as scraps, as trial run.

EG → Total production lifetime (useful lifetime) of plant.

F → BREAK EVEN POINT

FG → Total Profit

GH → Final Phase of Plant (Obsolescence, Maintenance Cost↑, Better Technology has come up).

How to calculate ROI ?

Say Z = Total Investment

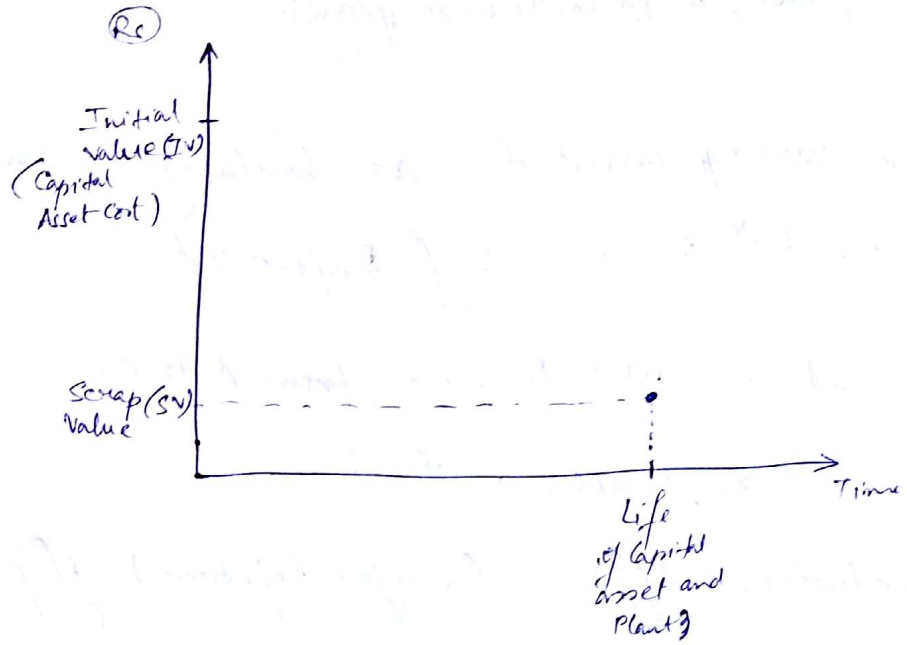
X = Total Revenue Generated.

Y = Time period when the plant gives return

$$\therefore ROI = \frac{X}{Y} \cdot \frac{100}{Z}\% = \frac{\text{Total Revenue}}{\text{Total Time} \times \text{Total Investment}} \times 100\%$$

Return on Investment

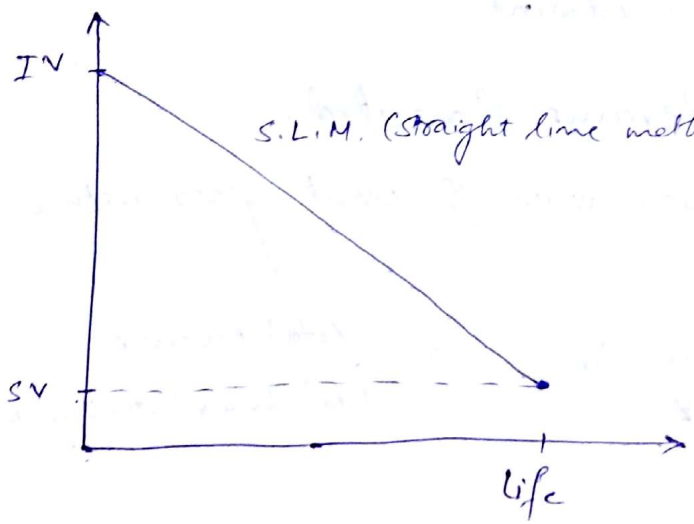
11/3/19



We need to keep some money aside as depreciation fund, so that, at the end of lifetime, we can replace the assets, after selling the scraps.

(1) Straight line method (of depreciation fund):

Each year, how much money to keep aside



∴ Depreciation fund kept aside year

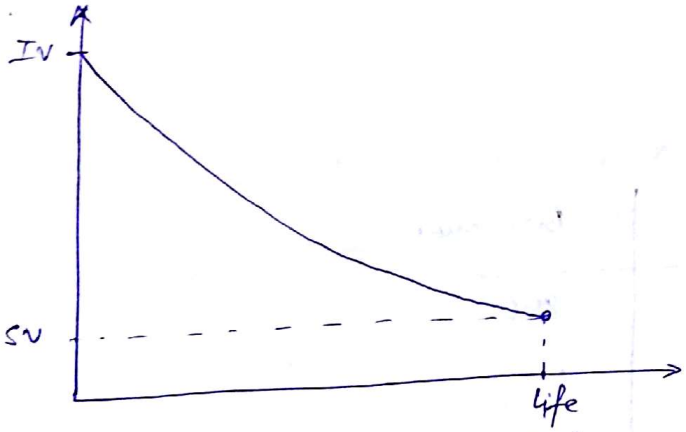
= Depreciation per year $= \frac{IV - SV}{Life}$

Say, IV = 1200, SV = 200, Life = 5 yr

$$D = \frac{1200 - 200}{5} = 200 \text{ /yr}$$

End yr	Depr.	Book value (Depreciated value) of asset
1	200	1000
2	200	800
3	200	600
4	200	400
5	200	200 = SV

② Written Down Value (WDV) (another method)



The slope is such that % of depreciation charged on the book value is constant

(Say depr. % = 20%)

End year	Depr.	Book value
1	240	960
2	192	

If life = 5yr, SV = 200, IV = 1200, dep^{n} % = ?

$$\frac{1200}{(1+x)^5} = 200$$

$$\Rightarrow (1+x)^5 = 6$$

$$\Rightarrow x = 0.43$$

$$1200 - x \times 1200$$

$$1200 \times \left(\frac{100-x}{100}\right)^5 = 200$$

$$\Rightarrow \frac{100}{100-x} = 6^{\frac{1}{5}} = 1.431$$

$$\Rightarrow 69.88 = 100 - x$$

$$\Rightarrow x = 30.12\%$$

Yr	Dep^{n}	Book Value
1	360	840
2	252	588
3	176	412
4	123	289
5	89	200

WDV is more beneficial than SLM as in 1st year, money kept aside for dep^{n} is 360 by WDV, which is more than SLM. So, by Time Value Of Money, 360 has more value than 200 and also, the tax burden is reduced in 1st year by WDV. So total tax is also less

Govt has 3 income through tax : Income tax, Corporate tax and Indirect tax. So, Govt won't want more depreciation charging as then tax is less. Corporate company would want to charge the depreciation as less no. of years as possible. But Govt sets minimum lifetime of several different assets.

Linear Programming

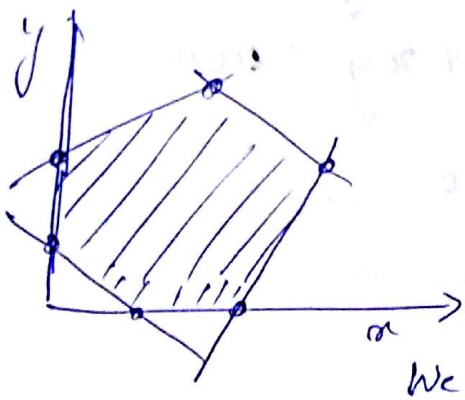
It is part of Operation Research. The origin was for Military Operation. It is basically resource optimization. Most fundamental is linear programming.

There is an objective func. which we want to minimize or maximize. This obj. func. is under some constraints. The term "linear" means both obj. func. and constraints are linear.

The constraints are inequality constraints. There are also some non-negativity constraints (i.e. $x \& y \geq 0$)

Obj func. → $f(x,y)$

Constraint → $g_i(x,y)$



The optimum pt will be one of the corner pt.

We take $f(x,y) = k$ and vary k, so that the k for which $f(x,y)$ touches the feasible region is the optimum.

We get the feasible region from the constraints.

9) A company is producing 2 products P_1 and P_2 in a batch process. There are 2 steps in the process.

	Step 1	Step 2
P1	25 hr	10 hr
P2	10 hr	20 hr

hrs spent for the steps of process.

Batch has capacity of 1000 kg. Step 1 total operating time available is 5000 hr/yr., and for step 2, 6000 hr/yr is available.

Selling price of P_1 = ₹30/kg, P_2 = ₹20/kg.

How many batch of P_1 and P_2 would you plan so that total revenue is max. (All produced is sold).

∴ x = no. of batch of P_1, y = no. of batch of P_2

$$F(x, y) = (30x + 20y) \times 1000$$

~~$(25+10)x + (10+20)y \leq$~~

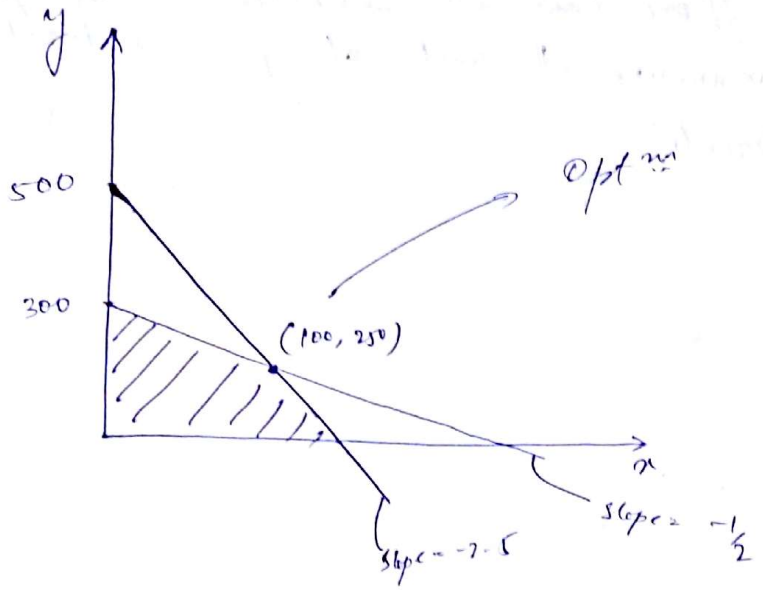
~~$(25+10)x \leq$~~ $25x + 10y \leq 5000$ —(1)

$10x + 20y \leq 1000$ —(11)

$x \geq 0$

$y \geq 0$

9)



$y = -2.5x + 500$

$y = -0.5x + 300$

$-0.5x + 300 = -2.5x + 500$

$\therefore 2x = 200 \Rightarrow x = 100$, $y = -50 + 300 = 250$

$\therefore F = (30 \times 100 + 20 \times 250) \times 1000$

$= \$8 \times 10^6$ Max. Revenue.

) A gold processor has 2 sources of gold ore, source A and source B. In order to keep his plant running, at least 80 tons of ore must be processed each day. Ore from source A costs $20 per ton to process, and ore from source B costs $10 per ton to process. Costs must be kept to less than $800 per day. Moreover, Federal Regulations require that the amt. of ore from source B cannot exceed twice the amt. of ore from source A. If ore from source A yields 2 oz of gold per ton and ore from source B yields 3 oz. of gold per ton,

how many tons of ore from both sources must be processed each day to maximize the amt. of gold extracted subject to above constraints?

9)

(1)

9) Hair oil → Active Ingredient / Coconut oil (base)

Jiva company wants to produce 400 lit. hair oil with 4% Active Ingredient. They want to blend some stock to make this.

① Stock 1 has 4.5% Active Ingredient, and minimum 40 litr. have to be used.

② Stock 2 has 3.7% Active Ingredients, no minimum constraint.

③ Base oil (0% A.I.) no constraint on amt.

Stock	Cost
Stock 1	$9/lt.
Stock 2	$7/lit
Base oil	$0/lit (free)

Find min. cost of producing final prod.

Stock 1 → x vol.

Stock 2 → y vol

Base Oil → z vol.

$x \times 4.5 + y \times 3.7 + (400 - x - y) \times 0 = 400 \times 4$ — ①

$\Rightarrow 4.5x + 3.7y = 1600$

$x \geq 40$

$x + y + z = 400$ — ①

$4.5x + 3.7y + 0.z = 4 \times 400 = 1600$ — (ii)

$4.5x + 3.7y = 1600$ — (II)

$x \geq 40$ — (III)

$x, y, z \geq 0$ — (IV)

$f = 9x + 7y$

① has to be converted into 2D.

Now, adding base oil is counter productive as adding any amt. of base oil would increase cost of production, as base oil has no active ingredient. If base oil is used, then the consumption of stock 1 will ↑, so more costly.

∴ ① ↦ $x+y = 400$

There are 2 poss. possible options, i.e. only ① 1 and 2, or only 1 and base. Check both.

Fuzzy Logic

12/3/19

What is meant by fuzzy ?

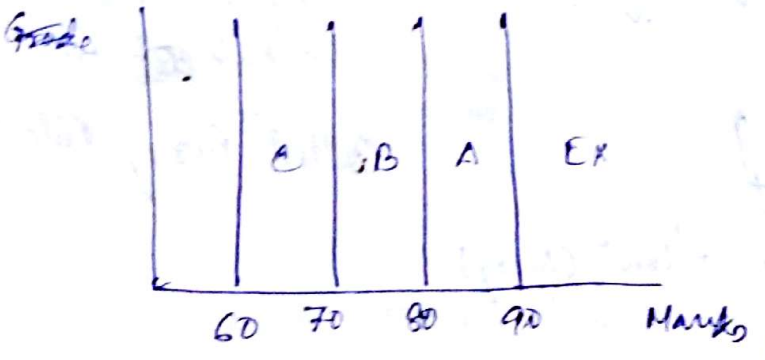
The words which are opposite in meaning to fuzzy are clear, distinct orderly, "well-defined", precise, crisp.

We would like to work more in the fuzzy domain than in the well-defined world, as we have a lot more freedom.

Our brain, in real life situations, in an unclear and not-well defined way. In most cases, our brain takes decisions without any calculations. For example, braking a cycle, how much force to apply. These actions are taken based on some pattern, or experience etc. By when we give the directions, we can't give the precise directions like Google Maps. The directions are incomplete.

The purpose of fuzzy logic is to process incomplete information. Like, how the Umpire decides to give LBW or not.

Example: To Grade Marks.



The lines are very crisp and a person getting 89.9 gets A. But 89.9 and 90 are not much different. The lines are very thin.

In fuzzy logic, we need to make the lines little blurry, i.e. define a band. This is still not fuzzy as the jump from B to A is still there. We need to get rid of the step or jump.

Like, we don't say what the temp. is but how cold or hot we

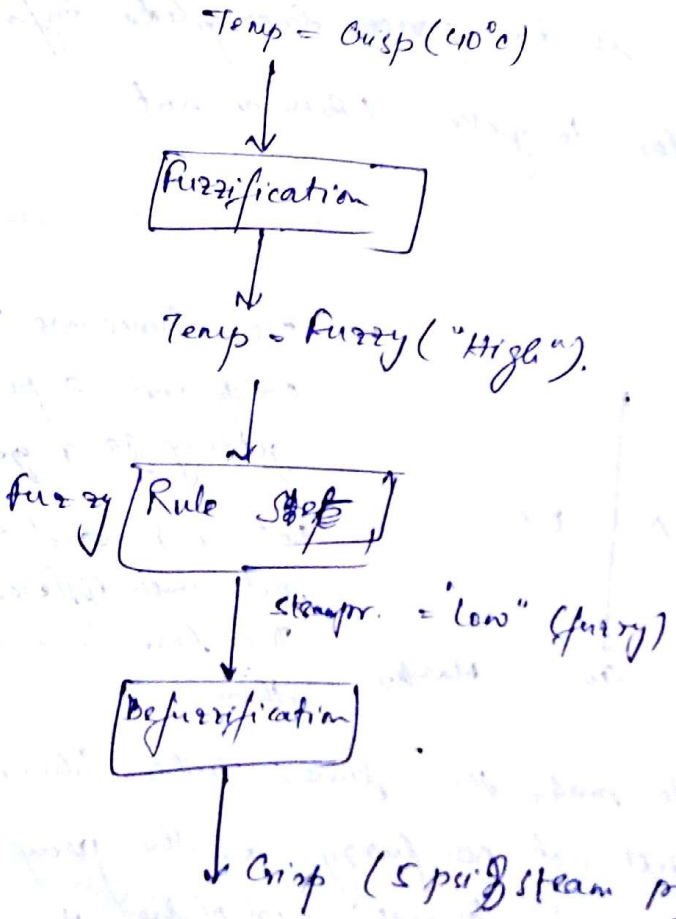
feel. These terms like high, medium, low, hot, warm, cold, are qualitative terms which are not crisp.

Then we go for linguistic variables, like good marks, very good marks, etc.. As the linguistic variables are not very clear, the linguistic variables are called fuzzy variable.

Eg:- If we say he is an avg. student, we don't know exact marks he has got.

This conversion of the crisp variable to a fuzzy variable is called fuzzification and it is the 1st step of the algorithm of fuzzy logic.

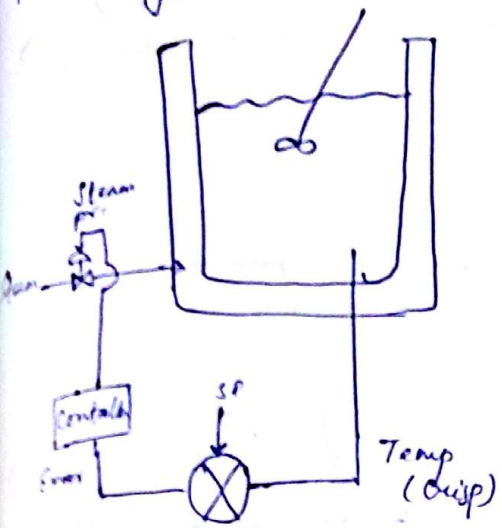
Eg:-

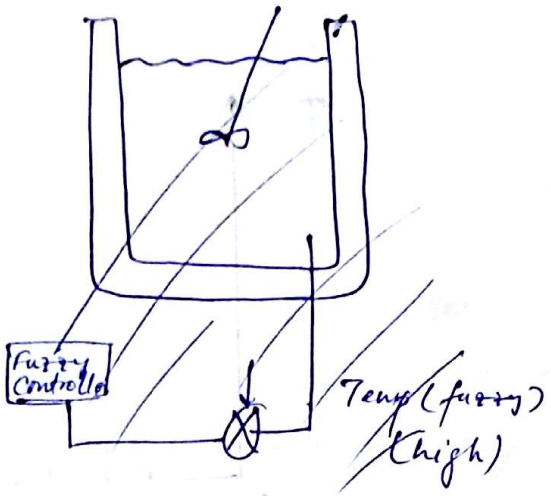


In IPC also, we have fuzzy controllers which act on a band.

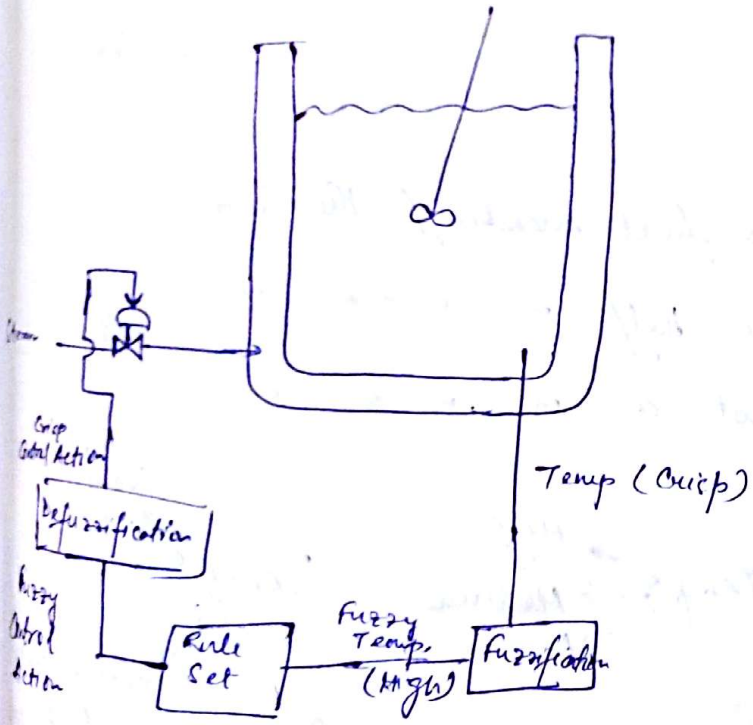
The fuzzified input goes to a rule set, where the action is determined based on the fuzzified inputs using the rule set called fuzzy rule set

Not Fuzzy Controller





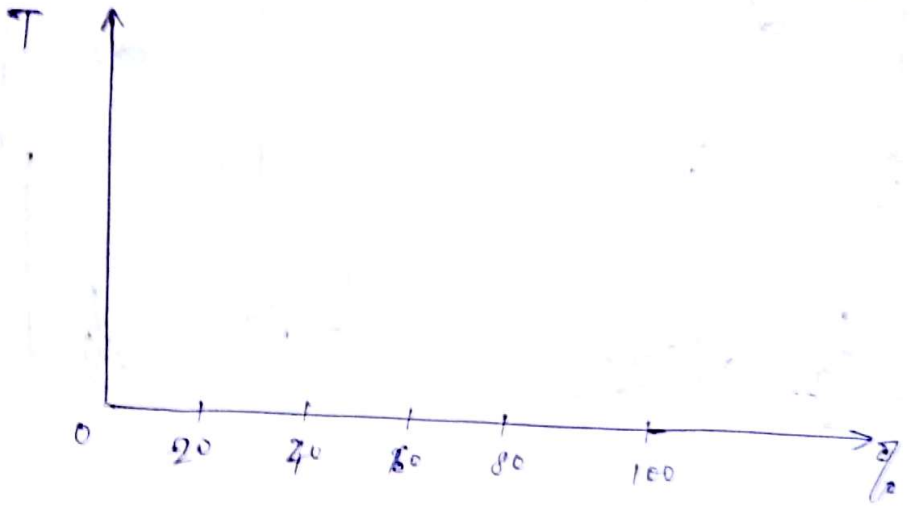
Fuzzy Controller



Any variable can be expressed in a range which can be mapped to 0-100%. Range to % conversion is

$$\% = \frac{X - X_{min}}{X_{max} - X_{min}} \times 100.$$

Say $T \in [20°C, 50°C]$., this is mapped into the graph



This give uniformity, so the variables are normalised to %.

Degree of membership (μ)

If $\mu_i \in [0, 1]$

If $\mu = 1$, it is a full member of the class

If $\mu = 0.5$, " " " half " " " "

If $\mu = 0$, " " " not a " " " "

What is class?

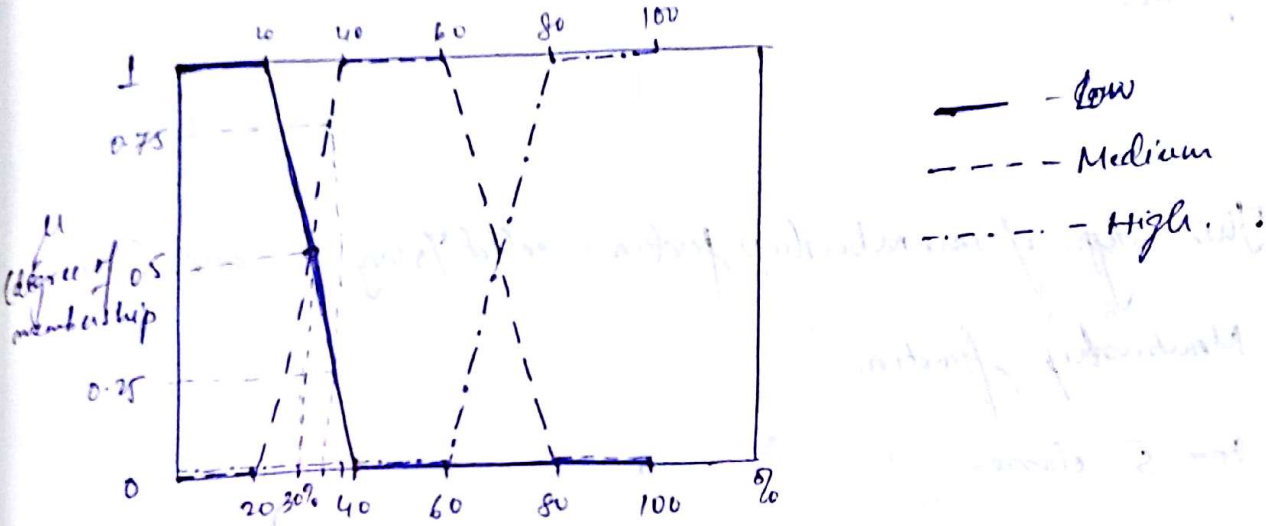
Temp → High, Medium, low

Temp → Very high, high, Medium, low, very low

These High, medium, low, etc. are the classes. No. of class is always odd (3, 5, 7, etc.).

T is mapped in %, so the X axis represents T.

We can plot a graph of μ vs %



Med. → $\% \in [0, 20] \Rightarrow \mu = 1$

$\% \in [20, 40] \Rightarrow \mu \in (1, 0)$.

Say

Say Temp = 30% after normalization.

Temp → High ($\mu = 0$)
Temp → Medium ($\mu = 0.5$)
Temp → Low ($\mu = 0.5$)

At 30% Temp., the temp is both some weight of low and medium, 50% low and 50% Medium.

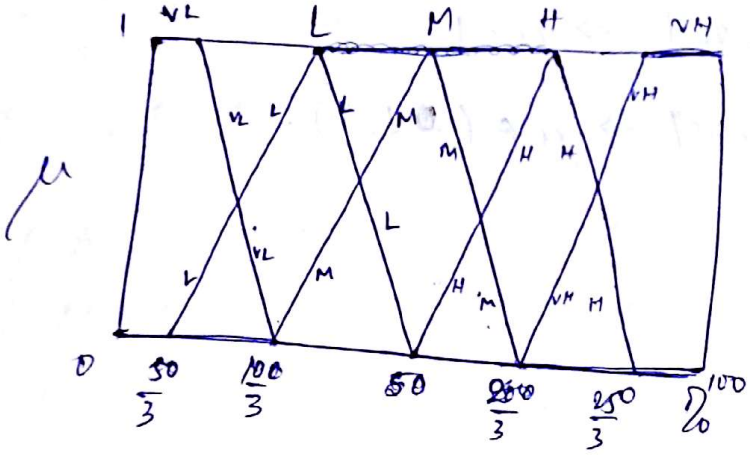
If say T = 35%, $\mu_{High} = 0$, $\mu_{low} = 0.25$, $\mu_{medium} = 0.75$

So 25% low, 75% medium.

So, from the Membership function we find out what's the degree of membership function of the variable in each class. There 10 or more different ways of defining membership func.

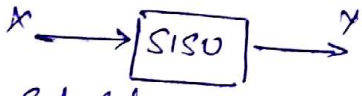
This type of membership function is called Triangular and Trapezoidal Membership function

for 5 classes.



In fuzzy logic, more than one control action are triggered, say for high and med., or med. and low, etc.

Q)

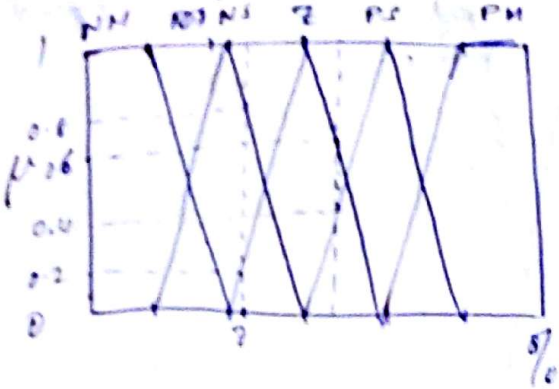


Rule set

If (X)	Then (Y)
1 → NM	NL
2 → NS	Z
3 → Z	Z
4 → PS	PL
5 → PM	PL

N = negative
P = positive
Z = zero
S = small
M = Medium
L = large

The input has 5 classes but output has 3 classes



Say x% is such that

x% → Z (0.2 → PS)
x% → NS (0.8)

Rest are all 0

So PS (0)
PM (0)
NM (0)

∴ Rule (2) and (5) are triggered.

* The membership value of input and output are exactly same for a particular rule.

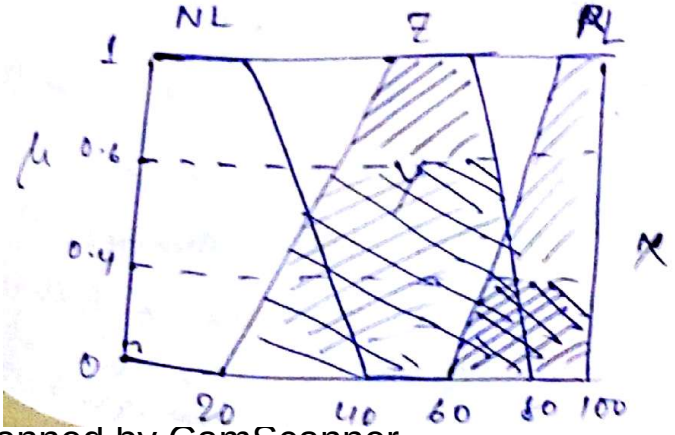
So my fuzzy output from rule evaluation is

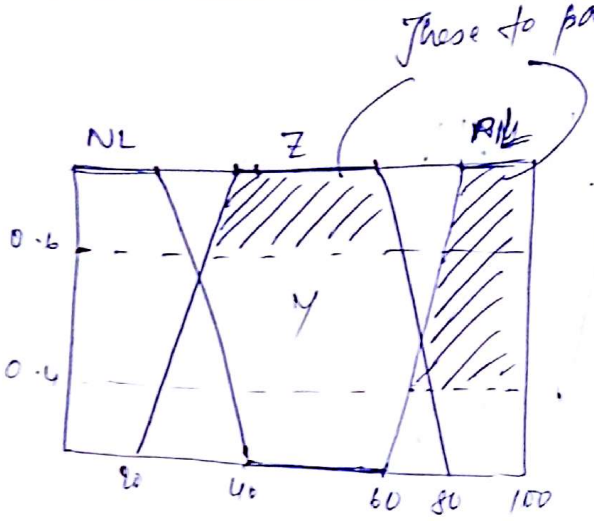
y → Z (0.2)
y → Z (0.8)

Say for another
x% → PS (0.4)
x% → Z (0.6)
Rule (3) and (4) triggered
∴ Fuzzy output
y → PL (0.4)
y → Z (0.6)

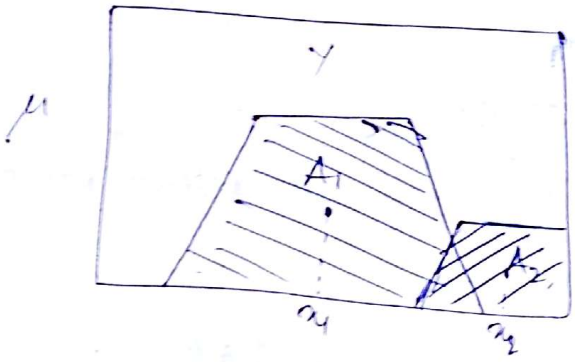
Now, we need to defuzzify our output.

For that, we need the membership function of y





We are left with



A = Area

We need to first find the centre of gravity of A_1 ($\bar{x}_1$) and cg of A_2 (i.e. $\bar{x}_2$), and find the total's centre of gravity of all the areas

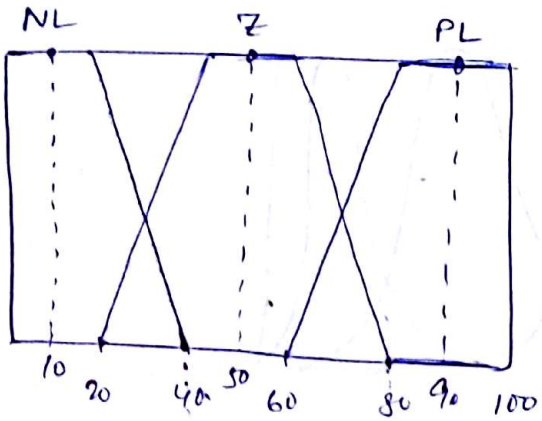
$$\bar{x} = \frac{\int x\, dA}{\int dA}$$

$$\bar{X} = \frac{\Sigma A_i x_i}{\Sigma A_i}$$

The last one is called Method of C.G.

A quicker, but less accurate method is:

Method of Height



For flat top, the max. height is taken at the mid-point % value. For Δ, it is only a pt., but for trapezium, we take the midpt.

Say Y → Z (0.5)
→ NL (0.1)
→ PL (0.4)

∴ Defuzzification is

$$Y\% = 0.5 \times 50 + 0.1 \times 10 + 0.4 \times 90$$

(Z_{mom}, NL_{mom}, PL_{mom})

$$= 25 + 1 + 36$$

$$= 62$$

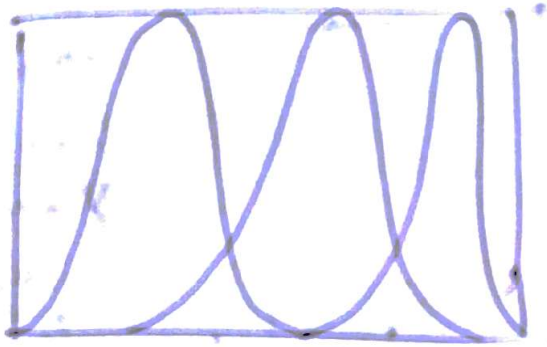
This is quicker, but inaccurate, as area is also a variable. The height method give equal weightage to all the areas, which is true only if the membership func. is symmetric, and will get same output from CG method as well as height method. But mostly we don't have symm. membership func., and hence CG method is most accurate.

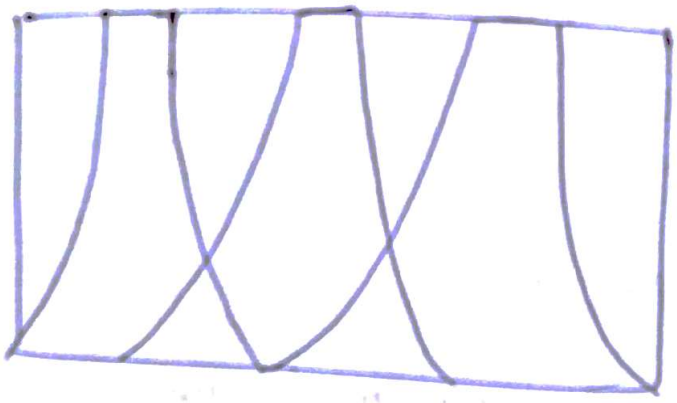
So, we can draw the triangular membership function. But the final objective is to get final output should match plant

Like

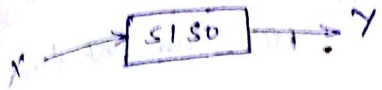
we can have a normal distr. or z-type membership

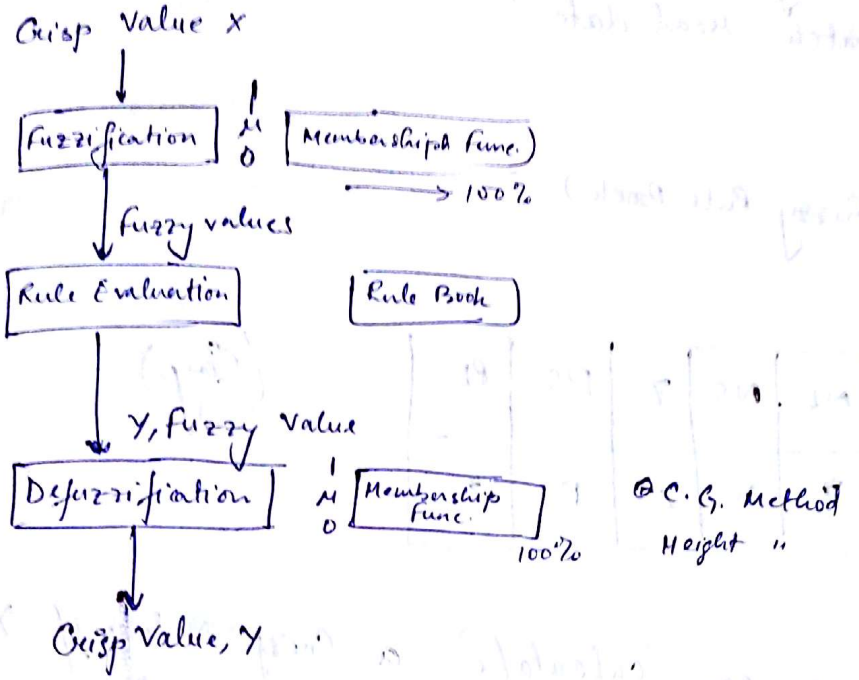
distr.



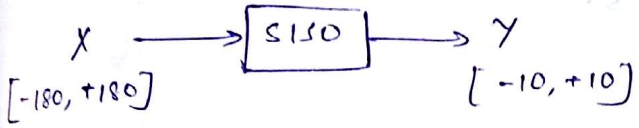


18/3/19



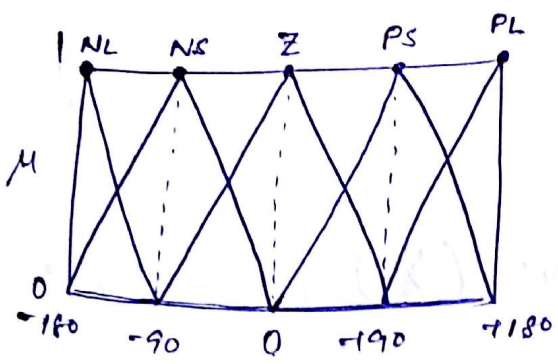


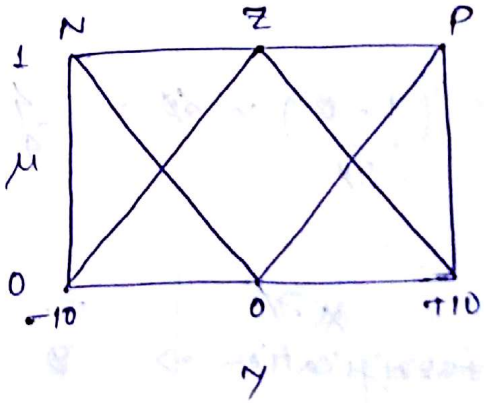
Eg:-



We want to identify the system ~~g.~~ doing Non-Linear Simulation using Fuzzy Logic.

Here, in x-axis, we don't normalize the X input (x-axis) as only on ~~the~~ 1 input





up

y

The no. of classes in our membership func. is one choice. We are designing the Fuzzy logic system. We can tune our system to match real data.

Rule Book (Fuzzy Rule Book)

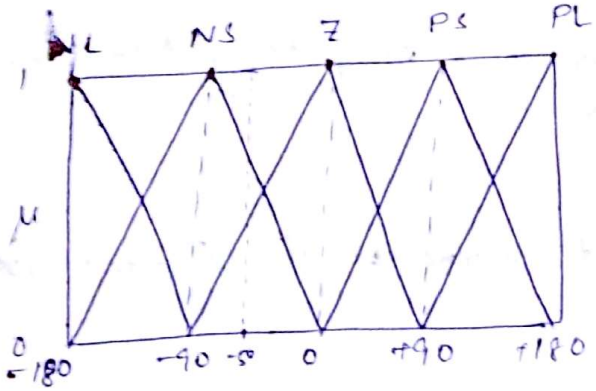
X	NL	NS	Z	PS	PL
Y	Z	N	Z	P	Z

(Say)

Using these, we can calculate a Crisp value of Y from a Crisp value of X.

Say the value of $X = -50$

Sol:



$$\left(\frac{1-0}{90}\right) \times 40 = \frac{4}{9}$$

Fuzzification → X → NS (5/9), Z (4/9)

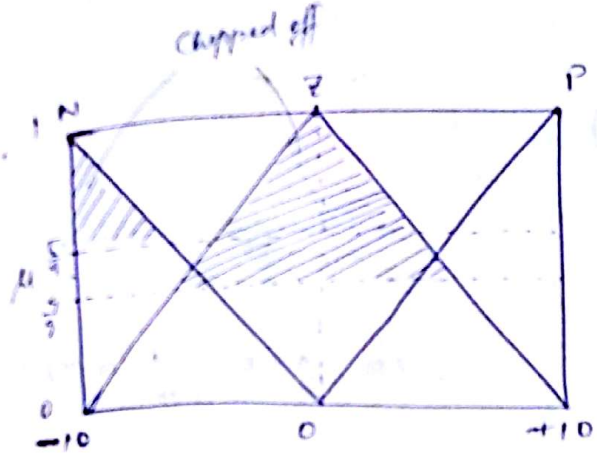
Rule Evaluation:

N (5/9)

Y → Z (4/9)

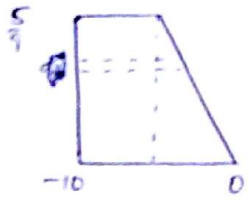
Min - Max Method of Rule Evaluation.

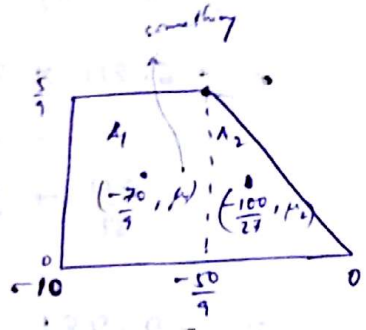
Min of input side and max of it



for a middle trapezium, 0 x-coordinate of CG = 0.

for left trap.





CG of triangle is $\frac{x_1 + x_2 + x_3}{3}$

$\frac{5}{9} - 0 = \left(\frac{1}{-10}\right)(x - 0)$

$\Rightarrow x = -\frac{50}{9}$

$\left(-10 + -\frac{50}{9}\right)$

$-5 - \frac{25}{9} = -\frac{70}{9}$

$\frac{x_1 + x_2 + x_3}{3} = \frac{-\frac{50}{9} \times 2 + 0}{3} = -\frac{100}{27}$

$A_1 = \left(10 - \frac{50}{9}\right) \times \frac{5}{9} = \frac{40}{9} \times \frac{5}{9} = \frac{200}{81}$.

$A_2 = \frac{1}{2} \times \frac{5}{9} \times \frac{50}{9} = \frac{125}{81}$.

$\therefore CG_1 = \frac{\left(\frac{200}{81} \times -\frac{70}{9}\right) + \left(\frac{125}{81} \times -\frac{100}{27}\right)}{\frac{325}{81}}$

$= -6.211$

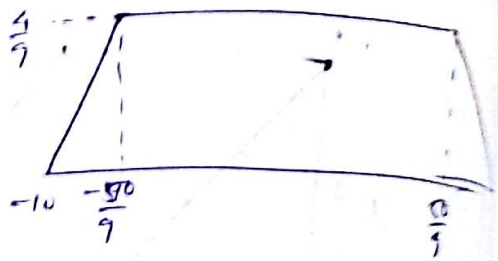
$CG_2 = 0$

$$A_1 = \frac{200+125}{81} = \frac{325}{81}$$

$$A_2 = \frac{1}{2} \times \left(\frac{20}{9}\right) \times \left(\frac{100}{9} + 20\right)$$

$$= \frac{2}{9} \times \frac{280}{9}$$

$$= \frac{560}{81}.$$



$$\frac{1}{9} - 0 = \frac{1}{10}(x + 10)$$

$$\Rightarrow x = \frac{40}{9} - 10 = -\frac{50}{9}$$

$$\therefore CG_1 = \frac{CG_1 \times A_1 + CG_2 \times A_2}{A_1 + A_2}$$

$$= \frac{6.211 \times \frac{325}{81} + 0}{\frac{325}{81} + \frac{560}{81}}$$

$$= -2.281$$

∴ The Crisp $Y = -2.281$ for $X = -50$
(calculated)

The actual system was $y = 10 \sin x$

For $x = -50$, $y = -7.66$.

$$\therefore \varepsilon_i = y_{a,i} - y_{c,i}$$

($y_{a,i}$: actual, $y_{c,i}$: calculated)

X	Fuzzification (X)	Rule Eval (Y)	Defuzz. Y_c
-50	NS (5/9) Z (4/9)	N (5/9) Z (4/9)	-2.28
-150	NS (1/3) NL (2/3)	N (1/3) Z (2/3)	
+45	PS (1/2) Z (1/2)	P (1/2) Z (1/2)	2.04
+100	PS (8/9) PL (1/9)	P (8/9) Z (1/9)	4.61
+120	PS (2/3) PL (1/3)	P (2/3) Z (1/3)	

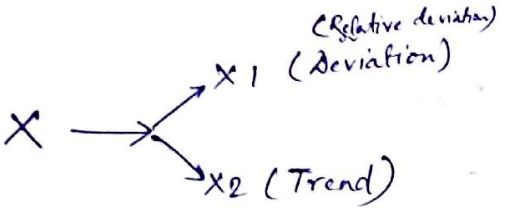
We need to calculate and optimize RMSE by tuning the membership functions.

$$RMSE = \sqrt{\frac{\sum \varepsilon_i^2}{N}}$$

19/3/19

2 Input, 1 Output Application

Apart of normalization of input from 0 to 100%, there are seve
other preprocessing methods, like:



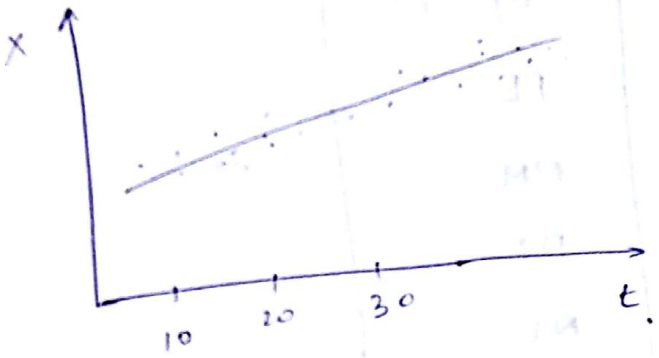
Bifurcated into 2 variable

$$\text{Rel. Deviation} = 1 - \frac{\text{Short Term Avg.}}{\text{Long Term Avg.}}$$

Eg:- In case of the form of a batsman, if he is in a good form
the short Term Avg. (say last 5 months) > Long Term. Avg. (Caree

$$\therefore \frac{STA}{LTA} > 1$$

Say, a new plant's performance degrades from the day 1 to say
after a year. This is drifting, not error. The controller will
correct only the error, not the drift.
A drift occurs slowly. But, whether it is a natural or due to som
anomalies in the system is eliminated by the Relative Deviation
as it involves both the short term avg. and long term avg.
It involves the entire history, not at input at a single time
instant.



$X = a + b't$

Trend = b (slope)

If b = 0, then system is alright

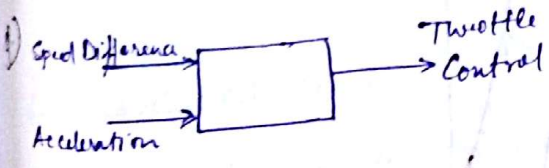
Trend and Rel. Dev. have 5 classes. These give us better control of the system.

Say we have a set point. If the current value decreases but the trend is +ve, then the control action should n't be very strong as it may self-correct. Fuzzy logic optimizes the control action.

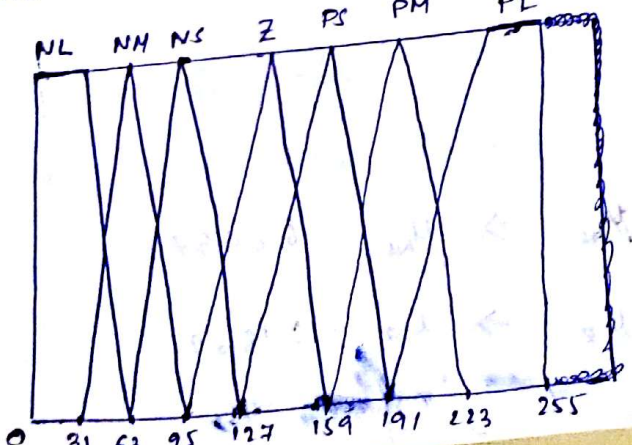
Fuzzy Tech software

Fuzzy Cruise Controller.

Speed Diff. = Actual - Set Pt. speed.



Speed diff. is diff. b/w set pt and Actual speed. Also, the current Acc. is input, so that the Throttle control action can correct the Acc. to reach set pt. speed.



Membership fun. for all (2 inputs and 1 output)

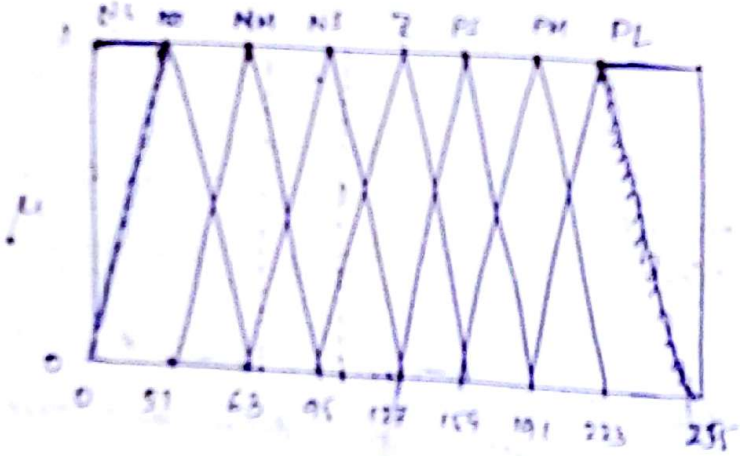
8 bit controller or micro controller

$2^8 = 256$

(Input Normalized b/w 0 - 255)

Rule	IF (SD) IS	AND (ACC) IS	THEN (TC) IS
1	NL	Z	PL
2	Z	NL	PL
3	NM	Z	PM
4	NS	PS	PS
5	PS	NS	NS
6	PL	Z	NL
7	Z	NS	PS
8	Z	NM	PM

If for some input, the office formulation, there is no rule information. Then we don't take any action, ignore it. It's a grey area for the controller.



Input : SD = 110, ACC. = 67

TC = ?

Fuzzification of SD

$\frac{1}{(95-127)}(95-100) = 1 - \mu_{NS} \Rightarrow \mu_{NS} = 0.84375$

$\frac{-1}{(95-127)}(95-100) = 0 - \mu_{PS} \Rightarrow \mu_{PS} = 0.15625$

$\therefore$ SD (100) → NS (0.8438), Z (0.1563)

Fuzzification of Acc.

$$\frac{1}{(63-95)}(63-67) = 1-\mu_{NM}$$

$$\Rightarrow \mu_{NM} = 0.875$$

$$\frac{1}{(95-63)}(63-67) = 0-\mu_{NS}$$

$$\Rightarrow \mu_{NS} = 0.125$$

ACC (67) → NS (0.125), NM (0.875)

SD (100) → NS (0.8438). . . .
Z (0.1563)

~~for the NS comp. of SD,~~

We try all possible combinations of SD and Acc.

For NS in SD, we don't have any rule (No rules for $(NS_{SD} + NS_{Acc})$ and $(NS_{SD} + NM_{Acc})$).

For Z in SD, last two rules are triggered.

$(Z_{SD} + NS_{Acc})$ & $(Z_{SD} + NM_{Acc})$

SD → NS (0.8438)
   → Z (0.1563)

ACC → NS (0.125)
    → NM (0.875)

For SD → NZ (0.1563) & ACC. → NS (0.125)

TC → PS (0.125) [The μ_i of TC will be the minimum of all the μ values for the different inputs].

↳ Analogy => The strength of a chain is the strength of its weakest link

For SD → Z (0.1563) & ACC → NM (0.875)

TC → PM (0.1563)

∴ TC → PS (0.125)
     → PM (0.1563)

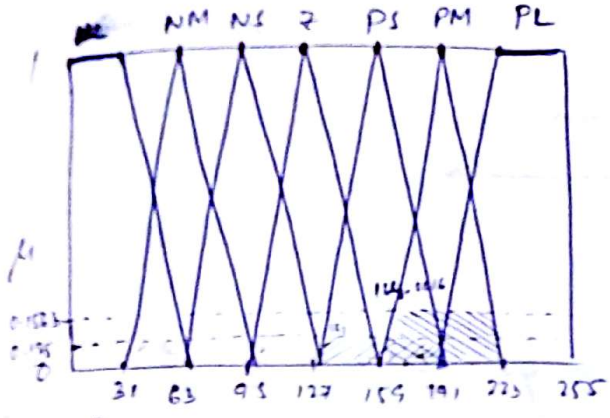
This method is called MIN - MAX method.

The MIN part is in the rule evaluation part.

Say another 3rd rule has been triggered and we get another TC → PS (0.5). Then we take the max. of the 2 PS membership values (among 0.125 and 0.5) and take only PS (0.5). Thus because so, in the defuzzification process, we chop of the part of PS above 0.5, not 0.125.

in the output (fuzzy) if a class appears more than once, then only the max membership value of that class

Defuzzification



$\bar{x}_1 = 159, \quad \bar{x}_2 = 191$

~~$A_1 \quad A_1 < A_2$~~

$\frac{1}{(191-159)}(127 - x_1) = 0 - 0.125$

$x_1 = 131$

$131 - 127 = 4$

$164.0016 - 159 = 5.0016$

$\frac{1}{(191-159)}(159 - x_2) = 0 - 0.1563$

$x_2 = 164.0016$

$A_1 = \frac{1}{2} \times (32 + 32 - 8) \times 0.125$

$= 3.5$

$A_1 = \frac{1}{2} \times (64 + 64 - 8) \times 0.125$

$= 7.5$

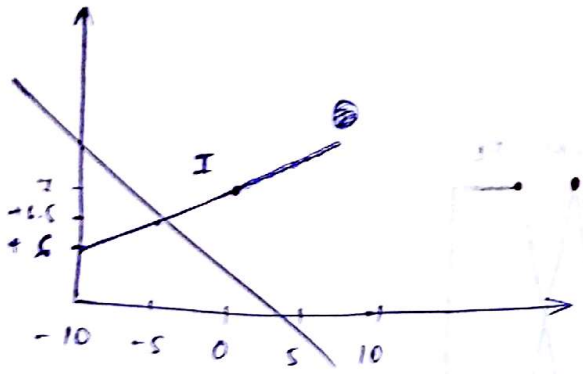
$A_2 = \frac{1}{2} \times (64 + 64 - 10.0032) \times 0.1563$

$= 9.22$

$\therefore CG = \frac{A_1\bar{x}_1 + A_2\bar{x}_2}{A_1 + A_2} = 176.65 = TC$

9) The petrol car engines are MPFI (Multi Point Fuel Injection), where the fuel injection control system needs to be optimized.

If $(-10 < x \leq -5)$ then y

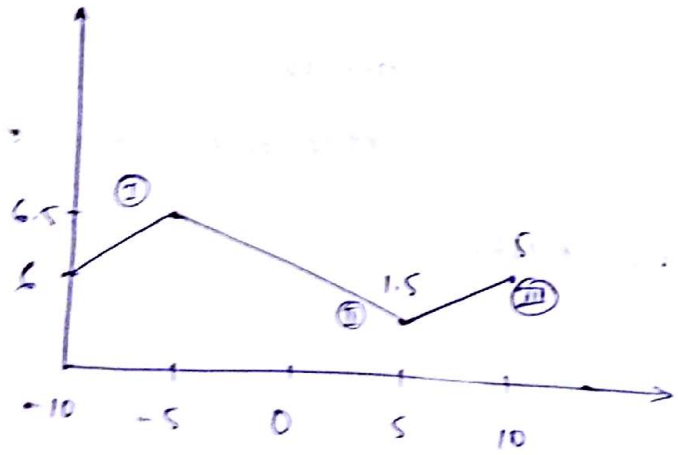


If $(-10 < x \leq -5)$ then $y = 0.1x + 7$ — (I)

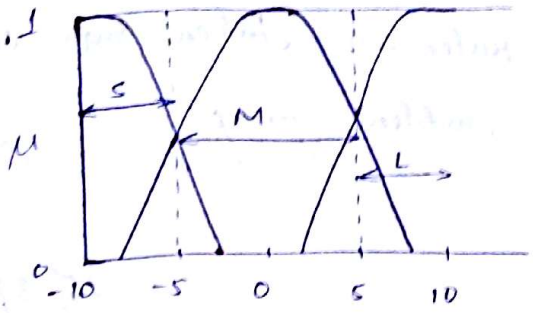
$(-5 < x \leq +5)$ then $y = -0.5x + 4$ — (II)

$(+5 < x \leq +10)$ then $y = 0.7x - 2$ — (III)

} Crisp Rule Set (Not Fuzzy)



But the problem was that during the transition from -5 to +5, or at the corner pt. (transition from (I) to (II) or (II) to (III)), the car was knocking (jerking). So, instead of crisp logic, fuzzy logic was used.



We need to know the crisp boundary b/w the classes using a membership func.

(3) Now, while transitioning from (1) to (2), there is a μ for both S and M.

Now, the Fuzzy Rule set becomes (IF part only)

If x is small, then $y = 0.1x + 7$ — (1)

If x is medium, then $y = -0.5x + 4$ — (11)

If x is large, then $y = 0.7x - 2$ — (111)

If x = -6 input, say

x → S (0.8)
→ M (0.2)

Then both rules (1) and (11) are triggered. We take a μ-weighted avg. of the y_1 and y_2 from rules (1) and (2)

i.e. (1) ⇒ $y = 0.1 \times (-6) + 7 = 6.4$ $\mu_S = 0.8$

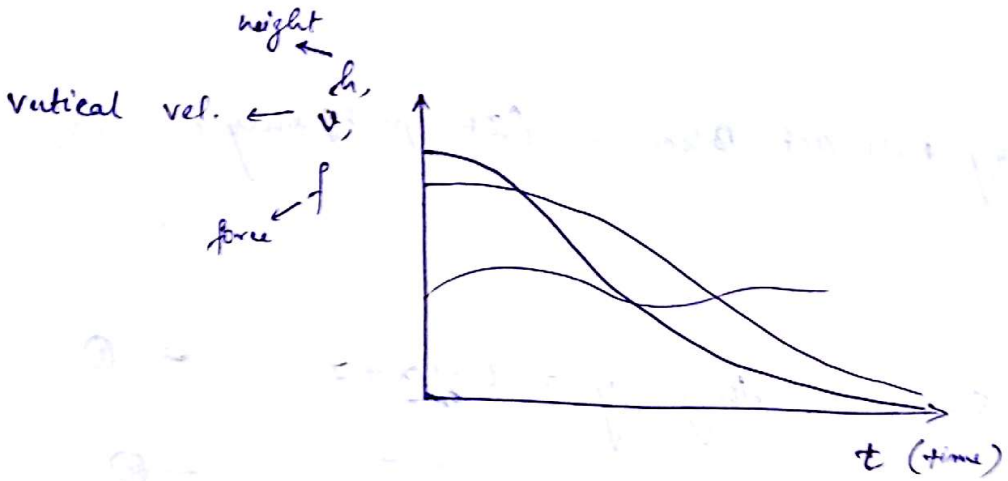
(2) ⇒ $y = -0.5 \times (-6) + 4 = 7$ $\mu_M = 0.2$

$y_{final} = \frac{0.8 \times 6.4 + 0.2 \times 7}{0.8 + 0.2} = 6.52$

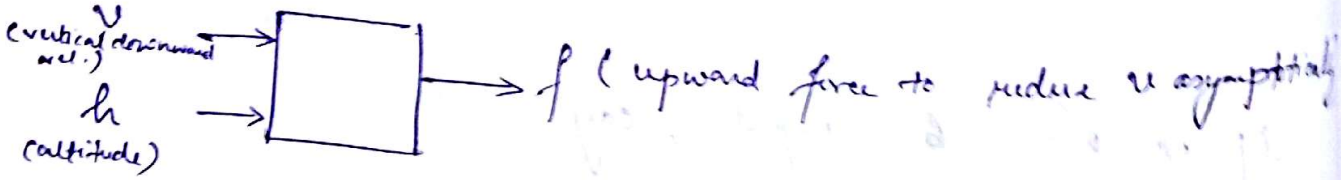
The jerk or knocking is eliminated as near the corner pt., if both the rules are taken into consideration so that there is not a sudden jump.

25/3/19

Example of Flight landing simulation



Fuzzy Logic + Some Physics Eqns.



$a = \frac{\Delta v}{\Delta t}$, $\frac{f}{m} = a$ (force per unit mass of air craft)

(acc.)

$\Delta v = v_{i+1} - v_i$, $\Delta t = t_{i+1} - t_i$

$\frac{f}{m} = F$ (normalized force)

$\therefore f_i = \frac{v_{i+1} - v_i}{t_{i+1} - t_i}$ . . . . . (1)

Let Δt = 1 sec. (chosen by us)

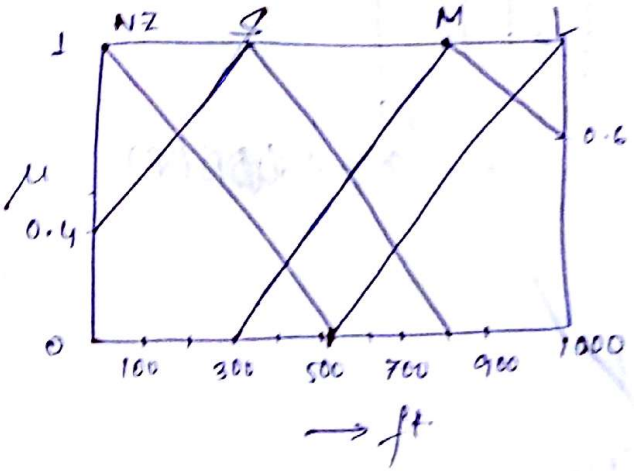
$\therefore \boxed{v_{i+1} = v_i + f_i}$ [for Δt = 1 sec].

from physics

Similarly, $\boxed{h_{i+1} = h_i + v_i}$ [for Δt = 1 sec].

Membership function:

h variable



- The landing algorithm (autopilot) kicks starts from 1000 ft (say).

$(\sum \mu_{(1.6)} > 1$ at $h = 0$, so this is possible too)

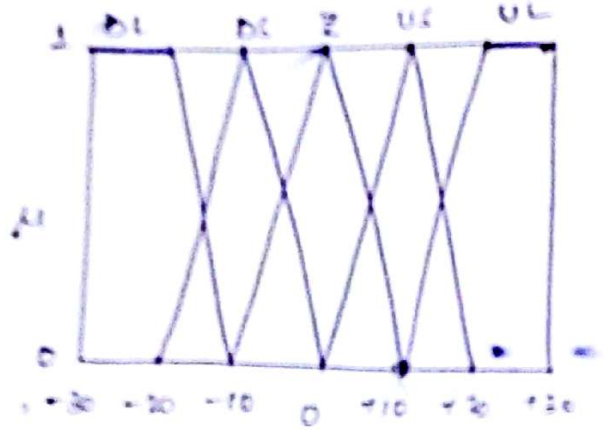
NZ = near zero [~~(0, 0.4) to (300, 0)~~] [(0, 1) to (300, 0)]

S = small [(0, 0.4) to (300, 1)]

M = medium [(300, 0) to (1100, 0.6)]

L = large [(500, 0) to (1000, 1)].

f and [illegible] variables



+ve = upward
-ve = downward

UL = upward large
DS = downward small
Z = Zero.

Rule Set

[illegible]	DL	DS	Z	US	UL
L	Z	[illegible]	DL	DL	DL
M	US	Z	DS	DL	DL
S	UL	US	Z	DS	DL
NE	UL	UL	Z	DS	DS

Input rpm (v)

Input [illegible] (h)

Output ([illegible]) (F)

Fuzzy logic sequence
Fact → Rule → Conclusion
eg:- Fact: Tomato is red.
Rule: If Tomato is red, it is ripe
Conclusion: Tomato is ripe

Initial values

$h_0 = 1000\ ft$

$V_0 = -20\ ft/s$

$f_0 = ?$

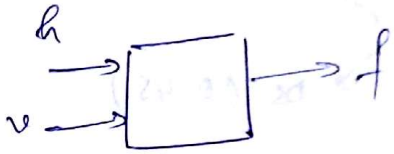
2 Rules will be triggered. Apply MIN-MAX, get f_0.

Then find v_1 and h_1.

Input

Given

t	h	v	f
0	1000	-20	4.5
1	980	-15.5	0.436
2	964.5	-15.064	



Step 1

h (1000 ft) → L(1), M(0.6)

v (-20 ft/s) → DL(1)

} Fuzzification

The rules fired are for L-DL and M-DL

f → Z(1), VS(0.6)

} Rule Eval.

In the rule eval. stage, for a multi-input, the membership value of the output is the min. of all the fs value of all inputs. Among all the outputs if an output is repeated, we take the max fs value in the output.

Min → input
Max → output

} Defuzzification

f(4.5)

$v_1 = v_0 + f_0 = -20 + 4.5 = -15.5$

$h_2 = h_0 + v_0 = 1000 - 20 = 980$

Step 2

h (980) → L (0.96)
h (980) → M (0.64)

$\frac{(980-1000)}{(800-1000)}\left(\frac{1-0.6}{}\right) = (y-0.6)$ } M

$\Rightarrow y = 0.64$

$\frac{1-0}{(1000-500)}(1000-980) = (1-y)$ } L

$\Rightarrow y = 0.96.$

V (−15.5) → DL (0.55)
V (−15.5) → DS (0.45)

$\frac{(1-0)}{(-20+10)}(x+10) = (y-0)$ } DL

$\Rightarrow \frac{-1}{10} \times (-5.5) = y$

$\Rightarrow y = 0.55$

$\frac{1-0}{(-10+20)}(x+10) = y-1$ } DS

$\Rightarrow y = 0.45$

Trick : ∵ Symmetric

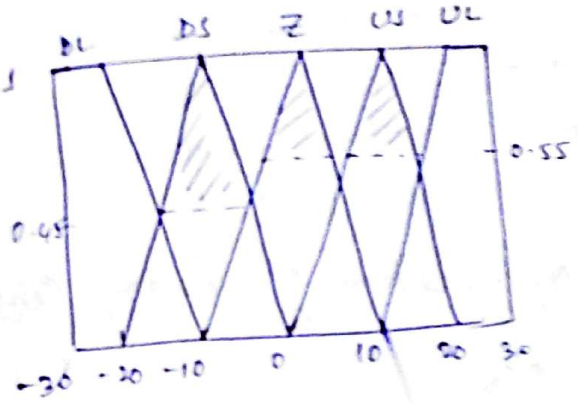
$\therefore y_{DS} = 1 - y_{DL} = 0.45$

∴ 4 rules will be triggered. [(L, DL), (L, DS), (M, DL), (M, DS)]

Z (0.55) → ∵ L (0.96), DL (0.55)
Z (0.45) → DS (0.45)
Z (0.55) → US (0.55)
Z (0.45)

min ← Here Z is repeating, so applying min-max. We take the Max. μ-value of the 2 Z clones while assigning the μ-value of each output class, we take the min. μ-value of the 2 inputs.

Final f → f → Z (0.55)
→ DS (0.45)
→ US (0.55)



DS

$\frac{1}{(1-0.45)} = \frac{20}{L} \Rightarrow L = 20 \times 0.55 = 11$

$\therefore A_{DS} = \frac{1}{2} \times 20 \times 1 - \frac{1}{2} \times 11 \times 0.55$

$= 6.975$

Z

$\frac{1}{1-0.55} = \frac{20}{l} \Rightarrow l = 9$

$\therefore A_Z = \frac{1}{2} \times 20 \times 1 - \frac{1}{2} \times 9 \times 0.45$

$= 7.975$

$A_{UL} = 7.975$

$\bar{Y} = \frac{A_{DS} \times (-10) + A_Z \times 0 + A_{UL} \times 10}{\sum A} = 0.436 = f_1$

$$\therefore v_2 = v_1 + f_1 = -15.5 + 0.436 = -15.064$$

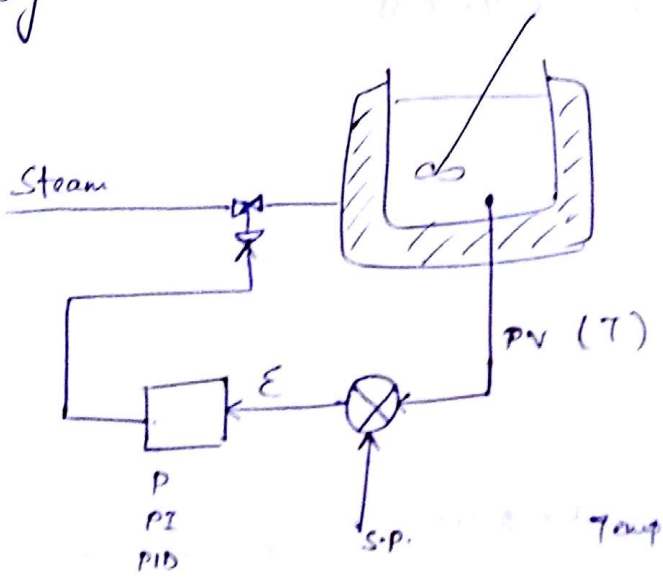
$$h_2 = h_1 + v_1 = 980 - 15.5 = 964.5$$

Fuzzy Controller Configuration

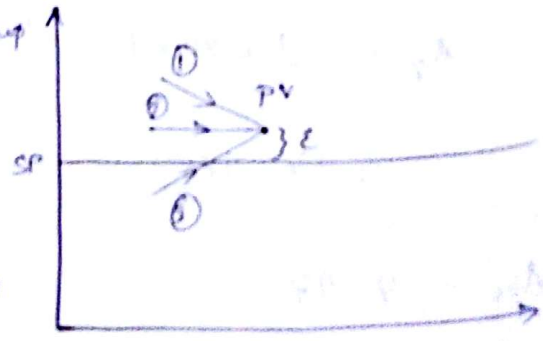
There are 4 popular configuration of Fuzzy Controller:

- Direct
- Supervisory
- PID Adaptation
- Intervention.

Say we want to control the reactor temp. of a jacketed CSTR



P, PI, PID essentially works with the instantaneous ε.



We don't know the past trajectory of the temp., But at that instant, ε value is same, so the traditional controller (P, PI, PID) actions will be same for all 3 trajectories. But logically, we know that ③ requires more control action than ② or ①, i.e. ③ > ② > ①

traditionals controllers don't take the trajectory into account and hence, inefficient. Hence, Fuzzy Controller is needed. It is a SISO system.

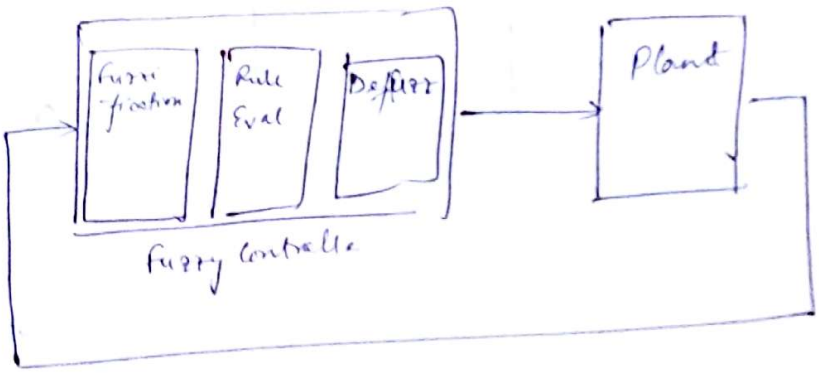
Direct → The traditionals controllers are not there, only Fuzzy Controller is there. The process var. is directly sent to Fuzzy controller and output to control valve.

Supervisory → The traditional controllers are still there, but the Fuzzy Controller controls the set point value.

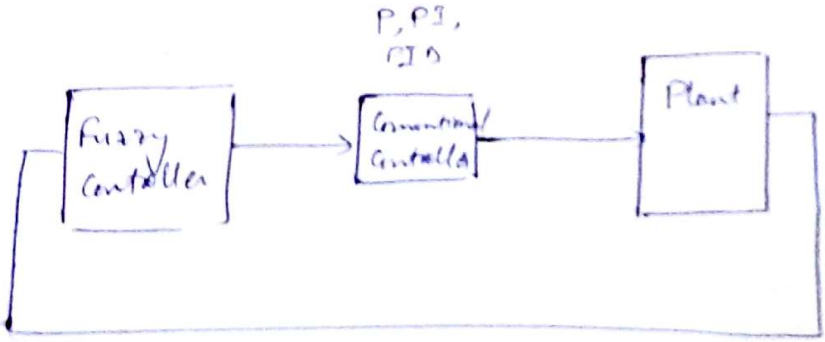
PID Adaptation → The Fuzzy controller adapts the K_c, T_i and T_D values (parameters)

Intervention → Both Conventional controller and Fuzzy Controllers work calculate control actions parallely. When the error is within a limit, the control action of the conventional controller is implemented. If beyond limit, intervention by Fuzzy Controller, and fuzzy control action is implemented. There is a switch over.

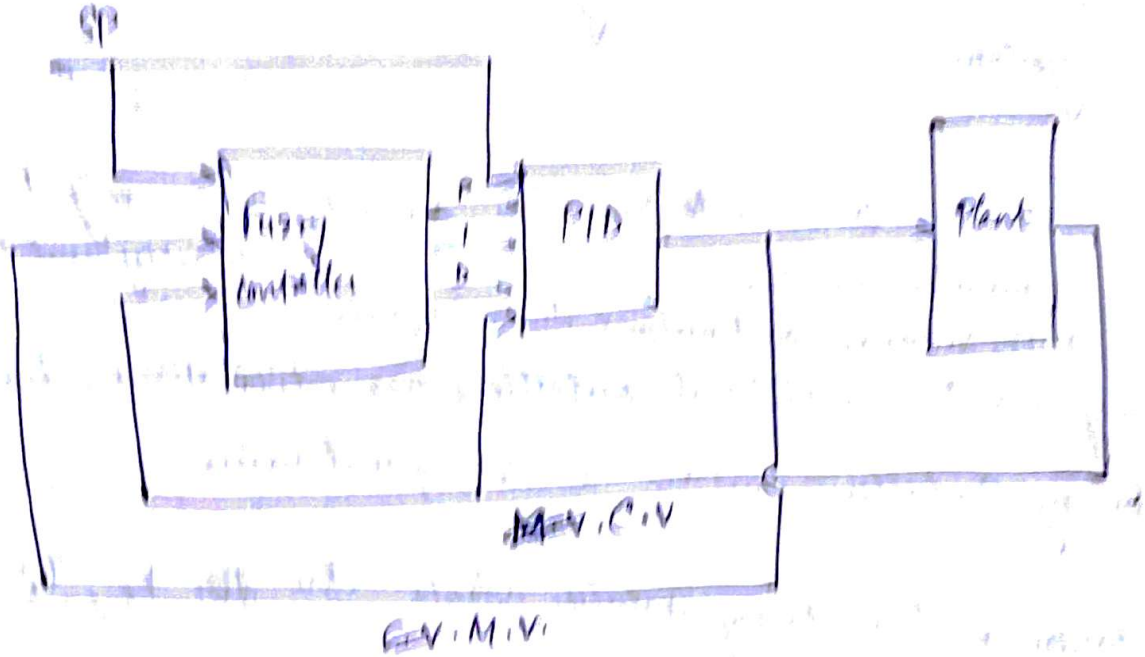
Direct



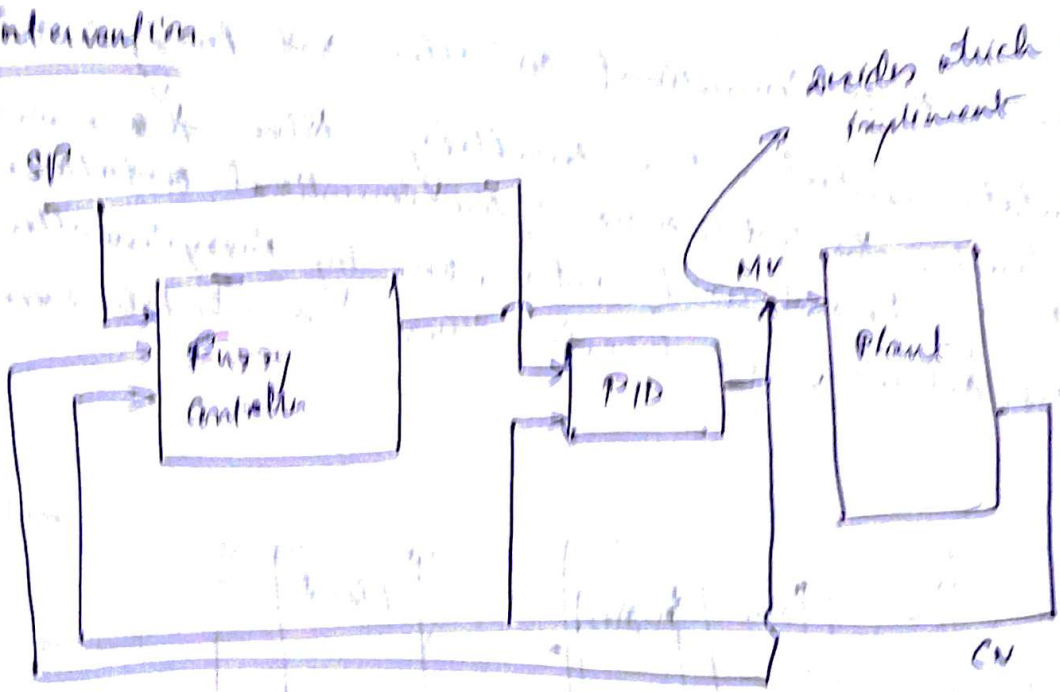
Supervisory



PID Adaptation

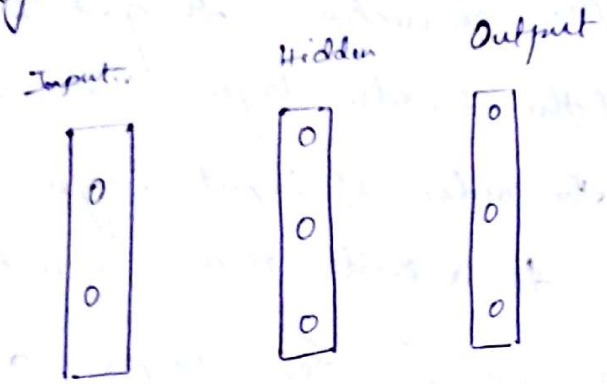


Intervention

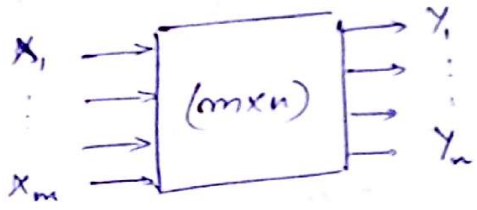


Neural Networks

Layers



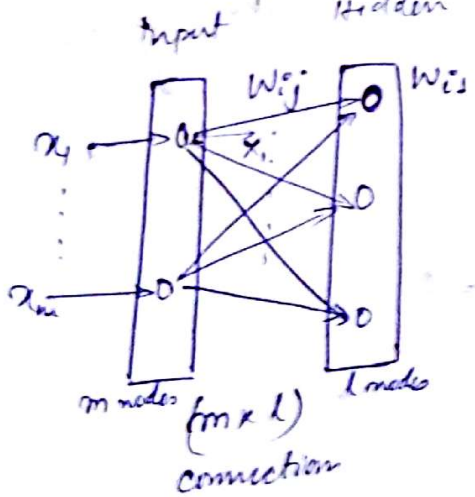
The no. of input and output nodes are fixed. But the no. of nodes in hidden layer can be manipulated. The no. of hidden layers can also be varied.



Each node is like a transfer function. There are <u>activation function</u> inside the nodes. The input data are normalized (from 0 to 1 or −1 to +1). Depending on the activation function, the outputs (y) are also lie $\in [0, 1]$ or $[-1, 1]$

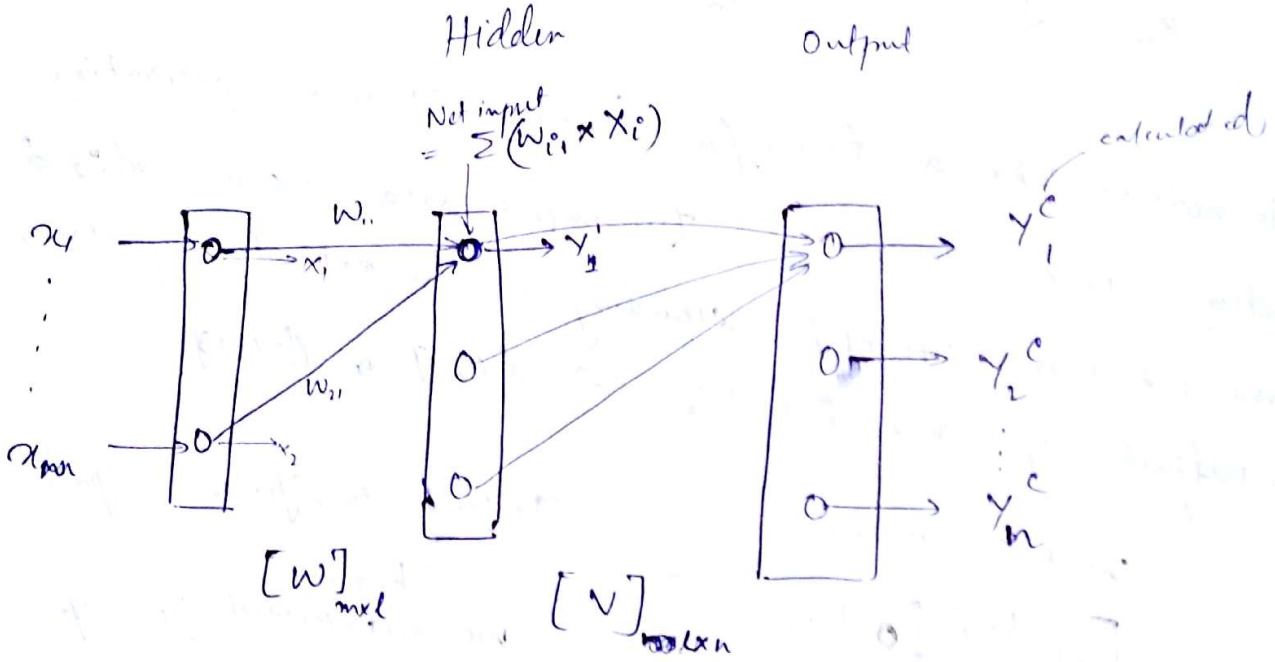
$x_1, x_2 \cdots, x_m, y_1, y_2 \cdots y_n$

However we want to predict $y_1 \cdots y_n$.



W_{ij} is a matrix of order $m \times l$. which has the weights of the connections

Say X_1 is the output of the 1st node of the input. The output of the node is multiplied with the corresponding weight of the connection to get input from a node to a node of the hidden layer. The sum of all the outputs from the nodes of input layer multiplied by their respective weights of all connections to one particular node of the hidden layer gives the net input to the node in the hidden layer. This is subjected to a transfer func. to get output from the node of hidden layer



$$\text{Error} = \frac{\sum (Y^c - Y)^2}{n} = E$$

$$E = f(W, V)$$

The error is a function of the weights. These weights are tuned to get the minimum error, and predict close to actual

output.

$$W_{i+1} = W_i + \Delta W,$$

$$V_{i+1} = V_i + \Delta V$$

$$\frac{\partial E}{\partial W} = 0, \quad \frac{\partial E}{\partial V} = 0$$

We keep refining the [W] and [V] until we get right error. This is called training the network. This needs to be done in a randomized manner, so that there is no bias.

Next comes the testing step, where we test how accurately our neural network can predict the y value.

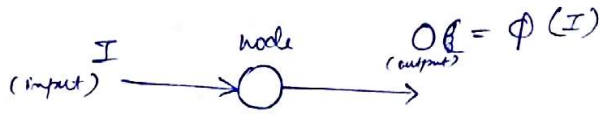
Say we have 1000 records, we take $\frac{2}{3}^{rd}$ for training and the remaining $\frac{1}{3}^{rd}$ for testing.

Neural Network is used generally when there is lot of data. Fuzzy logic is generally used when data is limited, because try to identify how humans would make decisions

1/4/19

Artificial Neural Network (Continuation)

Transfer functions

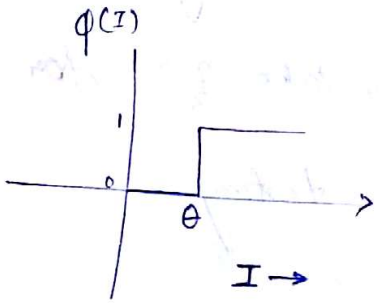


Types:

i) Thresholding Transfer functions :

$$O = \phi(I) = 1 \quad , \quad I > \theta$$

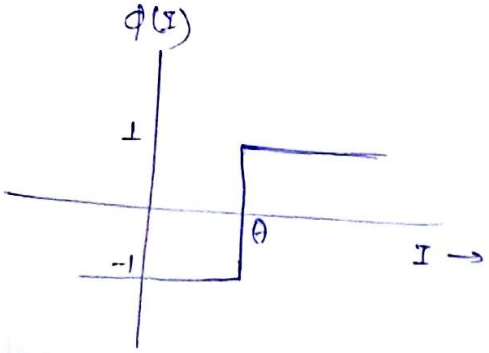
$$= 0 \quad , \quad I \leq \theta$$



ii) Signum function

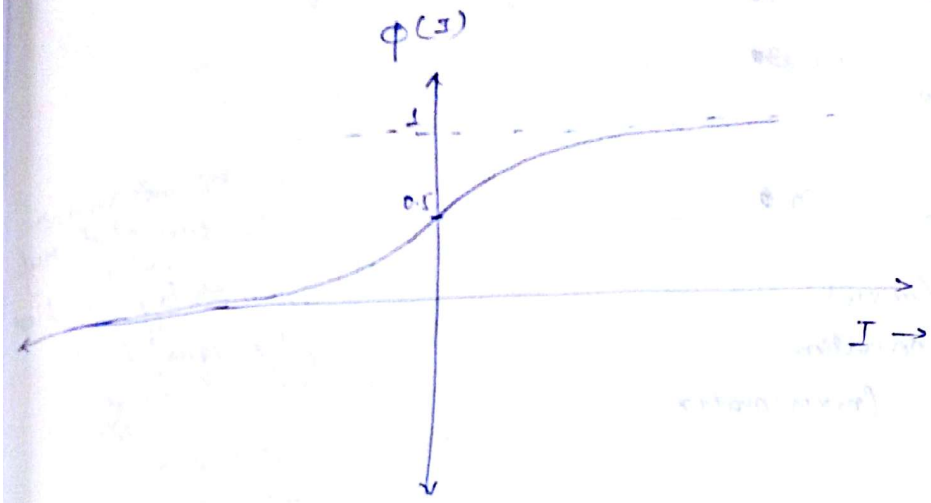
$$O = \phi(I) = -1 \quad , \quad I \leq \theta$$

$$= 1 \quad , \quad I \geq \theta$$



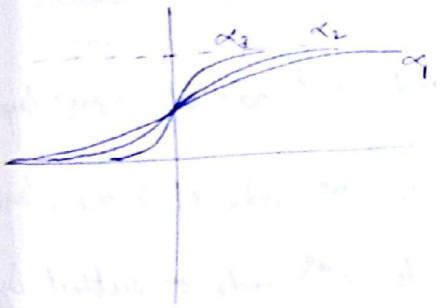
3. Sigmoidal function

$$\phi(I) = \frac{1}{1 + e^{-\alpha I}}$$

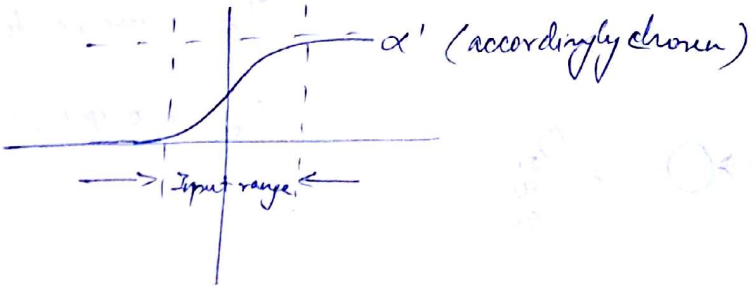


This is a popular TF in ANN

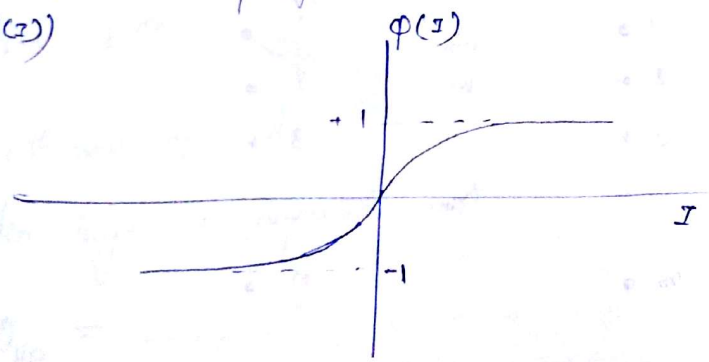
If $\alpha_1 < \alpha_2 < \alpha_3$



∴ The value of α determines the sensitivity zone of input; i.e. depending upon the range of our input, we choose the value of α for our sigmoidal TF.

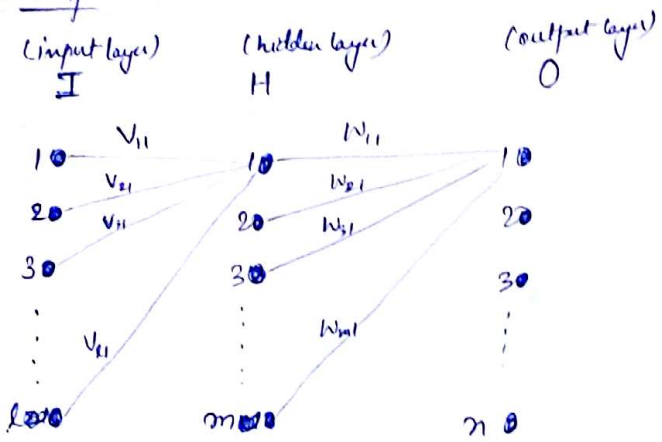


(iv) tanh (I) transfer function → The output ranges from −1 to +1 (≡ φ(I))



Example

● = node



l x m x n systems.

The sum of the weights for one layer need not be equal to 1

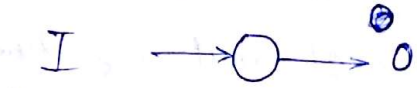
(l x m) connections

V_{ij} = weight of connection of i^{th} node of input layer and j^{th} node of hidden layer

(l x m) matrix

(m x n) connections

(m x n) matrix



~~I_{I1} = input to input layer's~~

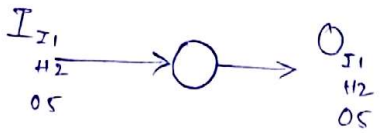
Input I has different subscripts

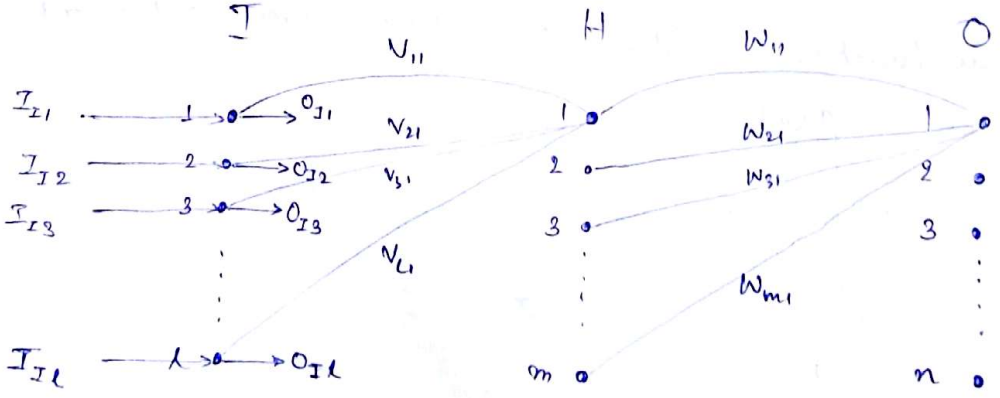
I_{I1} = input to 1st node of input layer

I_{H2} = input to 2nd node of hidden layer

I_{O5} = input to 5^{th} node of output layer

O_{O5} = output from " " " " "





$\{I\}_l$ → ANN → $\{O\}_n$

$I_{I1} \cdots I_{Il}\ O_{O1} \cdots O_{On}$ Process value (Actual values)

Output from Input layer

Let the transfer function of all node in input layer is 1.

$\therefore \{O\}_I = \{I\}_I$

output from input layer — input to input layer.

$(l \times 1)$ $(l \times 1)$

Input to Hidden layer

$\{I\}_H = ?$. Let's take the p^{th} node of hidden layer.

Input to hidden layer

$(m \times 1)$ a vector

$\therefore I_{Hp} = O_{I1} \cdot V_{1p} + O_{I2} \cdot V_{2p} + \cdots + O_{Il} \cdot V_{lp}$

$= \sum_{i=1}^{l} (O_{Ii} \cdot V_{ip})$

$\forall\ p \in [1, m]$

i.e. $p = 1, 2, \ldots, m$.

$\therefore \{I\}_H = [V]^T \cdot \{O\}_I$

$(m \times 1)$ matrix $(m \times l)$ $(l \times 1)$

Output from hidden layer : $O_{Hj} = ?$

Say p^{th} node of hidden layer. Let the transfer function of nodes of hidden layer is Sigmoidal function.

$\therefore O_{Hp} = \frac{1}{1 + \exp[-\alpha(I_{Hp} - \theta_{Hp})]}$

We start from some arbitrary value and tune the parameters

$$\therefore \underset{(m\times 1)}{\{O\}_H} = \left\{ \frac{1}{1+\exp\left[-\alpha(I_{Hj}-\theta_{Hj})\right]} \right\}_{\underset{(m\times 1)}{H}}$$

Input to Output layer

$$\underset{(n\times 1)}{\{I\}_O} = \underset{(n\times m)}{[W]^T} \cdot \underset{(m\times 1)}{\{O\}_H}$$

$[W]$ is $m\times n$
$[W]^T$ is $n\times m$

Let's take the q^{th} node in Output layer.

$$I_{Oq} = O_{H1}\cdot W_{1q} + O_{H2}\cdot W_{2q} + \cdots + O_{Hm}\cdot W_{mq}$$

$$= \sum_{j=1}^{m} O_{Hj}\cdot W_{jq}$$

Output from Output layer

Let the transfer func. of nodes of Output layer be sigmoidal

$$O_{Oq} = \frac{1}{1+\exp\left[-\alpha(I_{Oq}-\theta_{Oq})\right]}$$

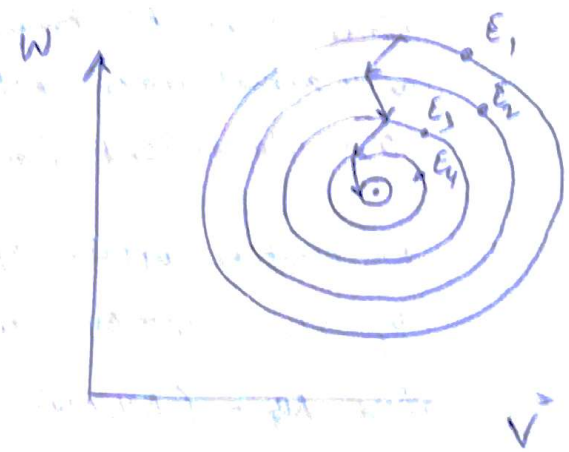
$$\therefore \underset{(n\times 1)}{\{O\}_O} = \left\{ \frac{1}{1+\exp\left[-\alpha(I_{Oj}-\theta_{Oj})\right]} \right\}_{\underset{(n\times 1)}{O}}$$

This is the Forward Calculation Part using the Training Data set for Input

Our ANN has $\boxed{I_{i1} \cdots I_{i2} \mid O_{01} \cdots O_{0n}}$

The gap/diff. b/w the actual output and calculated output value $\{O\}_0$. We calculate the RMSE and then do backward calculation using some algorithm to tune our V, W, etc.

Gradient Descent Method is used, i.e $\frac{\partial E}{\partial W} = 0$, $\frac{\partial E}{\partial V} = 0$

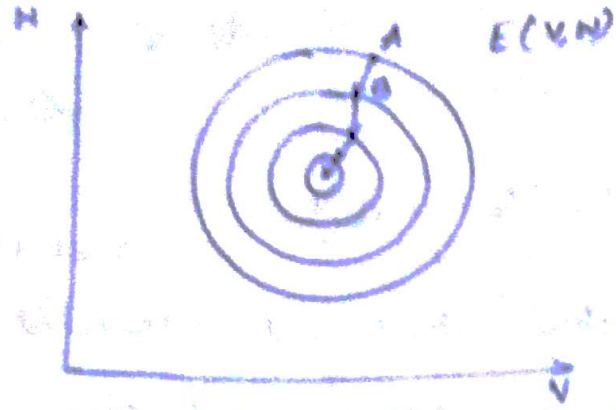


$E_1 > E_2 > E_3 > E_4$

We choose that path in which the gradient is the steepest. This is how the RMSE is reduced.

We try to fit the process data by brute force

2/4/19



We want to go from higher error contours to lower error contours

In vector form:

$\bar{G} = -\frac{\partial E}{\partial V}\cdot\hat{i} + \frac{\partial E}{\partial W}\cdot\hat{j}$ (unit vector)

$\overrightarrow{AB} = -\eta \cdot \bar{G}$ (η = magnitude of step AB)

$= -\eta\left[\frac{\partial E}{\partial V}\hat{i} - \frac{\partial E}{\partial W}\hat{j}\right]$

$E = \frac{1}{2}(T-O)^2$

where T is the training data output or actual process output, and O is the output from our ANN.

$\overrightarrow{AB} = (V_{i+1} - V_i)\hat{i} + (W_{i+1} - W_i)\hat{j}$

$= (\Delta V)\hat{i} + (\Delta W)\hat{j}$

We want to update V and W by $V_{i+1} = V_i + \Delta V$ and $W_{i+1} = W_i + \Delta W$

where $AB = (\Delta V, \Delta W)$

$\therefore \Delta V = -\eta\left(\frac{\partial E}{\partial V}\right)$

$\Delta W = -\eta\left(\frac{\partial E}{\partial W}\right)$.

We take $-\eta \cdot \bar{G}$, the -ve sign is because we want to go to minima, not maxima.

Let's take the k^{th} node of the output layer.

$\therefore E_k = \frac{1}{2}(T_k - O_k)^2$

∵ We are going in backward direction, we'll first update W and the V (i.e. hidden-output first and then input-hidden)

Updating w

$$\frac{\partial E_k}{\partial w_{ik}} = \left(\frac{\partial E_k}{\partial O_{ok}}\right) \cdot \left(\frac{\partial O_{ok}}{\partial I_{ok}}\right) \cdot \left(\frac{\partial I_{ok}}{\partial w_{ik}}\right)$$

$$\frac{\partial E_k}{\partial O_{ok}} = -(T_k - O_{ok})$$

$$\frac{\partial O_{ok}}{\partial I_{ok}} = \frac{1}{\left[1+\exp\left(-\alpha(I_{ok}-O_{ok})\right)\right]^2} \cdot \alpha \cdot \exp\left[-\alpha(I_{ok}-O_{ok})\right]$$

{ from sigmoidal function, $O_{ok} = \frac{1}{1+\exp\left[-\alpha(I_{ok}-O_{ok})\right]}$ }

$$= \alpha \cdot O_{ok} \cdot (1 - O_{ok})$$

$$\frac{\partial I_{ok}}{\partial w_{ik}} = O_{Hi}$$

$$\left\{ I_{ok} = \sum O_{Hj} \cdot W_{jk}^{o} \right\}$$

$$I_{ok} = W_{1k} \cdot O_{H1} + W_{2k} \cdot O_{H2} + \ldots + W_{ik} \cdot O_{Hi} + \ldots + W_{mk} \cdot O_{Hm}$$

$$\therefore \boxed{\frac{\partial E_k}{\partial w_{ik}} = -(T_k - O_{ok}) \cdot \alpha \cdot O_{ok} \cdot (1 - O_{ok}) \cdot O_{Hi}}$$

$$\therefore \Delta W_{ik} = -\eta \frac{\partial E_k}{\partial w_{ik}} = \eta\left[(T_k - O_{ok}) \cdot \alpha \cdot O_{ok}(1 - O_{ok}) \cdot O_{Hi}\right]$$

$$[\Delta W] = \eta \{O\}_H \cdot \langle \alpha \rangle$$

$[\Delta W]$: $(m \times n)$; η : $(m \times 1)$; $\langle \alpha \rangle$: $(1 \times n)$ same $1 \times n$ matrix.

$$\therefore \langle \alpha \rangle = \langle \alpha[(T_k - O_{ok}) O_{ok} \cdot (1 - O_{ok})] \rangle$$

Similarly, $[\Delta V]_{(l \times m)} = \eta \underset{(l \times 1)}{\{I\}} \underset{(1 \times m)}{\langle d^* \rangle}$

$d_i^* = e_i (O_{H_i})(1 - O_{H_i})$ { d_i^* = i^{th} element of $\langle d^* \rangle$ array }

where $e_i = W_{ik} \cdot d_k$ { d_k is the k^{th} element of the $\langle d \rangle$ array in last page }

η = Learning Rate (measure of how fast the ANN is learning)

If η is too large, we may jump the lowest error. It may converge and then diverge. If η is too small, large time required for convergence. ∴ η needs to selected carefully

$\langle d_i^* \rangle$

$\therefore \quad W^{n+1} = W^n + [\Delta W]^n ;$

$V^{n+1} = V^n + [\Delta V]^n ;$

We may sometimes be entrapped in a local minima, like, if we go below the least error (global minima) we can't go upwards as in Gradient descent method, we can only go downwards. So, we need to restart.

In our iterative process of gradient calculation, $x_{n+1} = f(x_n)$; instead of using x_n to find x_{n+1}, we can use a relaxation factor α to improve the time of convergence.

So $x' = \alpha x_{n+1} + (1-\alpha) x_n$

$x_{n+1} = f(x')$

If $\alpha \in [0, 1]$ $\Rightarrow$ Under relaxed.

" $\alpha \in (1, \infty)$ $\Rightarrow$ Over relaxed.

So, in ANN, we can use a momentum factor λ for faster convergence.

$$W^{n+1} = W^n + \lambda [\Delta W]^n ;$$

$$V^{n+1} = V^n + \lambda [\Delta W]^n . \quad [\text{Typically } \lambda \in (0, 1)]$$

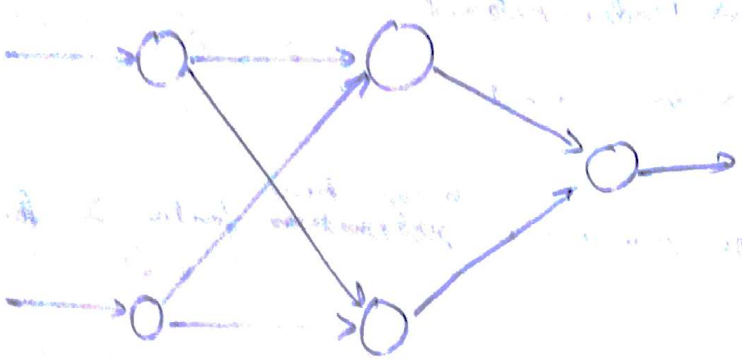
λ tries to capture the trajectory. Value of λ changes with iteration. We play with η and λ and see if convergence

Example

Sr. No.	Inputs		Outputs
	T_1	T_2	O
1	0.4	-0.7	0.1
2	0.3	-0.5	0.05
3	0.6	0.1	0.3
4	0.2	0.4	0.25
5	0.1	-0.2	0.12

No. of hidden layer nodes $\geq$ max. [no. of input layer nodes, no. of output layer nodes.]

Let's take so, we have 2 input node and 01 output node.
So let's start with 1 hidden layer with 2 nodes.



Step 1 : $\{O\}_I = \{I\}_I = \begin{Bmatrix} 0.4 \\ -0.7 \end{Bmatrix}$

Step 2 : Initialize the weights

$$[V]^0 = \begin{bmatrix} 0.1 & 0.4 \\ -0.2 & 0.2 \end{bmatrix}_{2\times 2} \quad ; [W]^0 = \begin{Bmatrix} 0.2 \\ -0.5 \end{Bmatrix}_{2\times 1}$$

Step 3: Find $\{I\}_H = [V]^T \{O\}_I$

$$= \begin{bmatrix} 0.1 & -0.2 \\ 0.4 & 0.2 \end{bmatrix} \begin{Bmatrix} 0.4 \\ -0.7 \end{Bmatrix} = \begin{Bmatrix} 0.18 \\ 0.02 \end{Bmatrix}$$

Step 4 : $\{O_H\} = \begin{Bmatrix} \frac{1}{1+e^{-0.18}} \\ \frac{1}{1+e^{-0.02}} \end{Bmatrix} = \begin{Bmatrix} 0.5448 \\ 0.505 \end{Bmatrix}$

$\alpha = 1$

$$\{I\}_0 = [W]^T \{O_0\} = \langle 0.2 \quad -0.5 \rangle \begin{Bmatrix} 0.5548 \\ 0.505 \end{Bmatrix}$$

$$= -0.14354$$

$$\{O\}_0 = \left(\frac{1}{1+e^{0.14354}} \right) = 0.4642$$

Error $= (T_0 - O_0)^2 = (0.1 - 0.4642)^2 = 0.13264.$

Let's adjust the weights.

First find $d = (T_0 - O_{01})(O_{01})(1 - O_{01})$

$$= (0.1 - 0.4642)(0.4642)(1 - 0.4642)$$

$$= -0.09058$$

$$[Y] = \{O\}_4 \langle d \rangle = \begin{Bmatrix} 0.5548 \\ 0.505 \end{Bmatrix} \langle -0.09058 \rangle$$

$$\left(\frac{\partial E}{\partial W}\right) = \begin{Bmatrix} -0.0493 \\ -0.0457 \end{Bmatrix}$$

$$[\Delta W]^1 = \alpha [\Delta W]^0 + \eta [Y]$$

(assuming $\eta = 0.1$, $\alpha = 1$)

$$= \begin{Bmatrix} -0.02951 \\ -0.02742 \end{Bmatrix}$$

Here η = learning rate

$\alpha = \lambda$ in our case

We are basically correcting of this step from the correction of the previous step.

0th step $[\Delta W]^0 = 0$ as no correction for 0th step.

Step 10: $\{e\} = [w]\{d\} = \begin{Bmatrix} 0.2 \\ -0.5 \end{Bmatrix} (-0.09058)$

$= \begin{Bmatrix} -0.018116 \\ 0.04529 \end{Bmatrix}$

Step 11: $\{d^*\} = \begin{Bmatrix} (-0.018116)(0.5448)(1-0.5448) \\ (0.04529)(0.505)(1-0.505) \end{Bmatrix} = \begin{Bmatrix} -0.00449 \\ 0.01132 \end{Bmatrix}$

Step 12: $[X] = \{O_I\}\{d^*\} = \{I_I\}\{d^*\}$

$= \begin{Bmatrix} 0.4 \\ -0.7 \end{Bmatrix} \langle -0.00449 \quad 0.01132 \rangle$

$= \begin{bmatrix} -0.001796 & 0.004524 \\ 0.003143 & -0.007924 \end{bmatrix}$

Step 13: $[\Delta V]' = \alpha \underbrace{[\Delta V]^0}_{0} + \eta [X]$

$= \begin{bmatrix} -0.001077 & 0.002716 \\ 0.001885 & -0.004754 \end{bmatrix}$

Step 14: $[V]' = \begin{bmatrix} 0.1 & 0.4 \\ -0.2 & 0.2 \end{bmatrix} + \begin{bmatrix} -0.001077 & 0.002716 \\ 0.001885 & -0.004754 \end{bmatrix}$

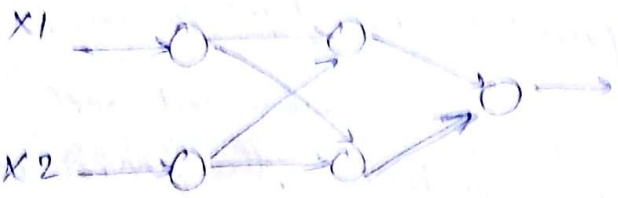
$= \begin{bmatrix} 0.0989 & 0.04027 \\ -0.1981 & 0.19524 \end{bmatrix}$

$[W]' = \begin{Bmatrix} 0.2 \\ -0.5 \end{Bmatrix} + \begin{Bmatrix} -0.02958 \\ -0.02742 \end{Bmatrix} = \begin{Bmatrix} 0.17042 \\ -0.52742 \end{Bmatrix}$

Step 15 : With updated [V] and [W], error is calculated again and next training set is taken and the error will be adjusted.

Example

XOR Gate (Bench mark problem)



Truth table

x_1	x_2	Y
1	1	0
1	0	1
0	1	1
0	0	0

Any neural package or software is loaded with this.

(a) ∵ no. of data is less (only 4),

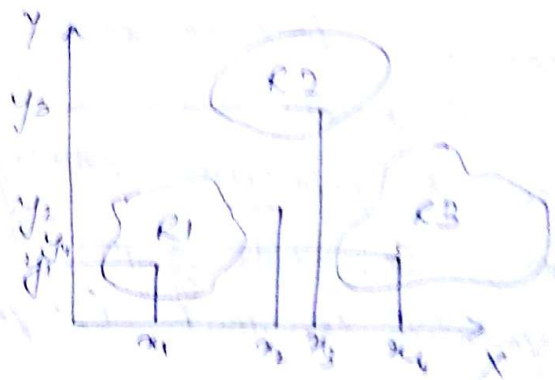
use table 2 or 3 rows of each (total 100),

then randomize, and test the data

Example

Identification of regimes in a DC system

X	Y	Z
x_1	y_1	1
x_2	y_2	0
x_3	y_3	2
x_4	y_4	3



We train the ANN from these types of data such to get the regimes, so that only with x, y data, the ANN can give the Z (0) value.

8/7/19

Genetic Algorithm

Darwin's Theory - "Survival of the fittest".

New breeds/class of living things come into existence through the process of "reproduction", "crossover", "mutation" among existing organisms.

They have to contend by oppose in a fitness test and if they fail, they perish. If they survive, they are becoming newer creature, by changing i.e. they are evolving.

Optimization techniques like Simulation Annealing (SA), Ant Colony Optimization, etc. are called Gradient - Free Optimization.

Also, they don't start with a single guess value but with multiple guesses, and one-by-one eliminate the poorer choices and strengthen the good choices.

All these are Nature inspired.

GA

- We start with a population, i.e. multiple guess values, and we expect 1 to be the optimum value.
- Then each member of population is subjected to a fitness test
- Those who fail are eliminated (out of population) and those survived pass.
- Now, the population strength has reduced (say from 1000 to 200). So, we'll give a breakup (of 200) to make the population constant (reproduction steps).
- The fitness test also creates a merit list, i.e. how well they performed. Say we choose the 1st 10 members from the merit list and copy them 20 times, so total 200. Then w

all these are the personal numbers (DNA). Now, population strength is 1000. When we keep on repeating this, we can find all the members pass the fitness test (i.e. the best values have in the population).

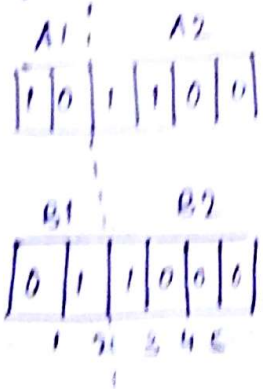
But if we go on doing this, we might find that at the end, the entire population is composed of 1000 copies of the 3rd performer or best performer. The drawback, here, is that if the best performer (given randomly) is not present in the initial population, we'll never get the optimum value by reproduction.

For this we need Crossover and Mutation steps, which makes a new member. We don't know if to be the best or worst

Now, let's go down to the bit-byte mode.

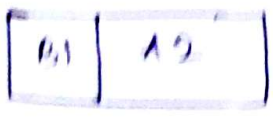
Say, every member of the population is a 6-bit string.

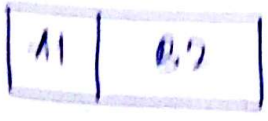
possible
Total numbers = $2^6 = 64$



We pick 2 random numbers. and choose a random location and split the strings. (Random string has (n-1) possible locations)

Upon crossover we get

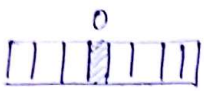




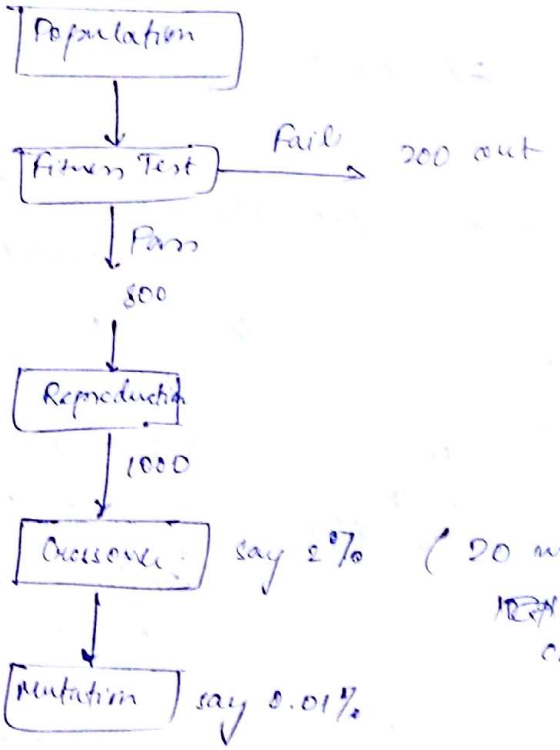
If we have a population of all 64 members then crossover is irrelevant. Generally, all possible members are not present in the actual population. Because of many give rise to a new member which may be poorer

- In mutation step, we randomly choose a member string, choose a random bit in the string, and flip the value, i.e. If 0 ⟶ 1

1 ⟶ 0

Say in our pop.  say all the member have the 4th bit = 0. But optimum value has 1 in the 4th bit. So, mutation is useful here.

We can't do this by cross over as there the location of every bit in a string is maintained.



(20 member randomly selected

The crossover operation is called 10 fineness

How to choose the string length?

The string length decides the accuracy or hard count.

Say our value range for search of optimum value is from 0 to 5. Now, if we use a 4 bit string and an 8-bit string to represent the members of our pop., the accuracy of the optimum value will be more if we use 8-bit strings.

 → 15 possible

$2^L - 1$ - no. of possible numbers

$$\text{Resolution} = \frac{x_{max} - x_{min}}{2^L - 1} = \frac{5}{15} = 0.33 \quad \left(\frac{\text{Range}}{2^L - 1}\right).$$

 → 255 possible

$$\text{Resolution} = \frac{5}{255} \approx 0.02$$

So, when we find the optimum string, the value associated with it will have higher accuracy