Intelligent Systems and Control

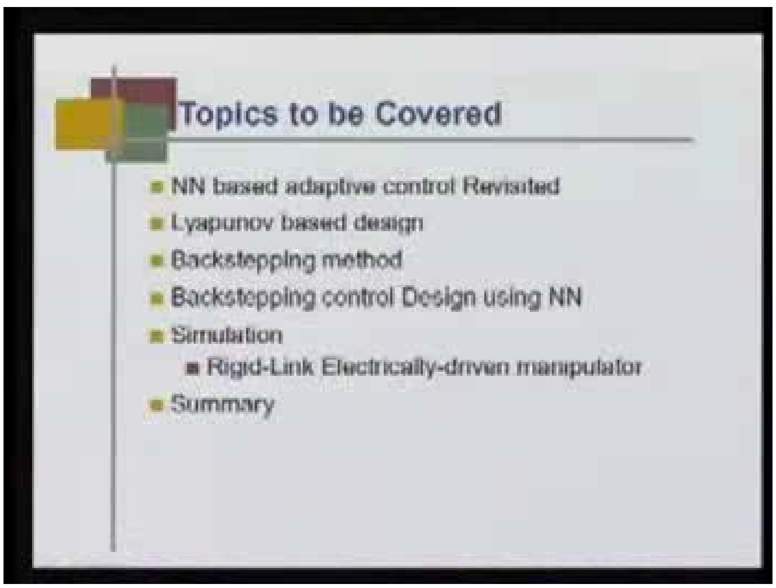
Prof. Laxmidhar Behera
Department of Electrical Engineering
Indian Institute of Technology, Kanpur

Module - 3 Lecture - 10

NN based Back stepping control

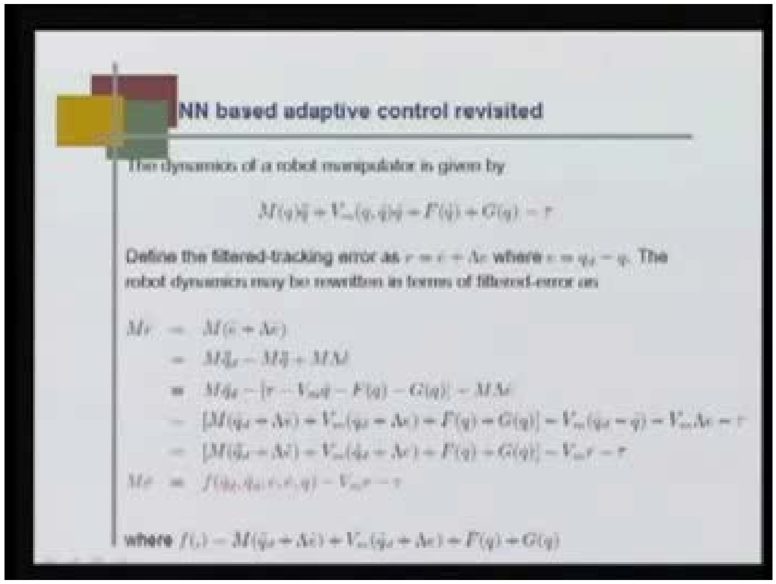
NN based back stepping control - This will be the topic that we will be discussing under module three, neural control, it is lecture ten. NN based back stepping control. In the last class, we learnt a very preliminary idea about neural network based control. Now, we will introduce a new concept called back stepping control using neural network.

(Refer Slide Time: 01:08)



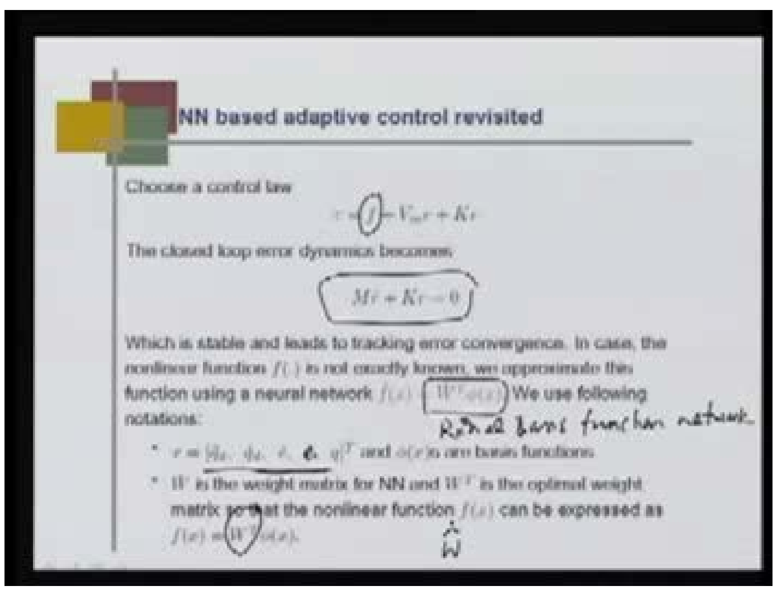
Following are the topics to be covered: NN based - Neural Network based adaptive control; Lyapunov based controller design; back stepping method and back stepping control design using neural network. We will apply this method to a rigid link electrically driven manipulator through simulation and final conclusion.

(Refer Slide Time: 01:35)



We already discussed this topic in the last class, Neural Network based adaptive controller.

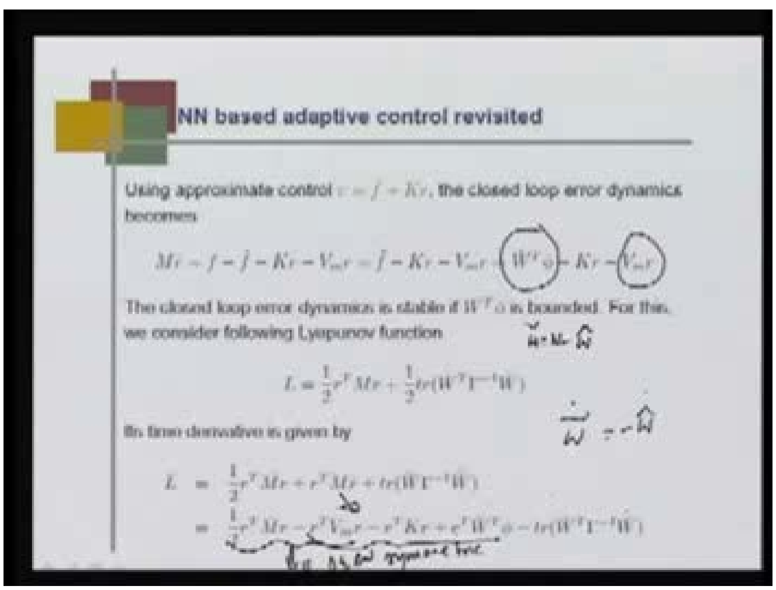
(Refer Slide Time: 01:46)



I will just give you little hint. For this if I select tau equal to f plus V_mr plus Kr. The closed loop error dynamics is Mr dot plus Kr equal to 0 which as well as stable dynamics, because M is always a positive definite matrix. So this is a stable dynamics, provided K is properly selected. You can easily see, in principle this actually is a computed torque control. But to implement this controller f must be known; f means this quantity; this quantity (Refer Slide Time: 02:37) must be known, but unfortunately we do not have any many parameters with us to compute this exactly.

Hence, we estimate f by f hat x. We approximate this as a radial basis function as a canon in which, W is the weight vector and phi x is the basis function. This phi x is the basic function of these quantities: q_d double dot q_d dot e dot and q. You can also easily see this quantity is a function of q_d double dot e dot q_d dot e and q. Thus exactly is here, q_d double dot q_d dot e dot e and q and phi x are the basis functions. This can be represented in terms of a radial basis function network. We have already discussed in neural network model f(x). W is the weight matrix for the neural network and W transpose is the optimal weight matrix so that the nonlinear function is f(x). It can be expressed as W transpose phi x. Now, the objective is that, we know what is phi x basis functions, because we know the input, but we do not know what is W transpose? We have to update the weights in such a way, we have to find out what is this W hat dot? Such that, we can again establish the system is stable so that we always do using a Lyapunov function method.

(Refer Slide Time: 04:50)

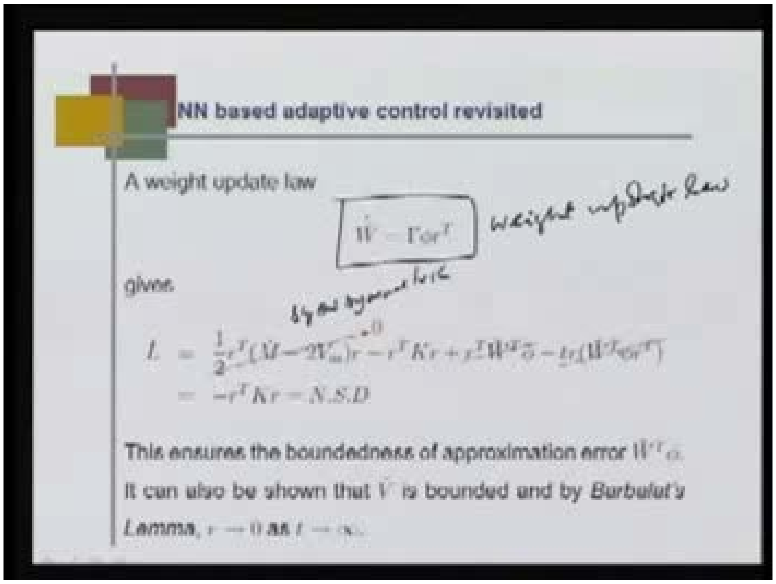


What you do now? Instead of tau equal to f plus Kr we write tau equal to f hat plus Kr you see (Refer Slide Time: 05:02) that tau is f V_mr is missing here. We also assume V_m is also unknown; in that case, tau is f hat plus Kr. The closed loop error dynamics is Mr dot is f minus f hat minus Kr minus V_mr, which is this quantity f hat minus Kr minus V_mr. Earlier we used to have tau is f hat Kr plus V_mr. In that case, V_mr used to cancel out. If f and f hat they are exact, then, simply Mr dot is minus Kr, but now we have extra a term here, this term and this term. We have to now find out W hat dot, such that the closed loop error dynamics is stable.

The closed loop error dynamics is stable if W hat tilde transpose phi is bounded. For this we consider a Lyapunov function, which is L is half r transpose Mr plus half trace W tilde transpose gamma inverse, gamma is a positive definite matrix, W tilde so it is time derivative of this Lyapunov function is half r transpose M dot r plus r transpose Mr dot plus trace W tilde gamma inverse W tilde dot and you know that W tilde dot is simply W hat dot, because if you look at here W tilde which is f minus f hat, so W tilde is W minus W hat; so W tilde dot is minus W hat dot.

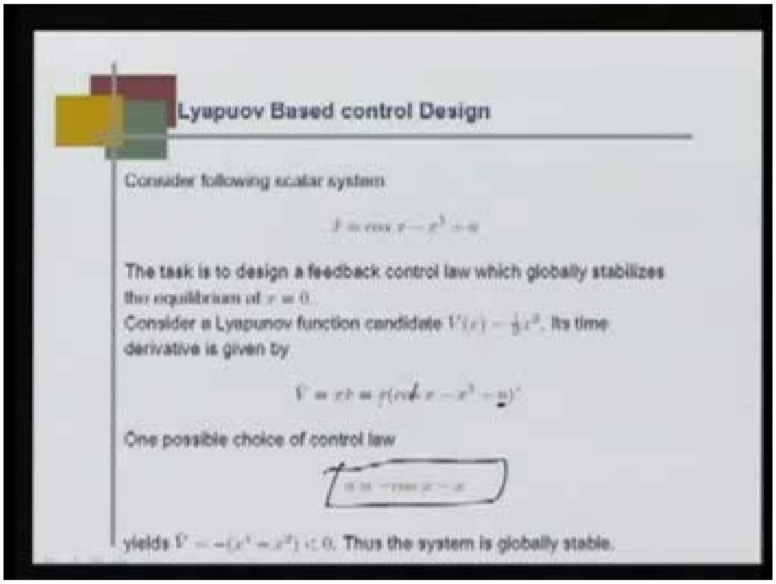
Going back here, you can easily see that r dot, if you replace r dot which is M inverse this quantities. So, this quantity r transpose Mr dot if I replace r dot from here, I finally get minus r transpose V_mr minus r transpose Kr minus r transpose W tilde transpose phi. You see that, because of skew-symmetricity these two quantities is 0. Finally, L dot is minus r transpose Kr and these two terms.

(Refer Slide Time: 08:22)



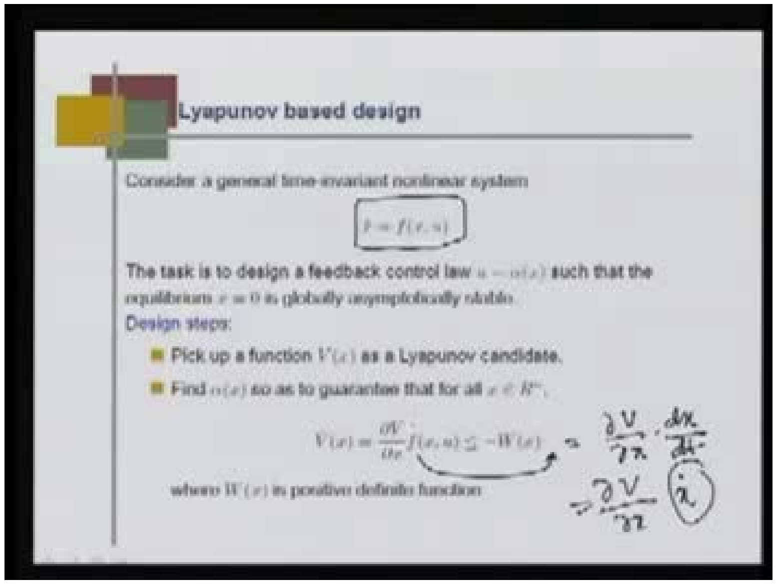
If I take W hat dot is gamma phi r transpose, then L dot is, as I said, this is 0, because of skew-symmetric. If I select this, this quantity is same as this quantity, they cancel out. Your rate derivative of Lyapunov function is minus r transpose Kr. This ensures the boundedness of approximation error W tilde transpose phi. It can also be shown that V double dot is bounded and by Barbalat's Lemma r tends to 0 as t tends to infinity. It is not simply stable; it also says that the tracking error will converge to 0. We showed that, this is a neural network based direct adaptive control, where my control law is given by this particular thing. Weight update law is given by this particular expression. W hat d dot is gamma phi r transpose.

(Refer Slide Time: 09:52)



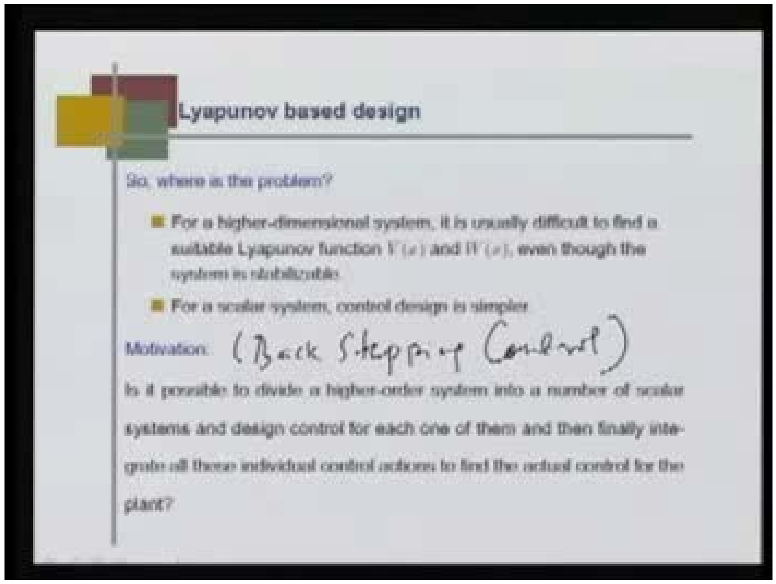
Now, we will go to the back stepping concept. How we can also utilize back stepping concept to design neural network based direct adaptive control for very complex systems. I just introduce little concept of Lyapunov based control and back stepping control. You see this is our simple scalar differential equation, x dot is a cross x minus x cube plus u; u is the control action and x is the state of the system. The objective is that, the task is to design a feedback control law which globally stabilizes the equilibrium at x equal to 0. Now, consider a Lyapunov function candidate V(x) which is half x square, because x is the only state, you can easily write down this. This rate derivative or time derivative V dot is x and x dot; x dot is cos x minus x cube plus u. If I select u to be minus cos x minus x so you get V dot is minus x to the power 4 plus x square, if I replace 1 here, this and this term cancels out and inside minus x cube minus x and outside x. If you multiply, you get, minus outside x 4 plus x square so this is always 0 so the system is globally stable. Tthis is my control law, u equal to minus cos x minus x. This I am finding out by simply applying the Lyapunov function.

(Refer Slide Time: 11:56)



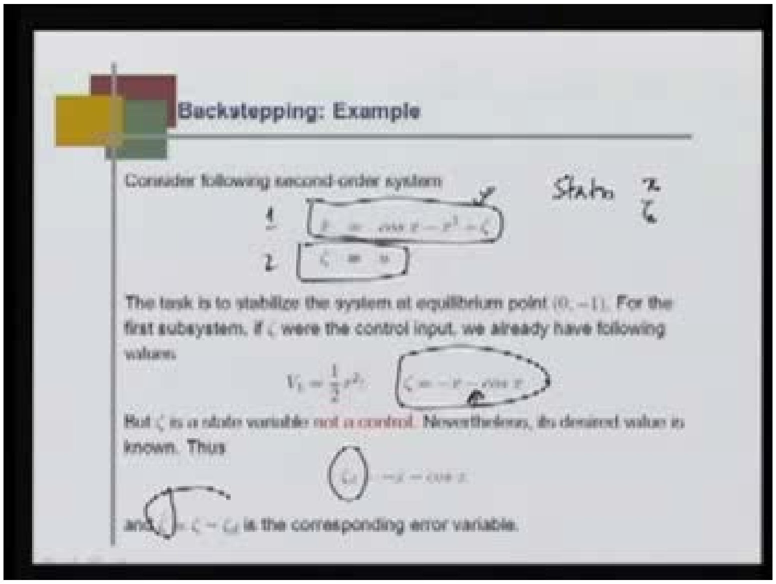
Now in a Lyapunov based design, if I am given a nonlinear system a vector equation x dot is f into x, u. x is a vector; f is a vector and u is also a vector. The task is to a design a feedback control law u equal to alpha x, such that, the equilibrium point x equal to 0 is globally asymptotically stable. The design step is always is to find out what is a V(x), as a Lyapunov candidate. Then, you find out what V dot x. The rate derivative which you can write as del V doe V upon doe x into f(x,u). Because this quantity is doe V by doe x into dx by dt. This quantity is doe V by doe x into x dot. So, this x dot is f (x,u), if I put… and if this is always less than some positive definite function W(x), then this is Lyapunov stable.

(Refer Slide Time: 13:10)



What is the problem? The problem is for a higher dimensional system, it is usually difficult to find a suitable Lyapunov function V(x) and W(x), even though the system is stabilizable. For a scalar system, control design is simpler. The motivation now for a back stepping control is. Is it possible to divide a higher-order system into a number of scalar systems and design control for each one of them and then finally integrate all these individual control actions to find the actual control for the plant. Now the example, so what I am trying to say here is that, because we saw that for a simple system like a scalar differential equation which was given earlier here. This is a scalar differential equation (Refer Slide Time: 14:11) for this designing a control action was very simple. But if (Refer Slide Time: 14:19) complex systems of vector differential equation is there, can I divide this vector differential equation to simple sub systems, such that, for each sub system, I can design a control action using Lyapunov function, it is possible.

(Refer Slide Time: 14:35)



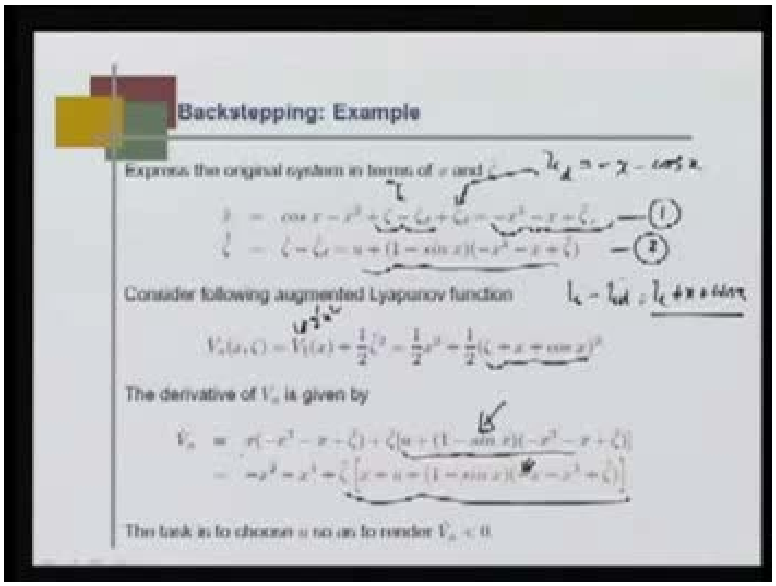
Now, we will see that how we can do that? Here is a vector differential equation. We have two states: One is x and another is z, where the differential equation is x dot is cos x minus x cube plus xi and xi dot is u. The task is to stabilize the system at equilibrium point, you can easily see that the equilibrium point here is 0 and minus 1. For the first subsystem, if xi were the control input. This is the equilibrium point. Now, we want to stabilize the system around the equilibrium point and we represent this subsystem 1 and this one as 2.

Let us design a control action for this. Let us stabilize the first subsystem, assuming xi to be a control action or as a state. If it is a state, then it is a double state there which is difficult. So, it is better that I always assume this to be… So, I already have solved this problem in the last example, where xi was u. Instead of xi, I say this xi is simply a virtual control action, let us imagine. If I accept xi as a control input then, we have already shown in the last example V_1 is half x square. Then xi has to be minus x minus cos x for which the system is stable. We found out the rate derivative of this is minus x to the power x to the power 4 minus x square.

This xi is not a control action; I cannot give this xi from outside the system, this is inside. But xi is a state variable not a control action, nevertheless, this desired value is known.

What is meaning of that? If xi follows this trajectory then, this is stabilizing. So, desired xi d is minus x minus cos x, if xi d follows these trajectory minus x minus cos x then, the first subsystem is stable that is very clear. We introduce a new state variable instead of xi tilde which is xi minus xi desired.

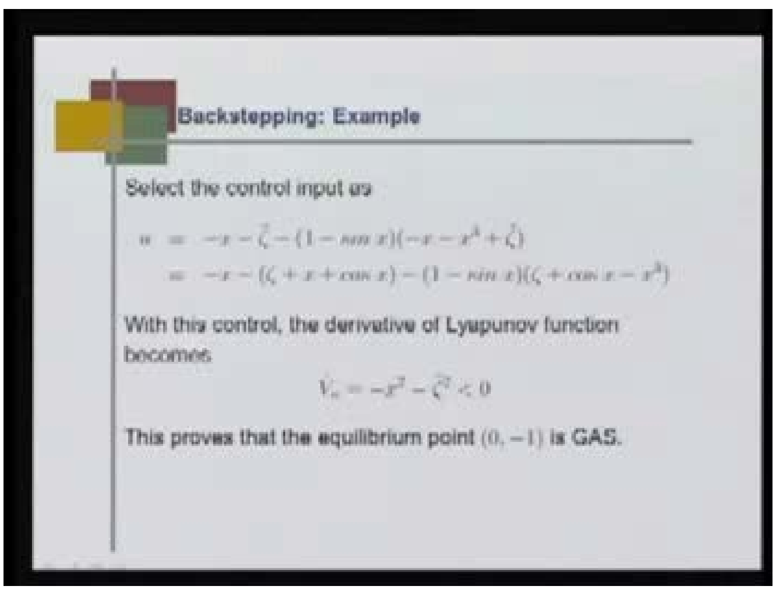
(Refer Slide Time: 17:30)



Now, what I do? I rewrite the first subsystem x dot as cos x minus x cube plus zeta, this was my original one, I add zeta d and subtract zeta d, by doing that, what I do? I write this equation… I know already that this zeta d is minus x. Let me write here we have already found out zeta d is minus x minus cos x. If I introduce this zeta d, cos x and cos x goes out. What is left is minus x and minus x cube. This particular term is xi tilde; this is what I told you. So, x dot in the new form is minus x cube minus x minus xi tilde and xi tilde dot, because xi tilde is a new state variables. I have to write down an expression for xi tilde dot which is xi dot minus xi desired dot. Which is xi dot; we already know u, from the original expression. So, xi d dot, if I differentiate this, of course this would become minus x dot minus cos x into x dot, if I write that, this will be 1 minus sine x into minus x cube minus x plus xi tilde. Now, you have this two: This is my subsystem one and this is my subsystem two (Refer slide time: 19:26). We have to show that whether the system is stable for and what control action I should find out u, such that, the overall system is stable.

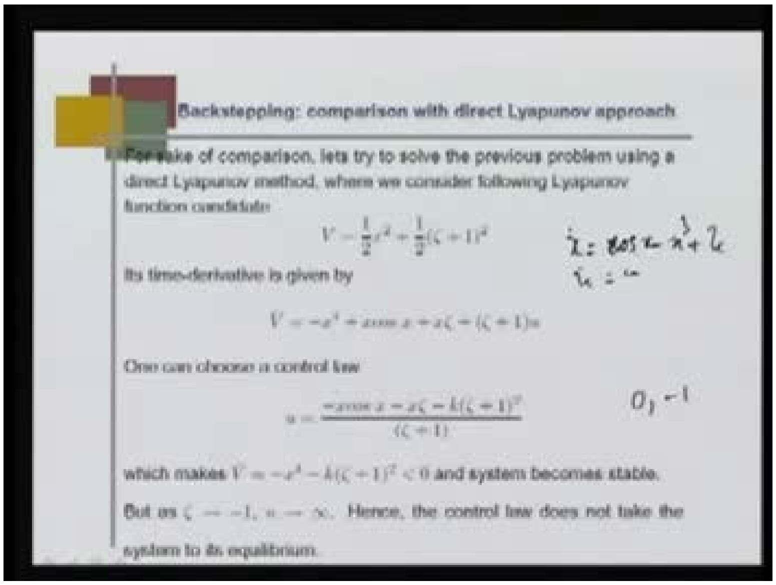
We first of all showed that x dot is stable provided this xi tilde is actually bounded or it is 0 or xi is following xi d. Now we have to find out the overall stability. To do that, we take the Lyapunov function, we acknowledge the previous Lyapunov function V_1x which is half x square and plus half xi tilde square. This is V_1x is half x square plus half xi tilde square can be written as xi plus x plus cos x whole square, because xi d is minus x minus cos x so xi minus xi d is xi plus x plus cos x, this is what is put it here. xi plus x plus cos x whole square. If I take the direct derivative of this Lyapunov function, I get x into x dot. So x dot is this quantity from minus x cube minus x minus xi tilde and plus this 1. If I differentiate this quantity xi tilde into xi tilde dot, this quantity xi if I (21:14) insert this quantity here, I get minus x square minus x whole square. You see that minus x square minus x whole square. I can take xi tilde common here and put x here and this quantity all enters here. I can make this quantity vanish if I select u to be x minus x minus 1 minus sine x into minus x minus x u plus xi tilde, then, (21:50) minus x square minus x to the power 4 that is guarantees robust stability.

(Refer Slide Time: 21:59)



Selecting u to be minus x minus xi tilde minus 1 minus sine x is quantity we get. Replacing xi tilde is xi plus x plus cos x, we get, V_a dot is minus x square minus xi tilde square. If I take this quantity and put it there, I get V_a dot is minus x square minus xi tilde whole square and this proves the equilibrium point is globally asymptotically stable because this is always negative definite.

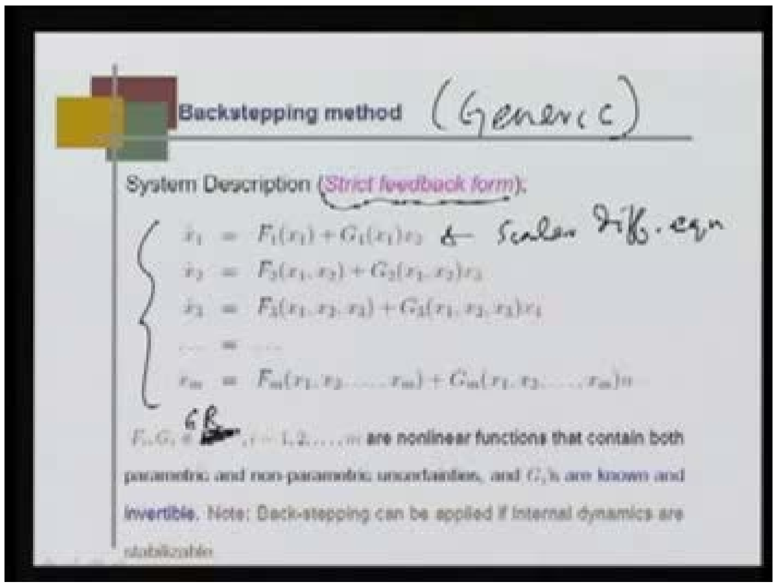
(Refer Slide Time: 22:45)



What we essentially did here in back stepping control is that, first we simplified our first stabilization problem by stabilizing the first subsystem and then augmented the Lyapunov function to define the global stability of the entire system. Instead of doing that, if I would have directly selected a Lyapunov function like this, then, I would have taken V Lyapunov derivative; this is what I am saying. Let us take the comparision and try to solve the previous problem using direct method where we consider following Lyapunov function candidate. You see that V dot becomes this quantity minus x 4 x cos x plus x xi plus xi plus 1 u you can easily see that this you can verify this to be the rate derivative of V dot because this is x x dot and we know already x dot is cos x minus x u plus xi. So, x dot becomes this quantity minus x 4 minus x cos x and plus x xi, this quantity and this quantity is xi plus 1 into xi dot and you know that xi dot is u. So, xi dot is u and if I select u to be this quantity then V dot is minus x 4 minus k xi plus 1 whole square and this is less than 0 and the system becomes stable but as xi tends to minus 1, because we are stabilizing around the equilibrium point 0 minus 1. When xi goes to minus 1, you can easily see u becomes infinite, so system is not stable; although it implies that here it is stable but it is not so.

Hence this control law does not take the system to equilibrium. This control law cannot be implemented by going directly. If I assume a global Lyapunov function from the very beginning designing control law is difficult; whereas, we had a very nice control law (Refer Slide Time: 25:29) that guarantees stability here.

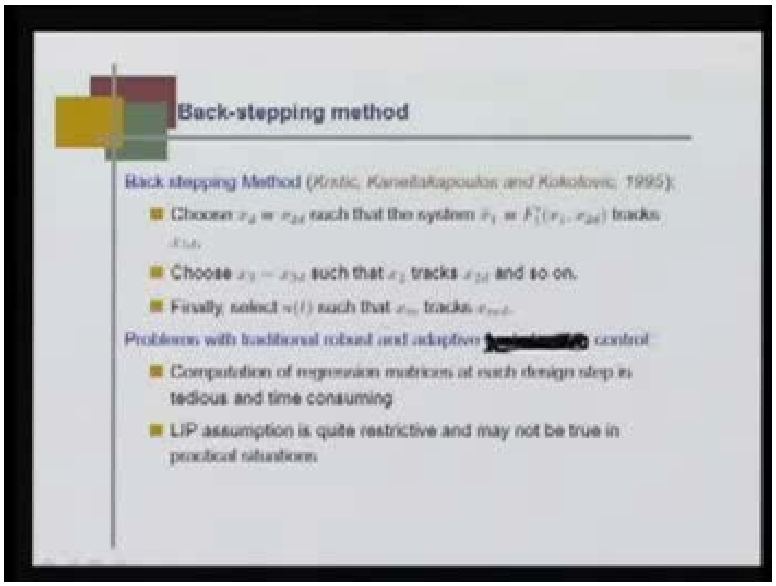
(Refer Slide Time: 25:32)



We will give back stepping method in generic; through an example we explained what the back stepping control. The general motion of back stepping is that the system description must be in strict feedback form. What is the strict feedback form? The strict feedback form means x_1 dot is F_1 x_1 plus G_1 x_1 x_2. To begin, you just assume this each element is simply a scalar differential equation. So x_1 dot is F_1 x_1 plus G_1 x_1 x_2 so this is a scalar; this is a scalar; x_2 is also a scalar. Similarly, x_2 dot is F_2 x_1 x_2 again a scalar function G_2 x_1 x_2 another scalar function into x_3. It can be also vector for the moment; you just try to understand simply individually each one is a scalar differential equation. If I can represent any vector differential equation, in terms of scalar differential equation then, this is called Strict Feedback Form. F_i and G_i, this is wrong, simply they are all scalars; I would say these are all scalars. At the moments I just assume are to be scalars. A nonlinear function that contain both parametric and non-parametric uncertainties, this F_1 G_1 F_2 G_2 F_3 G_3 F_m and G_m they are nonlinear functions that contain both parametric and non-parametric uncertainties and this G_is are known and invertible. We assume that

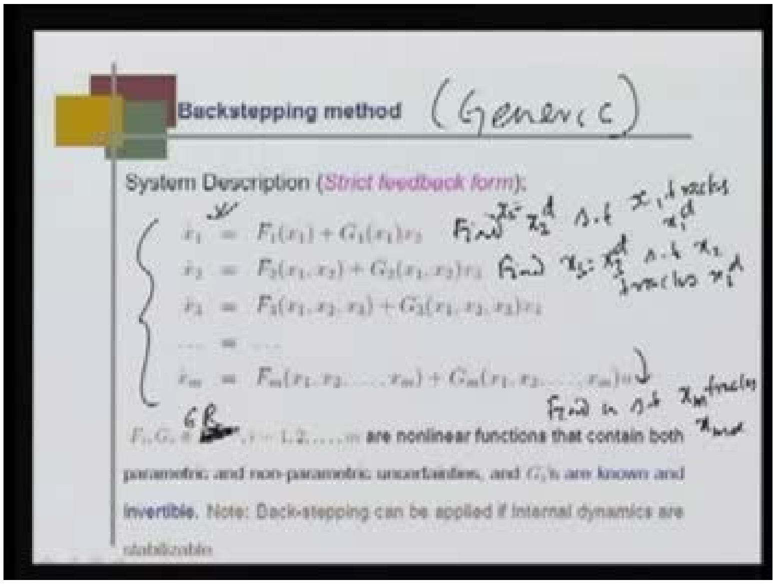
G_1 G_2 G_3 G_m. they are known and invertible. Note, back stepping can be applied, if internal dynamics are stabilizable, that is, if each individual subsystem are stabilizable then, we can design a back stepping control for this particular form which is called strict feedback form. You see that x_1 dot is represented in double x_2; x_2 is represented in x_3; x_3 dot is represented in x_4; until x_m dot is represented in terms of external controller.

(Refer Slide Time: 28:42)



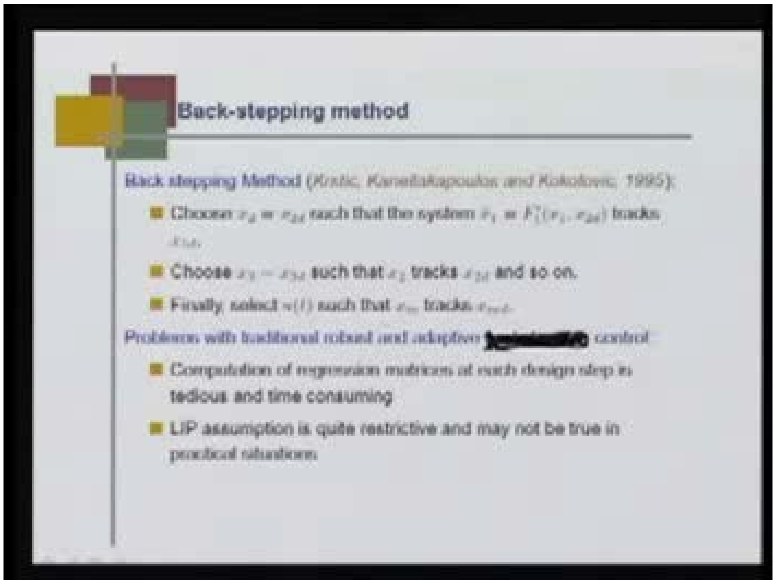
The back stepping method was originally given by Krstic, Kanellakopoulos and Kokotovic in 1995. The general idea in the back stepping control is you saw that (Refer Slide Time: 29:03), given this system, what I can always do? I can find out what is x_2 as a virtual control action. So, what is my x_2 equal to x_2d? Such that, this particular system is stable, which we already have shown, by using Lyapunov functions approach. Then again I find out what is x_3 equal to x_3d, the virtual control actions such that the second subsystem along with the first one is stable, and like that we go ahead. So choose x_2 equal to x_2d such that the system x_1 dot equal to F_1 hat x_1 x_2d tracks x_1d. Similarly, choose x_3 equal to x_3d such that x_2 tracks x_2d and so on. Finally, select the control action u_2 such that x_m tracks x_md.

(Refer Slide Time: 31:18)



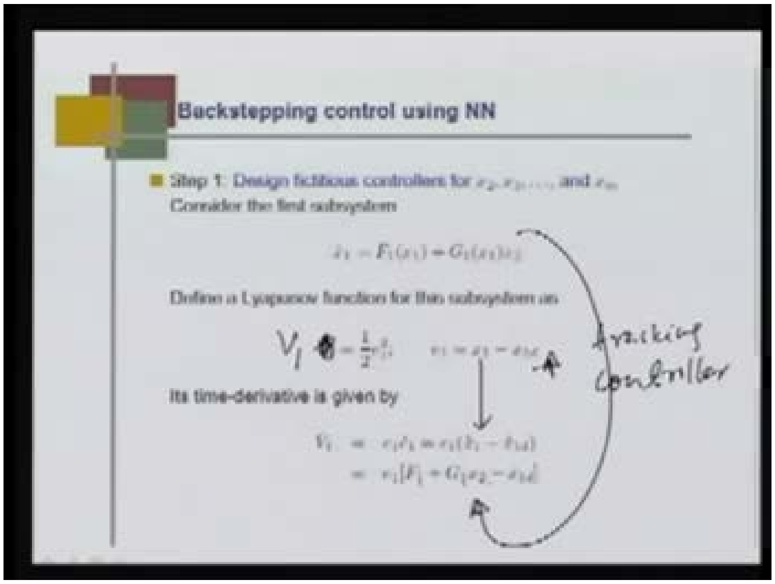
What we are trying to do here is that, find x_2 equal to x_2d such that x_1 tracks x_1d in this one. Similarly, find x_3 equal to x_3d such that x_2 tracks x_2d and so on. Finally here, find u such that x_m tracks x_md. This is why it is called back stepping.

Refer Slide Time: 31:29)



Once I told you what the methodology of back stepping is, I will tell you why this is normally done? The problem with traditional robust and adaptive control is computation of regression matrix at each design step is tedious and time consuming. Linear in parameterization assumption which is used in robot manipulator is quite restrictive and may not be true in practical situations.

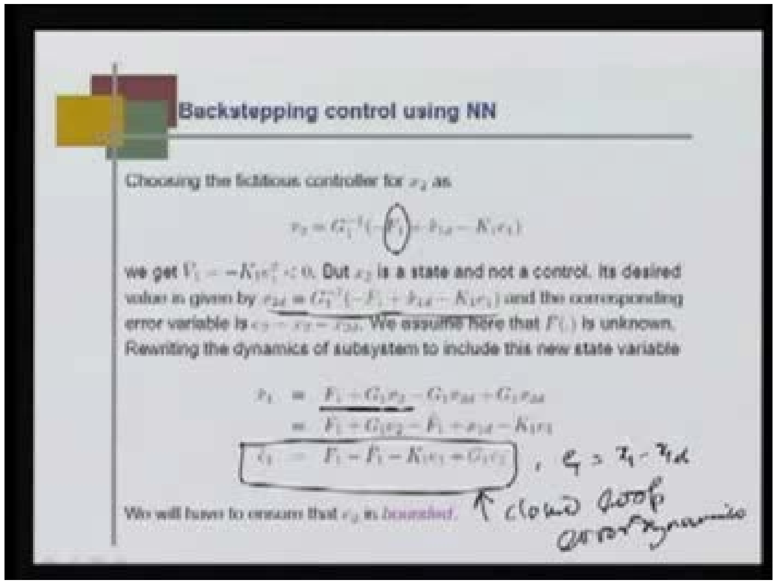
(Refer Slide Time: 31:50)



The back stepping control using neural network; this is the theme of this particular class today. So, I introduce, what is back stepping control? Now, we will go in detail design using neural network. Design a fictitious controller for x_2 x_3 and x_m is a first step. Consider the first subsystem in strict feedback form, that is, x_1 dot is F_1 x_1 plus G_1 x_1 into x_2.

Define a Lyapunov function for this subsystem, we have already seen that, for a subsystem V_1 dot is half e_1 square, where e_1 is x_1 minus x_{1d}, because we are trying to design a tracking controller. We can easily would have shown V_1 equal to half x_1 square, if I simply stabilizing around the equilibrium point. Instead, this is for tracking controller. The objective is to design a tracking controller that, my plant should follow desired trajectory. So it is time, derivative is given by V_1 is half u_1 square and V_1 dot is e_1 into e_1 dot, which is e_1 and e_1 dot you can easily see this is x_1 dot minus x_{1d} dot. What I can do? This x_1 dot I can bring it from here, which is F_1 plus G_1 into x_2 minus this quantity, which is x_{1d} dot.

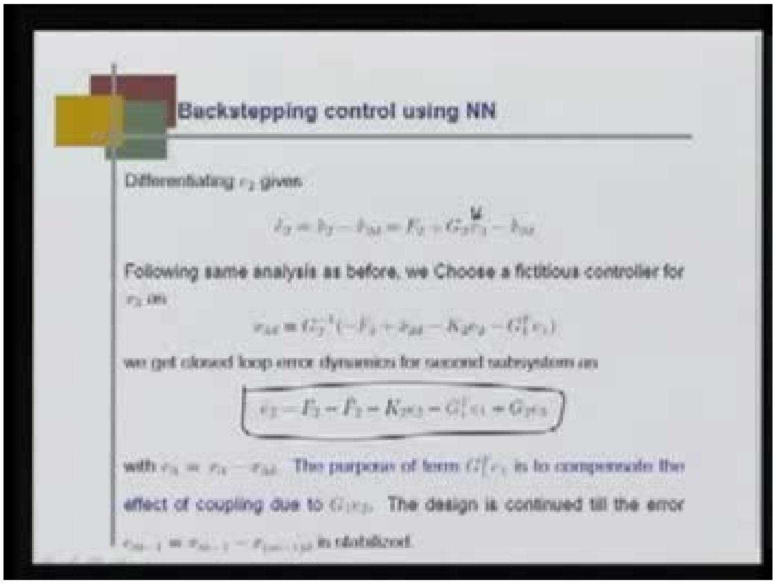
(Refer Slide Time: 33:48)



Choosing the fictitious controller for x_2 because you see that here (Refer Slide Time: 33:53) we found out V_1 is to be this. Now, I want to make V_1 dot negative definite; so x_2 is my virtual control action. So, what should be my x_2, so that, V_1 dot is negative definite that is possible if I select x_2 is G_1 inverse minus F_1 plus x_{1d} dot minus K_{1e} dot. If I select this and put this x_2 in this quantity, I get V_1 dot is minus K_1e_1 square, which is negative definite. But x_2 is a state not a control as we have already seen. So, its desired value is given by x_{2d}, so this is not a control action.

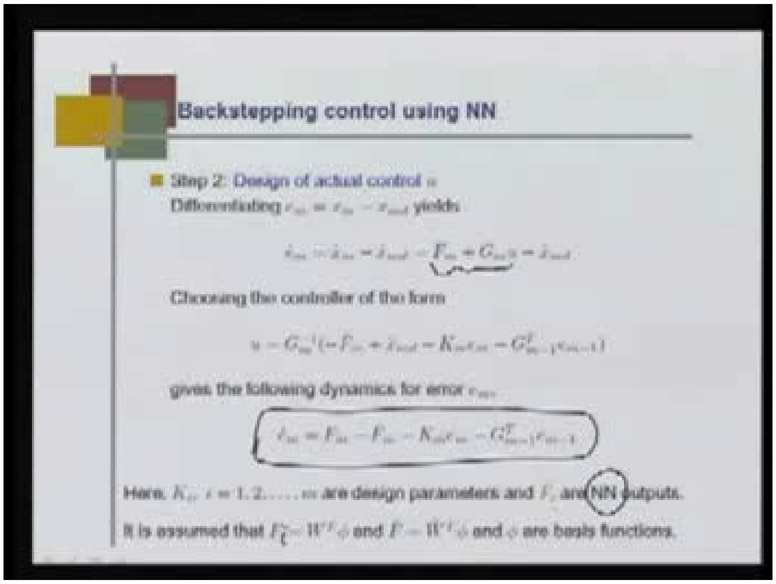
What I can say that, if my x_2 is following this desired value given by G_1 inverse minus F_1 plus x_1 desired dot minus K_1e_1 then, my first subsystem is stable. But you see that I normally do not know why we are utilizing new neural network because this F_1. You have assumed this F_1 to be unknown so because this F_1 is unknown I replace this F_1 by F_1 hat, which is x_{2d} desired, corresponding the error variable e_2 is x_2 minus x_{2d}. Then, we can now write x_1 dot is F_1 plus G_1x_1, this is my original equation and I add to this minus G_1 x_{2d} and subtract also the same quantity. I replace this x_{2d} by this quantity, by doing that, I get x_1 dot is F_1 plus G_1e_2 minus F_1 hat plus x_1 desired dot minus K_1e_1; K_1 is simply a positive constant. If you do that, I can take x_{1d} dot to this other side and I can write this to be e_1 dot, because my e_1 is x_1 minus x_{1d}. This is called closed loop error dynamics which F_1 minus F_1 hat minus K_1e_1 plus G_1e_2.

(Refer Slide Time: 36:45)



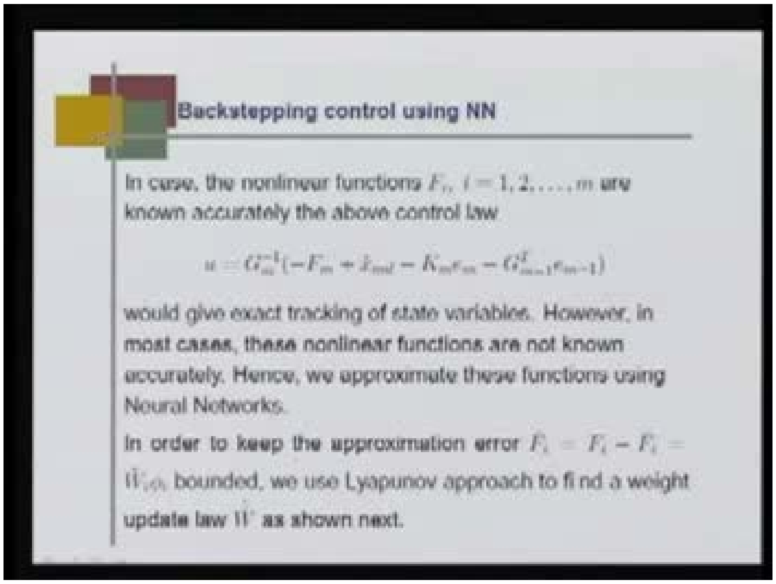
Differentiating e_2 gives e_2 dot is x_2 dot minus x_{2d} dot is F_2 plus G_2x_3 minus x_{2d} desired dot. Following the same analysis before, we choose a fictitious control for x_3, in such a way where x_{3d} is G_2 inverse minus F_2 hat x_{2d} dot minus K_2e_2 minus G_1 transpose e_1. Then, we get the second closed loop error dynamics. e_2 dot is F_2 minus F_2 hat minus K_2e_2 minus G_1 transpose e_1 plus G_2e_3. You see that this G_1 transpose e_1 is to compensate the effect of coupling due to G_1e_2. We can actually derive this from the first principle. The way we found out e_1 taking the Lyapunov function similarly, we can find out what is e_2 dot. The process of finding the closed loop error dynamics for each subsystem is scalar differential equation design is continued till the e_{m1} equal to x_{m1} minus x_m minus 1 desire stabilized. So, we found out the error dynamics.

(Refer Slide Time: 38:15)



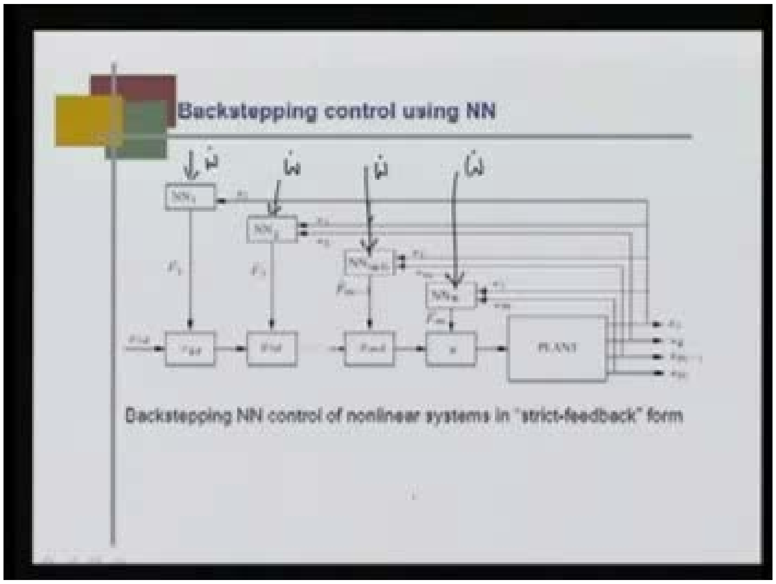
After the last equation, e_m is given by x_m minus x_{md} this time derivative of that is F_m plus G_{mu} minus x_{md} dot, because x_m dot is given by this quantity already by strict feedback form. Choosing u, again this is found out using again Lyapunov function approach, when we assume that F_m is known. Then this is actually a control action that will stabilize. So then, this is my final closed loop error dynamics e_m dot is F_m minus F_m hat minus $K_m e_m$ and minus G_m minus 1 transpose e_m minus 1. Where K_i, i is equal to 1 to m are design parameters; F_is are approximated by neural network. We assume that each F or F_i is approximated by W transpose phi, such that, phi hat the estimated one W hat transpose phi; phi are the basis function. You know that phi is the basis function of input. For example, if I go back to my original form, (Refer Slide Time: 39:53) if I want to estimate this F_1 then, my input is x_1 and F_2 my inputs are x_1 x_2 and F_3 I estimate then x_1, x_2 x_3 are the input.

(Refer Slide Time: 40:05)



In case the nonlinear functions F_is are known accurately the above control law is given by this, would give exact tracking of state variables. However, in most cases, these nonlinear functions are not known accurately. Hence, we approximate these functions using Neural Networks. In order to keep the approximation error bounded we use Lyapunov function to find a weight update law, W hat dot. Because if you at, previously this each neural network is represented by W hat transpose phi is an unknown basis functions; W hat transpose this weight vector is not known. We have to find out weight update law.

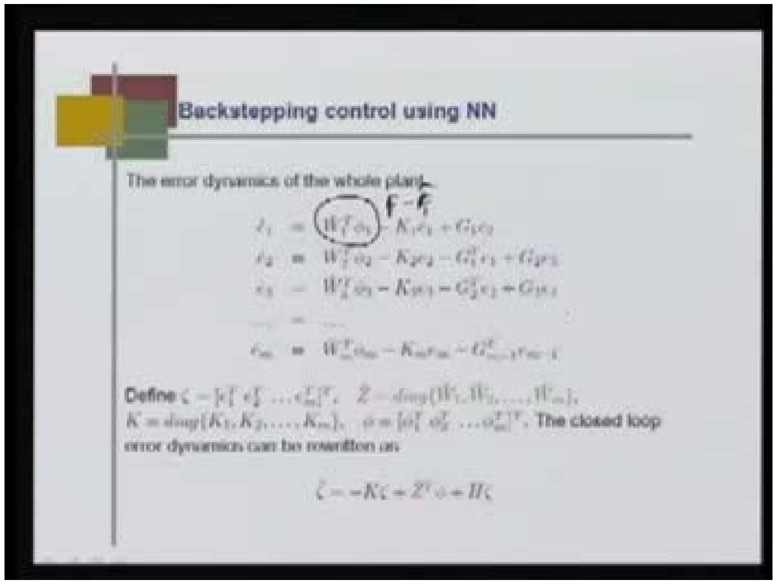
(Refer Slide Time: 40:52)



This is your back stepping controller. We design what is u and u utilizes the neural network NN_m whose output is F_m hat. Also it utilizes x_{md}, the information that is given by back stepping approach, because given x_{1d} I find out, what must be x_{2d}? This is my first F_1 is approximated by neural network 1, whose input is only x_1.

Second neural network NN_2 takes the input x_1 x_2 as I showed you earlier, so, the output is F_2 hat. I get x_{3d}, you can easily see that, (Refer Slide Time: 41:55) we said that given x_{1d}, I find out what is x_{2d}? x_{2d} is terms of x_{1d} is this input; neural network output F_1 hat. Similarly, you can easily see here also x_{3d} is a function of neural network output F_2 hat and x_{2d} and that is what we are seeing here. So, x_{2d} is a function of x_{1d} and F_1 hat. x_{3d} is a function of x_{2d} as well as F_2 hat and so on. u is finally a function of f_m hat, which is output of NN neural network m th neural network, whose input is x_1 to x_m and this is also u is a function of x_{md} dot. Then, if I give this to the plant, I have to find out, what should be the weight update law of all these neural networks? Such that the closed loop error dynamics is stable.

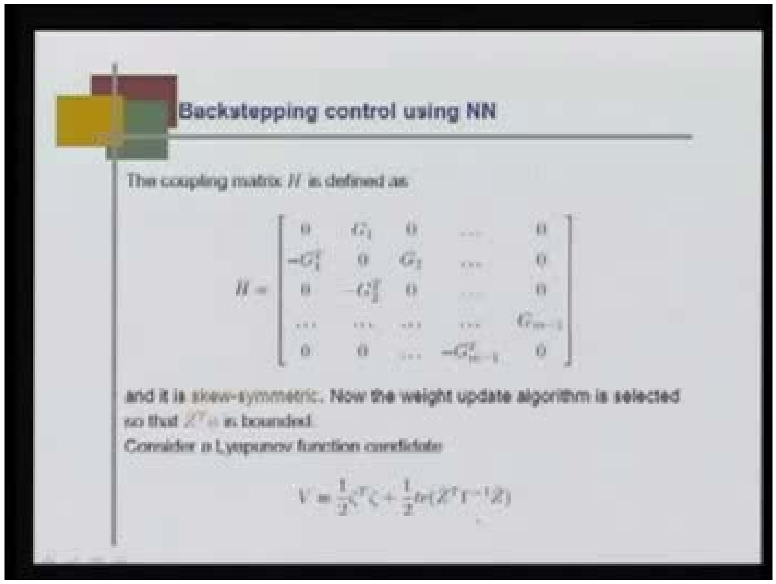
(Refer Slide Time: 43:20)



Now, I just represent the error dynamic that we have already seen u_1 dot is… this is F_1 hat minus K_1e_1 plus G_1e_2; e_2 dot rate derivative of second error differential is again F_2 hat minus K_2e_2 minus G_1 transpose e_1 plus G_2e_3 and so on. This is error dynamics. So, if I define to be this quantity this vector of all the parametric tracking errors. z tilde is… These quantities are actually F minus F hat. You can easily see that e_m dot is F_m minus F_m hat and so on. This is actually; the first one is F_1 minus F_1 hat; similarly, this is F_m minus F_m hat. This is my weight vector, the difference between the desired one, there exist the actual one. K is the controller parameters that as to be found out and phi is known quantities of basic functions of the input x_1, x_2, x_3 and so forth.

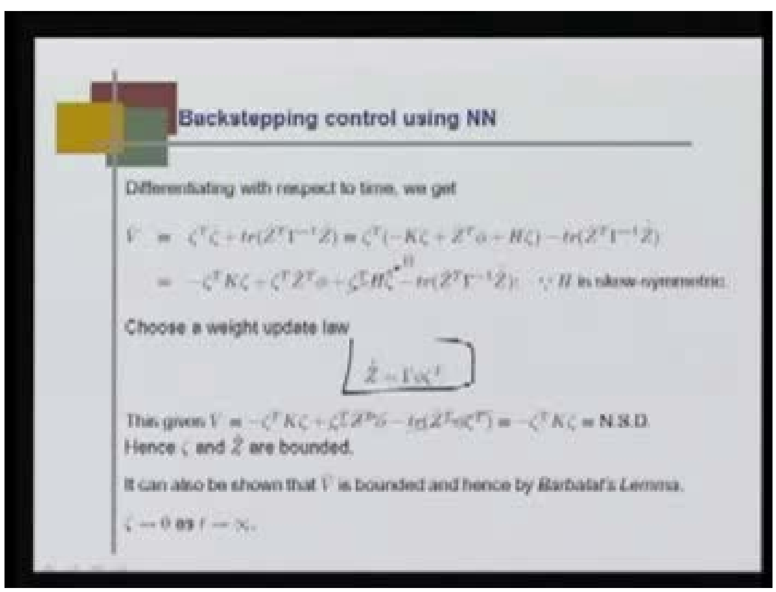
The each scalar differential equation is scalar error dynamics and like that m scalar differential equation can be put into one vector differential equation xi dot is minus K zeta. You can easily see K is the quantity; Z transpose phi this quantity and H xi is this quantity, because my xi is simply this quantity.

(Refer Slide Time: 45:30)



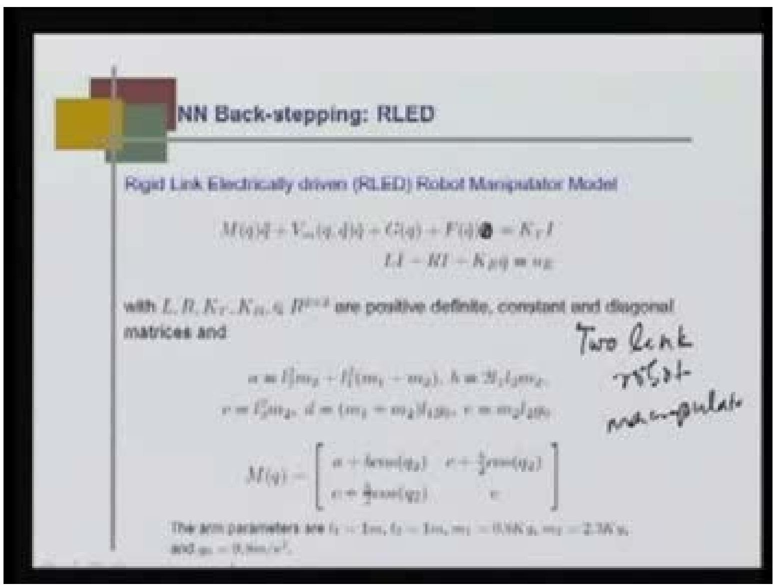
H is a matrix. H matrix is given by 0 G_1 0 0, minus G_1 transpose 0 G_2 and so on. This is my H, the property of this skew-symmetric. Now, the weight update algorithm is selected because if I write x transpose Hx that is 0. You can prove that.

(Refer Slide Time: 46:00)



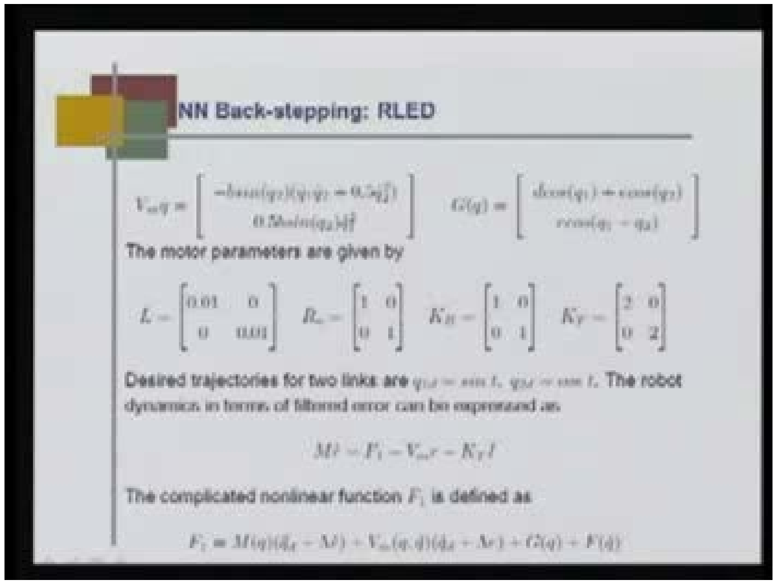
Let us take a Lyapunov function for this to be this. If I find out differential of V dot, finally, you get this quantity and if you further elaborate, you see that, this is due to skew-symmetric this quantity becomes 0. If assumed z hat dot is this quantity and you know the z is actually the weight vectors, so, if weight update law is taken by this quantity and this quantity they all are same. So V dot is simply xi transpose K xi and K is all positive quantity and hence this is stable. It can be shown that B double dot is bounded hence Barbalat's Lemma; xi tends to 0, means, 0 tracking error goes to 0. We just learnt that, this is weight update law for which this (Refer Slide Time: 47:15) particular dynamical system is stable or this closed error dynamic are stable, where neural network as been used to estimate the unknown functions F_1 to F_m.

(Refer Slide Time: 47:32)



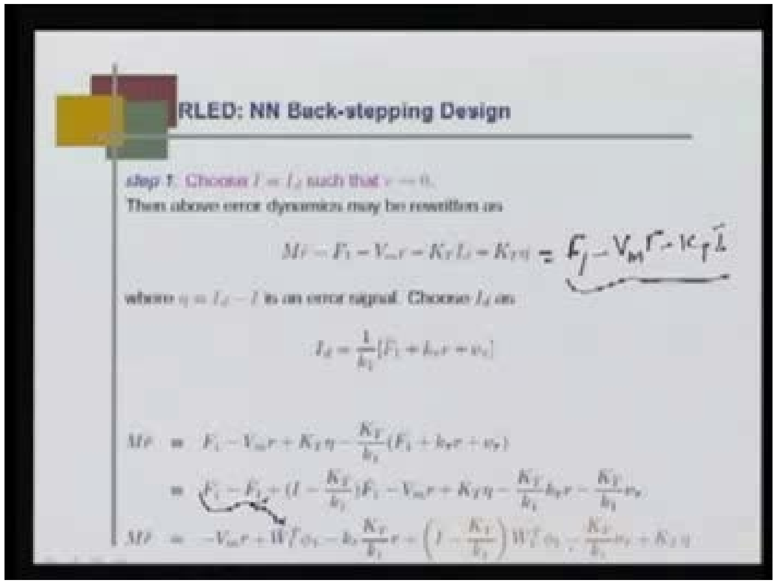
Now, we take a example of a rigid link electrically driven robot manipulator for which this is our normal expression M_q double dot; V_{mq} dot (47:54) this is gravity; this is friction; equal to K_tI. I is the current of the electrically driven motor. You know this is actually a torque. And torque is constant into current. Then we write for the motor of the equation is LI dot plus RI plus K_{bq} dot is U_e which is the applied voltage. By defining this parameters, a is this; b is this; this is taking two link robot manipulator and defining parameter a b c d and e as this. M becomes 2 into 2 matrix like this. The arm parameters are: L_1 is 1 meter; L_2 is 1 meter; m_1 is 0.8 kg; m_2 is 2.3kg; g_0 is 9.8 meter per Second Square.

(Refer Slide Time: 49:05)



You see that core lax matrix is force vector is given by this particular expression. G(q), the gravity forces is given by this particular term. The motor parameters are given L is again here; R is given by this; K_b is given by this particular matrix and torque constant is given, because we have two links so two motors, that is why you are seeing, these are all matrices. Desired trajectories for two links are q_{1d} is sine t and q_{2d} is cos t. So, robot dynamics in terms of filtered error, you see in the beginning we always represent the robot dynamics in terms of filtered error M_r dot is F_1 minus V_mr minus K_{tr}, where F_1 is given by this particular quantity.

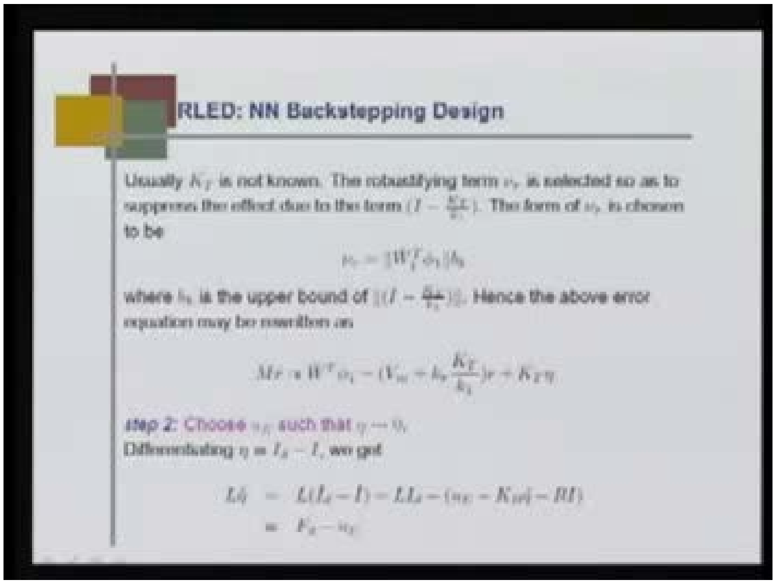
(Refer Slide Time: 50:10)



You see this is a kind of expression that we can always say this is stable, provided if I select I equal to desire I_d such that r tends to 0. The above, error dynamics may be written in terms of M_r dot is F_1 minus V_mr. Now, I just represented this, earlier it was simply minus K_tI instead of I, I write I_d plus K_t zeta and zeta is I_d minus I, so you can easily see this is same as original one F_1 minus V_mr minus K_tI. This is my original dynamics and that I am representing in this particular form. If select I_d to be of this particular form. Since, I do not what is K_t, so, I do not write 1 upon K_t I write 1 upon K_1. K_1 is the positive constant.

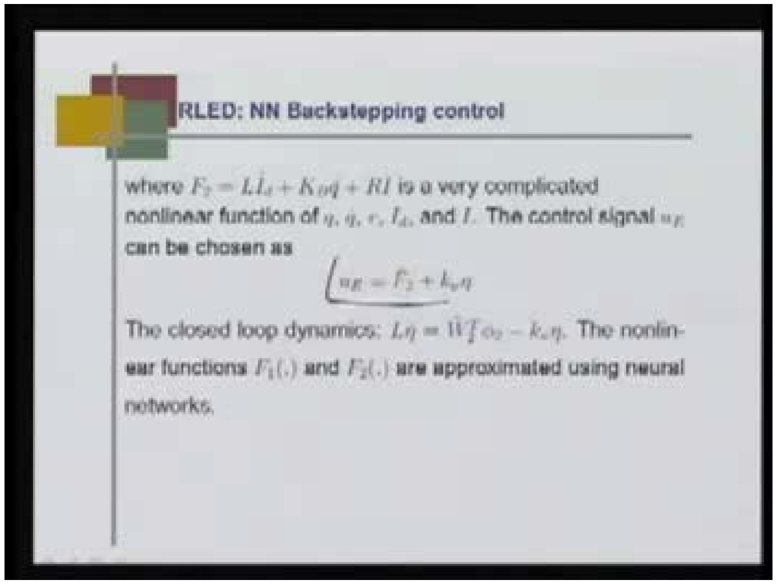
Then, M_r dot can be represented in this particular form. You see that if F_1 and F_1 hat there are almost known already, because let us think about the case where it is known, then, I can say this is almost 0. V_mr, this particular some constant into r they contribute to this stability. So all that I have to show that, if I can cancel this… Again I represent this particular term I minus V_mr bring it here and this minus F_1 this particular quantity is represented by this expression. This F_1 hat is represented by neural network. Then can represent this. This quantity is represented here. I always find some V a new tau in such a way this term and this term can be cancelled out. The objective is that to find this particular term that this term and this term first cancels out.

(Refer Slide Time: 52:35)



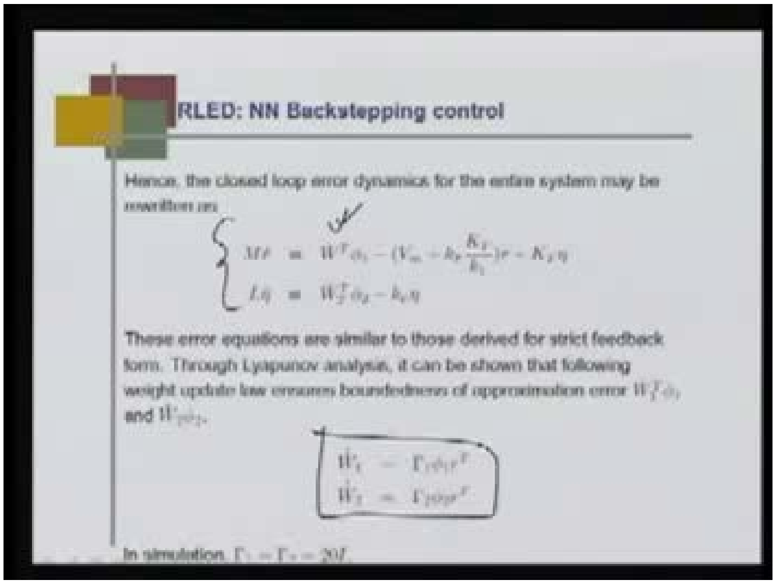
Usually, K_t is not known. The robustifying term U_t is selected so as to suppress the effect due to term I minus K_t upon K_1, V tau if select this, then V_k is the upper bound of this. The above error equation this M_r dot is this quantity, can be rewritten in this particular form (Refer Slide Time: 53:07). So you see this I representing by a neural network; this is actually, if M_r dot is this, if neural network approximately function, this is 0. Then, M_r dot is this quantity, this is stable. Then I have an extra term K_t zeta and if zeta is bounded, then system is stable. Now, I select U_e in such a way that zeta will be bounded, for that zeta already I know that, desired current minus I. So, L zeta dot is LI_d dot minus I dot, I can represent this in terms of F_2 minus U_v, this is the control action. What should be the control action such that theta is bounded, that is the second one.

(Refer Slide Time: 53:55)



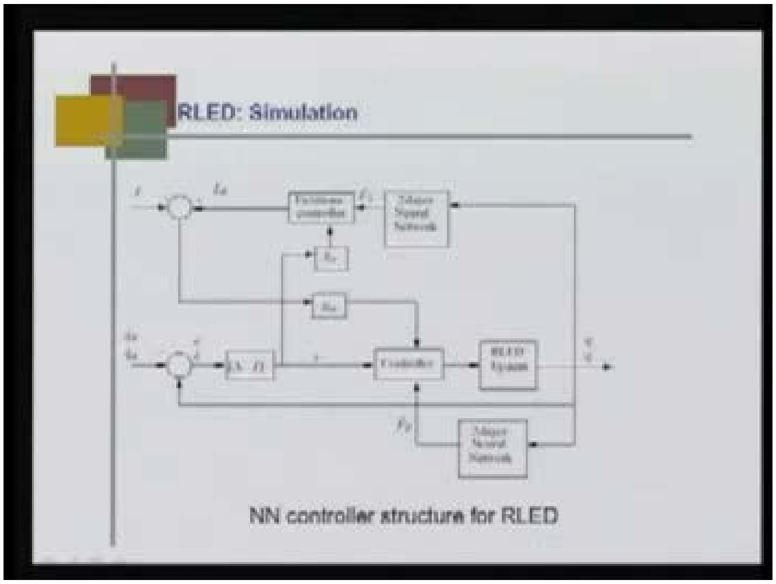
If I find U_e as, this is neural network approximation of F_2 and then if I take control action to be Kv theta then error dynamics is L theta dot is this quantity. You see that if this is 0 that means, neural network approximation is exact, then this is a stable dynamics. If zeta is bounded in previous term this is also stable. Now, I have to do is, I got two closed error dynamics. M_r dot is this quantity and L zeta dot is this quantity, so two closed loop error dynamics in the same manner that, we had earlier this one, set of similar error dynamics found out just now. This two error dynamics going by the same principle of designing the Lyapunov function, we can derive that for this (Refer Slide Time: 55:12)

(Refer Slide Time: 55:10)



The weight update law is gamma_1 phi_1 r transpose and W_2 hat dot, for this one, is gamma_2 phi_2 r transpose. If I do then the system is in Lyapunov stable.

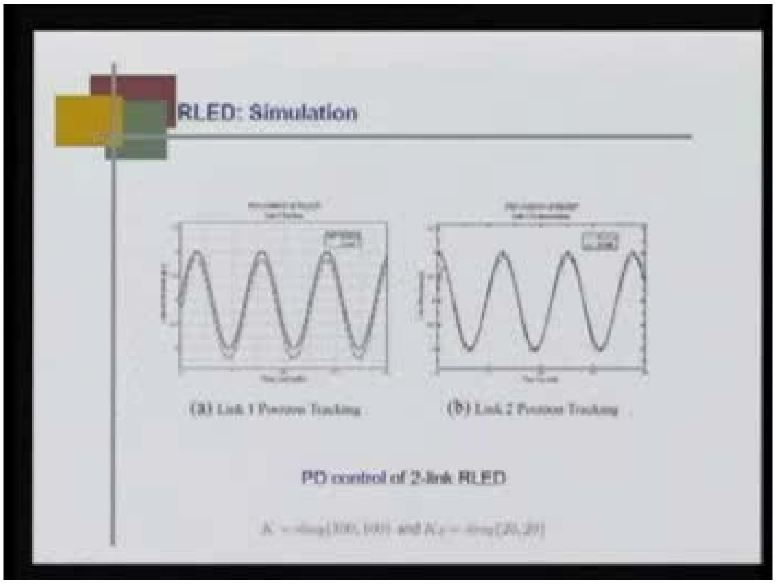
(Refer Slide Time: 55:29)



By doing that… This is obviously direct adaptive control architecture for rigid link electrically driven robot manipulator. Implementing the previous control law, (Refer

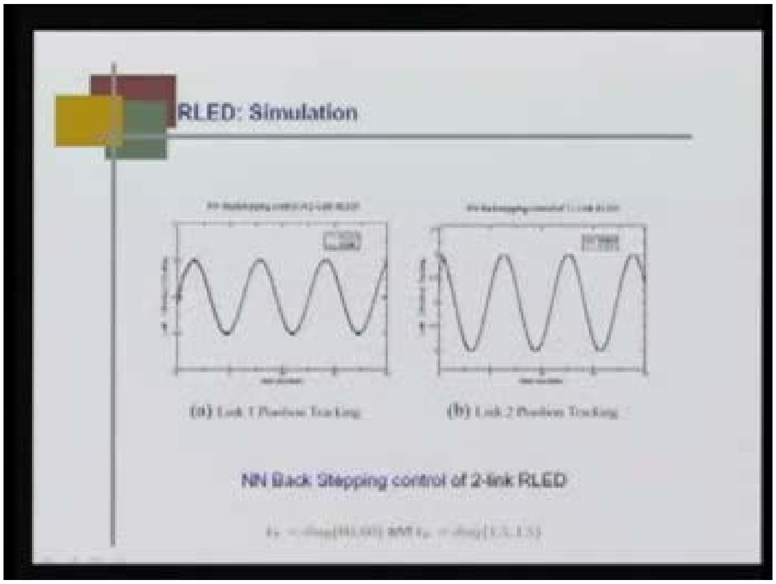
Slide Time: 55:47) this is weight update law and control law. This is control law (Refer Slide Time: 55:56).

(Refer Slide Time: 55:59)



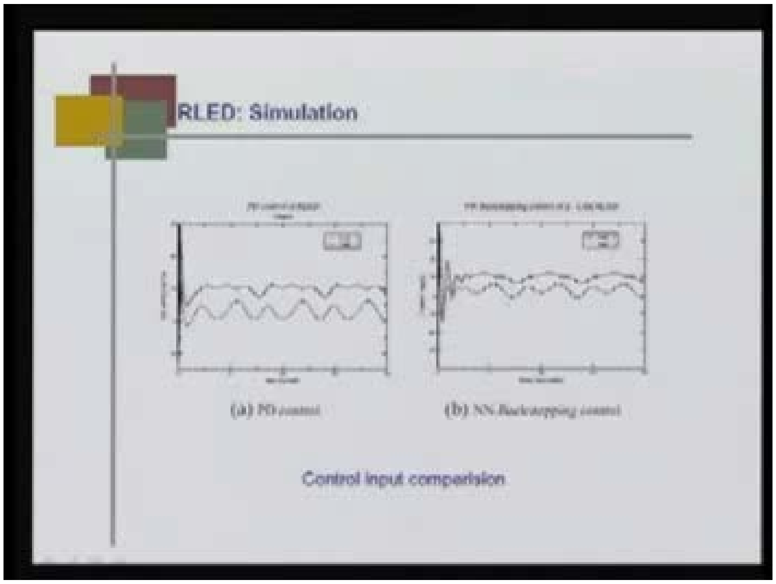
Doing that, I get this you see the desire actual link 1 position and tracking link 2 positions tracking here. This is PD control of rigid link electrically driven manipulator by taking….

(Refer Slide Time: 56:20)



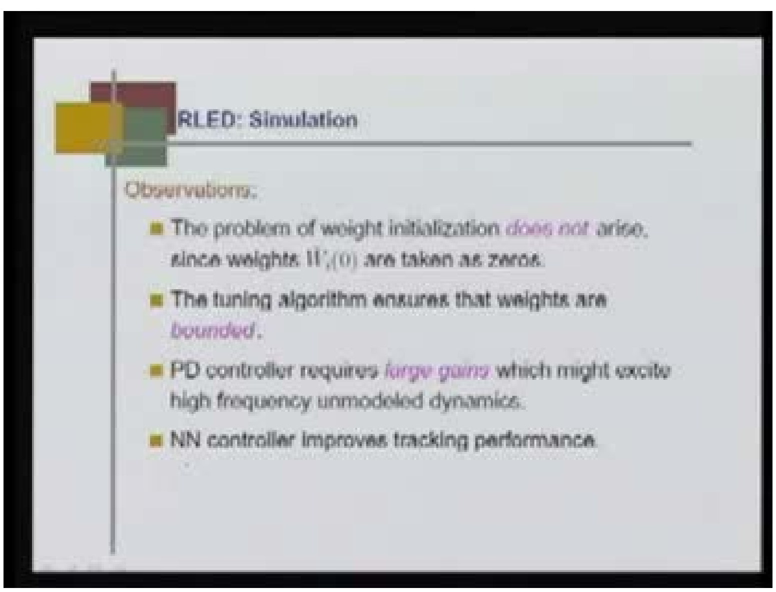
This is a PD control. You see that, the proposed back stepping control tracking error is vanished. If I do simple PD control lot of tracking error in both joint 1 and joint 2, but once I implement the neural network based back stepping control tracking error goes to 0.

(Refer Slide Time: 56:46)



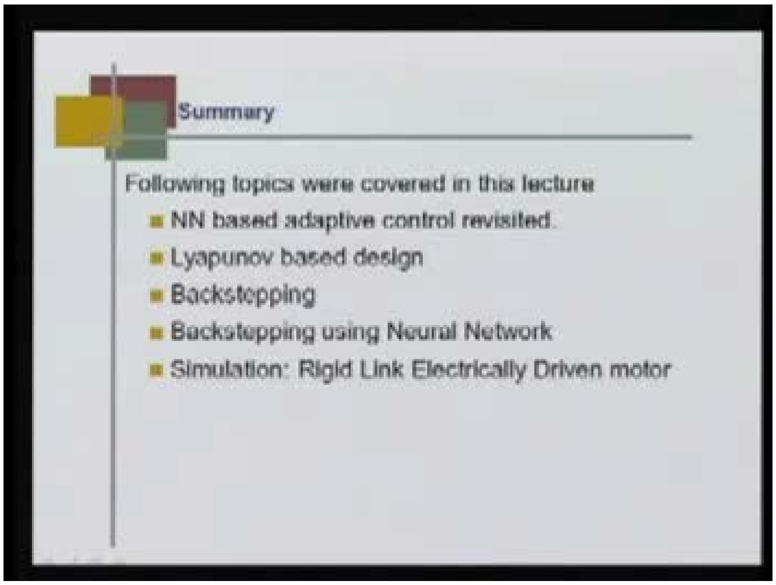
This is an error which is this control input and this is PD control actions and this back stepping control action.

(Refer Slide Time: 57:00)



The observations, the problem of weight initialization does not arise, since these are taken as zero; tuning algorithm ensures that weights are bounded; PD controller requires large gains which might excite high frequency unmodeled dynamics and NN control neural network based controller improves tracking performance.

(Refer Slide Time: 57:15)



The summary, what we discussed today is, NN based adaptive controller it was revisited, we discussed in the last class; Lyapunov based design; back stepping; back stepping using neural network and we demonstrated (57:34) of back stepping using rigid link electrically driven motor.

Thank you very much.