Intelligent Systems and Control

Prof. Laxmidhar Behera

Department of Electrical Engineering

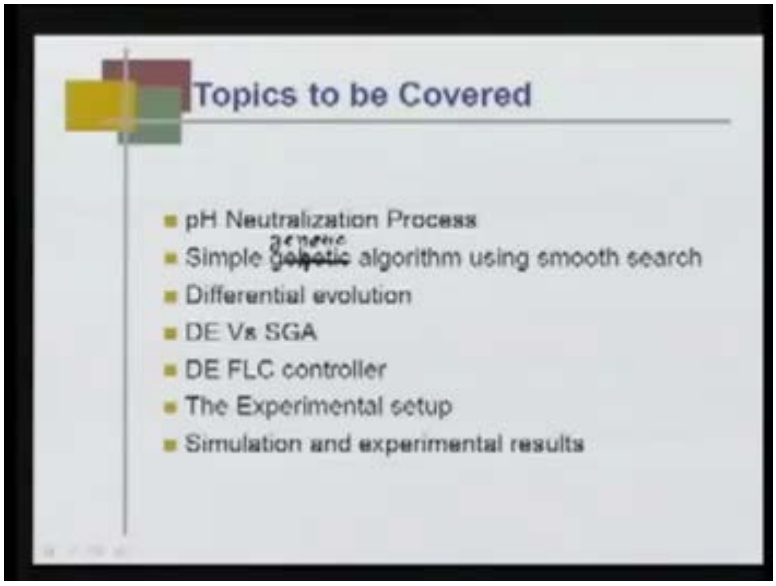
Indian Institute of Technology, Kanpur

Module – 4 Lecture – 3

Fuzzy control of a pH reactor

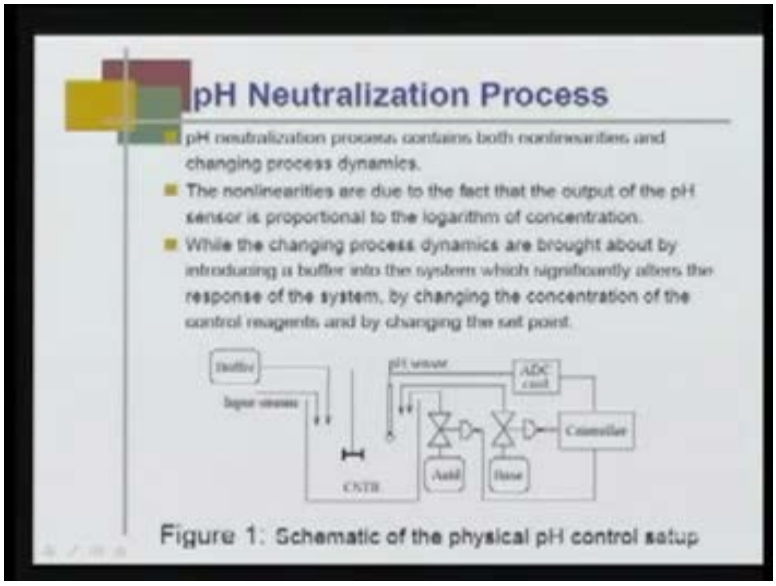
Fuzzy control of a pH reactor, this will be today's topic. In this module 4 on fuzzy control, in the last class we learnt how a Mamdani type of controller can be designed, and also can be optimize using genetic algorithm. Today, we will again consider the same Mamdani type of controller. But we will use a different algorithm for optimization, which is differential evolution as well as the same genetic algorithm. However, the interesting part of today's lecture is that, in this the fuzzy logic controller has been implemented on an actual pH reactor fabricated for a laboratory experiment.

(Refer Slide Time: 1:36)



These are the topics to be covered today: In pH neutralization process, I will describe what is a pH reactor? Simple genetic algorithm using smooth search, differential evolution, differential evolution verses simple genetic algorithm, differential evolution based fuzzy logic controller, the experimental setup, simulation and experimental results.

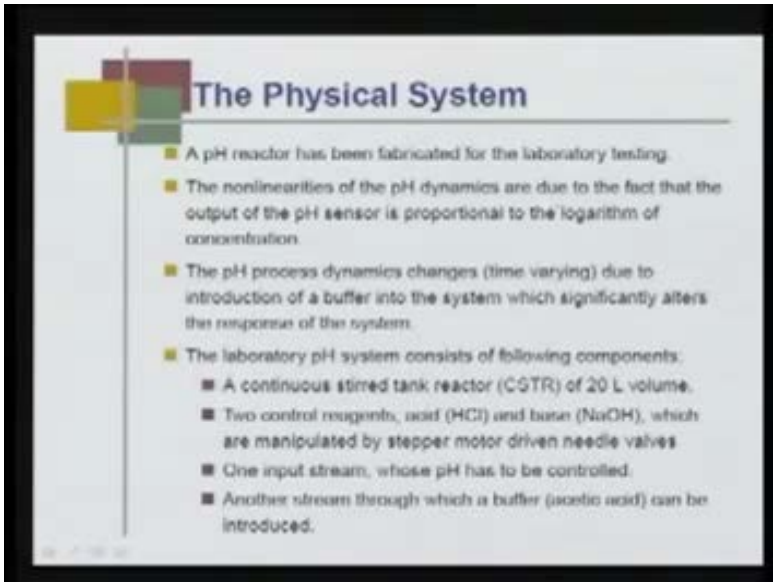
(Refer Slide Time: 2:19)



The pH neutralization process contains both nonlinearities and changing process dynamics. What are these things? The nonlinearities are due to the fact that the output of the pH sensor is proportional to the logarithmic of concentration. Although the dynamical equations as we see are linear, but linear in terms of this states of this pH reactor. However the pH the actual response of this reactor has the relationships with the states in terms of logarithmic. While the changing process dynamics are brought about by introducing a buffer into the system which significantly alters the response of the system, by changing the concentration of the control reagents and by changing the set point. You say that, normally you see what is pH reactor? Many chemical processes they have their waste product that has to be rejected into the environment, since these products are highly acidic, before it is rejected or it is thrown into the environment, the pH, which you know that the pH denotes, that normally the pH of seven means it is a neutral liquid, and more pH means more acidic, less pH means more alkaline. We want to keep the pH in a specific zone such that, the way output or the wastages that are being rejected or that is being released by a chemical plant doesn't do any injury to the environment. So it may seem that this is fairly simple process, but actually pH control of a reactant or reagent whatever you say is not an easy task because of nonlinearity, you see that nonlinearity is that pH is a nonlinear related to the concentration of the liquid, so this is continuous stirred tank reactor, this is your input stream with the flow that is coming from the plant

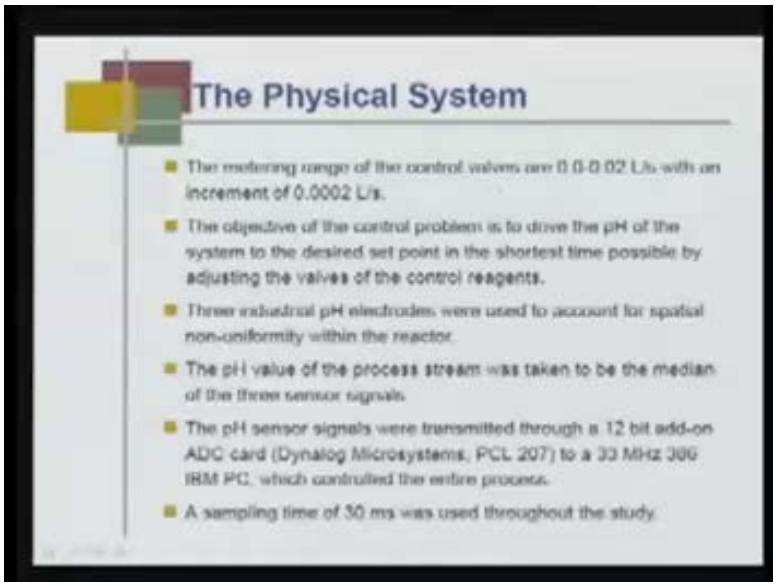
the waste products. This is called buffer (Refer Slide Time: 05:48). I will explain just now what the buffer is, this is the pH sensors and, so this is your controller, the sensors senses the pH of the reactor, and controller receives that sensor through ADC card, and you see that there are two ways you can add control the pH by either adding a acid or adding base you see, and either you add base are you can add the acid. Sometimes also simply have only one agent to be added that is either a acid or base but, here we have multiple input you can sometimes select a acid sometimes you can select base so two input here, and finally the response output is only single output that is the response of the pH of the liquid contained in this reactor. As I such change in process dynamics, what is changing the buffer is something that automatically flows into this system. So what is this buffer, buffer means once buffer is added to this liquid even if you are adding externally some acid or base to neutralize the pH content. Then because buffer is added, so buffer has the capability like it is an acidic buffer it has the ability to observe their acid more and more such that, even if you are adding more and more base into the system the pH does not change or your adding a acid into the system it does not change because, the buffer has the capability to absorb either a acid or the base depending on whether it is a acidic buffer alcholine buffer.

(Refer Slide Time: 7:55)



The pH reactor we have fabricated in the laboratory for the laboratory testing. The nonlinearities of the pH dynamics are due to fact that the output of the pH sensor is proportional to the logarithm of the concentration. The pH process dynamics changes due to the introduction of a buffer into the system which significantly alters the response of the system. That is why this variable this quantity automatically comes into through the input stream, nobody knowingly adds it, it is a quantity that automatically comes. The laboratory pH system that we have fabricated consist of following components, a continuous stirred tank reactor CSTR of 20 litre volume, two control reagents, acid as well as base, acid is (HCl) and base is (NaOH), which are manipulated by steeper motor driven middle valves. One input stream, whose pH has to be controlled. Another stream through which buffer (acetic acid) can be introduced.

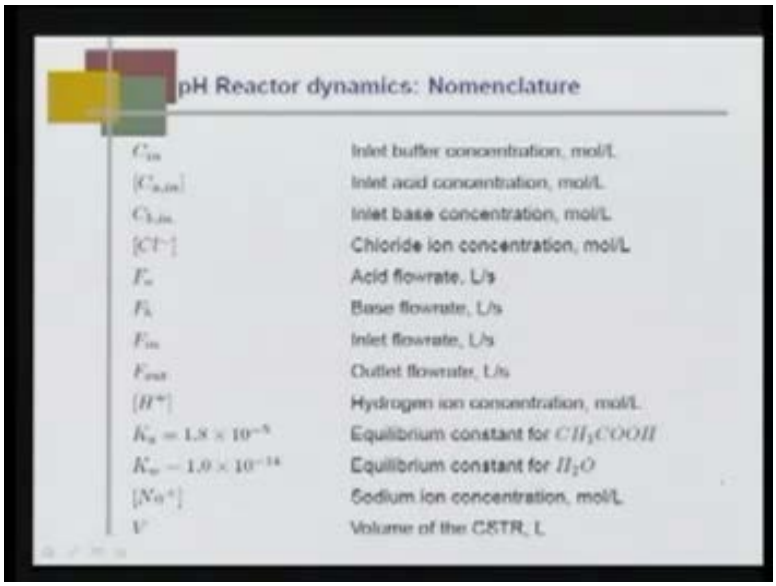
(Refer Slide Time: 9:21)



The physical system, the metering range of control valves are 0 to 0.02 liter per second with an increment of 0.0002 liter per second. This we have fabricated, we have fabricated the valves in such a way that, the input can be varied at any time between 0 to 0.02, and the incremental change in this input can be with the resolution of 0.0002 liter per second. This is the final resolution we have. The objective of the control problem is to drive the pH of the system to the desired set point in the shortest time possible by adjusting the valves of the control reagents. Three industrial pH electrodes were used to account for the

spatial non-uniformity within the reactor. What we have done here is that, you see here I have put only one sensor the schematically shown. But in actual experiment the sensors we have put in three different places in the CSTR. Such that, any kind of spatial non informative can be taken into account by averaging the sensor output of this three sensors placed in this three different location in this CSTR. The pH sensors signals were transmitted through a twelve bit add-on ADC card (Dynalog Microsystems, PCL 207) to the 33 Mega Hertz, 386 IBM PC, which controlled the entire process. A sampling time of 30 milli-seconds was used throughout the study.

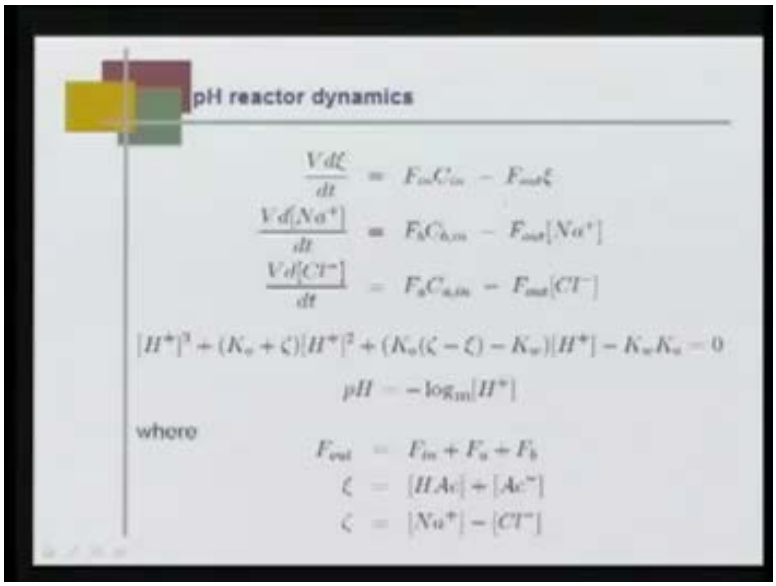
(Refer Slide Time: 11:16)



Here some of the parameters that we have utilized, we have used for describing the dynamics of the pH reactor. This is you see that obviously, you can easily see that concentration C stands for concentration this the inlet. This is inlet buffer concentration. Because you saw that to tank if this is my CSTR, so we have two inputs one is buffer, another is actual input stream, whose pH has to be computed of this. When this is the input stream and we have to control the pH of this input stream before this liquid goes to the environment or is released to the environment. The buffer the concentration is C_{in}. $C_{a,in}$ is inlet acid concentration, $C_{b,in}$ is inlet base concentration. So here in the input stream what is the concentration level of the acid, and concentration level of the base. This is chloride iron concentration Cl minus, F_a is acid flow rate, a stands for an acid, and

F stands for a flow rate. Acid flow rate liter per second, F_b is base flow rate liter per second, F_{in} inlet flow rate this is the acid flow rate base flow rate these are all control variables, F_{in} is input flow rate we have no control over it the input coming from the plant, F_{out} is the output flow rate, and this is the H plus is the hydrogen iron concentration, these are two different parameter K_a and K_w they represent the equilibrium constant for CH_3COOH, and this is the equilibrium constant for H_2O, Na plus is the concentration of the sodium ion, and V is the volume of the CSTR. So this CSTR has specific volume that is V. So given that you can follow any standard book to see how the dynamics…..

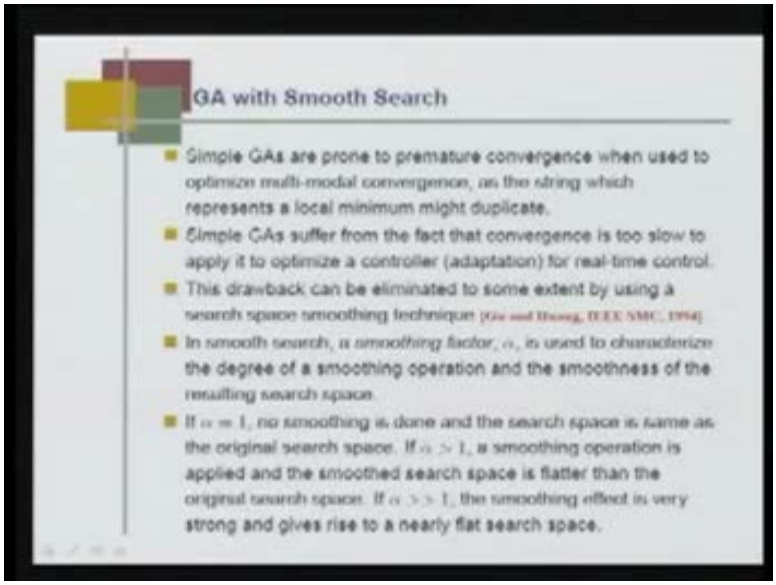
(Refer Slide Time: 14:16)



This is the concentration zeta, which is the concentration of HAc and Ac minus, and so V is the volume so V d zeta by dt is the F_{in} the input flow rate and input concentration minus F_{out} into zeta. Similarly, volume into sodium ion concentration, rate of change in sodium ion concentration is base flow rate into input base concentration in the input stream minus F_{out} output flow rate into sodium ion concentration, and here V…. rate of change in chloride ion concentration is F_a the acid flow rate into the acid concentration in the input stream minus F_{out} is the output flow rate in the chloride ion concentration. So you see that finally the solution of this we solve for the hydrogen plus concentration, hydrogen ion concentration and pH is defined as minus $\log_{10}$ hydrogen ion concentration.

Hydrogen ion concentration is related you see hydrogen ion concentration cube plus K_a plus xi into hydrogen ion concentration to be square into K_a xi minus zeta minus K_w into hydrogen ion concentration minus K_w into K_a equal to zero. If I solve this you see that, this equation is dependent on zeta which is the solution of this equation this zeta and xi, and so you get zeta from here and you know that Na plus you get from here Cl minus. So if I update this equation continuously, I get three variable three states zeta, a sodium ion concentration, and chloride ion concentration. Sodium ion concentration minus chloride ion concentration gives me xi, and zeta gives me the solution zeta. And this equation is a function of zeta and xi the states of this equation. I solve this equation and then, I take put this zeta and xi in this equation, and K_i is the external coefficient, K_w is another external coefficient by putting this I solve for H, which is a nonlinear equation, and then pH is minus logarithmic of H plus to the base ten. What we discuss until now is that, how the dynamics of a pH reactor is actually nonlinear, and also we described that the concentration and pH are nonlinearly related.

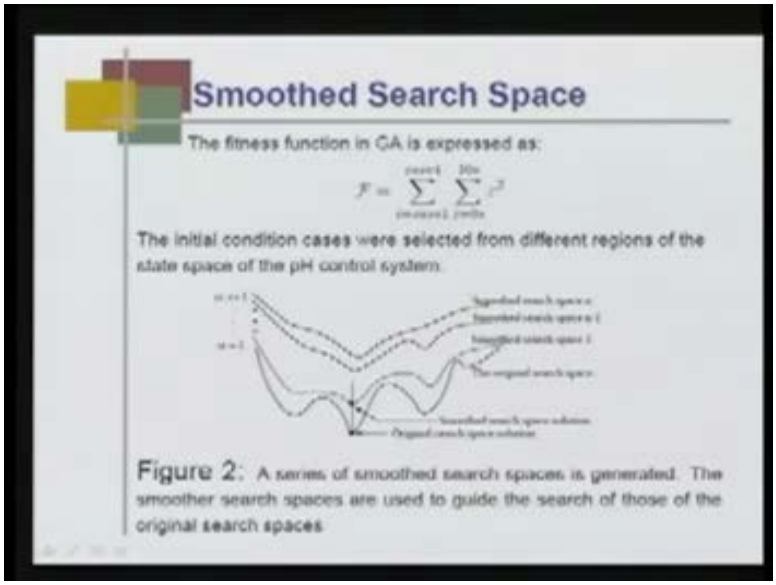
(Refer Slide Time: 18:42)



Now so we will be designing a fuzzy logic controller. We have to optimize the parameters as we said in the last class, we introduce in the last class two algorithm, one is SGA (simple genetic algorithm), another I hope you have remembered what is UMDA (univariate marginal distribution algorithm). There I have shown you that, how a

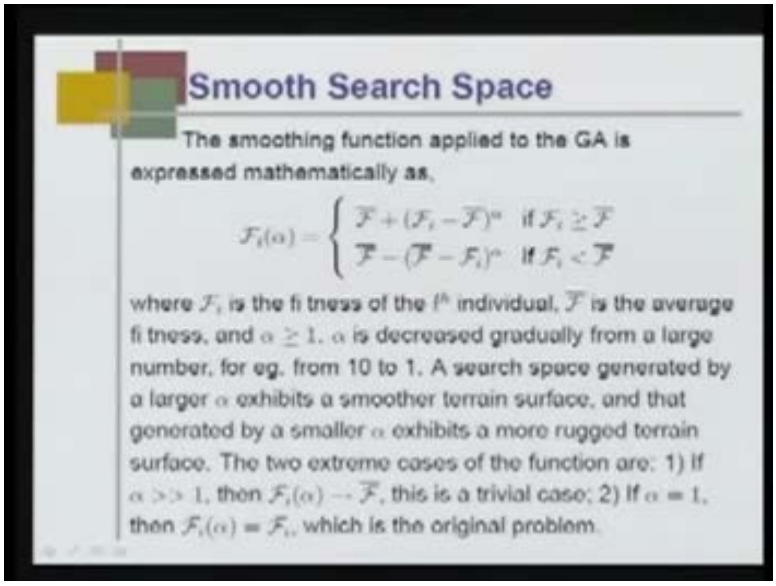
univariate marginal distribution algorithm does very well compared to simple genetic algorithm. So that is why we will be taking about a smooth search today that can have to improve this simple genetic algorithm. Simple GAs is prone to premature convergence when used to optimize multi-model convergence, as the string which represents the local minimum might duplicate. So if the error surface, this is my error surface (Refer Slide Time: 19:44), this is my parameter P, and this is my error cost function or error function. For example I am here, and most of my strings are duplicated in this zone some how, Where as my actual minimum is here or here. Then we are trapped in a local minimum. Simple GAs suffers from the fact that, convergence is too slow to apply it to optimize a controller for real-time control. In real-time we have to optimize it very fast. This draw back can be eliminated to some extent by using a search space smoothing technique, for search space smoothing technique you can follow this paper that appeared IEEE (systems men cybernetic in 1994). In smooth search, a smoothing factor alpha is used to characterize the degree of a smoothing operation, and the smoothness of the resulting search space. If alpha equal to 1 no smoothing is done, and the search space is same as the original search space if alpha is greater than one a smoothing operation is applied, and smoothing search space is flatter than the original search space. If alpha is much much greater than 1 the smoothing effect is very strong and gives rise to nearly flat search space.

(Refer Slide Time: 21:18)



Pictorially you can see this is my actual search space in which I am moving. That is what you are seeing is that, this is my error curve or cost function verses the parameter curve, the parameter for which I am trying to minimize or optimize the cost function. What is meaning of the smoothed search space is that, I am introducing a new parameter called alpha. So when alpha is equal to 1 I get back or I recover my original smoothed search space. But when alpha is greater than one, you see that the search space becoming flattening, and when alpha is very large, you see this almost flatten which has only one minimum. What is objective is that, what we do in this smooth search space technique, we increase the alpha to be very high in the beginning such that, my search is in the flat surface, where I have only one low minimum and then I decrease the alpha slowly such that, finally I recover my original search space. But my solution space will lie only in this zone. This is the understanding of a search space, the fitness function in genetically algorithm is expressed for the control; this is for the controller control. What you are seeing is epsilon square is my tracking error, this is my tracking error from zero second to thirty second, and case one to case four means, this is from different initial conditions. I can always control my pH from various initial conditions. But from any initial condition I start finally, from all initial conditions I should reach the actual pH with a minimal cost function that means, my tracking error should be minimal that is the cost function should be minimal. The initial conditions cases were selected from different regions of the states space of the pH control system. What you are seeing is series of smooth search space is generated, the smoother search space are used to guide the search of those of the original search spaces. Since in the original we have more local minimum as well as the global minimum, and I may be trapped here or here through smooth search space to the smooth search, I am aborting these things and reaching here right.

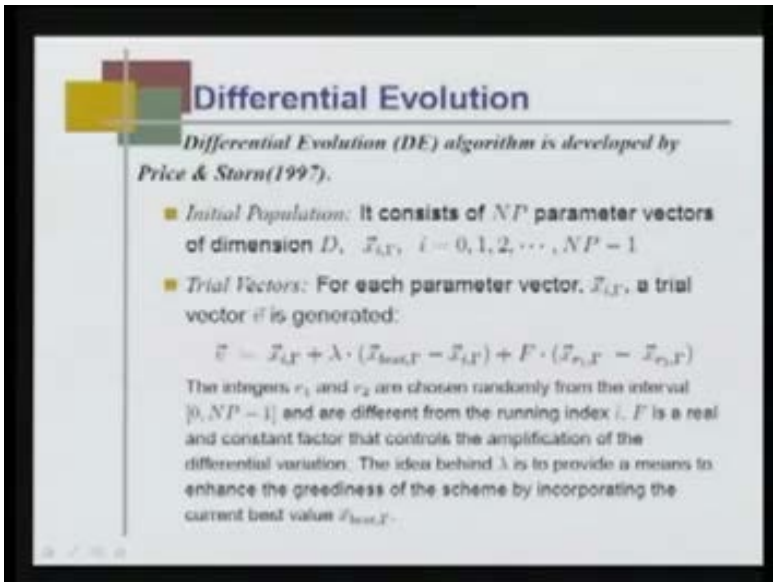
(Refer Slide Time: 24:47)



The smoothing function applied to the genetic algorithm is expressed mathematically, this is my cost function, this is my average cost function, average cost function of the individual cost function, and this is my average cost function. My individual cost function can be written in terms of the average cost function plus the individual cost function minus the average cost function to the power alpha. You see in genetic algorithm we have population consist many solution spaces. For each solution space we have a specific value of fitness function, and the fitness function is given by this, this is my actual F (Refer Slide Time: 23:38). I am representing this fitness function in terms of an alpha, I am introducing a parameter alpha because, and if you see the previous one this is independent this is not a function of alpha. I am introducing an external parameter alpha in such a way you see that there is an average fitness, average fitness is individual fitness of all the solution space and then I average it. That is called average fitness, and this is my actual individual fitness minus average fitness to the power alpha. If I put alpha equal to 1, you can easily see F_i is simply F_i, and here also similarly, alpha equal to 1 if I put here. So then F F cancels out and minus minus is plus F_i. So F_i alpha is F_i, and where as when I have alpha is very very large. I can easily say this quantity because this is normalized values F_i and F bar these are all normalized. F_i minus F bar is the fractional quantity. When alpha is very large obviously this two vanishes. F_i alpha becomes F bar. So it's a flat surface. So where F_i is the fitness of the i th individual F bar is the average

fitness, and alpha greater than equal to 1. Alpha is decreased gradually from a large number from ten to one. A search space generated by a larger alpha exhibition smoother term in surface, and that generate by smaller alpha exhibition more rugged terrain surface. The two extreme cases of functions are, if alpha is much greater than 1 then F_i alpha is F bar this is the trivial case, and if alpha is greater than 1 then F_i alpha is F_i which is original problem. This is what we wanted actually. We take a very large value in alpha in our experiment, we have taken alpha in the beginning around 9 and you reduced until alpha equal to 1.

(Refer Slide Time: 28:19)

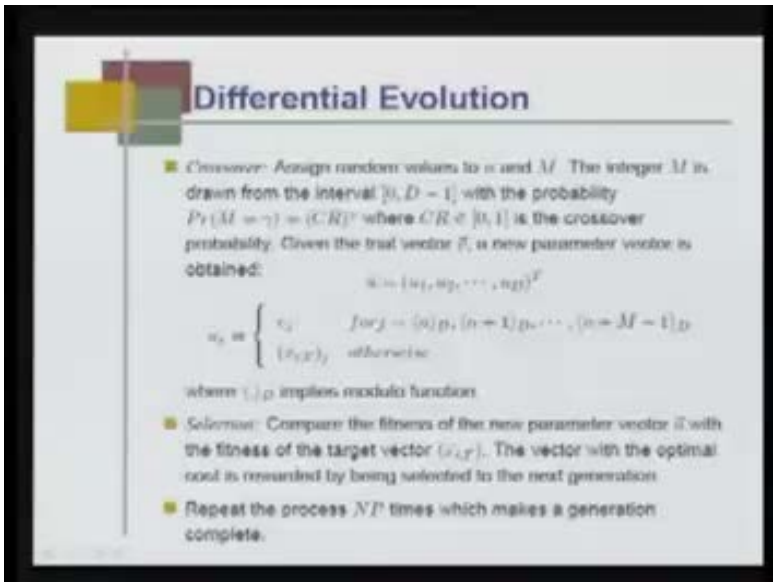


That is what we talked about that this concept of smooth search space idea can be augmented with genetic algorithm to make the genetic algorithm very fast. That is simple genetic algorithm augmented with search space smooth search space. And now here I will present, earlier we have discuss a simple genetic algorithm, invariant marginal distribution algorithm, today I am introducing you again another new kind of optimization algorithm, which is also called evolutionary computation algorithm. Which is here it is differential evolution; the differential evolution algorithm is developed by Price and Storn in 1997. The idea is that we have initial population as usual earlier it consist of NP parameter vectors of dimension D. So obviously, I am optimizing D parameter vectors, and my population size is NP. You see that each parameter vector is

represented by $x_{i,gamma}$, where I equal to zero to NP minus 1, this is my parameter vector, and its dimension is D. Unlike in genetic algorithm, what we do given initial population we make a selection and according to fitness, and then we create a new population by cross over and mutation. But here this variation and new population is generated using trial vectors you see this is the trial vectors. What is the trial vector, for each parameter vector, in the initial population $x_{i,gamma}$, a trial vector v is generated for each $x_{i,gamma}$, and what is the v vector the trial vector, is the original parameter vector plus lambda. This is the best parameter vector means the parameter vector having optimized fitness and or highest fitness minus the present the original parameter vector for which we are creating a trial vector or we can see this is the target vector. Target parameter vector for which we are creating the trial vector plus F into x_{r1} some arbitrarily randomly selected another parameter vector from the population and r_2 represents another parameter vector from the population. So r_1 and r_2 they represent two randomly selected parameter vector for the population. It is like this is my zone, in this zone in this space, this is my r_1 vector, this is my r_2 vector. What I am trying to do is that, this is my say reference point is at r_1. I am subtracting this vector r_2 vector from r_1 and multiplying this with a real constant F, and then I am adding this difference with the original target vector to create a trial vector. And this second term that is being added, the main objective of trial vector is that I select two random parameter vectors from the initial population from the population, and then magnify, I multiply that difference with a real constant F and add that to the actual target vector to create a trial vector. The second term is simply is the idea behind like lambda is to provide a means to enhance the greediness of the scheme by incorporating the current best value which is expressed the parameter vector that has the maximum fitness. This is just an additional thing, but usually the parameter vector is the target vector plus difference between two random vectors, randomly selected parameter vectors from the population multiplied by a real constant. The integer r_1, r_2 are chosen randomly from the interval 0 to NP minus 1, and are different from the running index I, it can not be the same index I, F is a real and constant factor that controls the amplification of the differential variation. The idea behind lambda is to provide a means to enhance the greediness of the scheme by incorporating the current best value. The current best value is represented by the parameter vector in this population. Once I create a trial vector, you

see that if my population size in NP, then my trial vector size is also NP. Then what I do, I do a crossover.

(Refer Slide Time: 34:09)

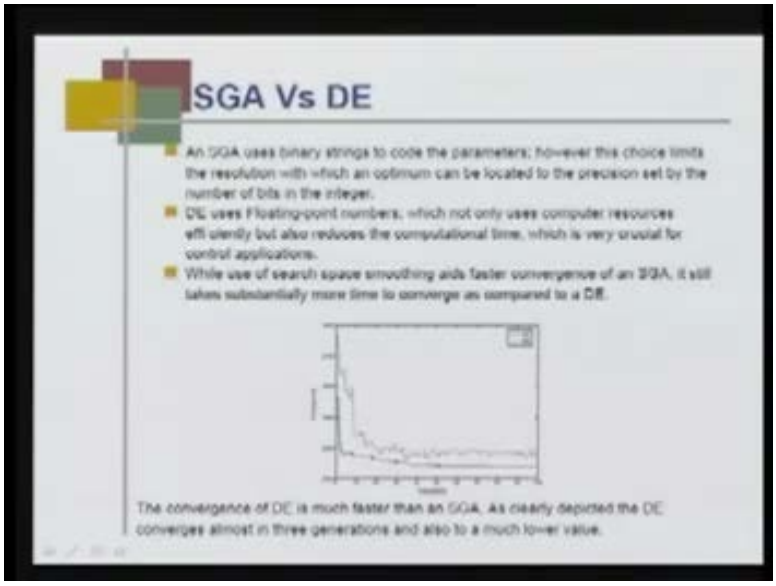


Once I create this trial vector, I perform crossover on this trial vector in this manner, what I do is that, I create define two parameters n and M, and they are random integers n and M they are random integers, n can be any integer but the integer M is drawn from the interval from 0 to D minus 1. Because each parameter vector in the solution space has consist of D parameters, because the actual we are considering a parametric space of size D. So integer M is drawn from this with the probability M is some value, this is you see the small gamma which is lies between 0 to D minus 1 which is CR this is crossover probability, CR is known as crossover probability to the power gamma. The normal CR is less than 1. If you take 0.9 to the power gamma, higher value of M means less probability CR is the crossover probability. What I am trying to do is that given the trial vector a new parameter u vector is obtained, where my the crossover output of u_1,u_2 until u_D because, I have D parameter vector the parameter vector consist of D elements, and the j th element is v_j means, I retained the trial vector element, if j is equal to n modulus D, n plus 1 modulus D until n plus M minus 1 modulus D. What is this modulus you remember that, this is the quantity I divide D, whatever the modulus means the modular function means the remainder is 0. This close bracket D implies the modular function. If j equal to

modular D means, the remainder if I divide n by D, I must get j. If that happens then I retain v_j otherwise, I take the element from my target vector the j th element of the target vector. What we are finding with this is for $x_{i,gamma}$ I created a trial vector v, then I am doing a crossover this target vector and the trial vector to find the actual new parameter vector in the population.

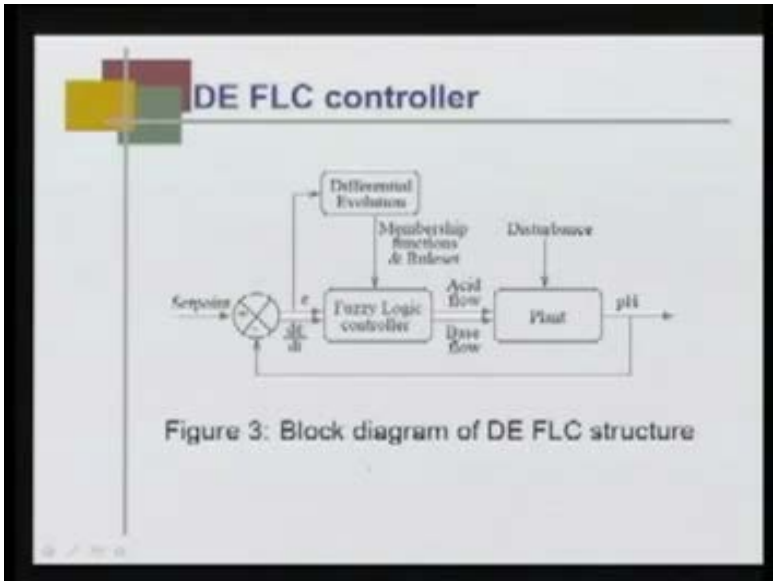
What I do is that, I for that I select two parameter n and M in such a way that, I retain the elements of v as long as the modular functions, any of this modular function is equal to j otherwise, I take the element from x into this. You see that the bigger is M less likely I will select and this is the objective. Objective is that is a very probabilistic way of mixing the elements of v the trial vector and the target vector in such a way, to create a new solution vector. New solution vector is obtained by the crossover between the original target vector and the trial vector. Once the crossover is over, the selection is comparing the fitness of the new parameter vector which is u bar. What we found out that u bar is my target vector, and I can put an i for indicating target vector crossover CR, I am getting a new parameter vector. The vector with the optimal cost is rewarded by being selected to the next generation. Unlike the selection that we did for genetic algorithm or for univariate marginal distribution, we saw that here the selection is not the comparison among all the parameter vectors, but the comparison between the target vector parameter vector from the population and corresponding the new vector generated u from the making the crossover between the target vector and corresponding trial vector. Now, we have to find out the fitness of x_i and the new parameter vector u. So u fitness is better than the target vector, then u becomes new parameter vector because the parameter in the new population. Among this two, one of them will be selected to the new population based on the fitness. Only two, only target vector and the corresponding new parameter vector their fitness is compared. Repeat the process and NP times make a generation complete, see if I find like that NP new parameter vectors then I get a new population, and again I start from beginning for each trial vector, parameter vector in new population again I find a new trial vector. Now we will have a little discussion on the simple genetic algorithm and the differential evolution.

(Refer Slide Time: 41:09)



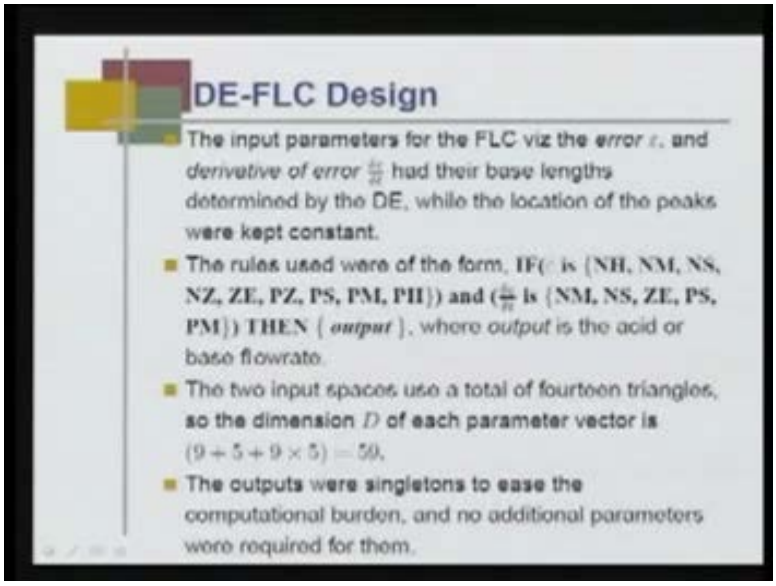
I wish that you are clear about what is differential evolution. Simple genetic algorithm uses binary strings to code the parameter vectors we talked about that, and however the choice limits the resolutions with which an optimum can be located to the precision set by the number of bits in the integer. In contrast differential evolution uses floating-point numbers, which not only uses computer resources efficiently but also reduces the computational time, which is very crucial for control application. While use of search space smoothing aids faster convergence of a simple genetic algorithm, it still takes substantially more time to converge as compared to differential evolution. Here we can see that this is differential evolution very fast convergence. Where as this one is simple genetic algorithm this dotted line and this is pretty slow convergence. The convergence of differential evolution is much faster than simple genetic algorithm as clearly affected the differential evolution convergence almost in three generation in this particular example, and also to a much lower value. Where as simple genetic algorithm takes almost to twenty generation and so.

(Refer Slide Time: 42:44)



Given that, now we will be talking about the FLC controller architecture, this is my fuzzy logic controller, this is my plant pH reactor that the disturbance is actually buffer output of the plant is the pH, and this is my set point pH normally the pH is set as seven, and here the error and derivative of the error. These are the two input variable to my fuzzy logic controller, and I utilize differential evolution to optimize the parameter of fuzzy logic controller. The output of fuzzy logic controller is either acid flow rate or base flow rate to the plant.

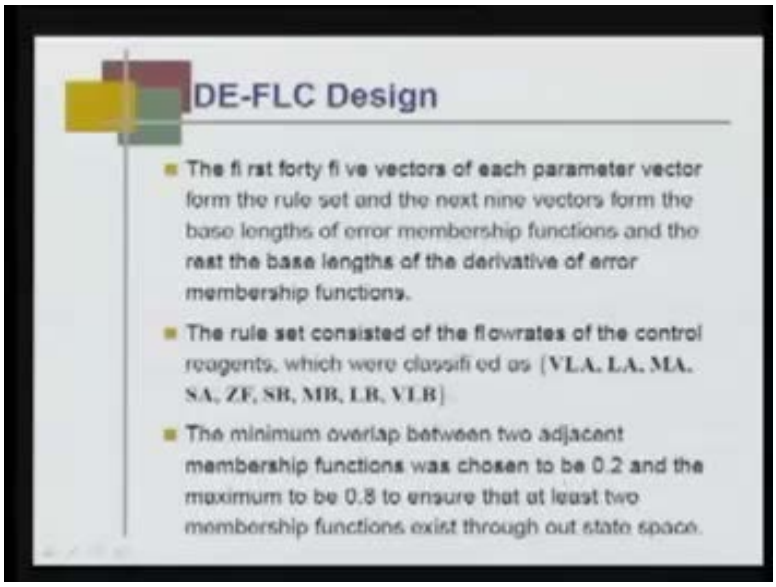
(Refer Slide Time: 43:26)



DE-FLC design the input parameter for FLC the error, and derivative of the error had had their base lengths determine by the differential evolution, for the location of the peaks were kept constant. What is meaning of that this is my epsilon. What I do given any fuzzy membership function, I fix the peak for that linguistic variable, where the base what is the distance between this point to this point this is optimized using, what is the base length is optimized using the differential evolution. The rules used were of the form if error is either negative high, negative medium, negative small, negative zero, zero, positive zero, positive small, positive medium, positive high, and the derivative of the error is negative medium, negative small, zero, positive small, positive medium, then output where output is the acid or base flow rate. You see that the rule base size has to be 1,2,3,4,5,6,7,8 and 9 into 1,2,3,4 and 5. So 45 rules, we have in this particular fuzzy logic controller 45 rules, and we have assumed our output is simply singleton. So you see the outputs are singletons to use the computational burden, and no additional parameters were required for them. And for each fuzzy partition you have two variables, and for fuzzy partition of each variable I need only, say for example my error is fuzzy partition to nine, fuzzy linguistics variable and for differential evolution has to find only nine parameter corresponding to that, so that is the base length. This length is determined by the differential evolution. Total number of parameters that I need is forty five parameters for the output, and nine parameters, nine base lengths for the fuzzy variable error, and

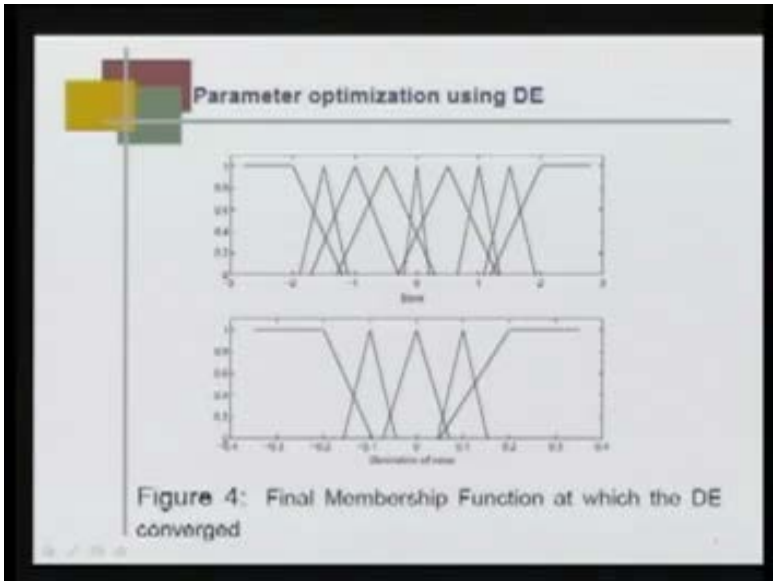
five base lengths for the fuzzy input variable the change in rate derivative of the error. So the total is nine plus five into 45 is 59 parameters that we have to optimize. The outputs were singletons that I have already told.

(Refer Slide Time: 46:24)



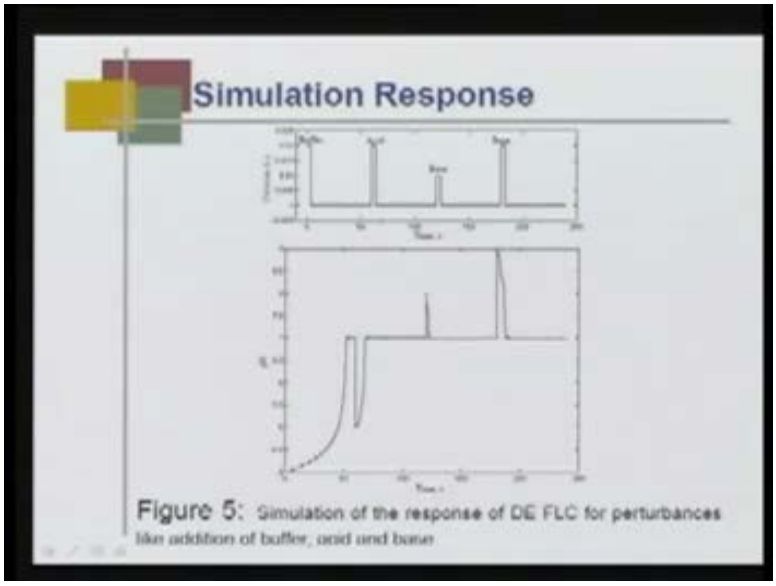
The first 45 vectors of each parameter form the rule set and the next nine vectors of the base lengths of error membership function and the rest the base lengths of the derivative of error membership functions. So that's how we get 59. The rule set consisted of the flow rates of the control reagents, which were classified as, that is the output is classified as, very large acid flow rate, large acid flow rate, medium acid flow rate, small acid flow rate, zero flow rate, small base flow rate, medium base flow rate, large base flow rate, and very large base flow rate. This is how the output is fuzzy. The minimum overlap between two adjacent membership functions was chosen to be 0.2 and the maximum to be 0.8 to ensure that at least two membership functions exists through out the state space.

(Refer Slide Time: 47:33)



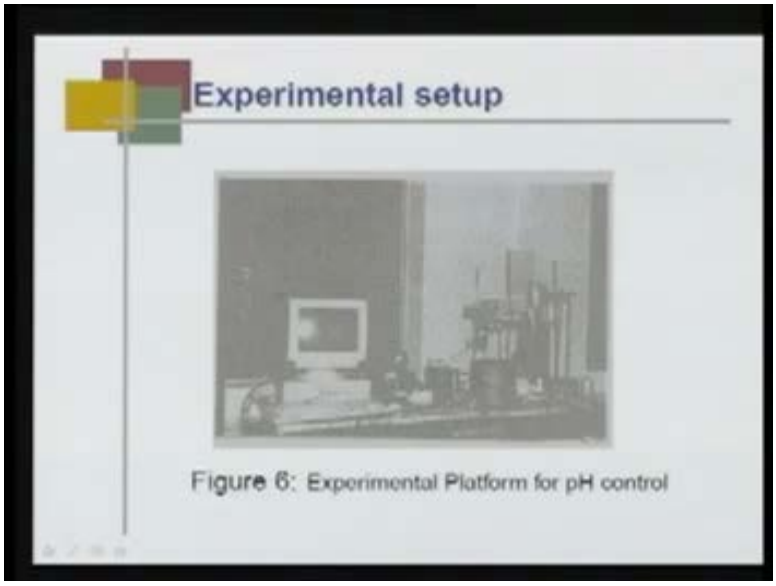
Doing this, when we did the optimization you see that, I have a variable base length the base length were desired by differential evolution, my error fuzzy partition into nine linguistic variables 1,2,3,4,5,6,7,8 and 9 the peaks are fixed. Only base you see this is the base that is obtained from the differential evolution like for this particular one, so from here this is the base length that is obtained, and this particular value was actually the peak, where is the peak that has fixed prior. Similarly, the derivative of the error also the base lengths were obtained in this particular manner, the final membership function at which the differential evolutions converged. Now simulation response we first did the simulation for the theoretical model that we obtained.

(Refer Slide Time: 48:41)



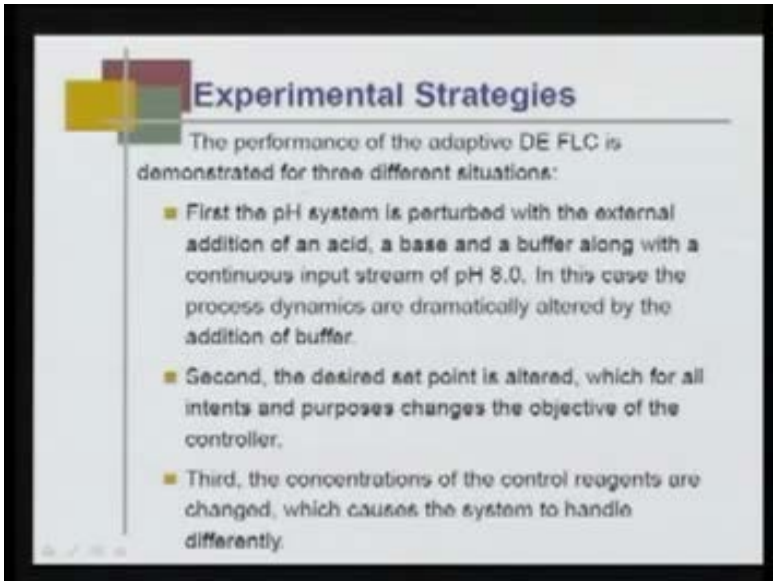
Here what you are seeing the flow rates, the buffer, acid, base, and base. That is how the input stream and here you see that what we are trying to do is that the pH is controlled at its neutral value 7. So you can see this is, the value is 7 and you see that in the beginning the actual input stream has a pH of 4, there goes to increase 7 and you see that because acid is added here, (some deep…, mistake the figure should have been well expanded,…this is not to be scale) and then to actually this dip is due to this particular effect. And this effect is due to the base, and again this effect, this effect some hike is due to the base. Simulation of the response of DE FLC for perturbances like addition of buffer acid base, in the beginning when I add buffer it takes long time to jump to the actual pH 7, and then acid pH goes down, base pH goes up and addition of base this is the disturbance added. But always finally the pH is maintained at 7.

(Refer Slide Time: 50:26)



This is our experimental setup you see that we control using a pc, this is our reactor which is fabricated in the lab, and the entire controller was implemented using pc experimental strategies.

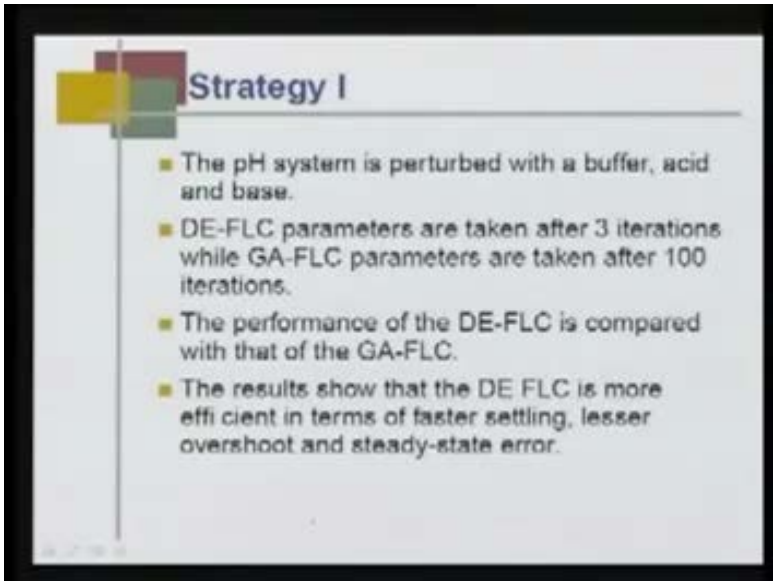
(Refer Slide Time: 50:47)



The performance of the adaptive differential evolution fuzzy logic controller is demonstrated for three different situations; first the pH system is perturbed with the

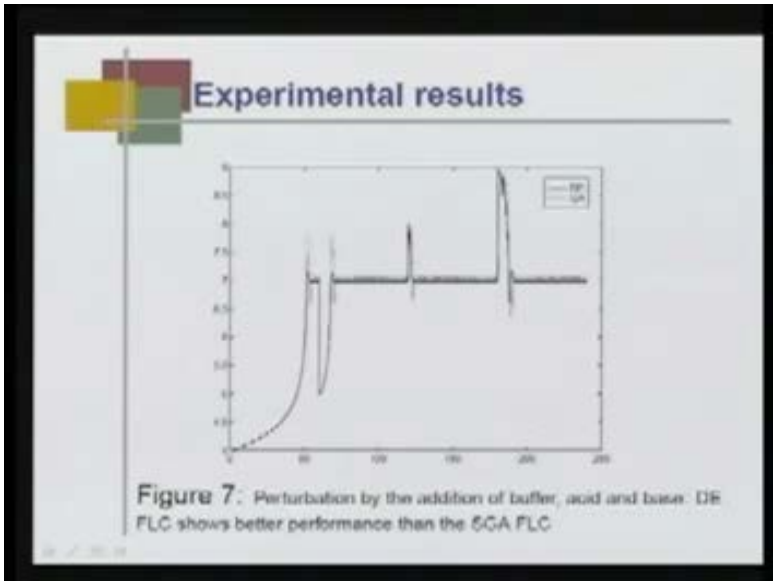
external addition of an acid, a base and a buffer along with a continuous input stream of pH 8. In this case the process dynamics are dramatically altered by the addition of the buffer. Second, the desired set point is altered, which for all intents and purposes changes the objective of the controller. Third, the concentration of the control reagents are changed, which causes the system to handle differently.

(Refer Slide Time: 51:25)



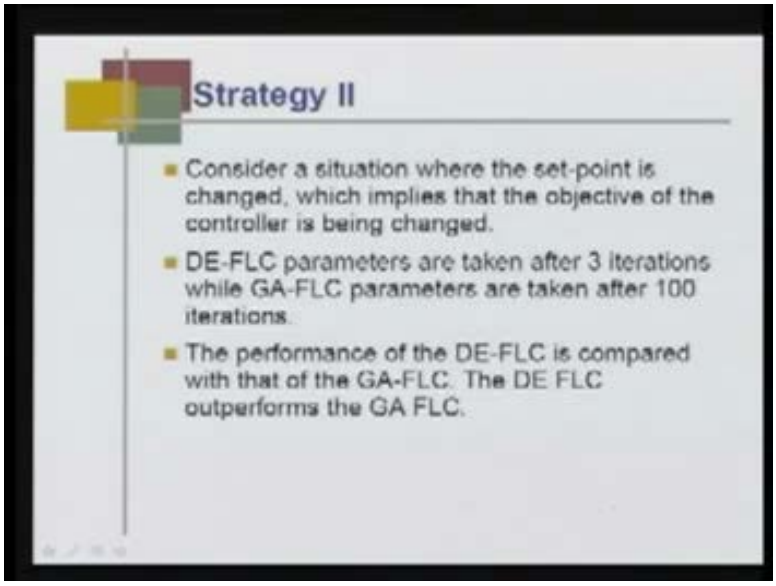
Strategy 1, the pH system is perturbed with a buffer, acid and base. DE-FLC parameters are taken after 3 iterations, and the performance is compared with the GA-FLC base controller whose parameters are taken after 100 iterations. So usually, you see that while optimizing I am going only for three generations for DE-FLC, while hundred generations for GA-FLC. The performance of DE-FLC is compared with that of the GA-FLC. The results show that the DE-FLC is more efficient in terms of faster settling, lesser overshoot and steady-state error.

(Refer Slide Time: 52:10)



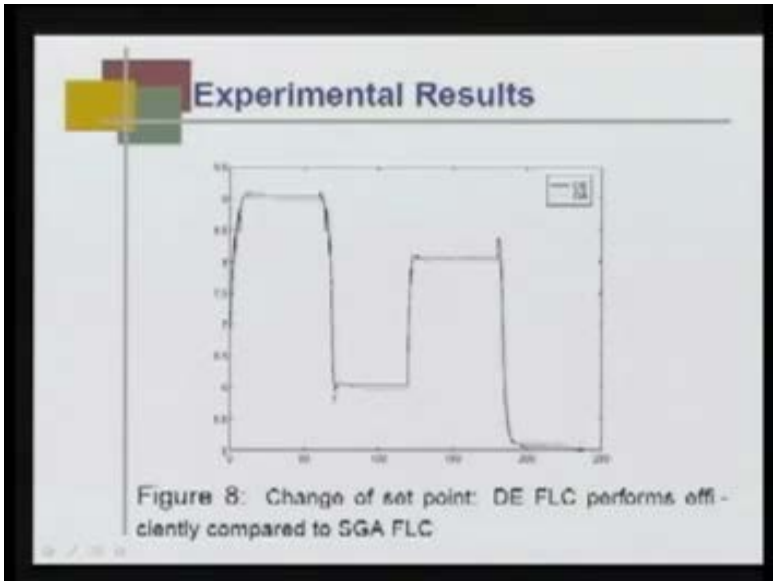
You see here that, here the acid is added, here buffer, here base, and here again base. You see that the dotted one is genetic algorithm, where as D differential evolution is the solid line. You see the genetic algorithm has base control has more overshoot, here and here also more overshoot, here is more undershoot, here is more undershoot, and also there is some error steady state error in case of GA, where in case of differential evolution. We get constant pH this is the experimental result, so perturbation by the addition of buffer, acid and base. DE FLC shows better performance than the SGA FLC.

(Refer Slide Time: 53:17)



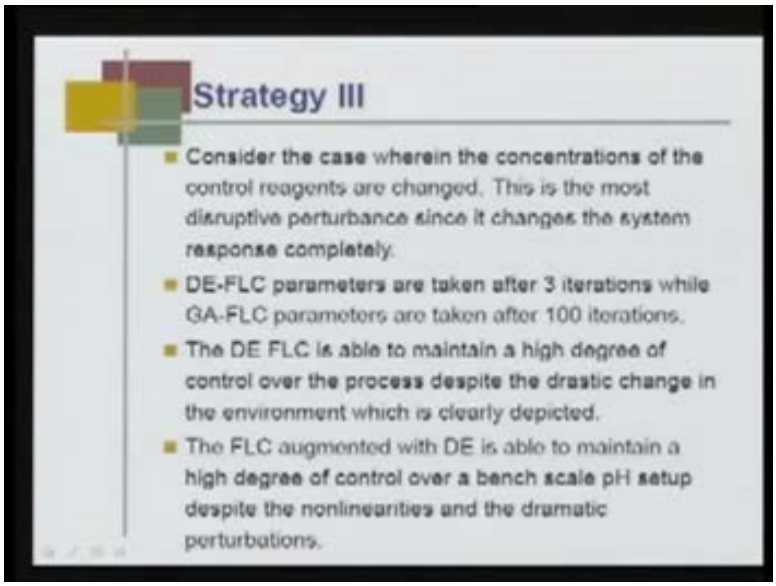
Now strategy 2, consider a situation where the set point is changed, which implies that the objective of the controller is being changed. DE-FLC parameters are taken after 3 iterations while GA-FLC parameters are taken after 100 iterations. The performance of the DE-FLC is compared with that of the GA-FLC. The DE FLC outperforms the GA FLC.

(Refer Slide Time: 53:39)



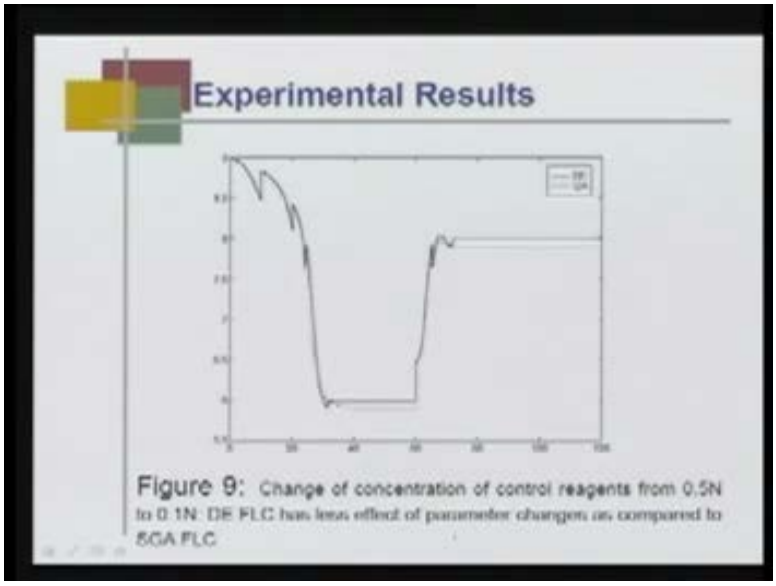
You can see here now, set point earlier here was nine, and then it was changed to six to eight and then five. So you see that the comparison between GA and DE. DE is the straight line a solid line, and GA is the dotted line is the experimental result. In this case in this zone GA is little better than DE. But you have to see that GA parameters are optimizing hundred generation, where d is already three and here if you look at 6 DE differential evolution is much better than the GA, and when it is 8 also the DE is better than GA, and when it is five again DE is better than GA. Here DE FLC performs efficiently compared to SGA FLC.

(Refer Slide Time: 54:55)



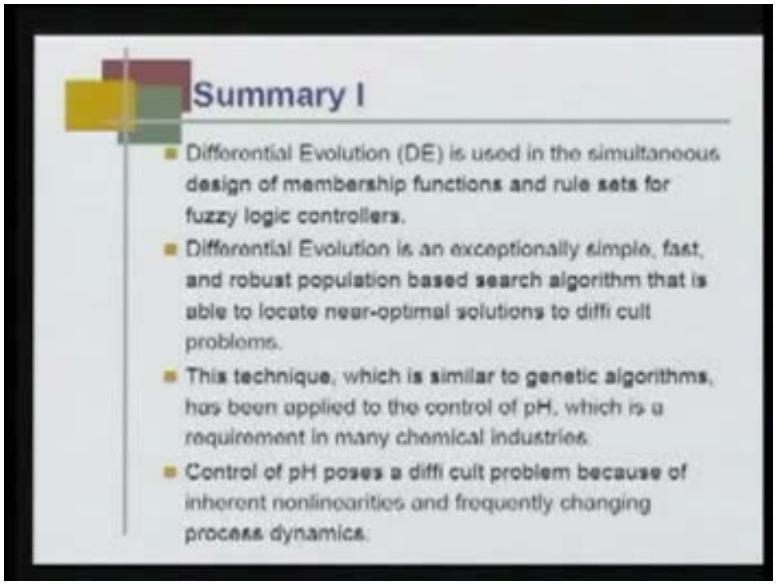
Strategy 3, consider the case wherein the concentration of the control reagents are changed. This is the most disruptive perturbance since it changes the system response completely. DE-FLC parameters are taken after 3 iterations while GA-FLC parameters are taken after 100 iterations. The DE-FLC is able to maintain a high degree of control over the process despite the drastic change in the environment which is clearly depicted. The FLC augmented with DE is able to maintain a high degree of control over a bench scale pH setup despite the nonlinearities and the dramatic perturbations. You see here how the response is, from nine to six again from six to eight.

(Refer Slide Time: 55:44)



You see that, change of concentration of control reagent from 0.5N to 0.1N. DE FLC has less effect of parameter changes as compared to SGA FLC. So SGA certainly gives some steady state error, here also some steady state error. These are all experimental results.

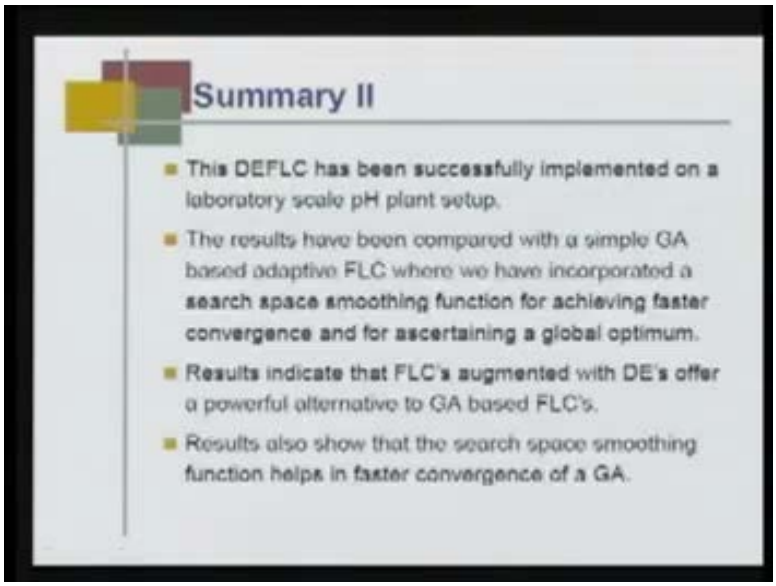
(Refer Slide Time: 56:12)



In summary you would say differential evolution is used in the simultaneous design of membership functions and rule sets for fuzzy logic controllers. In the last class we

showed optimizing using simple genetic algorithm, univariate marginal distribution algorithm. Differential Evolution is an exceptionally simple, fast, and robust population based search algorithm that is able to locate near-optimal solutions to difficult problems. This technique, which is similar to genetic algorithms, has been applied to the control of pH, which is a requirement in many chemical industries and not only had we done it for simulation, in simulation also for experiment. Control of pH poses a difficult problem because of inherent nonlinearities and frequently changing process dynamics.

(Refer Slide Time: 56:58)



The DEFLC has been successfully implemented on a laboratory scale pH plant setup. The results have been compared with simple GA based adaptive FLC where we have incorporated a search space smoothing function for achieving faster convergence and for ascertaining a global optimum. Today the d is compared with the SGA. But SGA has been augmented with smooth surface still DE better than SGA. Results indicate that FLC's augmented with DE's offer a powerful alternative to GA based FLC's. Results also show that the search space smoothing function helps in faster convergence of a GA. That is in this lecture series intelligent control you will strict to this two lectures on Mamdani type of fuzzy logic controller, and their optimization. From the next lecture onwards we will be concentrating on its fuzzy model before we go to ts fuzzy model I will give a trajecent and in tranjecent that is the next lecture. In the fourth lecture I would

show you that how intelligently; we can actually generate the rule base for fuzzy Mamdani type of controller. Such that the control system as a whole is stable this is very interesting.

Thank you very much.