Intelligent Control

Prof. Laxmidhar Behera

Department of Electrical Engineering

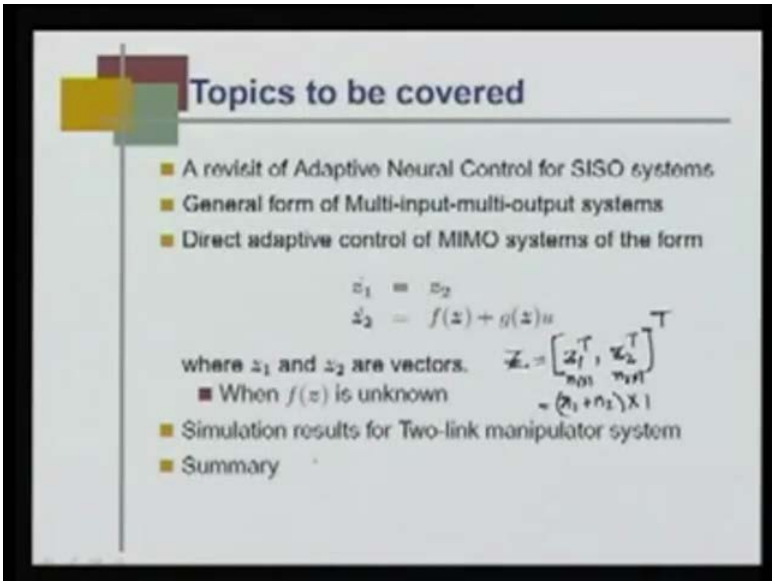
Indian Institute of Technology, Kanpur

Module 3 Lecture 6

Adaptive Neural Control for Affine Systems (MMO)

Today's lecture is on the topic adaptive neural control for affine systems - multi-input multi-output. Last class, we talked about single-input single-output system. We will see how the techniques we learnt in the last class can be extended to multi-input multi-output system. This is the lecture on the module 3 which is neural control and in this course on intelligent control.

(Refer Slide Time: 01:08)

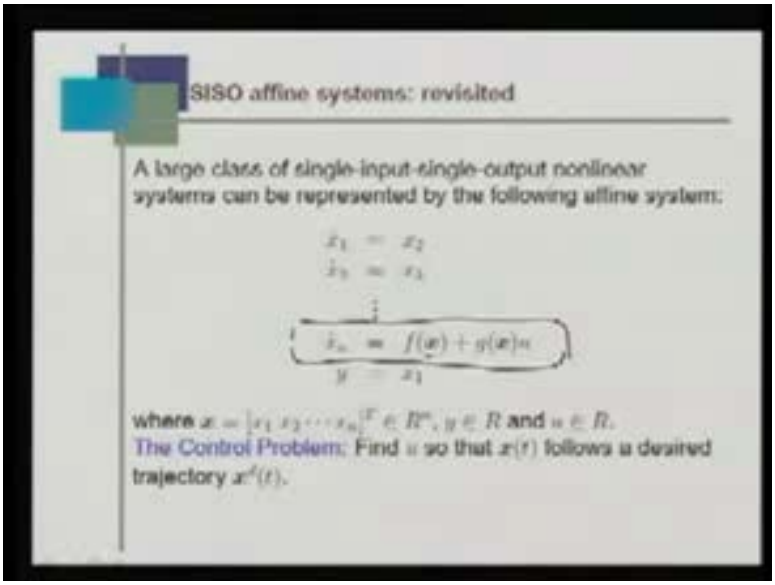


Topics that will be covered today are a revisit of adaptive neural control for single-input single-output system. I just want to refresh your memory with what we discussed in the last class, general form of multi-input-multi-output systems. Then, direct adaptive control of MIMO system of the form. So, we are only considering a specific form of the multi-input multi-output system, where this can be written in the form z_1 dot is z_2, z_1 is z vector and z_2 dot is equal to f (z) plus g (z) u.

Where z_1 and z_2 are vector and of course z is z_1 transpose, z_2 transpose. So, if z_1 is n_1 dimensional vector and z_2 is n_2 dimensional vector, then z is n_1 plus n_2 dimensional vector. We will talk about when f (z) is unknown g (z) is known and when g (z) is unknown, we do not know the solution.

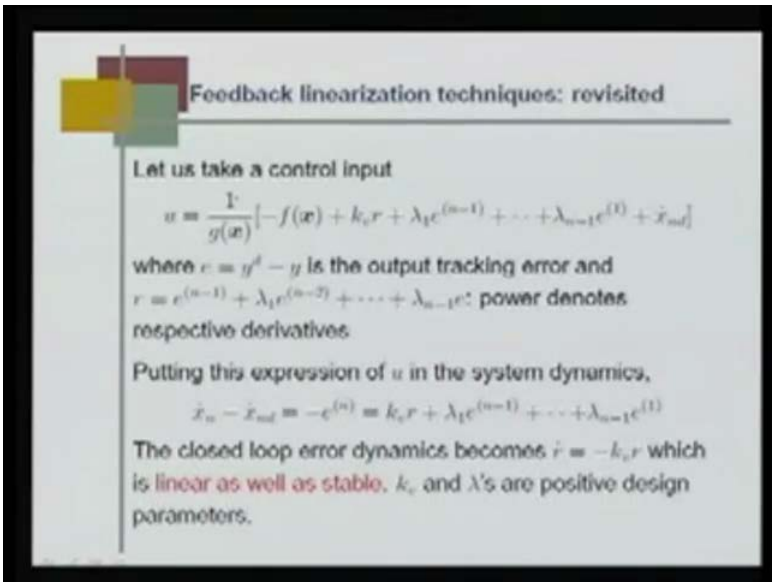
Simulation results for two-linked manipulator system: We will present whatever solution we will get for this kind of non-linear system, where we assume object to be unknown and we will solve the control problem. Finally, the summary, single-input single-output affine systems; we will be revisiting that subject again today.

(Refer Slide Time: 03:07)



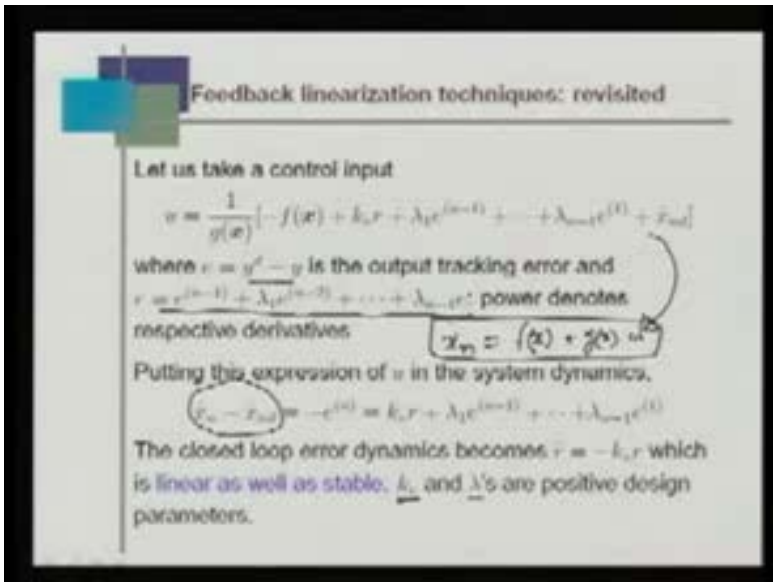
A large class of single-input single-output non-linear system can be represented by the following affine systems: an affine system can be written as x_1 dot is x_2 x_2 dot is x_3 and so on until x n dot is f x, where x is the complete vector consisting of the elements x_1 x_2 x_3 until $x_{n.}$x is x_1 x_2 x_n and plus g x_u u is a singular input u and y are single-input system so belongs to real line. The control problem is find u so that x t follows a desired trajectory x desired.

(Refer Slide Time: 03:48)



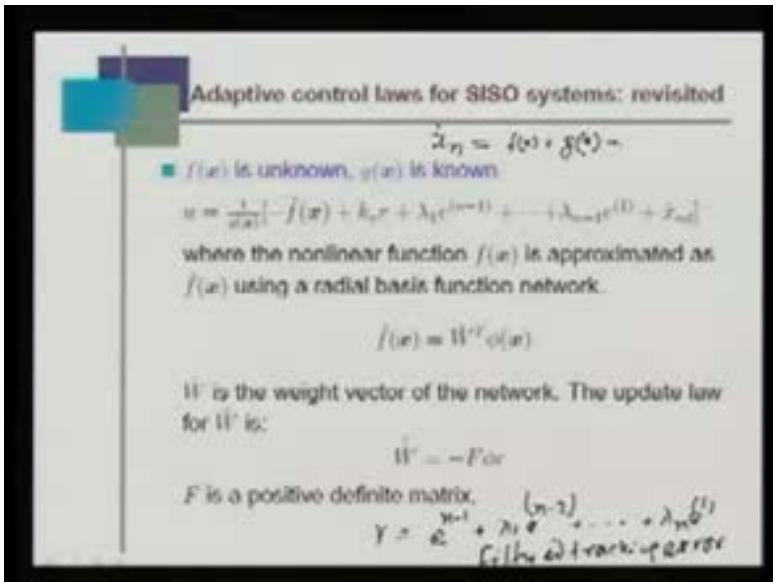
Feedback linearization technique we discussed last class that, in general if I have this particular non-linearity here which we call affine. Then if we select this control law u to be of the form 1 upon g x minus f (x) k_v r lambda 1 e n minus 1 and so on until this is n minus 1th derivative of the error. This is the first derivative of error plus x n desired, desired x_n dot, where e is y d minus 1 is the output tracking error, r is we said filtered tracking error which is n minus 1th derivative of error plus n minus 2, 2th derivative of error and so forth. Power d denotes respective derivatives; putting this expression of u in the system dynamics. System dynamic is simply your x n dot is f (x) g (x) u. If I replace this u here so I get x_n dot this side and this side is in u you have this x_n d dot. When you multiply this u with g (x) you get x_n d dot so bring to this side then by definition this term is minus if you look at e is this one then this particular term by definition becomes minus nth derivative of the error and equal to this side if you look at this minus f (x) will cancel with this f (x).

(Refer Slide Time: 05:58)



Hence this is k_v r plus lambda 1 e n minus 1 until lambda n minus 1 e first derivative. The closed loop error dynamics becomes finally if I look at here, if I take this 1 to this side, then this term will become r by definition here this is my r; so this will become if I take to this side this will become r dot. r dot becomes minus k_v r which is linear as well as stable given k_v and lambdas are positive parameters. These parameters are all positive. This is speed back linearization. What you see is that, if my system is distract by this form, x_n dot is f (x) plus g (x) u and if I select control input is like this and I am able to show that such a controller will stabilize the system. Also tracking will be achieved and the closed error of feedback error dynamics becomes r dot equal to minus k_v r.

(Refer Slide Time: 08:00)

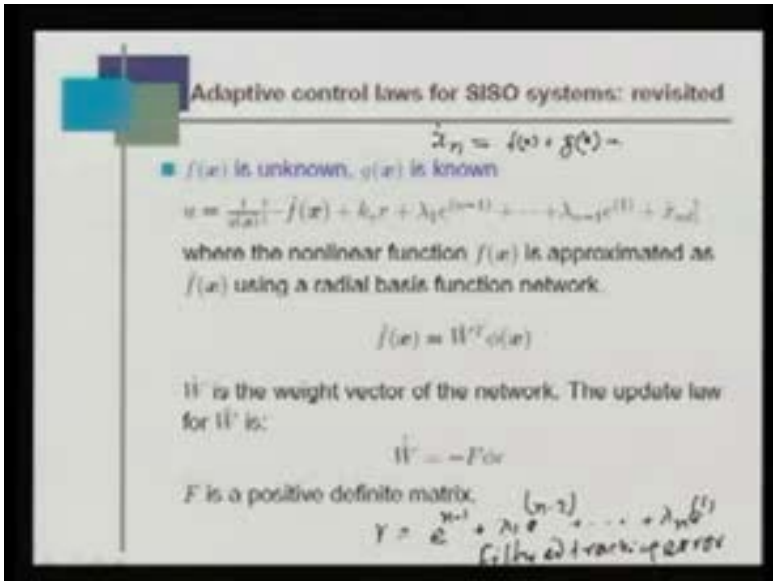


We extended this feedback linearization principle, for adaptive control technique. Why we are doing adaptive control? Because, we assume that in the dynamic which is given x_n dot is f (x) plus g (x) u if this f (x) and g (x) are known, then there is no need for adaptive control.

But most of the situations or in many situations, we do not know what is f (x) and g (x). How do we solve this problem using the same principle, the feedback linearization concept? What we are trying to do here is that, last class we discussed two cases, where 1 is f x is unknown and g x is known and the other is that both are unknown. We are considering the first case now: f x is unknown g x is known. Then if we select the control law which is 1 upon g x, you can see that this form is exactly as that of a feedback linearization control law, but with the exception that instead of f x which is known we have written f hat x which is an estimate of f x. We are saying direct adaptive control because we will not be estimating f (x) using the system identification principle, but we will be identifying f x hat using the principle of tracking error convergence directly. Hence this control technique is known as direct adaptive control. What I am saying here is that, where the non-linear function f (x) is approximated as f hat x using a radial basis function network, f hat x is W transpose phi x; W hat is the weight vector of the network. The update law for W hat is W hat dot is minus phi F phi into r. F is a positive definite matrix, phi is the radial basis functions easing which the unknown function F hat x is is estimated and r is the filtered tracking error which you have already defined; r is e to the

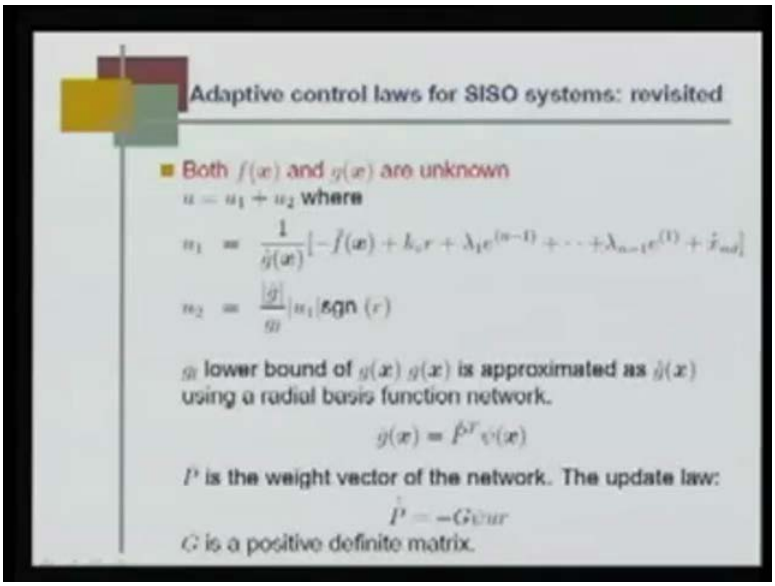
power which is there here - e to the power n minus 1 plus lambda 1. This is our filtered tracking error and using this filtered tracking error we have weight update rule for estimating F hat x.

(Refer Slide Time: 10:18)



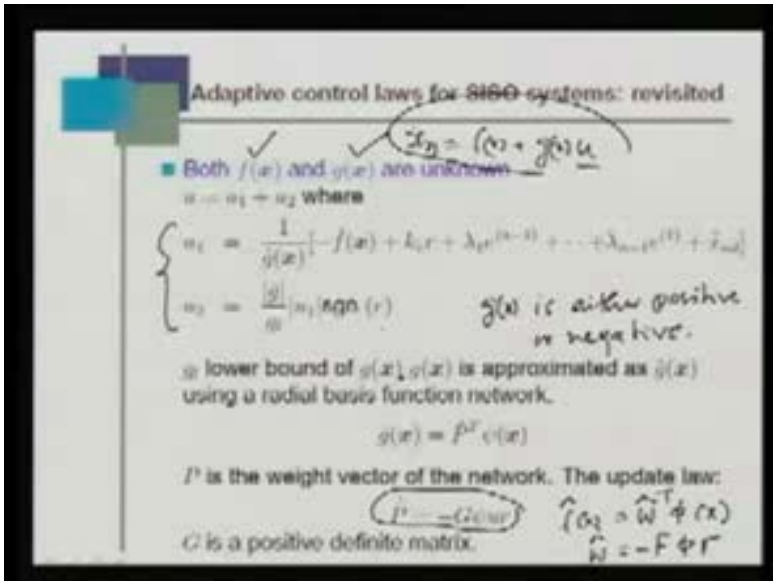
In this method the idea is not to exactly identify what is F hat x but to estimate F hat x in such a way the tracking error is converged. We proved this theorem we are not going to prove this theorem in this class because we have already done it; I am just refreshing your memory.

(Refer Slide Time: 11:05)



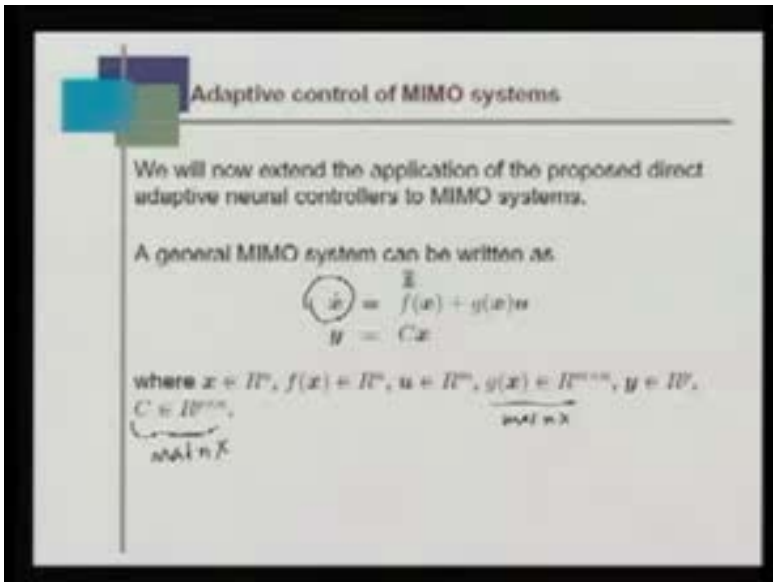
Similarly, we also proposed a control law for when this affine system both f (x) and g (x) are unknown. Again for your memory here I rewrite it as f (x) plus g (x) u so in this the control law is given by u_1 plus u_2; where u_1 is similar structure except that here we have instead of g (x) we have g hat x and here instead of f (x) we have f hat x the estimate of f x because we do not know f (x) and g (x) and u_2 is a sliding mode term which is the absolute value of g hat upon g_l absolute value of u_1 sign um sine r, sine of the filter tracking error. g_l lower bound of g (x).

(Refer Slide Time: 12:07)



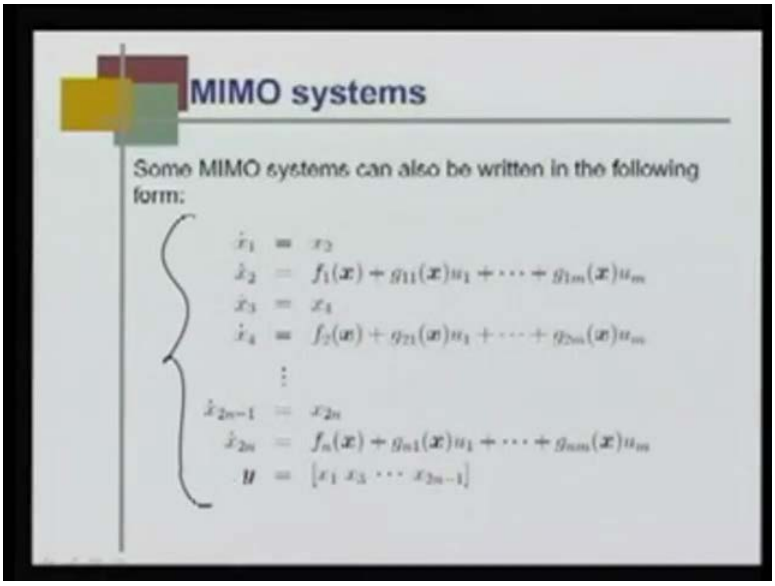
What we are also assuming here g (x) is either positive or negative. g x is approximated as g hat x using radial basis function network; g hat x is P transpose psi x. P hat is the weight vector of the network the update law P hat dot is minus G psi u r and G is a positive definite matrix. Similarly, the weight update law for f (x) which is f (x) hat is W transpose phi x so this is f (x) and the W hat dot is also the same what we derived F phi r. We have two weight update laws; one update law for the weights of the neural network that approximates f (x) and there is another update law here, which is the weight update law for the neural network that estimates g (x). If these are the two update laws, then the control law given by u equal to u_1 plus u_2 will stabilize this non-linear system. Now adaptive control of MIMO system using these two theorems, we have already proved in the last class today we will extend these concepts to MIMO systems.

(Refer Slide Time: 14:21)



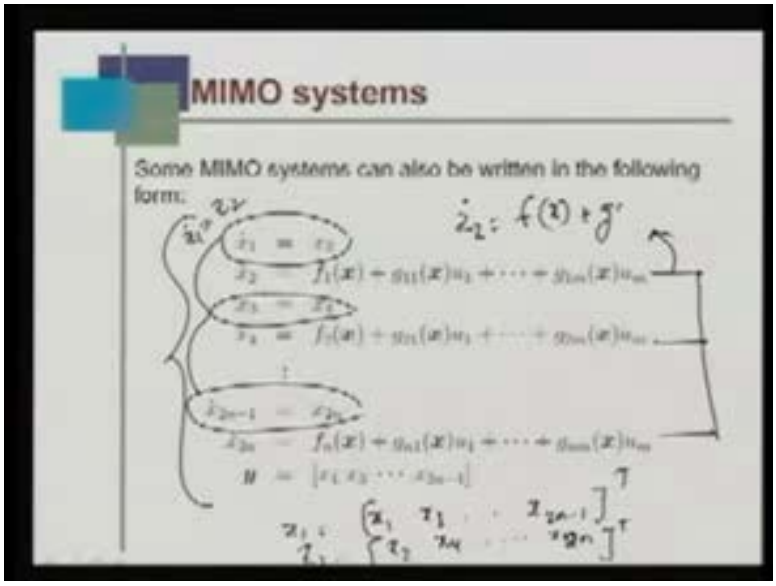
Let us see what a MIMO system is. We will now extend the application of proposed direct adaptive neural control to multi-input multi-output system. A general multi-input multi-output system can be written as x dot is f (x) plus g (x) u and y is Cx. Unlike the earlier case, what we are writing here is, that here x dot is a vector a whole relation f (x) plus g (x) u, where x belongs to the n dimensional vector, f (x) also is an n dimensional vector, u is m dimensional vector, g is m into n dimensional vector, y is p dimensional vector and C is p into m dimensional matrix. In fact, this is a matrix g (x) is a matrix and this is also a matrix. We are talking about a multi-input multi-output system.

(Refer Slide Time: 15:40)



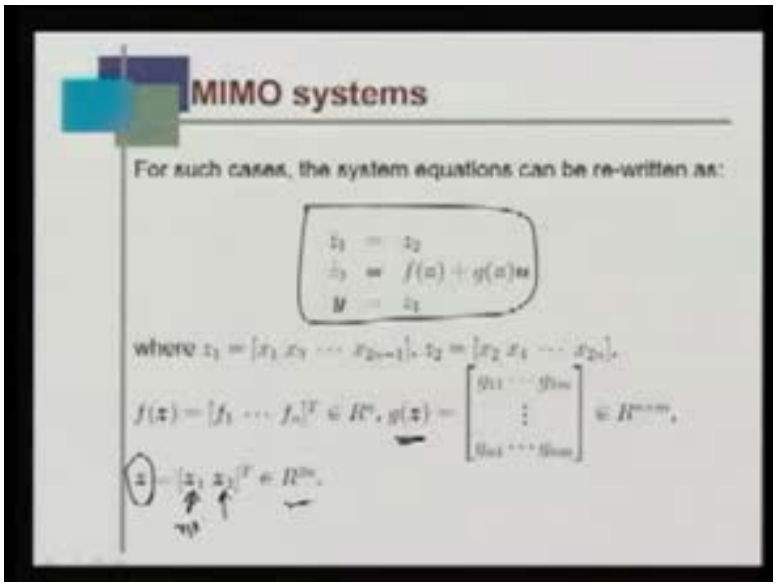
Now, this is a very general form and out of this form we will consider a very specific form which is this one. This structure where this MIMO system we can write any of the system dynamics particularly multi-link robot manipulators. All categories of multiple robot manipulators are two-link or more than two-link robot manipulator. The dynamics can be written as x_1 dot is x_2 dot is f_1 x plus g_{11} u_1 until g_{1n} u_m x 3 dot is x 4 x 4 dot is f 2 x plus g 2 1 x until g_{2m} (x) u_m and finally, $x2_n$ minus one dot is x 2_n and x 2_n dot is f (x) f_n (x) plus g n 1 x u 1 until g_{nm} (x) u_m. So, what you are seeing is that here that we have written is a generic form of a specific multi input multi output system where outputs are x_1 x_3 the odd number state vectors x_1 x_3 until x_2 n minus 1. For such cases the system equation can be rewritten as z_1 dot is z_2 and z_2 dot is f z plus g z u.

(Refer Slide Time: 17:25)



We are clubbing these equations and representing by z_1 dot is z_2 and then obviously we are saying z_1 is x_1 odd number things x_{2n} minus 1 transpose and z_2 is x_2 x 4r x_{2n}. You see that a given 2 n state vectors can be represented as z_1 dot is z_2 and the next one (Refer Slide Time: 18:23) z_2 dot can be written as f (z) plus g (z) u which you can see here, this one this one and this one. So this representation is z_2 dot is f (x) plus g x u. So this is again the n dimensional vector z_2 and $f_x(x)$ is 2 n dimensional and g x also is n into m dimensional, because u is m dimensional.

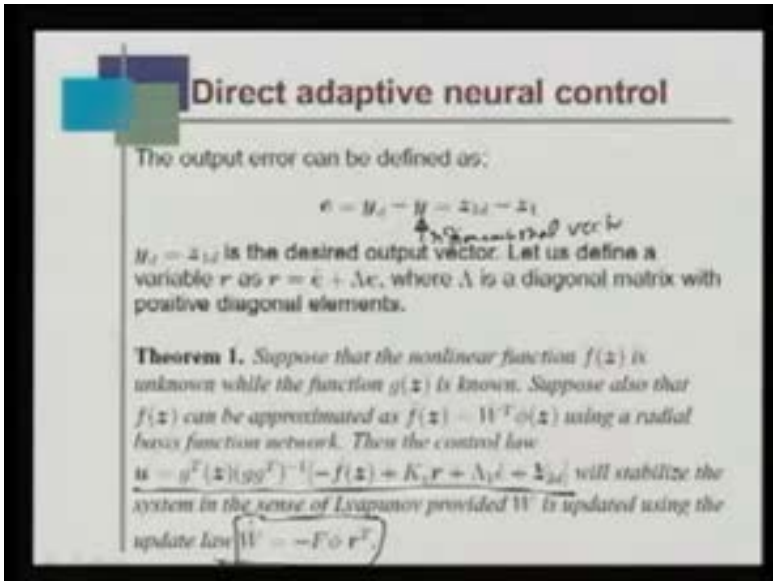
(Refer Slide Time: 19:21)



Here, this is our class of multi-input system; multi-input multi-output system that we will be discussing today. Where z_1 is x_1 x_3 the odd number of state vectors and z_2 is even number state elements. f (z) is f_1 until f_n transpose and g (z) you can easily see g (z) is g_{11} so this is (Refer Slide Time: 19:42) g_{11} until g_1 m $g_{2\,1}$ until g_2m g_n 1 until g_n m.

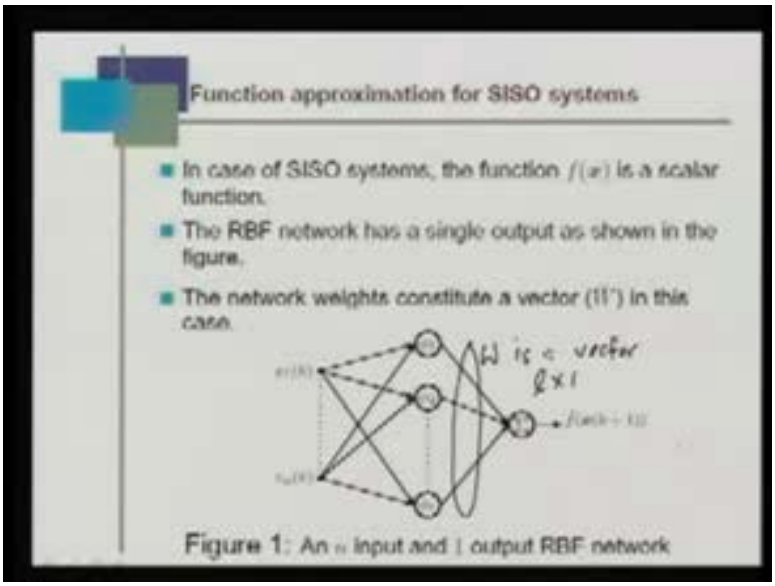
This is the elements gather that forms g (z) and then your final state vector z is actually 2_n dimensional where each one here is n dimensional.

(Refer Slide Time: 20:06)



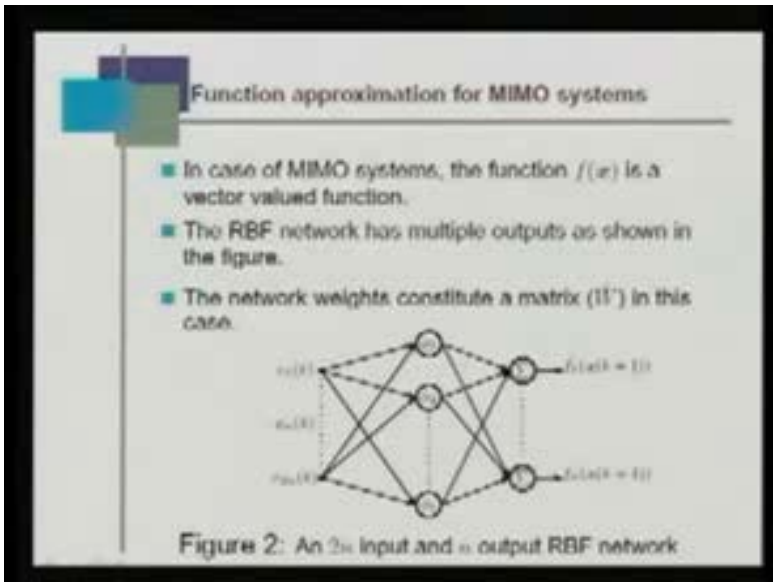
Direct adaptive neural control, the output error can be defined as e is y d minus y in the as you know that y is again n dimensional vector because we are assuming that this is z_1 d desired minus z_1 since z_1 is the n dimensional vector so this is valid. y d is z_1 d is the desired output vector. Let us define a variable r which is again in the form of a filtered tracking error. r is e dot plus lambda e where lambda is a diagonal matrix with positive diagonal element. Theorem one - suppose that the non-linear function f (z) is unknown while the function g (z) is known, suppose also that f (z) can be approximated as f hat z upon equal to W hat transpose phi z using a radial basis function network then the control law given by this expression u is g transpose into g g transpose inverse whole multiplication minus f hat z plus k_vr plus this term into e dot plus z_2 d dot so z_2 dot. The desired one will stabilize the system in sense of Lyapunov provided W hat is updated using the update law minus F phi r transpose. You see that in here in-single-input single-output case r was a scalar and in this case this is a vector n dimensional vector.

(Refer Slide Time: 22:32)



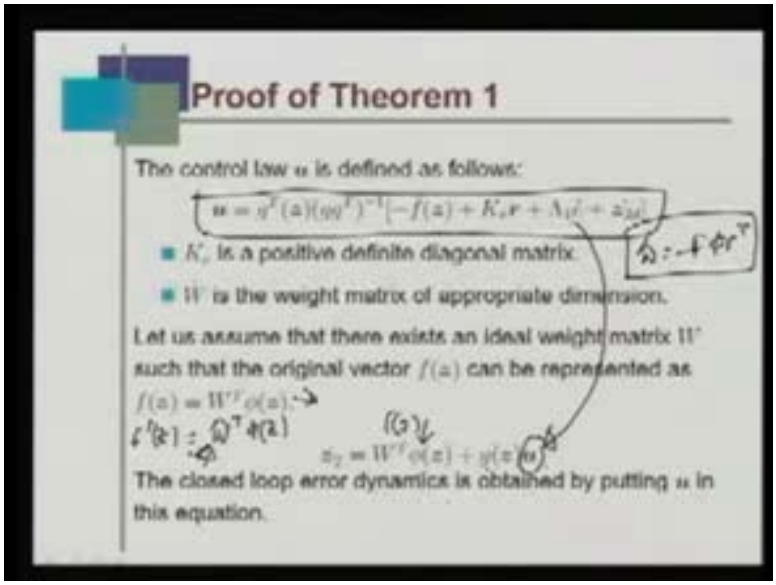
You see that when we did with using single-input single-output system we had a radial basis function network that was estimating what is f (x). If you see the weight vector here which is W this is a vector and how many dimensions? Dimension is l into 1 dimensional vector W. In case of single-input single-output system the function f (x) is a scalar function the R B F network as a single-output as shown in the figure. The network weight constitutes a vector W hat in this case while when we do it in multi-input multi-output system.

(Refer Slide Time: 23:25)



We have n outputs here. In this case this is no more a vector W hat is a matrix. What is the meaning of this W hat here? In case of MIMO system the function f (x) is a vector valued function. The R B F network has multiple outputs as shown in the figure. The network weights constitute a matrix W hat in this case. Now, we will go to the proof with this basic idea we have already actually described.

(Refer Slide Time: 23:55)

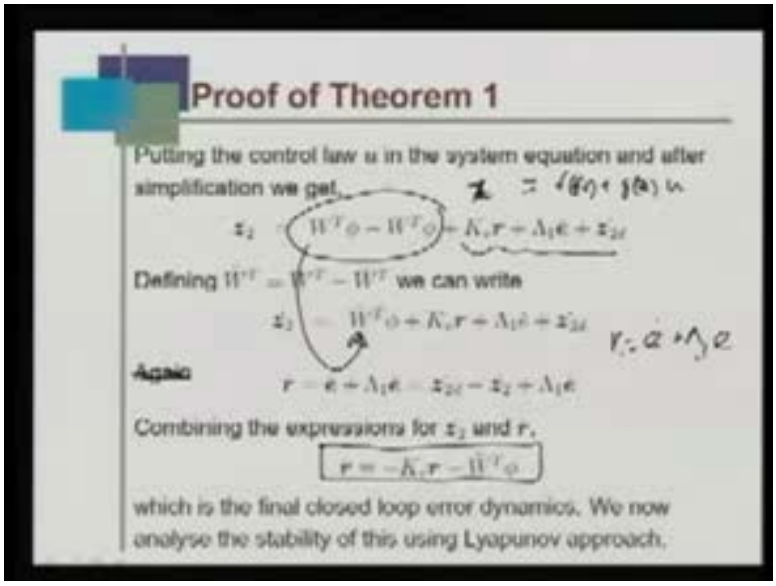


We described that this control law, with the weight update algorithm for F hat z which is W hat dot is minus F phi r transpose. If I have this weight update law, for estimating F

hat z then, this controller will give me the results that the actual output will follow the desired output.

In this control law k v is a positive definite diagonal matrix W hat is the weight matrix of appropriate dimension. Let us assume that there exists an ideal weight matrix W such that the original vector f (z) can be represented as W transpose phi z. We can say here f (24:54) (z) is W hat transpose phi z for this W hat as to be updated. We have already given this rule of W hat dot is minus F phi r transpose so can we say that given this as the weight update law these controller will stabilize the given affine system. Now what I do is that my dynamic is because f (z) is this one so z 2 z is f (z). This f z is now W transpose phi z plus g (z) u in this u is replaced by this equation. The closed loop error dynamics is obtained by putting u in this equation.

(Refer Slide Time: 25:52)

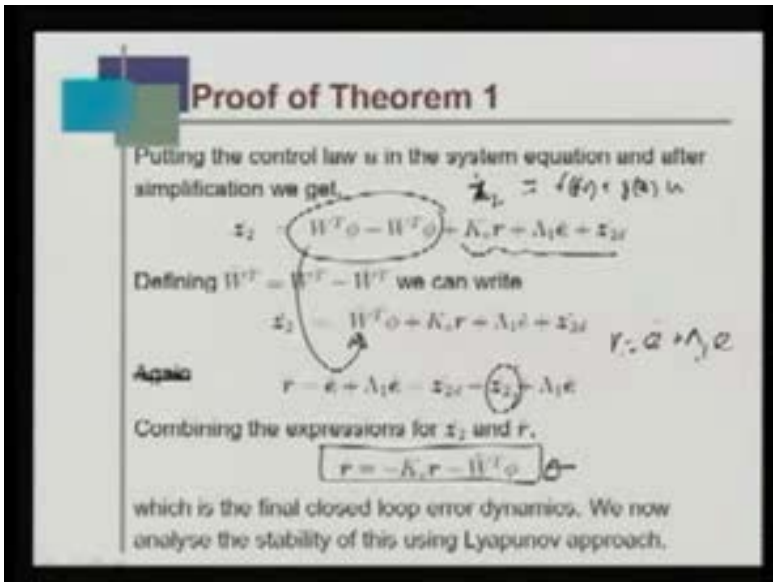


This is putting the control law u in the system equation and after simplification we get z_2 dot is W transpose phi minus W hat transpose phi plus k v r plus lambda 1 e dot plus z_2 dot. If f x would have been known, then these two terms would have cancelled. Then we would have remained with only this term which is a stable closed loop error dynamics. Since these two are not exact, they are different. How do we stabilize it?

We have proposed a control law for weight update. Defining the error in weight vector is… earlier I told that in this case W is a matrix so this is very important and here W is actually matrix; defining the weight matrix W tilde is according to this. We can write z_2

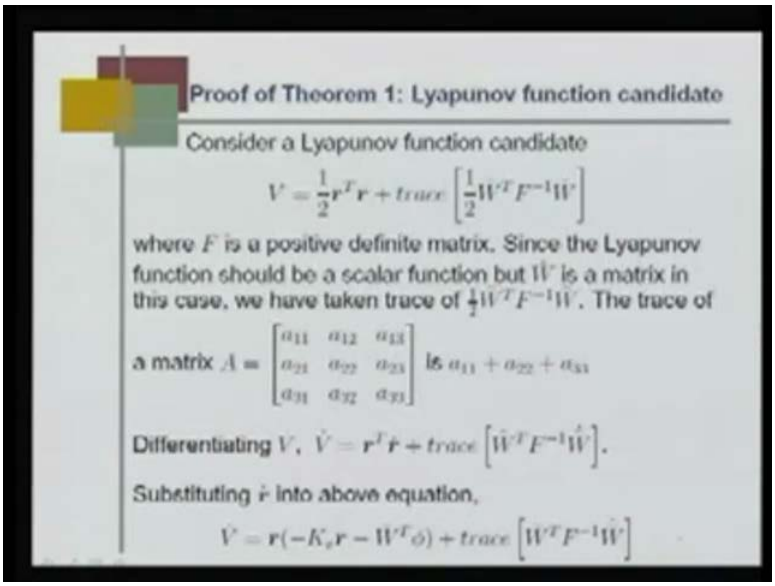
dot this particular term is W tilde transpose. This is written as W tilde transpose phi plus this 3 terms k_v r plus this term into e dot plus z dot 2 desired. We have already defined r dot. r dot we know, we have defined r to be e dot plus so if I re-compute r dot is e double dot plus 1 into e dot. The symbol this is simply a constant into 1 suffix 1 e dot which is z_2 dot minus z_2 dot plus 1e dot. This particular expression we derive from r dot simply differentiating then replacing, in this case what is e double dot. So e double dot is simply z_2 d dot minus z_2 dot. Combining this expression we get the close error dynamics is r dot is minus k_vr minus W tilde transpose phi which is the final closed loop error dynamics. We now analyze the stability of this using Lyapunov function so what we get is that this is our closed loop error dynamics. By replacing u in our system dynamic which is f ($z_{2)}$ plus g (z) u is z_2 dot. This is our dynamics we refer in this dynamics here z_2 dot here, then I get this expression.

(Refer Slide Time: 29:50)



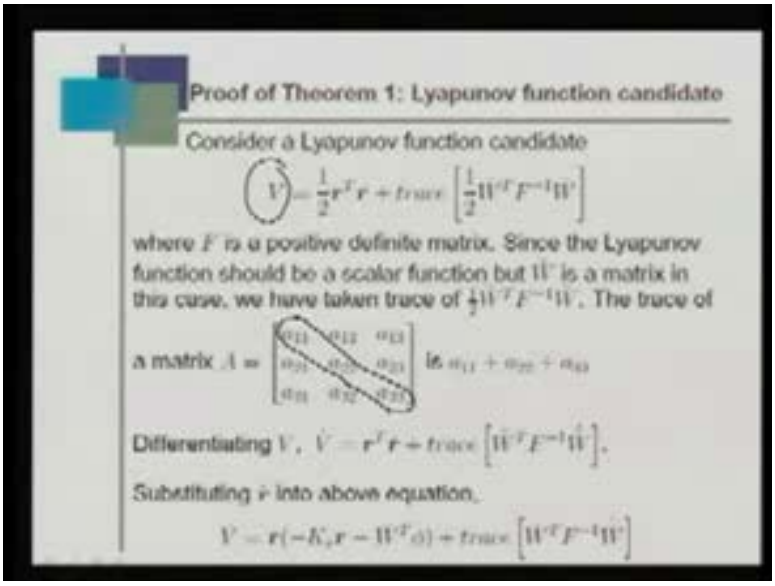
This (Refer Slide Time: 29:50) is our closed loop dynamics r dot is minus k_v r W tilde transpose phi.

(Refer Slide Time: 29:55)



Now consider a Lyapunov function candidate V is half r transpose r plus trace half W tilde transpose F inverse W tilde F is the positive definite matrix. Since the Lyapunov function should be a scalar function but W hat is a matrix. In this case we have taken trace because earlier we simply wrote this is W tilde transpose F inverse W tilde. But now we have taken trace because of the matrix. Because this Lyapunov function has to be scalar.

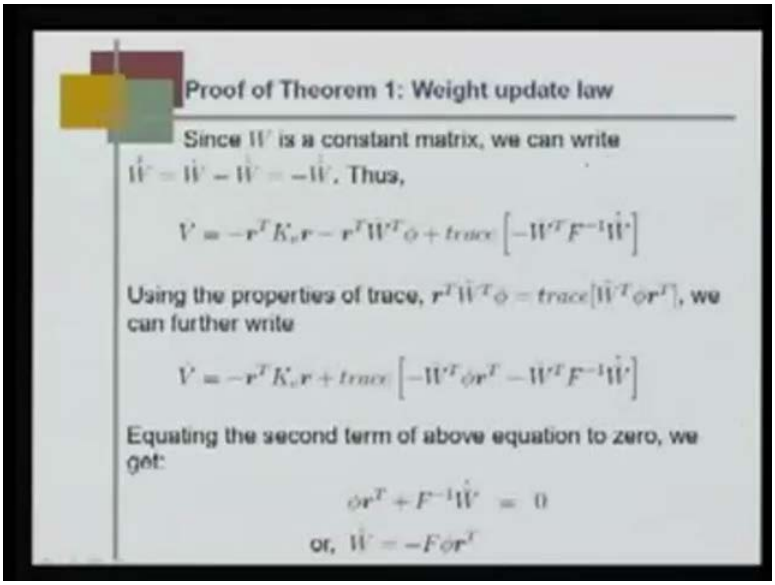
(Refer Slide Time: 30:44)



Trace of a matrix those of you do not know, if I take a 3 dimensional matrix the diagonal elements is given by a 1 1, a 2 2, a 3 3. Trace of a matrix is a 1 1, plus a 2 2 ,plus a 3 3. Differentiating V, V dot which is r transpose r dot plus trace of W tilde transpose F inverse W tilde dot. Substituting r dot into above equation, so this r dot which is the closed loop dynamics this is my r dot (Refer Slide Time: 31:23). If I give this equation or if I replace this equation in V dot which is r transpose r dot, we may get a nice solution for this which we have derived now.

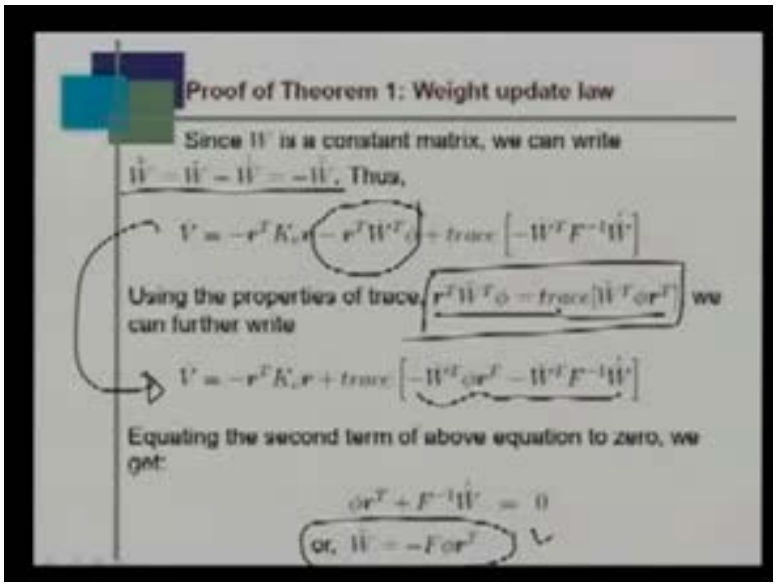
You can see that r transpose r dot plus trace of this thing with W tilde dot. Substituting r dot into the above equation in this equation you get the rate of Lyapunov function to be r minus k_v r minus W tilde transpose phi plus the trace of this particular term.

(Refer Slide Time: 32:10)



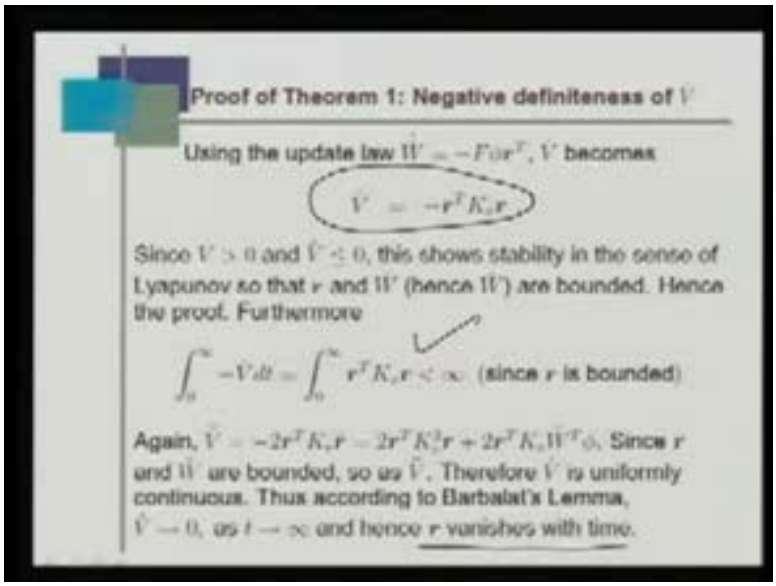
Since W is a constant matrix we can always write W tilde dot to be minus W hat dot. V dot, which we have already computed to be minus r transpose k_v r minus r transpose W tilde transpose phi plus trace minus W tilde transpose F inverse W hat dot.

(Refer Slide Time: 32:40)



By taking this example, using the properties of trace we will utilize this theorem, which says, r transpose W tilde transpose phi is trace of W tilde transpose phi r transpose. We can further simplify this V hat V rate derivative of the Lyapunov function to minus r transpose k_v r trace. This is our trace; this particular thing has been replaced by this here trace W tilde transpose phi r transpose. This is trace and this is also another trace so you can write trace is minus W transpose phi r transpose minus W tilde transpose F inverse W tilde W hat dot. Equating the second term of the above equation to 0, this one we want to eliminate, I can take out W tilde transpose to the left side as a common, then I get simply phi r transpose; you see that phi r transpose plus F inverse W hat dot is 0 or this is my weight update law W ha dot is minus F phi r transpose. If this is my update law then direct adaptive control for non-linear systems is solved.

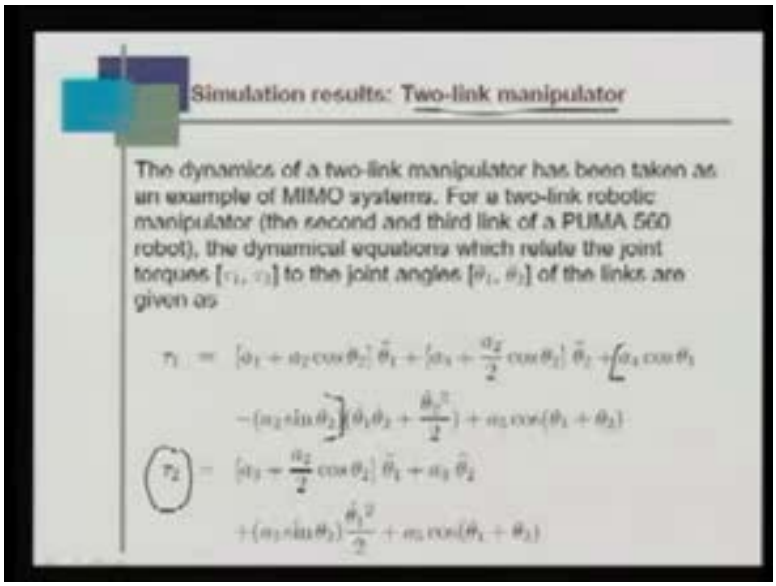
(Refer Slide Time: 35:31)



Further the proof of the theorem again using the update law W hat dot which we just derived minus F phi r transpose. Now let us see whether the algorithm is convergent rate derivative Lyapunov function becomes according to our definition… this V dot becomes V dot is minus r transpose k_v r. Since V is greater than 0 and V dot is less than equal to 0 this shows stability in the sense of Lyapunov so that r and W tilde are bounded hence the proof. Furthermore, you can easily check this one. Again V double dot if I have found V dot then V double dot is minus 2 r transpose k_vr dot is 2 r kv square because r dot is minus k_v r. This was negative this becomes positive plus 2 r transpose k v W tilde transpose phi; because this r dot is replaced by…

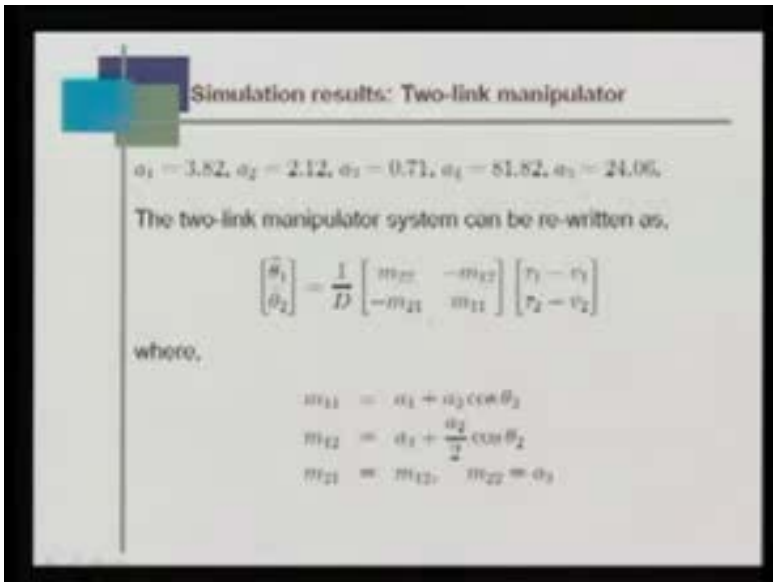
Since r and W tilde are bounded as V double dot double differential of the Lyapunov function. Therefore, V dot is uniformly continuous. Thus according to Barbalat's Lemma, V dot tends to 0 and t tends to infinity; hence r vanishes with time. We showed here (Refer Slide Time: 36:15) by saying that this is my update law, I found out this is my rate derivative of Lyapunov function which is negative definite which is always negative for the values r not equal to 0 when r is equal to 0; this is 0. Then further we are showing that V dot is uniformly continuous considering this term. Thus according to Barbalat's Lemma if this is continuous, then this term will go to 0 as t tends to 0. So r would vanish with time and what is r? If you look at here r (Refer Slide Time: 37:03) we have defined r is this particular term which is the filtered tracking error. So finally error will converge to 0.

(Refer Slide Time: 37:19)



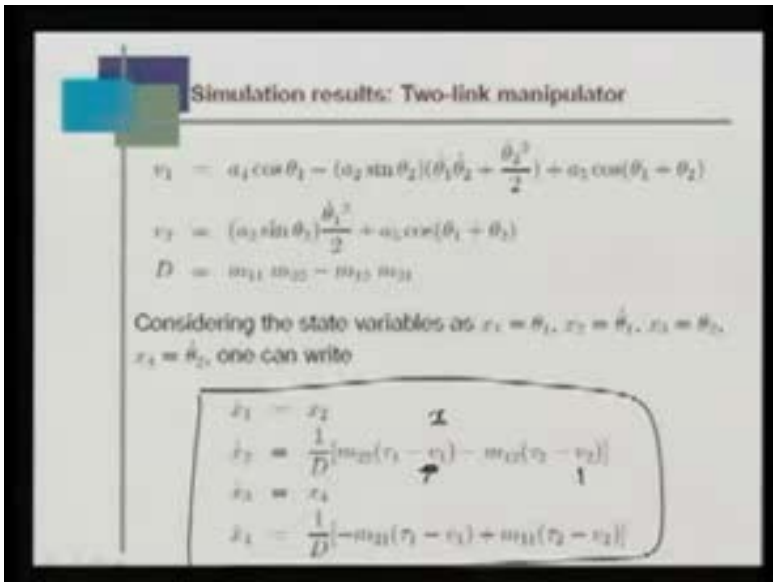
Now we will apply this particular application to this adaptive control theory to an actual system in simulation and we take two-link manipulator. The dynamics of a two-link manipulator has been taken as an example of multi-input multi-output systems. For a two-link robotic manipulator the second and third link of a PUMA 560 robot that we have taken. The dynamical equation which relate the joint torques tow 1 and tow 2 to the joint angles theta 1 theta 2 of the links are given as where tow 1 is a 1 plus a 2 cos theta 2 theta 1 double dot plus a_3 plus a_2 by 2 cos theta 2 so this is the coefficient of theta 2 double dot plus a 4 cos theta 1 minus a 2 sine theta 2; this whole term multiplication with this term. The joint torque 1 is related with the joint velocities acceleration velocity which is the co-relate force theta 1 dot theta 2 dot plus theta 2 dot whole square by 2 and the gravity term F i cos theta 1 plus theta 2. Similarly, the joint torque in second joint which is a 3; this is the acceleration term this is co-relation term and this is your gravity term.

(Refer Slide Time: 39:08)



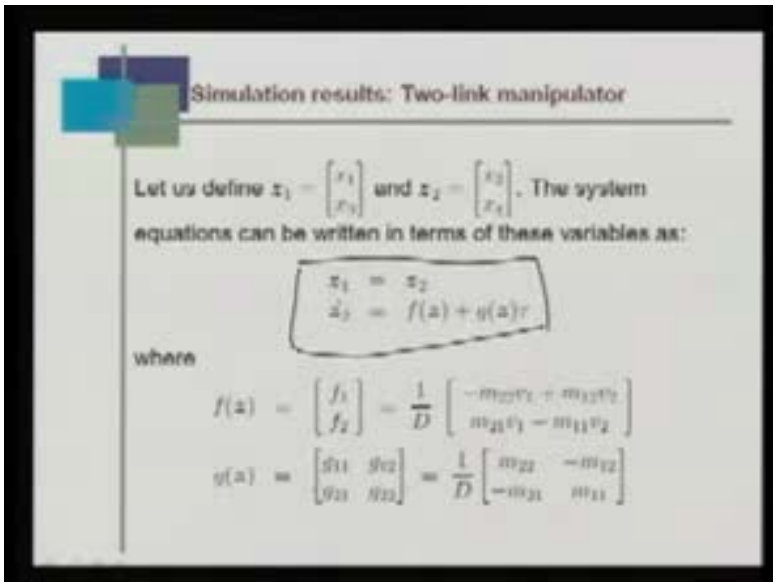
Where a_1 is 3.82, a_2 is 2.12, a_3 is 0.71, a_4 is 81.82, a_5 is 24.06. The two-link manipulator system can be re-written as (Refer Slide Time: 39:24) theta 1 double dot is something and theta 2 double dot is something. But you see that each equation is written in terms of theta 1 double dot and theta 2 double dot. So I have to eliminate and then I have to find out the expression for theta 1 double dot. Similarly, I have to find the expression of theta 2 double dot using other terms. Doing that what you are getting theta 1 double dot theta 2 double dot equal to 1 upon D m_{22} minus m_{12} minus m_{21} m_{11} tow 1 minus v_1 tow 2 minus v_2 where, m_{11} is given by this expression, m_{12} is given by this expression, m_{21} is m_{12} and m_{22} is a_3.

(Refer Slide Time: 40:21)



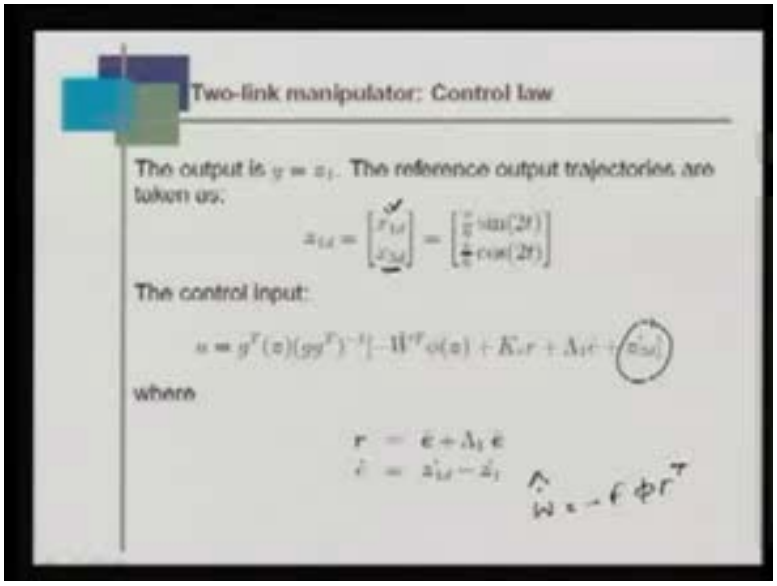
Where v_1 is given by this expression this big expression which is a_4 cos theta 1 minus a_2 sine theta 2 theta 1 dot theta 2 dot plus theta 2 dot whole square by 2 plus a_5 cos theta 1 plus theta 2. Similarly, you can see that d_2 here another expression a_2 sine theta 2 theta 1 dot whole square upon 2 a phi cos theta 1 plus theta 2 and D is m_{11} m_{22} minus m_{12} m_{21}. This is m_{11} m_{22} minus m_{12} m_{21} so m_{11} m_{22}, diagonal element multiplication minus the other diagonal element. Considering the state variable as x_1 is theta 1 x_2 is theta 1 dot and x_2 is theta 2 x 4 is theta 2 dot. One can write x_1 dot is x_2 and x_2 dot is 1 upon D and then this quantity m_{22} t 1 tow 1 minus v_1 minus m_{12} tow 2 minus v_2 and similarly x_3 dot is x_4. Similarly x dot 4 is in this particular format 1 upon D minus m_{21} tow 1 minus v_1 plus m_{11} tow 2 minus v_2. You see that we said in the beginning of the class that some of the system can be represented in this particular form. You see that this v_1 is function of the state vector x and similarly v_2 also is a function of state vector x this is also function of state vector. You see that v_2 is function of theta 1; theta 2 theta 1 dot, v_1 is theta 1 theta 1 dot theta 2 dot and theta 2. All the states are there here one state is missing. But anyway in general they can be a function of this entire state vector.

(Refer Slide Time: 42:34)



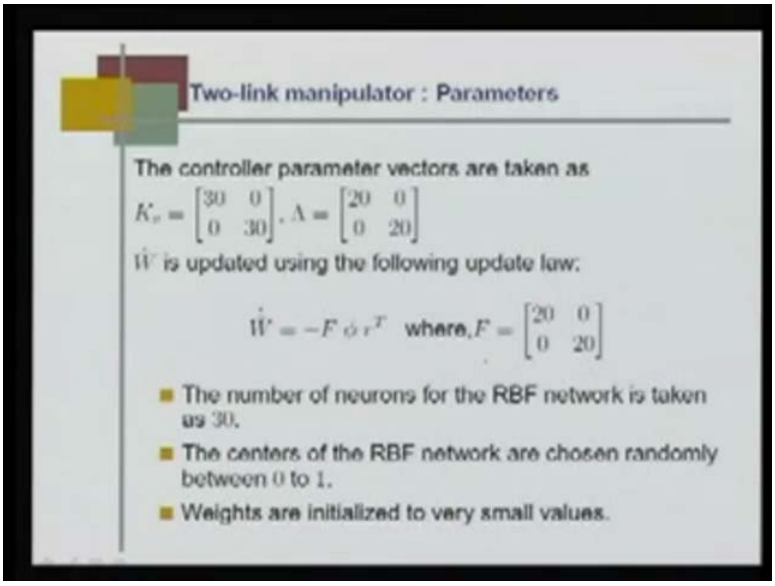
This can be represented if we define z_1 to be x_1 x_3 and z_2 is x_2 and x_4 then the system can be written in our general notation that we talked which is z_1 dot is z_2 and z_2 dot is f (z) plus g (z) tow and going back and reformulating the problem we get f (z) is f_1 f_2 which is 1 upon D and this is my first term minus m_{22} v_1 plus m_{12} v_2 and the second term is m_{21} v_1 minus m_{11} v_2 and g z is a 2 by 2 matrix $g_{11,}$ $g_{12,}$ $g_{21}$and g_{22} and this is 1 upon D m_{22} minus, m_{12} minus m_{21} and m_{11}.

(Refer Slide Time: 43:30)



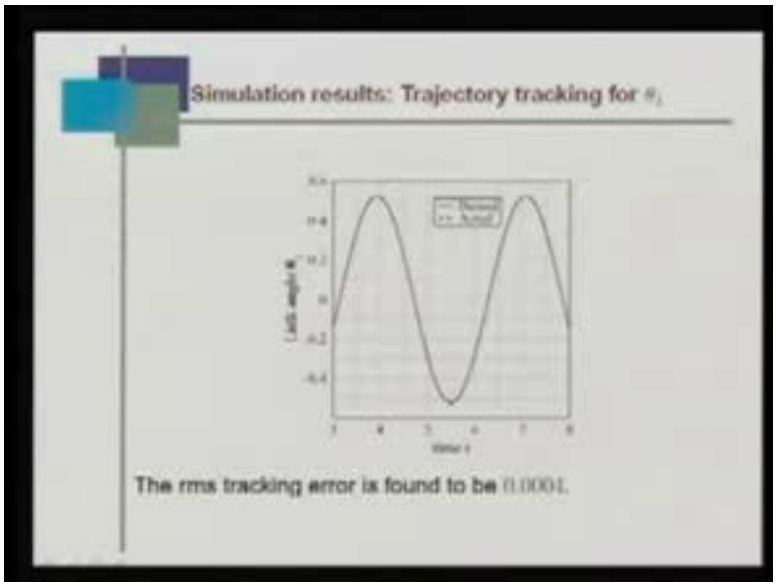
Output y is z_1 and z_1 is our first link position and second link position. The reference output trajectories are taken as which is a z_1 d is x_1 d and x_3 d is pi, so this is a angular position of the joint 1 and angular position of the joint 2. pi by 6 sine 2 t pi by 6 cos 2 t the control input which is ug transpose z g g transpose inverse minus W hat transpose phi z plus k v. This term and this the desired trajectory z_2 d dot so where r is our tracking error e double dot plus lambda 1 e dot and e dot is z_1 e dot minus z_1 dot. This is our control law we have already said this control law with the weight update law for W hat will stabilize. The weight update law is we have already seen F phi r transpose. This weight update law would surely stabilize the system.

(Refer Slide Time: 45:14)



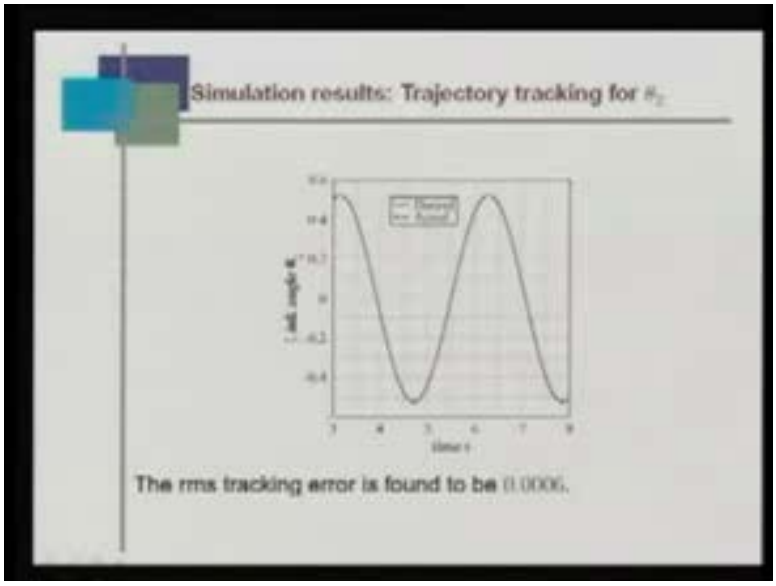
We have selected kv is 30, 00, 30 which is a 2 by 2 matrix and lambda is 20, 20, 00. W hat is updated using the following update law, minus f phi r transpose, where f is taken as 20, 00, 20, the number of neurons for the radial basis network function is taken as 30, the centers of radial basis function networks are chosen randomly between 0 and 1.

(Refer Slide Time: 45:59)



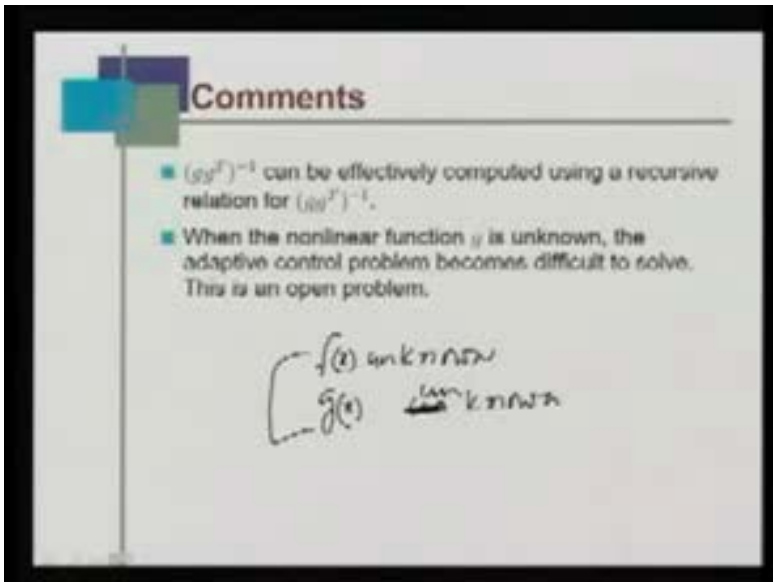
Weights are initialized to very small values. If we do that the simulation results would show the trajectory tracking force theta 1 is desired and absolutely no difference. Tracking is so perfect so we see that we have achieved this tracking and the RMS tracking error is found to be 0 point 0004 assuming f (x) to be unknown. Of course, in the initial period we are not showing instead when the trajectory has settled, once the transients are gone in their steady state that tracking is perfect because of very small value 0 point 0004. This is a desired perfect tracking.

(Refer Slide Time: 46: 49)



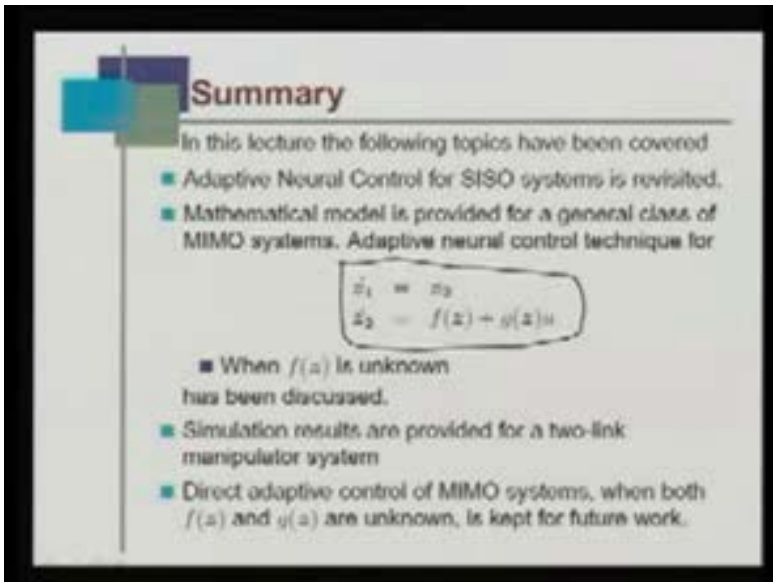
Similarly, here this is a trajectory tracking for theta 2. Again this is link angle theta 2 and according to time and steady state from time 3 to 8, if we compute the RMS tracking error this is 0 point 0006 and very efficient tracking. Correspondingly, the controlled torque tow 1 and tow 2 that were found out to be like this that you can see again in steady state the controlled torque is very smooth without any kind of fluctuations. This implies that the proposal algorithm is very efficient. If we go back to the control law, we have this g, g transpose mac inverse; you know that this is a matrix. In this case the g is two-dimensional matrix, so gg transpose is again two-dimensional matrix. But in general, if I have n link matrix; if it is two-link then it is two dimensional. If I have a six link matrix it is a 6 by 6 inverse matrix. Inverse means computation is more; this point gg transpose can be computed using a recursive relation. One can work out on this that instead of doing this inversion we can find a recursive solution for it. That this easily be computed.

(Refer Slide Time: 48:50)



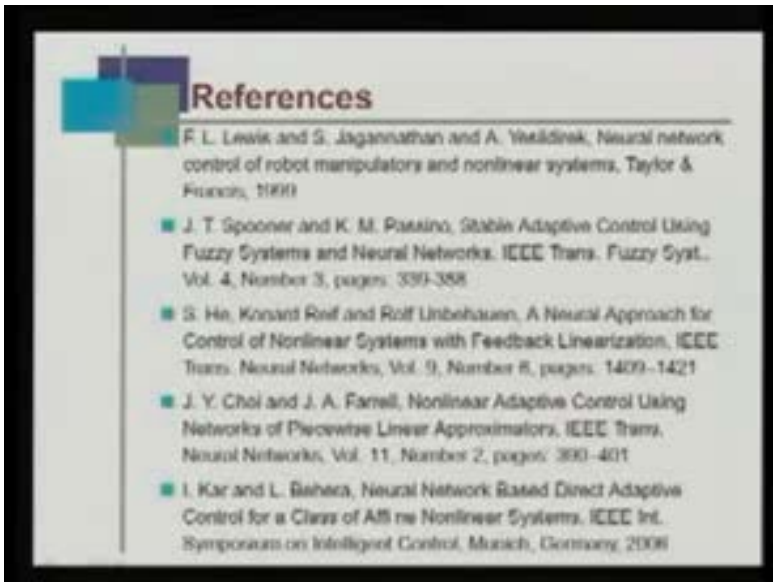
Second thing is when we consider f (x) is unknown and g (x) is known, if g (x) is also unknown or both are unknown this problem is still not solved in the control literature. Non-linear function g is unknown the adaptive control problem becomes difficult to solve. This is an open recursive problem.

(Refer Slide Time: 49:56)



In the summary, the following topics have been covered: adaptive control system for single-input and single-output systems is revisited, mathematical model is provided for general classes of multi-input multi-output system, where we could represent even practical systems like multi-link robotic manipulators. They can also be represented in this particular format: z_1 dot z_2 z_2 dot z_u plus g (z) u, and f (z) is unknown then this solution to this control problem is already provided but we found that in this control law includes an inversion which must be replaced by a recursive solution. Simulation results are provided for a two-link manipulator system where we saw the tracking order is in the range of 10 to the power minus 4, implying tracking is very perfect. Direct adaptive control of multi-input multi-output system, when both f (z) and g (z) are unknown is kept for the future work. This problem is still not solved.

(Refer Slide Time: 51:09)



Those who are further interested to work on this problem, I would like that you can follow these references in the Lewis and Jaganathan and Woodwreck, neural network control of robot manipulators and non-linear systems. A book published by Taylor and Francis in 1999. Spoonor and Passino they have a paper on Stable Adaptive control and Fuzzy systems and Neural Networks, S He Konnald Reif and Rolf have published A neural approach for the control of Non-linear systems with Feedback linearization, Choy and Farnell have published Non-linear adaptive control using networks of linear approximates volume 11. This is our paper; Indrani Kar and I have published Neural Network Direct adaptive control for all non-linear systems.

Thank you very much.