Intelligent Systems and Control

Prof. Laxmidhar Behera

Department of Electrical Engineering

Indian Institute of Technology, Kanpur

Module – 4 Lecture – 2

Mamdani type FLC and parameter optimization

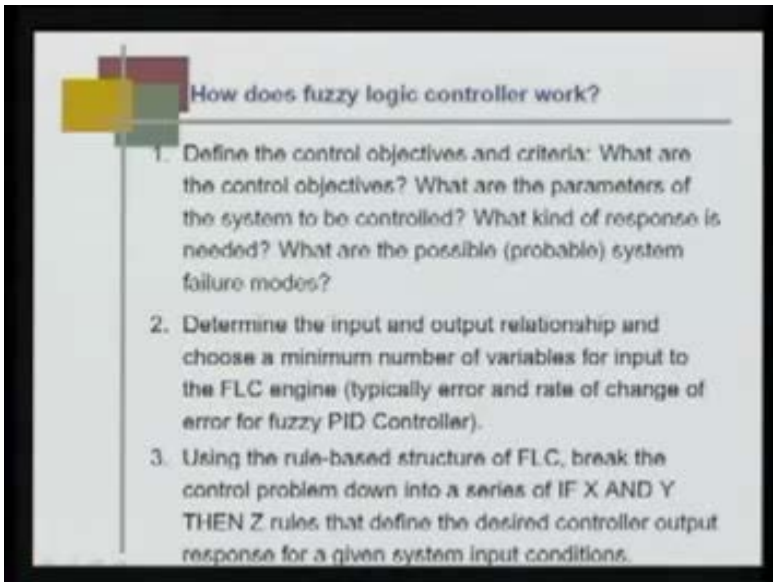
We are on the last module of this course, intelligent control module four. The broad topics is fuzzy control, in the first class we described this. Then, we took a loop on the various fuzzy controls schemes that are prevailing in the literature. Today, we will now be discussing all those schemes in depth. Today, we will be discussing on Mamdani type fuzzy logic controller and parameter optimization.

(Refer Slide Time: 01:06)

Basic architecture of a Mamdani type fuzzy logic controller, we have already discussed in the previous, in the second model on fuzzy logic about this we will revisit that. The issue always with Mamdani type is that, how do you optimize the parameters? Very briefly we will discuss genetic algorithm, fuzzy logic controller for a single link manipulator. Now, univariate Marginal Distribution Algorithm, this is another genetic algorithm (01:41) evolutionary computation approach for optimization. We will compare both this genetic

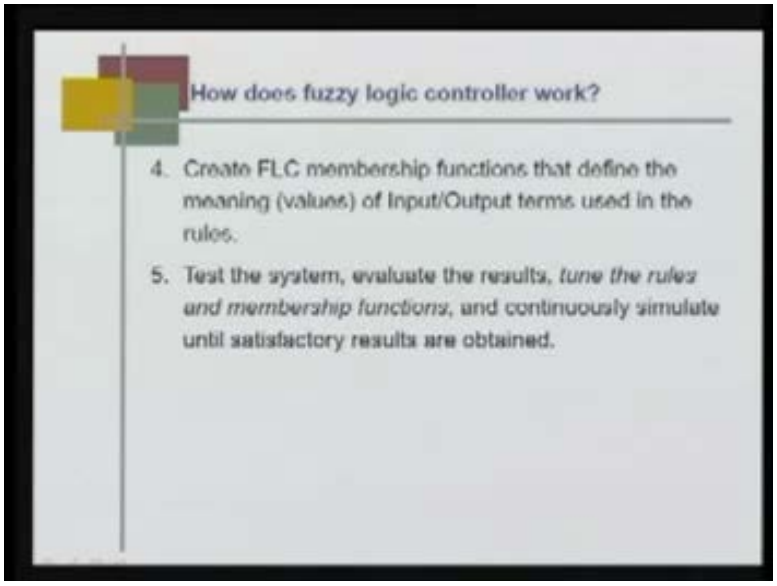
algorithms, simple genetic algorithm and univariate marginal distribution algorithm in a generic example of a robot arm control. So how does fuzzy logic controller work?

(Refer Slide Time: 02:00)



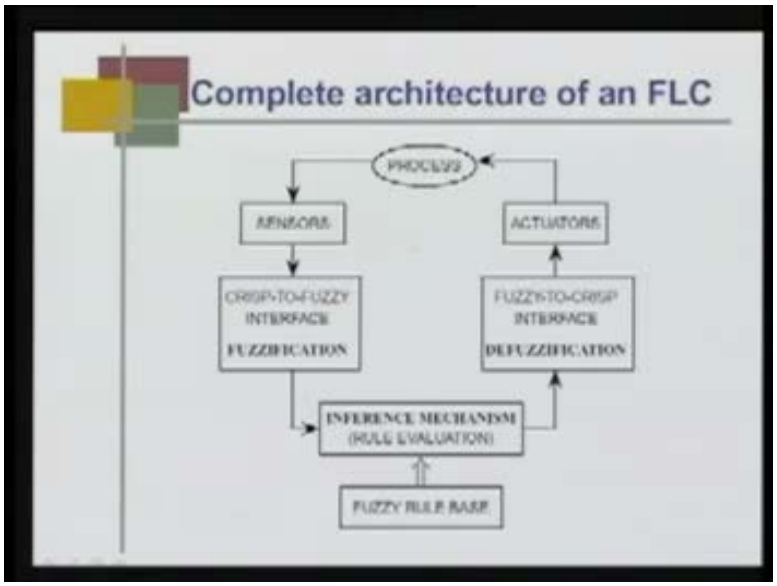
Define the control objectives and criteria. Given a system; we have to say what the control objectives are, what parameters of the system to be controlled, what kind of response is needed, what the possible system failure modes are. Now, for this system, determine the input and output relationship and choose a minimum number of variables for input to the fuzzy logic controller engine, typically error in rate of change of error for fuzzy PID controller. I hope that you remember in our last class, in fuzzy logic not in the last class, in previous one of our previous classes. We have discussed that, how the PI or PID type of controller can be represented in terms of error and change in error or their function of error or change in error. Using the rule based structure of fuzzy logic controller, break the control problem down into a series of IF X AND Y THEN Z rules that define a desired controller output response for a given system input conditions. Create fuzzy logic controller membership functions that define the meaning values, linguistic values of input output terms used in the rules because; a fuzzy logic controller resists the crisp input.

(Refer Slide Time: 03:19)



But, the computation is through linguistic variables and after that the output is again crisp because, the system which is out there is processing crisp input and crisp output. The next is the test of the system, evaluate the results, tune the rules and membership function and continuously simulate until satisfactory results are obtained.

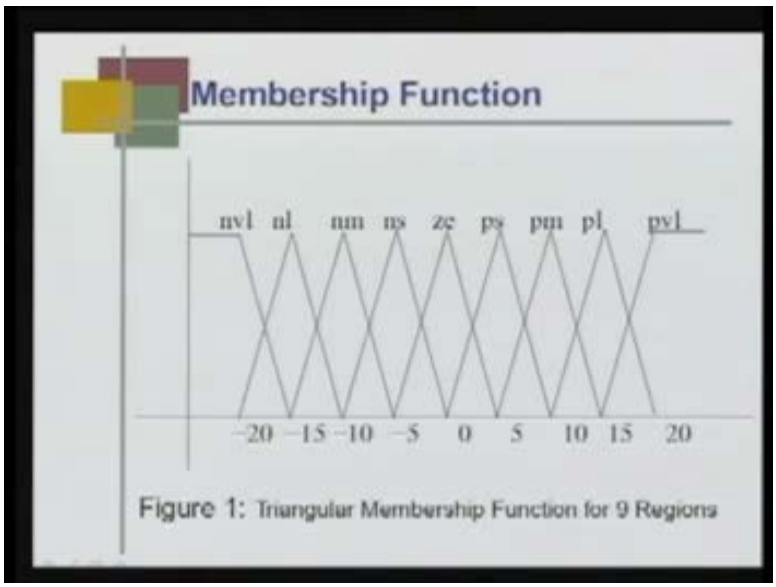
(Refer Slide Time: 03:59)



So, complete architecture of a fuzzy logic controller. This is our plant of process. You have sensors and then you get the plant feedback through sensors which are converted to

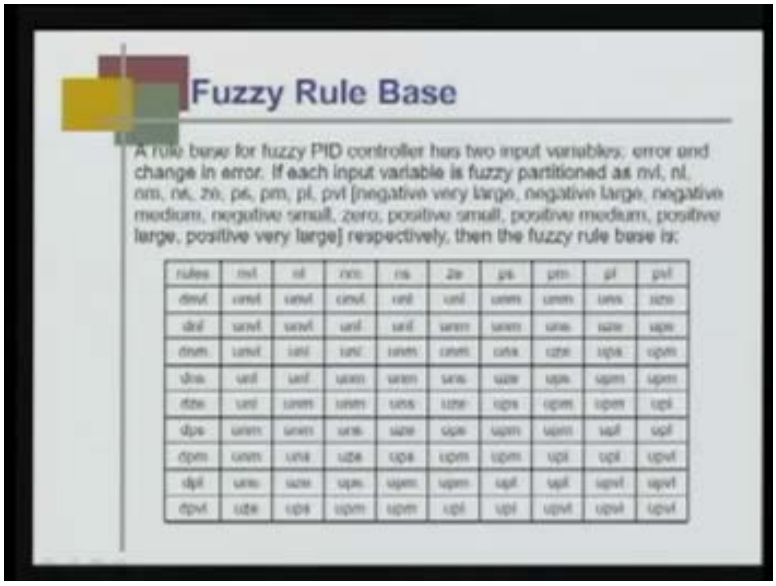
fuzzy linguistic variables which, is known as fuzzification. That goes to the rule base and then inference mechanism. Then, information mechanism actuates the control signal in fuzzy variable which is converted to crisp values through defuzzification and the driver or actuator gives to the control signal the actual process. This is our architecture.

(Refer Slide Time: 04:45)



You see that, normally if say for example, this is my error and this is my membership function mu and maximum value is 1. You see that, if my change in error, this is in unit, so we can define here, it shows that, this is positive very large, positive large. This is thus giving an idea what pvl values is positive very large. Similarly, nvl is negative very large. So, you say that, this is the way. This are the linguistic values, negative large negative medium, negative small 0, the 0 is around the value 0, positive small, positive medium, positive large and positive very large. This way you see that, in this case the variable e the error has been fuzzy partitioned into 9 fuzzy linguistic variables from negative very large to positive very large. So, once you fuzzify a crisp variable then we are almost very clear about the rule base.

(Refer Slide Time: 06:19)

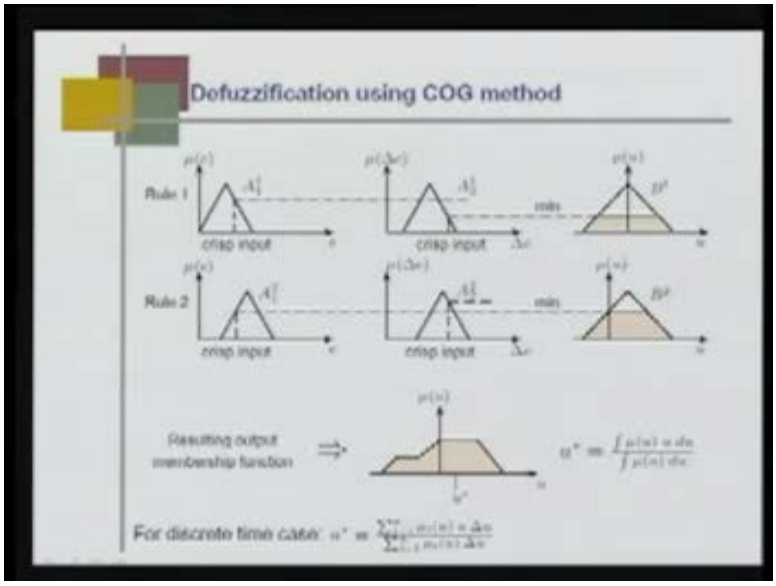


For example here, a rule base for fuzzy PID controller has two input variables, error and change in error, if each input variable is fuzzy partitioned as negative very large, negative large, negative medium and so forth until positive very large respectively then, the fuzzy rule base is that hardly found at the rule. So this is my error and this is my change in error (Refer Slide Time: 06:49). What I am saying that error is negative very large, negative large until positive very large and change in error is this is d is differential term. So, that is also negative very large, negative large, negative medium, d simply refers here to change in error. That is also fuzzy partitioned into 9. So, 9 into 9 obviously the number of rules are 81. So, how do I do value is, if I say error is negative very large and change in error is negative very large then, my control actions will be negative very large.

We can take any other example, let us say positive medium my change in error where error is a 0. The action should be positive medium. So, this is way. You see that, all 81 rules are possible maximum number of rules given 8 fuzzy partitioning zones, in 2 variables. Then, 9 into 9 total 81 rules, 81combinations are possible. Total is 81 rules. Once rule is given then, given a situation we see which the rules are. Like you know if you look at here, given a situation all the things will not fire. For example, you see here if my crisp values are here. Only 2 rules will fire 0 error and positive small, in that sense, maximum time only two rules fire, otherwise it all depends how you fuzzify, I cannot say

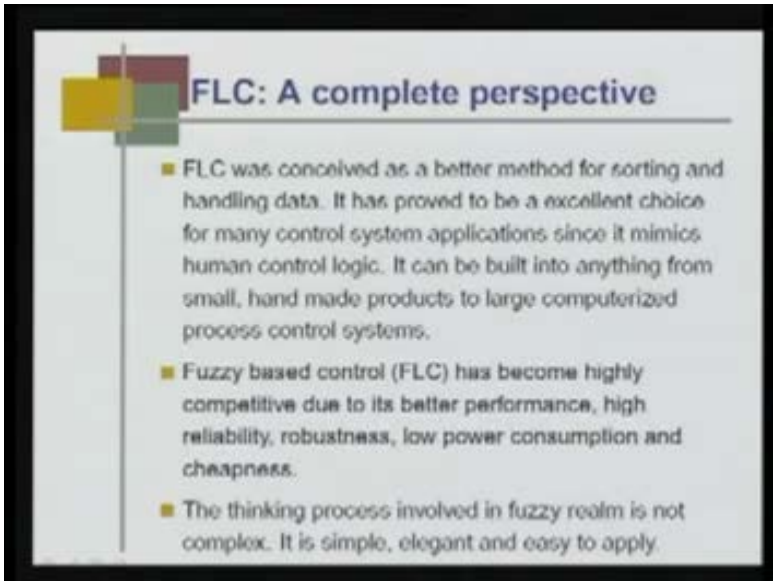
that only two rules firing. Also you can make rules of only one rule; atleast one rule has to fire. Otherwise your scheme is not right. You have to design in such a way atleast one rule will fire, given a situation but in general more than one rule fires. But in general fuzzy partition is done in such a way that two rules fire. That is why, you are saying two rules: Rule 1 and rule 2.

(Refer Slide Time: 09:15)



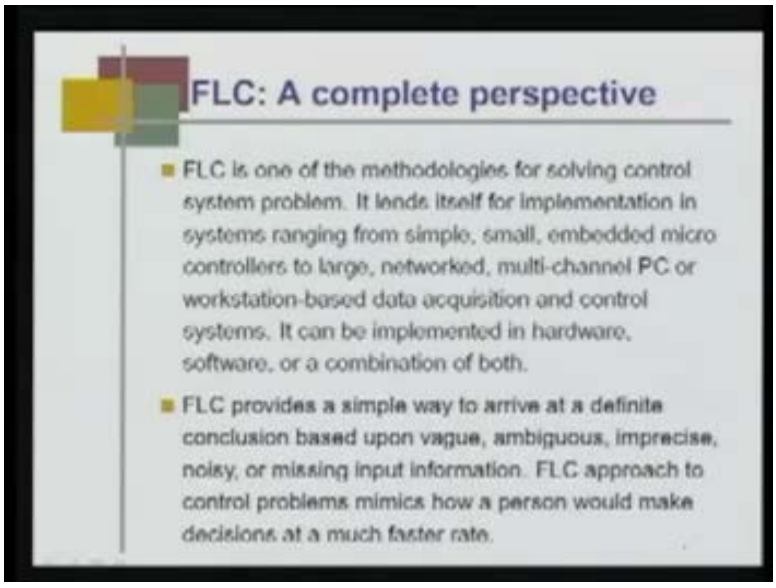
Given this, we find out this is my crisp input. This is rule 1 fires and gives, this is my control action and rule 2 fires and gives me the control action like this (Refer Slide Time: 09:35). Then, what we do is, the sided portion is the control action. This is given by mean principle and now we add this to shaded area. Then, we get this complete area and this is my complete action control action. How do I find out this is fuzzy control action? This fuzzy control action is converted to the crisp control action using center of gravity method. This we have already discussed, center of gravity method. In general now I would like to turn your attention, why we are doing fuzzy logic controller? We are doing because; fuzzy logic controller was conceived as a better method for sorting and handling data.

(Refer Slide Time: 10:27)



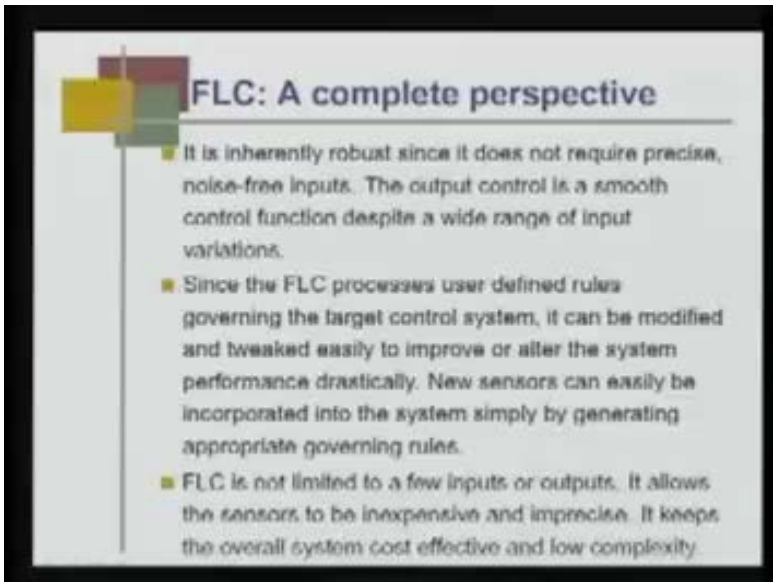
It has proved to be an excellent choice for many control system applications since, it mimics human control logic. It can be built into anything from small hand made products to large computerized process control systems. Fuzzy based control has become highly competitive due to its better performance, high reliability, robustness, low power consumption and cheapness. The thinking process involved in fuzzy realm is not complex it is simple, elegant and easy to apply.

(Refer Slide Time: 10:59)



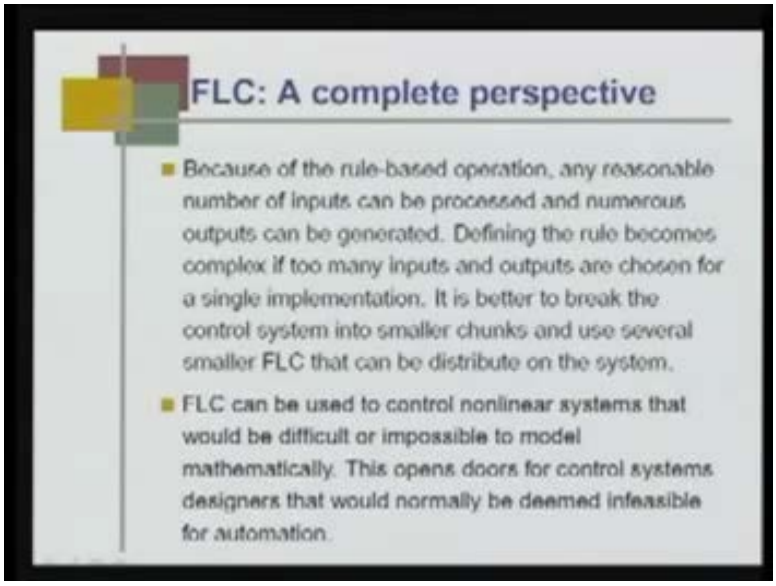
Fuzzy logic control is one of the methodologies for solving control system problem. It lends itself for implementation in systems, ranging from simple small embedded micro controller to large networked multi-channel PC or workstation based data acquisition and control system. It can be implemented in hardware software combination of both Fuzzy control directly also can be implemented in fuzzy hardware, fuzzy chips. A fuzzy logic controller provides the simple way at a definite conclusion based up on vague ambiguous imprecise noisy or missing input information. Fuzzy logic controller approaches to control problem, mimics how a person making a decision at a much faster rate.

(Refer Slide Time: 11:51)



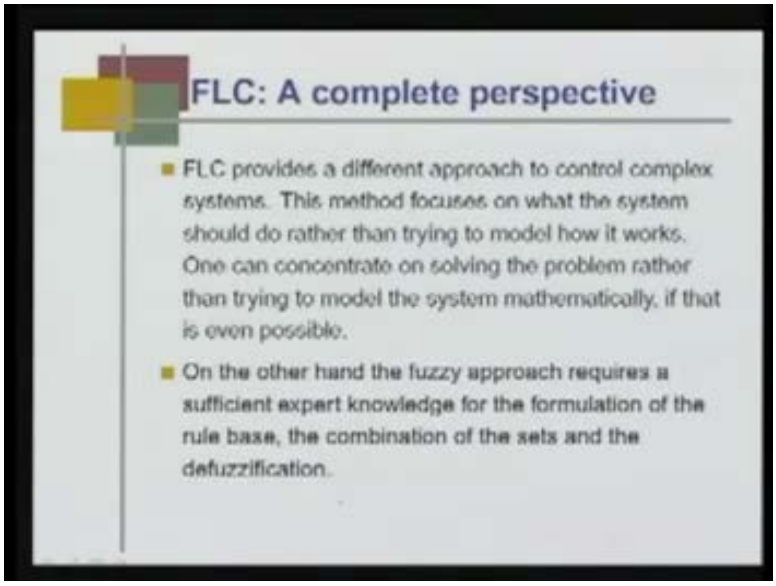
It has inherently robust, since it does not require precise noise free inputs. The output control is a smooth control function, despite a wide range of input variations. Since the fuzzy logic controller processes user define rules, governing the target control system. It can be modified and tweaked easily to improve or alter the system performance drastically. New sensor can easily be incorporated into the system by generating appropriate governing rules. Fuzzy logic controller is not limited to a few input or outputs. It allows the sensor to be inexpensive and imprecise. It keeps overall system cost effective and low complexity.

(Refer Slide Time: 12:36)



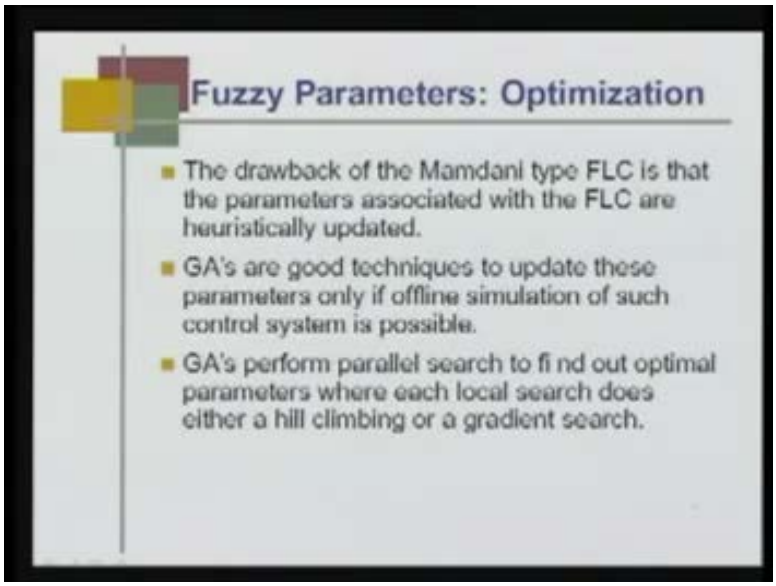
Because of the rule based operation, a reasonable number of inputs can be processed and numerous outputs can be generated. Defining the rule becomes complex if too many inputs and outputs are chosen for a single implementation. It is better to break control system into smaller chunks and use several smaller fuzzy logic controllers that can be distributed on the system. Fuzzy logic controller can be used to control non linear system. That would be difficult or impossible to model mathematically. This opens doors for control system designer that would normally be deemed infeasible for automation.

(Refer Slide Time: 13:15)



Finally, fuzzy logic controller provides a different approach to control complex systems. This method focuses on what the system should do, rather than trying to model how it works. One can concentrate on solving the problem rather than trying to model the system mathematically, if that is even possible. On the other hand the fuzzy approach requires a sufficient expert knowledge for the formulation of the rule based on the combination of the sets and the defuzzification.

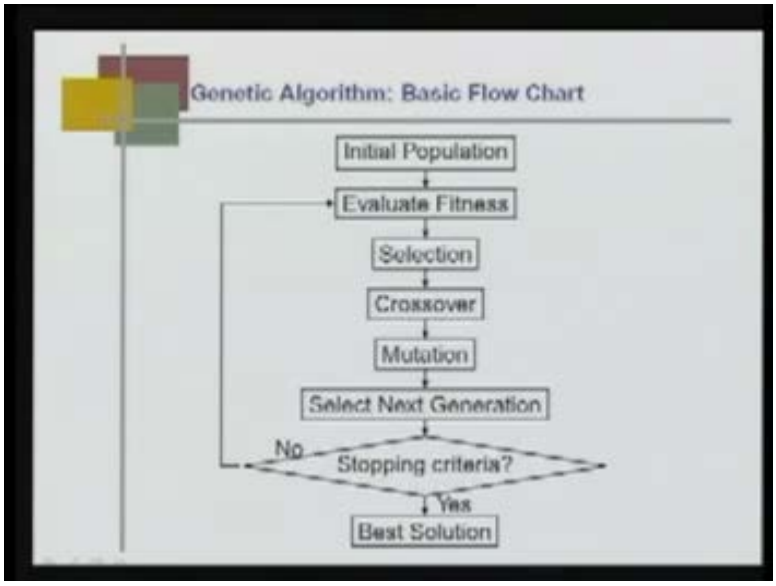
(Refer Slide Time: 13:42)



The drawbacks of the Mamdani types, so now we talked about fuzzy logic controller, it is in a kind of found the positive aspect of fuzzy logic controller. Now we are only discussing Mamdani type of fuzzy logic controller. The problem is that, you have rule base and this is associated with many parameters, the membership parameters. These membership parameters because; how many rules, how many fuzzy zone portioning and all those things and membership function, whether you can select… if you are selecting Gaussian membership function then a Gaussian membership function is characterized by two parameters. One is the mean another is the sigma.

How do we optimize? Because normally what the engineer does is that he kind of heuristically tunes his rules. But can there be proper method of tuning? Normally the genetic algorithm is good technique to update these parameters. But, this is only possible if you are doing it offline simulation. That is the model that is being controlled. We have some kind of mathematical form. The system that we are trying to control we have a mathematical model for it. Then, we can always design a fuzzy logic controller using Mamdani type. We can optimize the parameters using genetic algorithm. Normally what is done is that, given a physical system you take some approximate mathematical model and do offline simulation. Optimize the parameters and take to the real time system and their heuristically tune the parameters. So, GA's perform parallel search. What is GA and GA's perform parallel search? Find out optimal parameters where each local search does either a hill climbing or a gradient search.

(Refer Slide Time: 15:55)

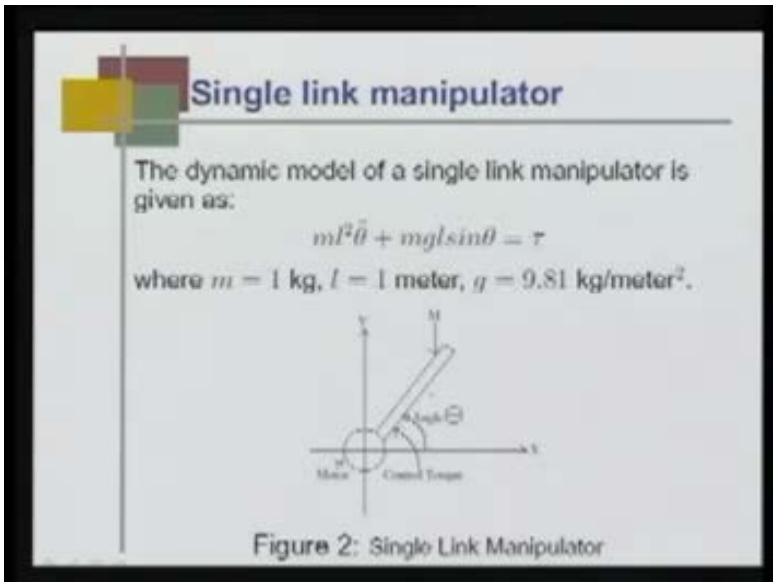


A genetic algorithm you see the first initial population means this is when any population means in the case of a control systems solution random solution space of parameters. If I know every parameter has certain range. In that range I create random numbers than I take each solution set and then I take each parameter set and then evaluates the fitness. Based on the fitness value but in this case what is the fitness obviously it is tracking error. If the tracking error is minimum in a specific case that as in the top. Similarly, there are various, so we reorganize, rearrange the population of fitness, in other various methods selection and specifically for control system, you are very comfortable simply doing proportional selection. Proportional selection means, according to the fitness the selection is done. If I say the initial population is 100 and I will always select after every iteration, only 30% of them. That means 30% of the top best candidates are selected and from those candidates 30% from this 100. So, I have 100 I categorize then according to their fitness value and then 30% I take and these 30% through crossover and mutations, are again converted 100. I started with 100 using fitness, evaluate through fitness I did a selection in terms of proposal of selection 30% best out of 100 is taken and through crossover and mutation, I duplicate this 30 to 100 again.

Initial population was 100.We went through the evaluation process using a fitness cross function. Then, we did the selection using the best values and then using cross over

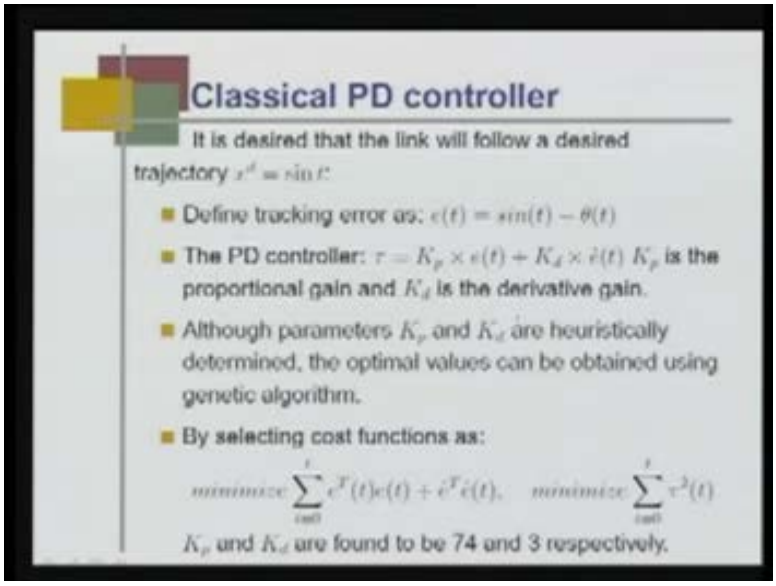
mutation we again got 100. What is the stopping criterion? Stopping criteria is that how much tracking error I need. Normally we select a stopping criteria is that what is the minimal requirement for tracking error or root mean square of the tracking error something like that. Or you can always say also I do just 100 iterations or 1000 iterations. One of them, whatever stopping criteria you get out of those solutions, what is the best you take and that best is your optimal solution. Now we will utilize this concept to control. We take a very simple example. It is better to learn as subject through simple example.

(Refer Slide Time: 19:44)



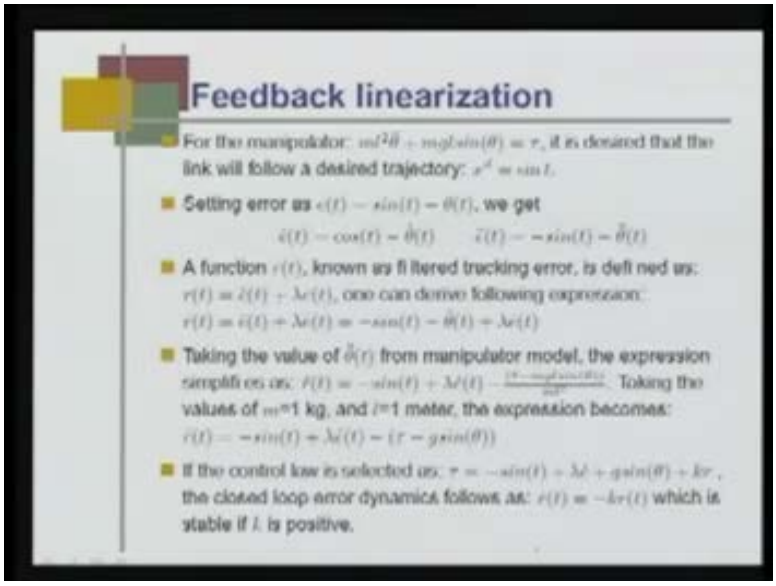
Again consider a single link manipulator is what you are seeing here. This is my link, this is the motor. I apply the control torque this is my angle theta (Refer Slide Time: 19:54) and this if I assume this theta is actually from this angle theta. This is angle theta. Then, ml square theta, double dot plus mgl sin theta is tau. So, m is 1 kg l is 1 meter g is 9.81 kg per meter square.

(Refer Slide Time: 20:22)



It is desired that the link will follow. We are talking about how to implement for remote manipulator PD controller. It is desired the link will follow a desired trajectory x d equal to sin t. What we are saying that this will simply oscillate around its particle position like this. The PD controller tracking error is obviously sin t. The desired trajectory minus theta t the PD controller tau equal to K_p plus e (t) plus K_d into e dot t. So this is our PD controller. K_p into error plus the derivative gain into the differential of the error K_p, is the proportional gain and K_d is the derivative gain. Although parameter K_p and K_d are heuristically determined the optimal values can be obtained using genetic algorithm. For certain purpose, what we did that this is the multi objective function. We want to minimize the cost function in error as well as the change in error. Here also the total controls effort the square. We got the quadratic function in error in quadratic function change in error in the actuated control input. We want to minimize both. So, doing that running a genetic algorithm for this, we got the K_p value is 74 and K_d value is 3.

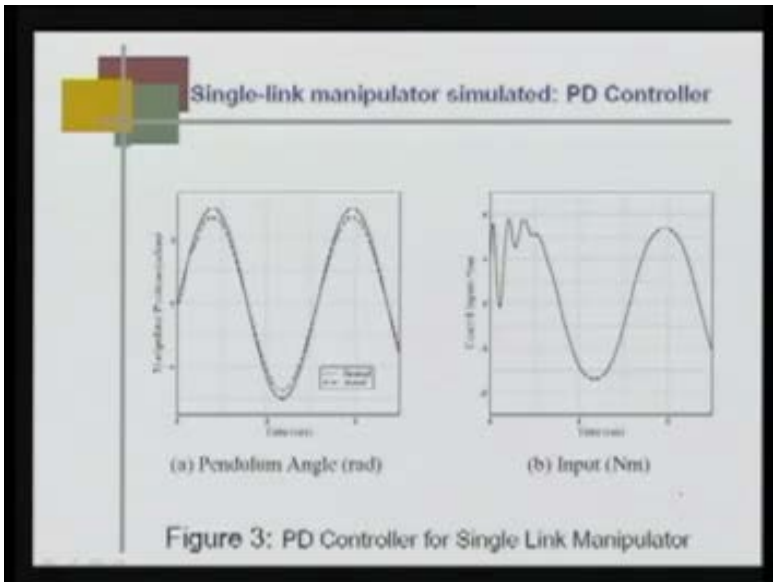
(Refer Slide Time: 22:25)



We will see that, for same remote manipulator also can be control a feedback linearization. For this again the same manipulator ml square plus double dot plus mgl sin theta tau. It is desired that, it will track sin t again setting the error sin t minus theta t as the error. We get the derivative of error is cos t minus theta dot(t) and the double derivative of e error is minus sin t minus theta double dot(t). If I select a filter tracking error r (t) to be e dot(t) lambda e(t) then, I can write r dot(t) is e double dot(t) plus lambda e dot(t). This e double dot(t) can be written here minus sin t minus theta double dot t plus lambda e dot t. So here, this theta double dot(t) (Refer Slide Time : 23:27) can be taken from here from this expression which is tau minus mgl sin theta upon ml square theta is theta double dot. This is theta double dot minus lambda e dot is retain here minus sin t is written here. This is my closed loop error dynamics. So to make, if I take the value of m equal to 1, l equal to 1, my final close loop dynamic becomes this one r dot(t) r dot(t) is minus sin t, plus lambda e dot(t) minus tau minus g sin theta. How do I select my control law tau such that, this is stable? So if the control law is selected, tau is minus sin t plus lambda e dot plus g sin theta plus kr. Then, the closed loop error dynamics becomes r dot(t) is minus kr. This is stable, if k is a positive constant. This is a very simple feedback linearization because it is a non linear system we selected tau, the control law which is this one. This is the control law.

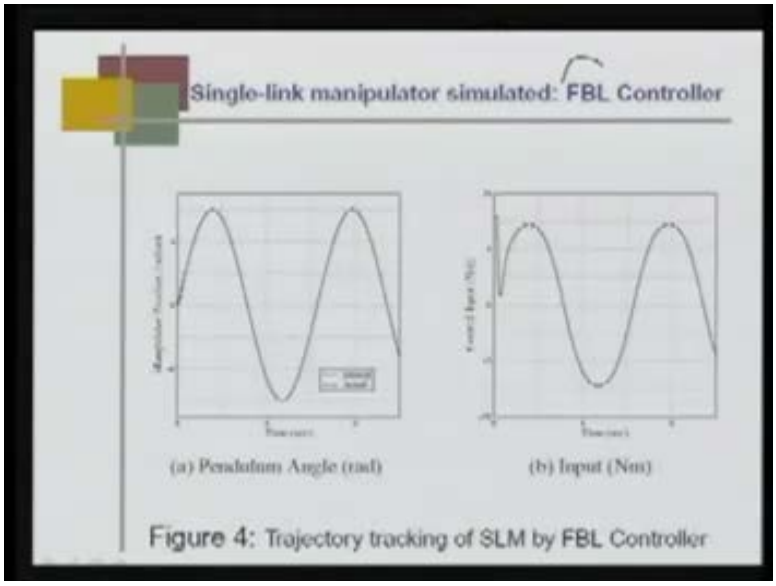
If we select this control law, we are able to show that, the close loop error dynamics is given by r dot(t)is minus k dot. This is a stable dynamics.

(Refer Slide Time: 25:16)



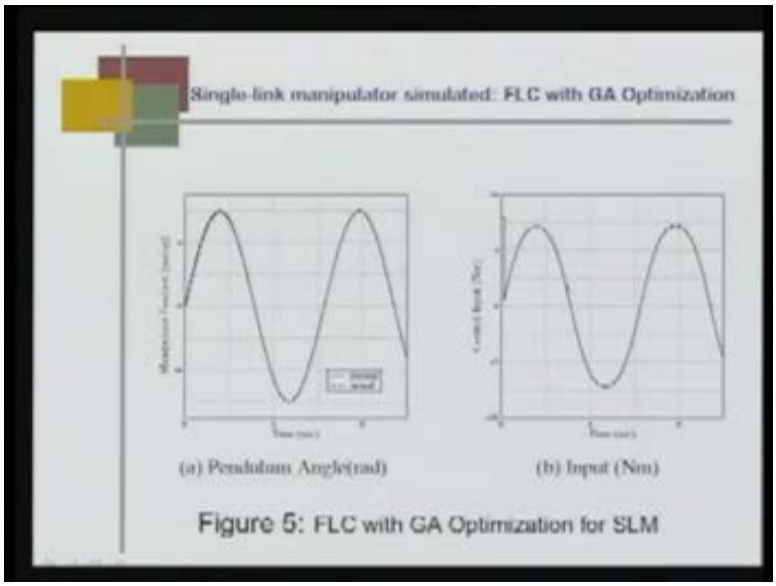
What we did is that, we did a feedback linearization. Before that, we did classical PD controller and also using the earlier fuzzy controller also, we implement a fuzzy controller. Now let us see the performance. PD controller what is the performance you see that, this is my solid line is my desired and the broken line is actual. So, this tracking is not so good which is natural because, simple PD controller and hence the control access are given here. This is most optimal performance of the PD controller because you have optimized the parameter of K_p and K_d using genetic algorithm.

(Refer Slide Time: 26:06)



Now feedback, this is FBL is feedback linearization technique. So using this controller you easily see that the tracking is very good, as well as this is the oscillation in the input torque is reduced here and it is very smooth torque. Now, we have implemented as fuzzy logic controller g optimization.

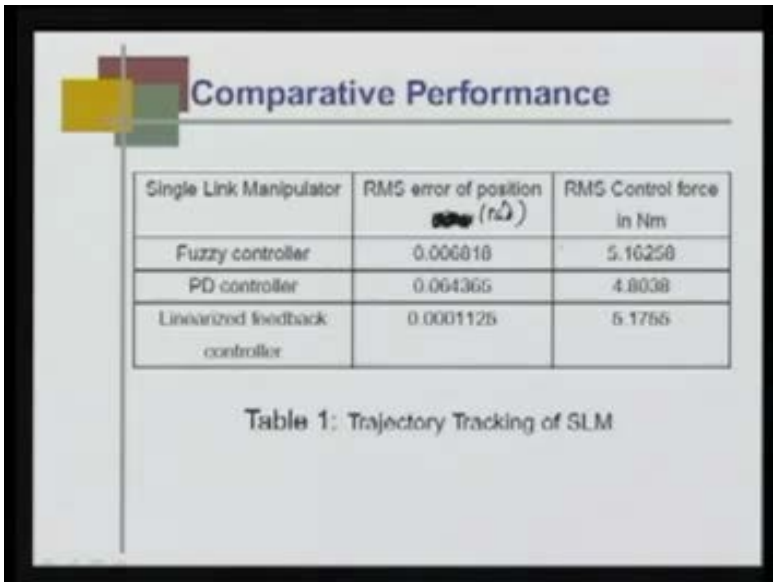
(Refer Slide Time: 26:37)



Using this, when we again did the performance is good with the control actuation is also very smooth just like Feedback linearization, it is the best achieves, the very good

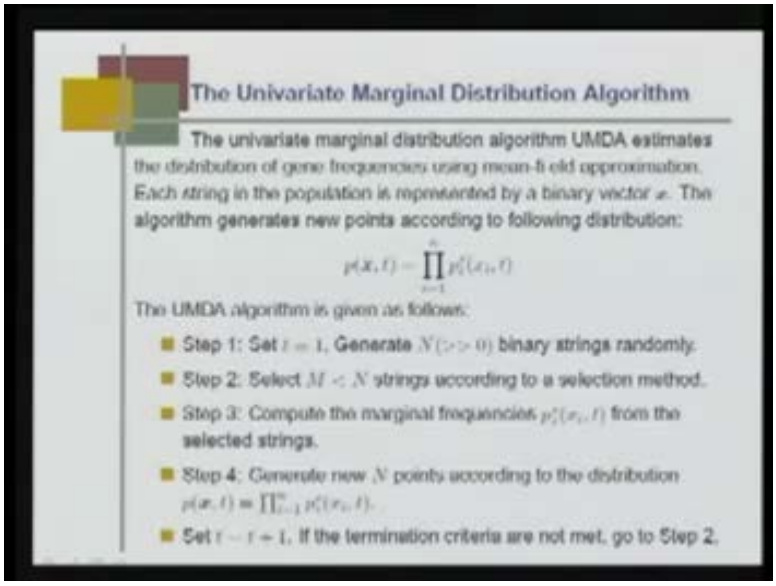
controller. So, fuzzy logic controller, it was started with this g of optimization also provide the similar.

(Refer Slide Time: 27:12)



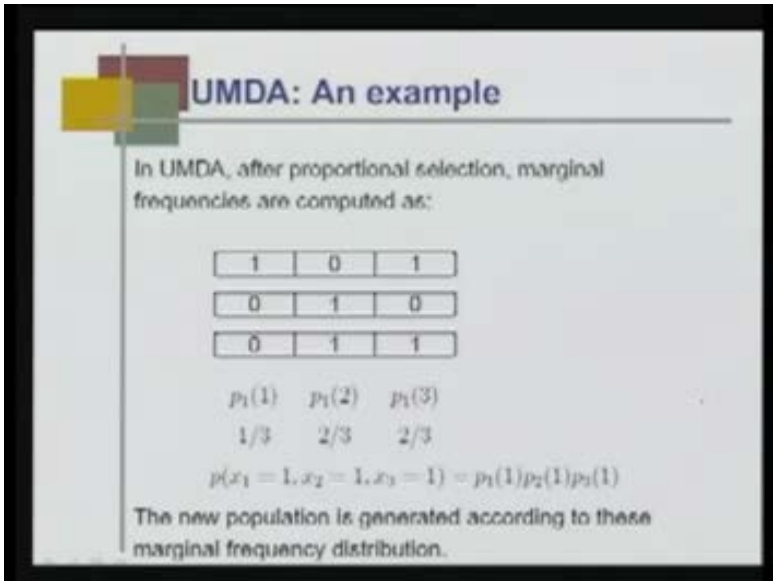
If I want to compare comparative performance, you see that, fuzzy controller is now with this same. You can see that same controller input effort. You see that, this is the comparative because; we are almost applying same control input 5.1, 5.1, 4.8 Newton meter. This is root mean square control source. So, total control at every time. If I have u(k) or if I have u(t) ut square into dt. Find out and then average it. Then, take the root means square. I get this control effort and correspondingly you see fuzzy controller has the order of error is 10 to the power minus 3. But, PD controller has 10 to the power of minus 2 and linear feedback controller say 10 to the power minus 4, so fuzzy controller is almost equivalent to linearization feedback controller. You will now of course, this is just an example. Using some simple example the fuzzy logic controller also can be further improved. The invariant you will now discuss about another evolutionary computation approach for optimization the Univariate marginal distribution algorithm.

(Refer Slide Time: 28:59)



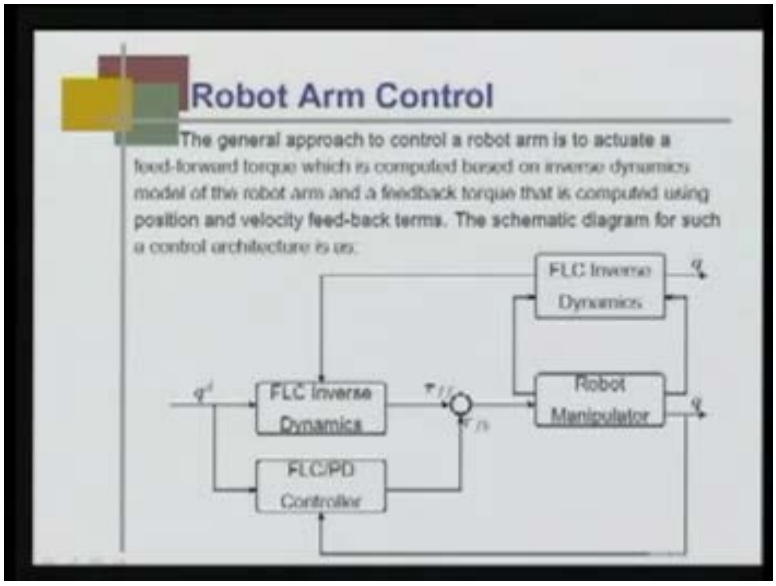
UMDA estimates the distribution using mean field approximation. Each string in the population is represented by a binary factor x. The algorithm generates new points according to following distribution. This is called mean field approximation. This is the distribution of a total string and this is the individual bits in the string. So, the UMDA algorithm that is what is meaning of this is that, if x is a factor of x_1, x_2 and x_3, then, p(x) is simply $p(x)_1$, $p(x)_2$, $p(x)_3$. This is called mean field approximation in probability theory. So, probability joint distribution is computed simply multiplication of individual or marginal distribution. So, the step 1: set t equal to 1 and generate n binary strings randomly and this binary streams represents certain parameters. Step 2: select m less than n strings according to a selection method, normally proportional method. Step 3: compute the marginal frequencies for each one, from the selected strings. Generate new n point according to this distribution which we have derived. Set t equal to t plus 1. If the termination criteria not met, go to step 2.

(Refer Slide Time: 30:48)



The example I will say, let us say I have 3…, the consist 3 bits and I have such 3 such strings. Then, you can easily simply the marginal frequency of 1 in the first bit is 1 upon 3. Marginal frequency of 1 in the second bit is 2 by 3 and marginal frequency of 1 in the third bit is 2 upon 3. Obviously, the probability join distribution of x_1 equal to 1, x_2 equal to 2, and x_3 equal to 1 is simple multiplication of this, which is 1 upon 3 into 2 upon 3 into 2 upon 3. Then, easily see this is 4 upon 3 cube or 4 upon 27. How do we do it? New population is generated according to this marginal frequency distribution. What I do in the next time when I generate, I generate here. So, I generate, what I had say 10 strings. I did a selection and I proved to 3 best and this 3 best, are converted again to 10 number of population to 10 using this. Why I do is that the first strings, they are generated. I will generate 1 with the probability 1 up on 3 and with probability 1 up on 3 1 is selected probability 2 up on 3 0 is selected in the first string, first bit and second bit with the probability 2 up on 3 I select 1, otherwise 0 and similarly third one with probability 2 up on 3 I selected 1 and with probability 1 up on 3 as select as 0.

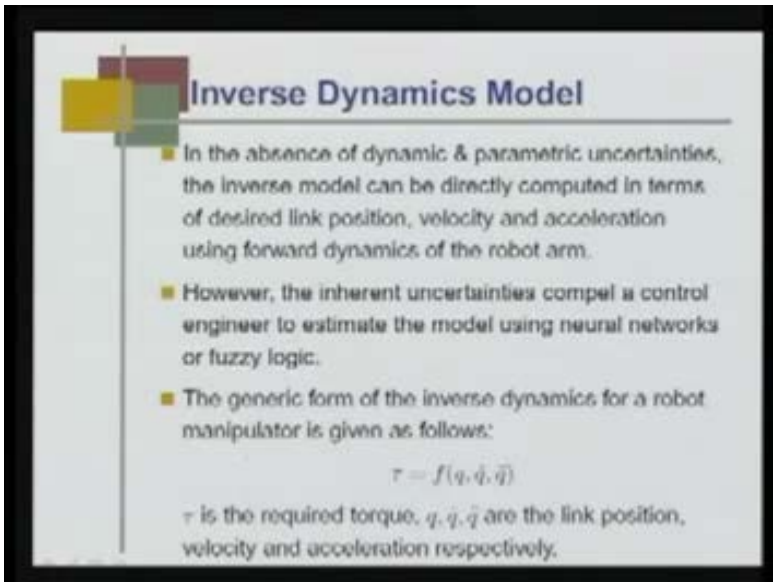
(Refer Slide Time: 33:12)



This is the UMDA univariant marginal distributional problem. Now we will show you in this particular lecture, the application of this particular invariant marginal distribution algorithm to a robot arm control. The general approach to control a robot arm is to actuate a feed forward torque which is computed based on inverse dynamics model of the robot arm and a feedback torque that is computed using position and velocity feedback terms. The schematic diagrams for such control architecture are as follows: What you are seeing is here that given a robot control robot arm, this is my robot manipulator. The normal approach to control a robot arm is that, I have an inverse dynamics which gives me given the desired trajectory gives me what is the feed forward torque. Given the desired trajectory using the feedback, actual output feedback I have a PD controller and that actuator is tau feedback. This is the total tau and the objective is that, the robot manipulator should follow the desired trajectory. That means q should follow q d. This is the generic architecture for robot manipulator. This is inversion, this is for stability. Now you see that, given if I assume that robot manipulator dynamics are not exactly known or there are uncertainties what I can do is that, I can learn inverse dynamics of robot manipulator.

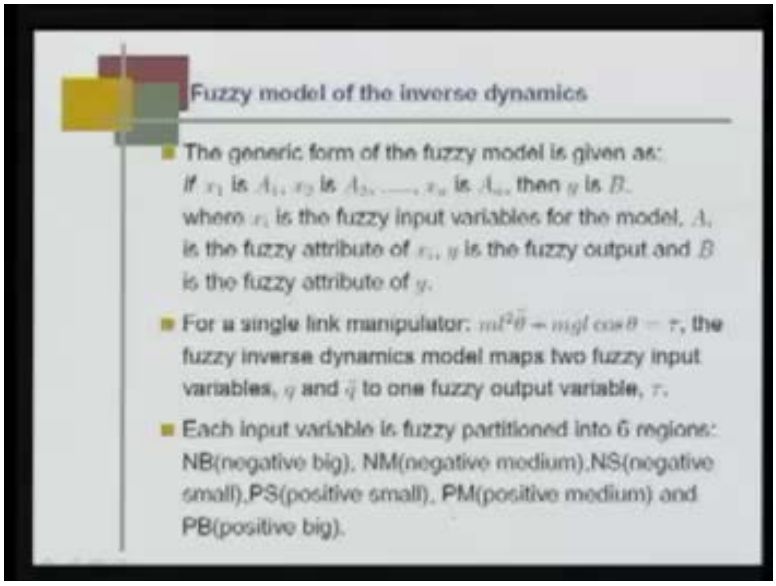
I put this inverse dynamics here. I can also design instead of PD controller a fuzzy PD controller. Now I am controlling the same system using a different approach that is fuzzy logic controller.

(Refer Slide Time: 35:26)



What is inverse dynamics model? In the absence of dynamic and parameteric uncertainties the inverse model can be directly computed in terms of desired link position the velocity and acceleration using forward dynamics of the robot arm. This is very simple in the general robot dynamics is m theta double dot c theta dot plus g is tau. All that I have to do is that, if I know mlg. Then, given theta d theta dot d and theta double dot d. I can compute tau. This is the best way to control the robot manipulator because this is the inverse dynamics but, unfortunate most of the time the parameters that we identify m c and g. They may not be exact. However, the inherent uncertainties compile a control engineer to estimate the model using neural networks or fuzzy logic. The generic form of the inverse dynamics for a robot manipulator is thus given as tau is the function of q desired velocity desired and acceleration desired. So, tau is the required torque and position velocity and acceleration as the link position the velocity acceleration respectively.

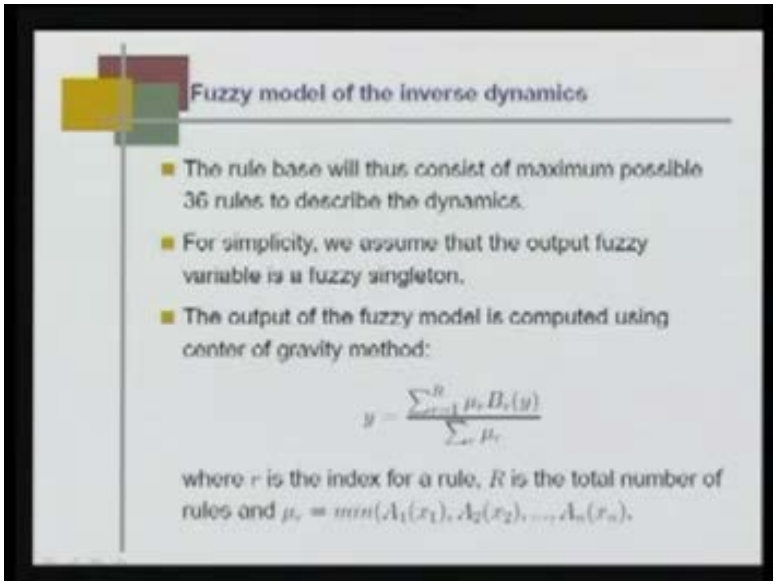
(Refer Slide Time: 37:09)



The fuzzy model of the inverse dynamics you see that, we talked about this model. First we will solve this one. How to derive a fuzzy logic, fuzzy inverse dynamics of a given robot manipulator? So, the generic form of the fuzzy model is given as, if x_1 is A_1 x_2 is A_2 and x_n is A_n then, y is B. Where x_i is fuzzy input variables for the model A_i is the fuzzy attribute of x_i and y is the fuzzy output and B is the fuzzy attribute of y. For a single link manipulator, if I take ml square double dot plus mgl cos theta is tau. The fuzzy inverse dynamics model maps, 2 fuzzy input variables q. This is not q this is theta and theta double dot to one fuzzy output variable. Sometimes we represent this angular position and angular velocity and acceleration either in terms of theta or q. So that, we have 2 variables here, theta and theta double dot.

Because we know the torque does not depend on the other variable whichhis is theta dot. How many variables then this inverse model are dependent on in this particular example two input variables. So each input variables are fuzzy partitioned into 6 regions, negative big, negative medium, negative small, positive small, positive medium and positive big. Once you have 2 input variables and each input variable is fuzzy partitioned into 6 zones. Then obviously, we have the rule base that will consist of maximum possible 36 rules.

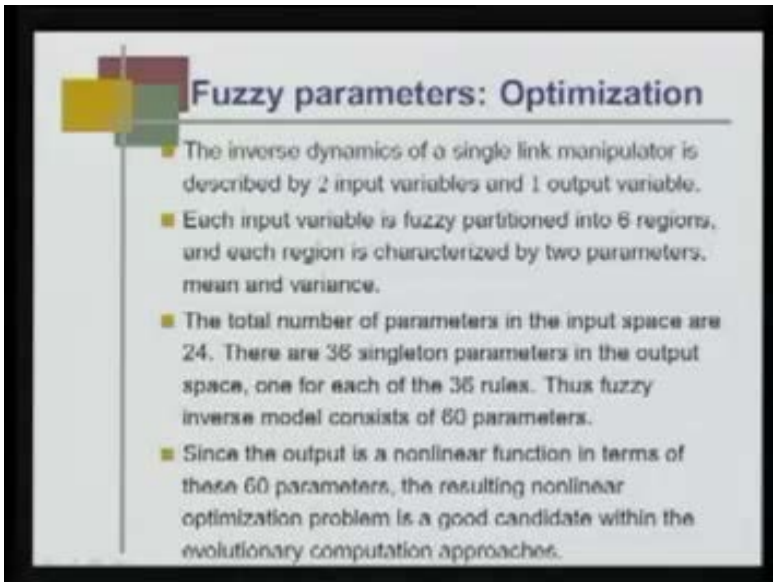
(Refer Slide Time: 39:12)



To describe the dynamics for simplicity, we assume that, the output fuzzy variable is a fuzzy singleton. It is a fuzzy singleton means it is like this (Refer Slide Time: 39:30). This is called fuzzy singleton. But, if the moment I represent than this is triangular members. We can Gaussian membership. But, singleton means it has a single value. This is the x_1 whatever output is here. This is fuzzy singleton (Refer Slide Time: 39:55) output of the fuzzy model is computed using center of gravity method which have already discussed, where r is the index for the rule, R is the total number of rules and mu_r here is minimum of all this fuzzy index. That is A_1 of x_1 of A_2 of x_2 and $A_n$of x_n. When I fuzzify this given crisp values x_1, x_2, x_n. There corresponding fuzzy values are fuzzy memberships are computed using define fuzzy membership function A_1, A_2 and A_n so fuzzy parameters optimization. We saw that if I am trying to control a single link robot manipulator, I have to design two things. One is inverse dynamic model and another is a fuzzy PD controller. We are now doing fuzzy inverse dynamics. In this we saw that, the torque, actuating torque is a function of two variables which is angular position and angular acceleration. The missing element is angular velocity so 2 input fuzzy partitions into each input is fuzzy partition into 6 variables, 6 linguistic variables. Then, we have 36 rules and if I say that, each membership function is a Gaussian membership function, which we have actually taken in this case. Then, each membership function is

characterized by 2 parameters. One is mean any Gaussian is by mean as well as then variants sigma.

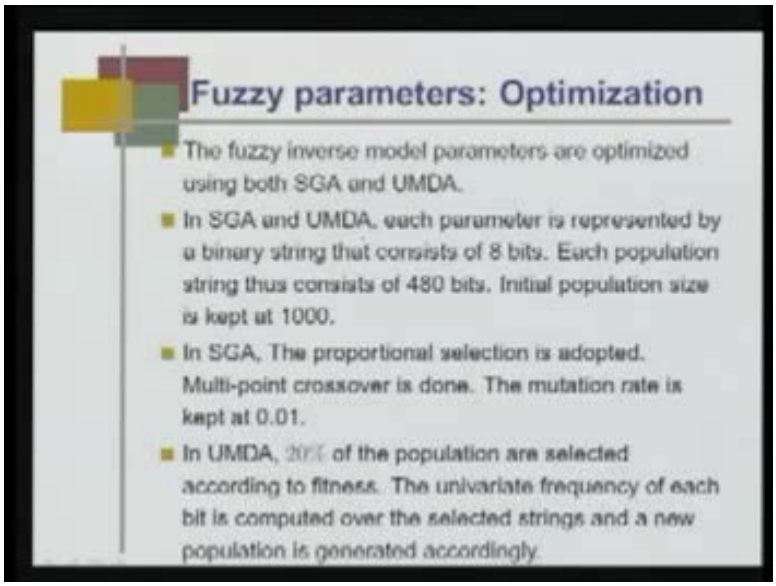
(Refer Slide Time: 41:54)



The inverse dynamics of the single link manipulator is described by 2 input variables, and one output variable. Each input variable is fuzzy partition into 6 regions and each reason is characterized by 2 parameters mean and variance, because of Gaussian membership function. The total numbers of parameters in the input space are 24 because I have 6 partitions in the first input variable and another 6 partition in the second input variable, each input linguistic variable is characterized by 2. So, 6 plus 6 is 12 into 2 is 24 and there are total number of 36 rules. Each rule is associated with one fuzzy single term. So, that is 36 parameters means if x_1 is A_1 x_2 is A_2. Then, y is a specific value y is 10 unit and y is 20 units, y is 40 units like that. Singleton value means it is a precise value. There are 36 singleton parameters in the output space one for each 36 rules. Thus, fuzzy inverse model consist of 60 parameters 24 plus 36. Since the output is a nonlinear function in terms of these 60 parameters. The resulting nonlinear optimization problem is a good candidate within the evolutionary computation approach. What we will do is that, we will now use simple genetic algorithm as well as UMDA univariant marginal distribution algorithm to learn this inverse dynamics. Quickly compare which is doing better because,

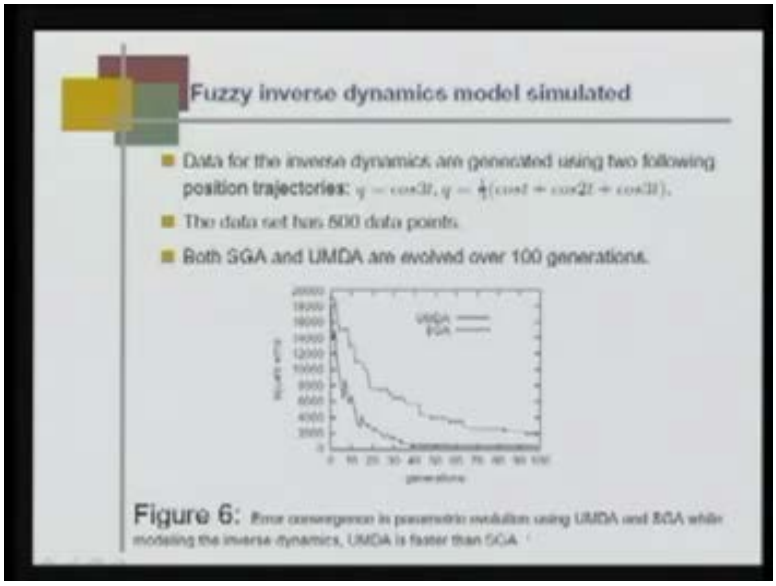
we have learned. The fuzzy inverse model parameters are optimized using both simple genetical logarithm and univariate marginal distribution algorithm.

(Refer Slide Time: 44:02)



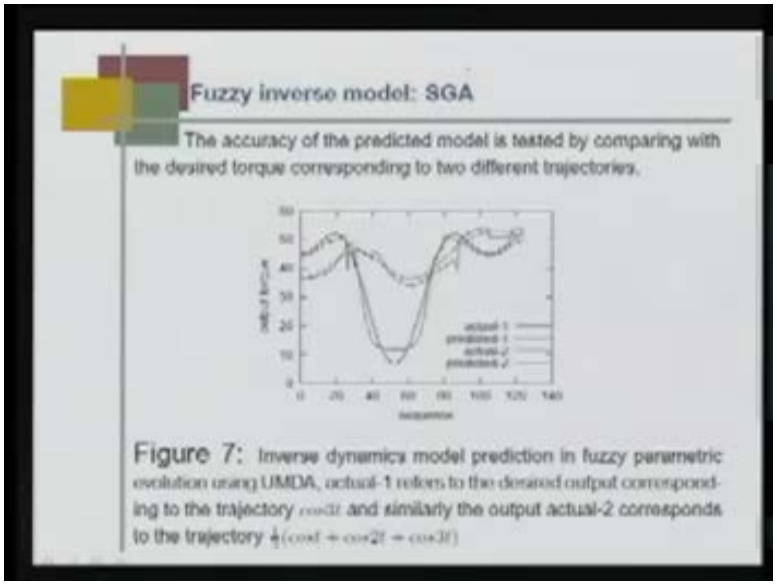
In simple genetic algorithm and univariate marginal distribution algorithm each parameter is represented by a binary string that consists of 8 bits and you have 60 parameters. So, 60 into 8 are 480 bits. It is if I have a string in the population these strings consist of like that. 480 bits and out of that 8 bits represent 1 parameter and total number of 480 bits will represent 60 parameters. Initially, the population size is kept at 1000. So, in simple genetic algorithm the proportional selection is adapted multipoint cross over is done the mutation rate is kept at 0.01; whereas, in univariate marginal distribution algorithm 20% of the population is selected according to fitness. The univariate frequency of each bit is computed over the selected strings and a new population is generated accordingly.

(Refer Slide Time: 45:18)



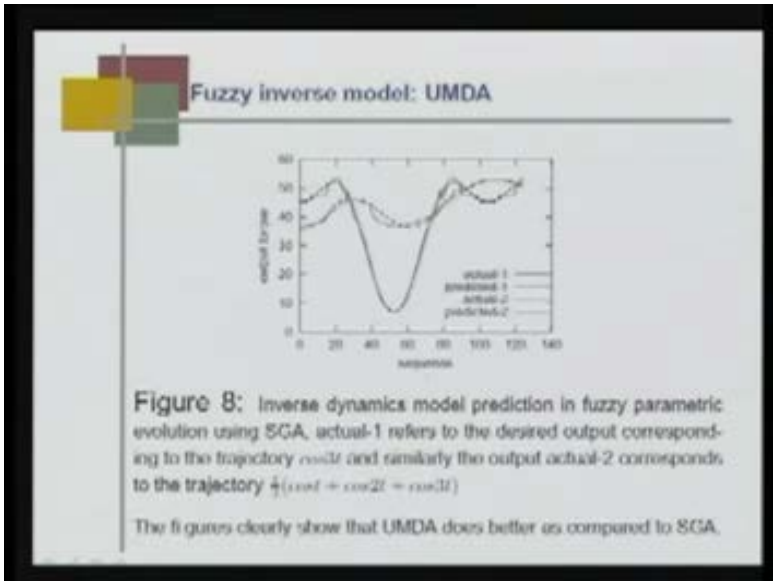
That has been the universe dynamics as generated in two following position trajectories. We took in one case desire trajectory to cos 3 t another case the desired trajectory q is 1 up on 3 cos t plus cos 2 t plus cos 3 t. Using this 2 trajectory, we generated 500 data points. For all this 500 data points with every step we took a parameter set. We computed the output and we matched whether given these values we can find out tau. Using SGA and UMDA both simple genetic algorithm and univariate marginal distribution algorithm are evolved for 100 generations. The cost function is always evaluated for these 500 data points. If we see that; the learning of UMDA which is the solid line and simple genetic algorithm which is the broken line, you see that, error convergence in parametric evolution using univariate marginal distribution algorithm and simple genetic algorithm while modeling the inverse dynamics we are showing it and the conclusion is UMDA univariate marginal distribution algorithm is faster than simple genetic algorithm. The interesting point is that, this univariate marginal distribution algorithm is actually a very simplified form of a probabilistic approach purely probabilistic approach to evolve the solution base.

(Refer Slide Time: 47:22)



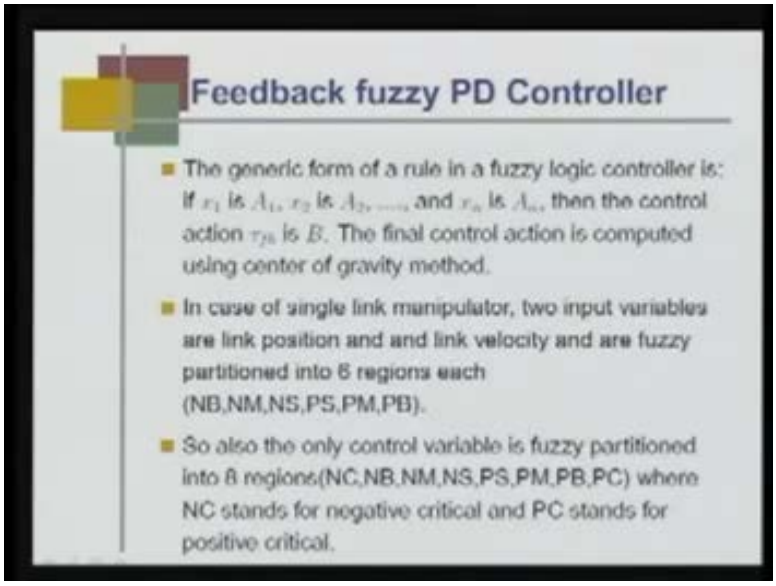
Now we will test the fuzzy inverse model using simple genetic algorithm. The accuracy of the predict model is tested by comparing with the desired torque corresponding to two different trajectories. What you are seeing is that this first trajectory what you are seeing corresponds this is the desired torque. This is the torque output torque tau and the desired torque and this is the given trajectories. If you see that this is the desired torque at different instance along the trajectory cos t is this one desired torque. SGA is estimating the values like this you see that this is SGA estimation. You see this is not so accurate. Similarly, the second trajectory corresponding to the tau is computed like this value; whereas, the predicted through inverse model. There is certainly discrepancy this model learning is not good.

(Refer Slide Time: 49:04)



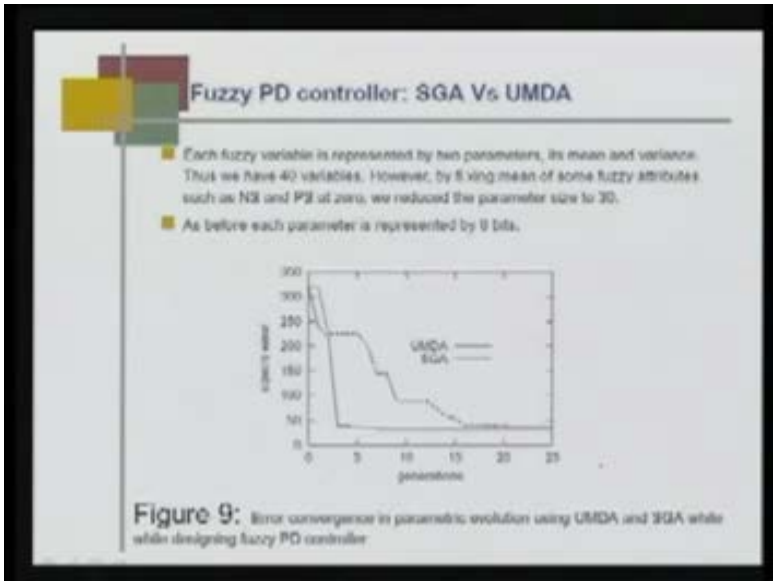
But, when you set to univariate marginal distribution algorithm you can check that, the prediction is much better here, using univariate marginal distribution algorithm here also the prediction is much better. This is the first trajectory cos 3t and second trajectory is the summation of 3 different trajectories. The figure clearly shows that the univariate marginal distribution algorithm does better when compared to the simple genetic algorithm is not that is a blanket statement. We are only considering the simple genetic algorithm for there also very advanced genetic algorithms that may also do better than UMDA. So, we showed until now is an inverse dynamics fuzzy inverse dynamic model.

(Refer Slide Time: 50:04)



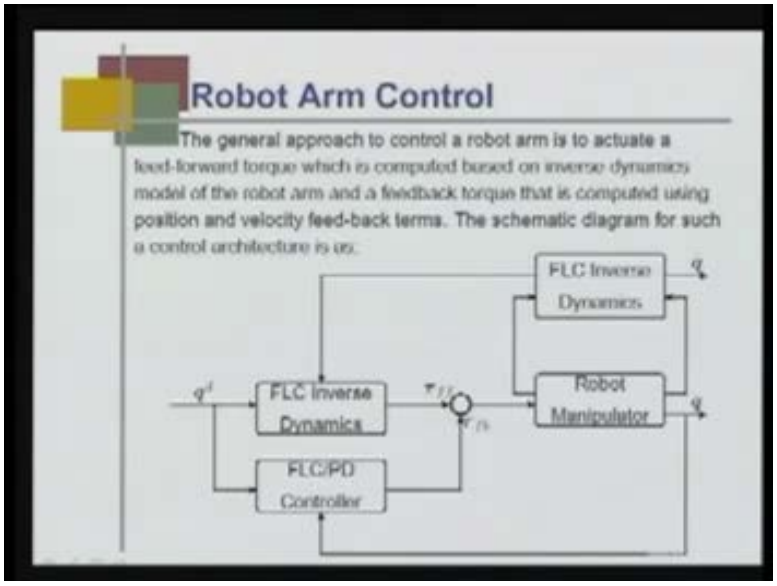
Now, would be talking about feedback fuzzy PD controller. The generic form of a rule in a fuzzy logic controller is again the same if x_1 is A_1, x_2 is A_2 and x_n is A_n then, the control action tau is B. The final control action is computed using center of gravity method in case of a single link manipulator. 2 input variables are link position and a link velocity and are fuzzy portioned in 6 reasons as negative big, negative medium, negative small, positive small, positive medium, and positive big. Also the only control variable is fuzzy partitioned into 8 reasons: Negative critical, negative big, negative medium, negative small, positive small, positive medium, positive big and positive critical.

(Refer Slide Time: 50:57)



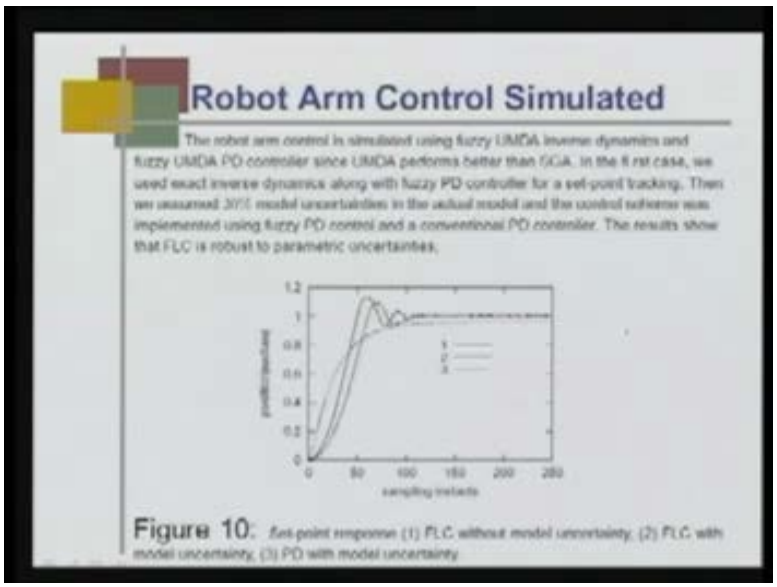
Each fuzzy variable this is represented by two parameters taking Gaussian membership function, its mean and variants. Thus, we have 40 variables. However, by fixing means of some fuzzy attributes not negative small, positive small 0. We reduce the parameters tries to 30. You have to do little because, we are very sure you can fix certain parameters and then you reduce those parameters size to 30. As before each parameter is represent by 8 bits so naturally is now consisting of 240 bits. Now we see that, error convergence in parametric evolution using UMDA and SGA while designing fuzzy PD controller is that, UMDA is pretty fast compare to simple genetic algorithm.

(Refer Slide Time: 51:54)



Finally, we first design two aspects separately. Now, you will put both of them together and then, we will design a robot arm control using both inverse dynamic and PD controller. So, if we do that which proposed earlier.

(Refer Slide Time: 52:20)

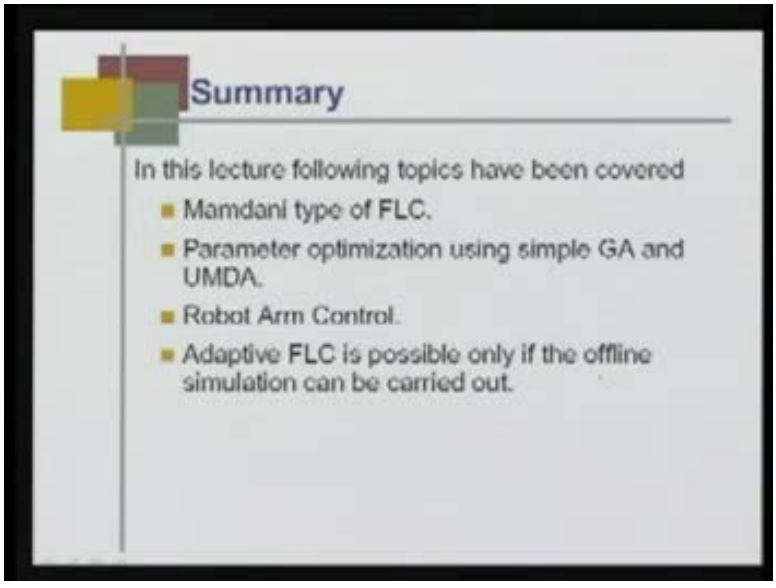


See that, the robot arm control is simulated using fuzzy UMDA inverse dynamics and fuzzy UMDA PD controller. Since, UMDA performance is better than SGA. In the first

case, we used exact inverse dynamics are along with fuzzy PD controller for a set point tracking. Then, we assumed 20% model uncertain is in the actual model and the control schemes are implemented using fuzzy PD controller and a conventional PD controller. The results show that, FLC is robust parametric uncertainties. So, in the beginning you see that, this solid line which is this one, this particular one (Refer Slide Time: 53:02). So, this solid lines represents when we have the model is exact for which we have derived the fuzzy inverse model and we have derived the fuzzy controller. Now introduced 20% model certainties and doing the model uncertainties, we again are running on the chain fuzzy logic controller which we derived for the exact model.

Now you took the 20% uncertainties in the parameters, we introduced that, arbitrarily randomly and then we saw the controller performance has not degraded, the second one which is inversing this. But, if you do simple the PD controller with the classical PD controller and assume the 20% model uncertainties obviously, due to model uncertainties you cannot go to the exact set point. This is your set point desired set point. There is an error, the actual set point and desired set point. So, desired set point is 1 radian which is this. What we are saying that, set point response FLC without model uncertainty first one and this is my third one. FLC with model uncertainties does very nicely and PD with model uncertainties, you see that, it has some error.

(Refer Slide Time: 55:03)



Finally, in this lecture following topics have been covered. We covered first of all Mamdani type of fuzzy logic controller then, parameter optimization using simple genetic algorithm and UMDA. We demonstrated that, how these two optimization schemes work for robot arm control. But, we give a caution here the adaptive fuzzy logic control is possible only if the offline simulation can be carried out. Today, we gave you some idea about how we do parameter optimization of fuzzy logic controller. In the next class which will be interesting for you, for which we will show you how to generate rules from actual crisp data in a manner that, the fuzzy logic controller is stable. Here, we are optimizing and as long as our performances criteria are limiting then we say system is stable. But, there is no readymade. We do not actually generate rules such that, system is stable. Next fuzzy logic controller the type that we will be describing how to generate rule base using stability concept.

Thank you.