Intelligent Systems and Control

Prof. Laxmidhar Bahera

Department of Electrical Engineering

Indian Institute of Technology, Kanpur

Lecture 9

Direct Adaptive Control of Manipulators: Introduction

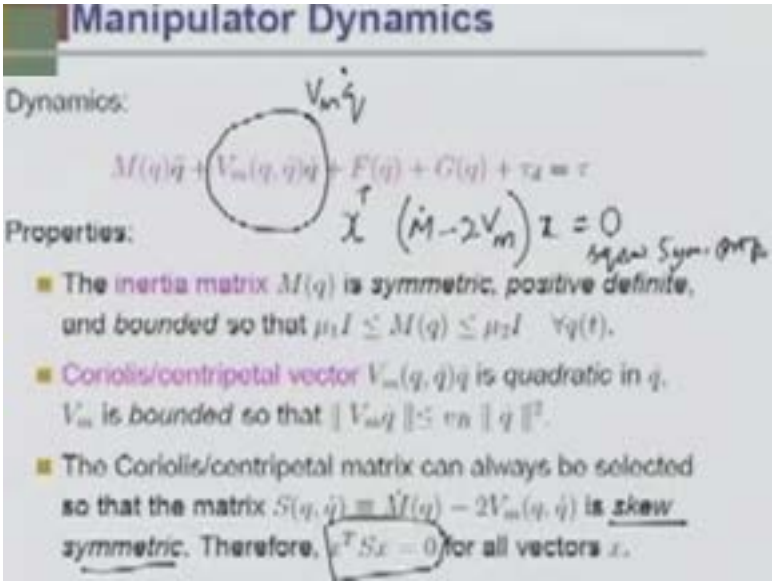
Direct adaptive control of manipulators - introduction: This will be the topic we will discuss today. We discussed various control systems while discussing various aspects of neural control. From today the series of lectures that we will be delivering, they will be unidirectional –it means the focus would be application to manipulators.

And we have already discussed what direct adaptive control is. Now this direct adaptive control of manipulators this will be specific that means this physical system is fixed. So how do we design control schemes for manipulators? So we will go step by step today. The notion of direct adaptive control for manipulators that would lead to a neural network based direct adaptive control and then we will continue to back stepping and singular perturbation based neural network control.

(Refer Slide Time: 02:02)

Topics that will be covered are: manipulator dynamics, direct adaptive control scheme, computed-torque control, adaptive control, robust control, neuro-adaptive control and summary.

(Refer Slide Time: 02:17)

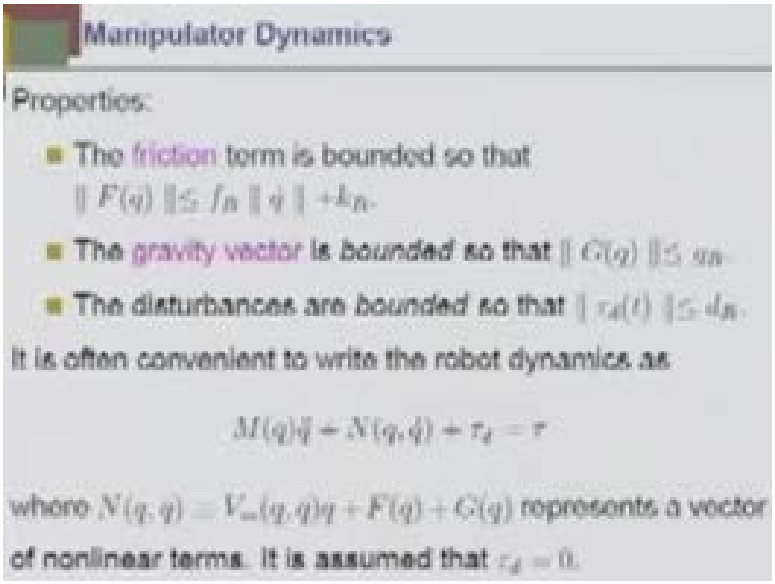


Manipulator dynamics: we have already been talking about this dynamics for a long time, so the usual dynamics is M q q double dot plus V_m, q dot F q dot G q plus tau_d is tau. In this, the inertia matrix M q this one is symmetric, positive definite and bounded. This is a property for all rigid link manipulator; this property is satisfied, so that M q the inertial matrix can be written as less than mu_2 I and greater than mu_1 I for all q t, q is angular position, q dot is angular velocity and q double dot is angular acceleration.

The second is Coriolis centripetal vector V_m is quadratic in q dot; V_m is bounded so that we can write V_m q dot norm is less than v_B norm q dot whole squared, so, here this V_m q dot is the total quantity. The Coriolis centripetal matrix can always be selected so that matrix S q q dot is M dot q minus 2 V_m q q dot is skew symmetric. Skew symmetric means if a matrix is skew symmetric -if S is a skew symmetric matrix, then this condition is satisfied: x transpose S_x is 0. In this case, if you compute M dot minus $2V_m$, this quantity and we multiply it by anything, it is 0.

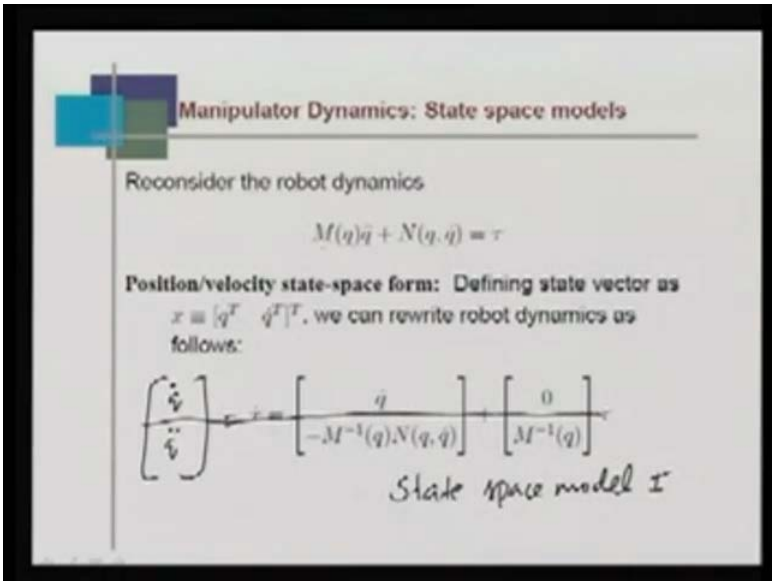
This is called skew symmetric property

(Refer Slide Time: 04:55)



The friction term is bounded; this is the friction term F q dot and it is bounded by this quantity $f_{B,}$ which is some constant q dot plus k_B. The gravity vector is bounded, so that G q norm is less than or equal to g_B; the disturbances tau_d -this is a disturbance term and this is a actual torque actuated at every joint, so disturbances are bounded so that this norm is less than d_B. It is often convenient to write the robot dynamics as M(q) q double dot plus N(qq) dot plus tau_d equal to tau. For practical purpose, we will just not consider disturbance torque for the moment during the analysis. Of course in simulation, we will use tau_d, but in analysis we will just remove tau_d. So, in practical purpose M(q) double dot plus N equal to tau, where N is Coriolis force due to friction and the force due to gravity. So this represents a vector of non-linear terms; it is assumed that tau_d is 0.

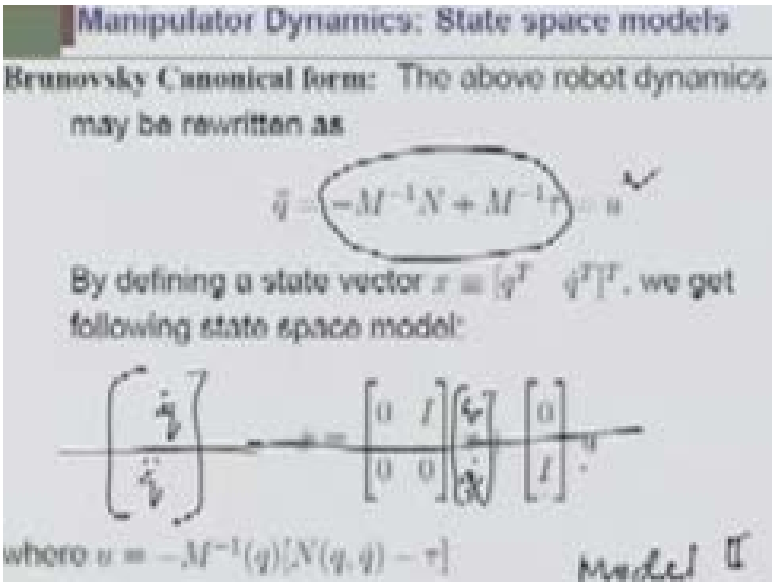
(Refer Slide Time: 06:18)



Reconsider now the robot dynamics, which we said M(q) q double dot is N(q) equal to tau where N consists of Coriolis forces, gravitational forces and forces due to friction.

Now, let us define the state space x as the vector consisting of position angular position vector and angular velocity. We can rewrite robot dynamics as in the state space form; this is one state space form and state space model you can easily see x dot is q dot; that is we can write here q dot q double dot, a vector. So x dot is q dot q double dot, so the q dot part I can put a line here. On the top line q dot q dot 0; that is always true. The second part is, q double dot you see the q double dot is this one obviously minus N(q) into M inverse. So M inverse minus N(q) which is this one and plus M inverse into tau, thus very straight forward.

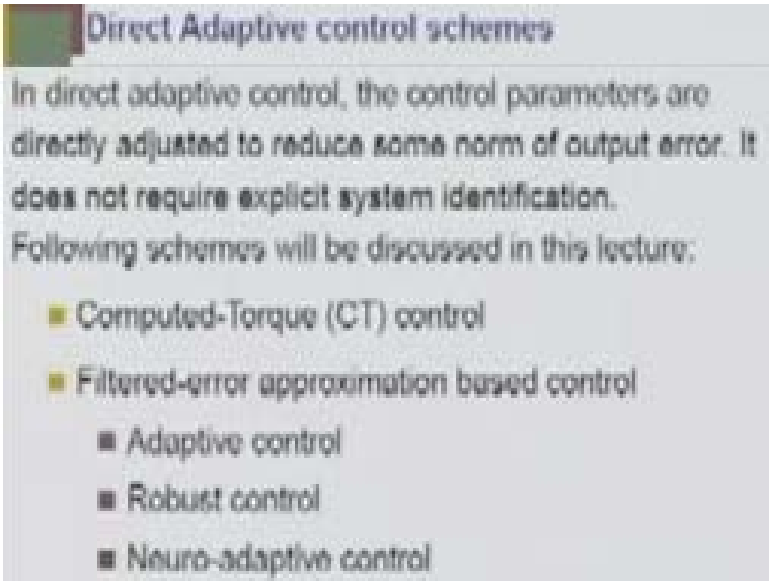
(Refer Slide Time: 08:24)



There is another state space model, this is model 2. In this model the Brunovsky Canonical form the same robot dynamics can be written as q double dot minus M inverse N plus M inverse tau equal to u. This quantity, if I say u simply, then this is my input now. How do I write q double dot? x is q dot q double dot equal to u.

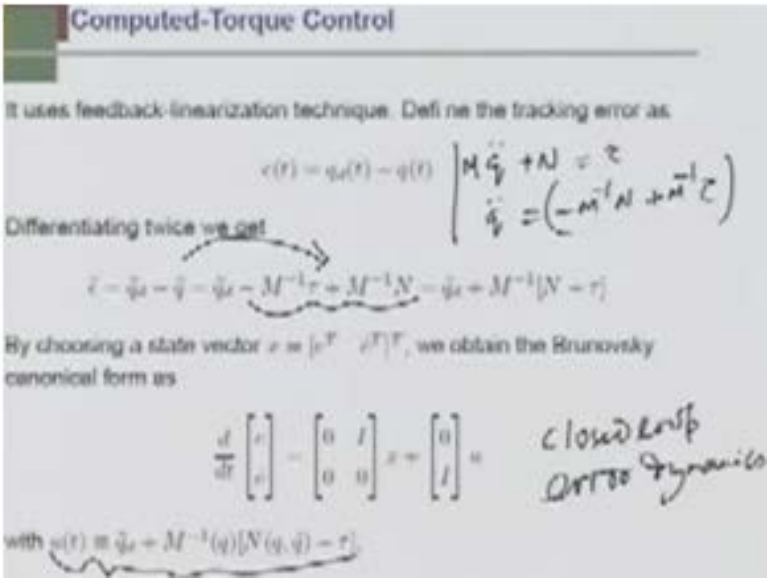
So, q double dot equal to u, you see that if I put this line here, q double dot is simply u, which is q double dot is u and q dot is simply. This x is again q and q dot. Obviously, q dot is, you can easily see, 0 q is 0 and I q dot is q dot. So, q dot is this and 0 and the top 1 is 0 u is 0, so q dot is q dot and q double dot is u. Simply by partitioning I can explain that this is and where u is this quantity, this is the virtual control u, this is the second model.

(Refer Slide Time: 10:29)



Direct adaptive control schemes: In direct adaptive control, the control parameters are directly adjusted to reduce some norm of output error. It does not require explicit system identification, so we do not do explicit system identification direct adaptive control that what is the system model parameter exactly in the same subsystem identification -we do not do. We will first talk about computed torque control, which is non-adaptive control, an exact control. We assume that the all the parameters of the robot manipulators are known, what is the meaning of computed torque control? We will go from that, we are assuming – relaxing the condition that some of the parameters are unknown, and then we go to adaptive control. I will not really focus on a robust control, because that is not our topic today. But I will just give you a little idea about robust control and then I will from adaptive control I will go to neuro-adaptive control and the difference between these two, we will try to understand today in this class.

(Refer Slide Time: 11:50)

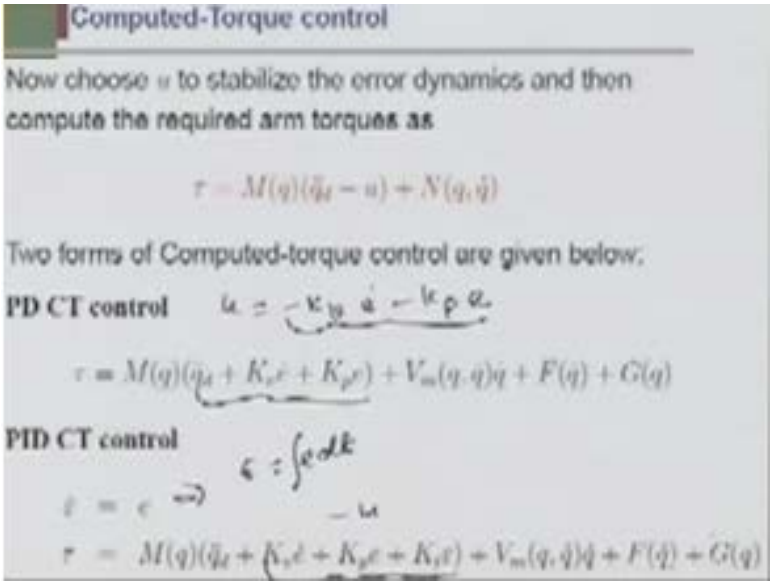


Computed torque control: It uses feedback linearization technique, define the tracking error e t, it is a desired trajectory minus the actual trajectory. Differentiating this quantity twice, we get e double dot is q_d double dot minus q double dot is q_d double dot minus M inverse tau. So, q_d double dot and q double dot is, you know that q double dot, writing properly here, we can write - we know that - the robot dynamics M q double dot plus N is tau. This is our simplified form; so I can write q double dot is minus M inverse N plus M inverse tau.

If I take this quantity and put it here instead of this, q double dot is placed here, you get q_d double dot M inverse N minus tau. By choosing the state vector x is e t e double dot t, we obtain the canonical form like this: d by dt e e dot 0 I 0 0 x 0 I u. This is state closed error dynamics. This is normally we say, closed loop error dynamics where the control action u t is q_d double dot plus M inverse q N q q dot minus tau. This is virtual control action, but from here we can easily see the tau can be computed from this. You can see that, u t is defined as this one, where the tau is computed.

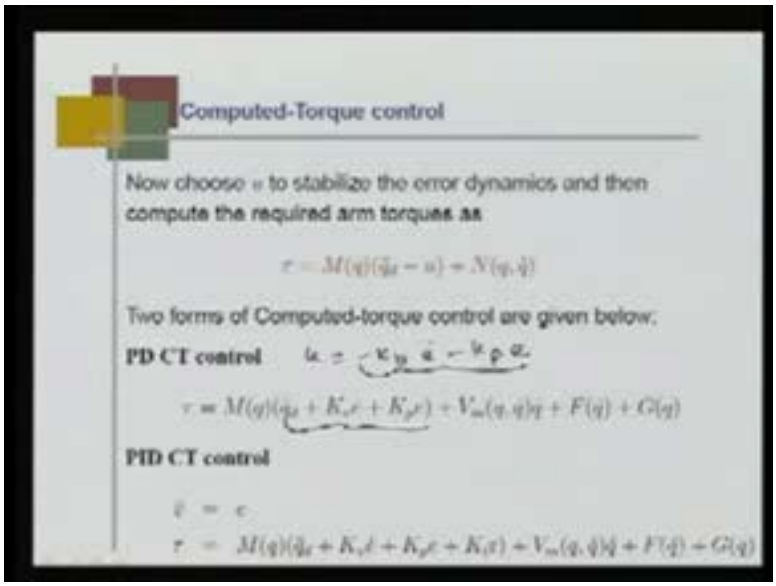
What we have to see is that, now the objective would be what should be the u for which the system is stable. You take that u and put it here and then find out what is tau. This is the reverse process.

(Refer Slide Time: 14:41)



Now, choose u to stabilize the error dynamics, and then compute the required arm torque as tau is M q q_d double dot minus u N q q dot. If I write now here (Refer Slide Time 14:54), I want to compute tau, then I take q_d double dot minus u this side and if I take the other side, M inverse q N q q minus tau, from there you find out tau is M q q_d double dot minus u plus N q q dot.

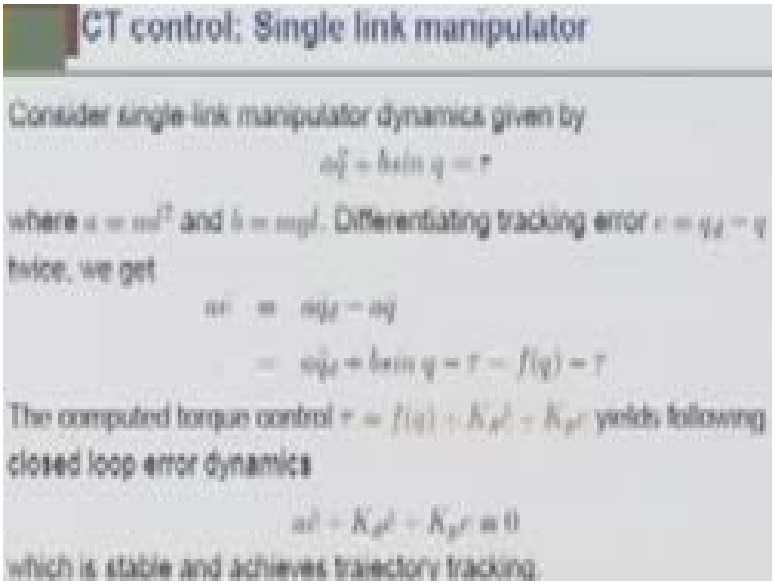
(Refer Slide Time: 15:25)



Now what is this u? If I want to stabilize this particular system using a PD controller, then my u is simply, u is minus K_v e dot minus K_p e, this is called PD controller, where my e is q_d minus q t. This will stabilize the system. By plugging these values here, I get tau is M q q_d double dot plus K_v e dot plus K_p e plus N q q dot is V_m q dot F q dot plus G q. But this control action I can actuate -provided all the parameters that is M, V_m, F, G all are known. The parameters that I had only designed are K_v and K_p. Similarly PID computed torque control would be, I just create another one for PID epsilon dot is e.

That is e d t integral, then your torque is this is my minus u. So M q is this is minus u. So u is minus K_v e dot minus K_p e plus minus K_i epsilon. This is the integral term because epsilon is e d t. This implies epsilon is, so this is the integral action. This is the proportional action and this is the derivative action. This is the PID controller. The PID computed torque control is given by this function.

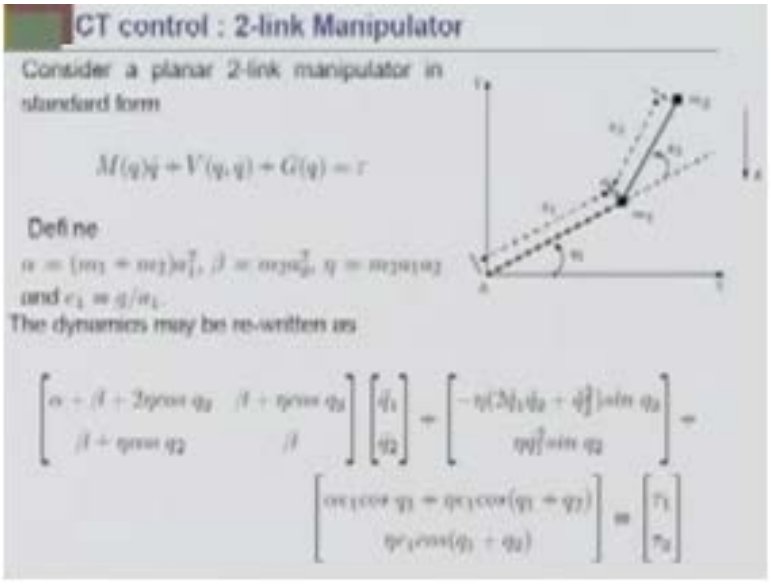
(Refer Slide Time: 17:47)



Now let us take a simple example: a q double dot b sine q is tau, which is a single link manipulator, a is ML squared, b is MGL. Now if I differentiate the tracking error e you get a e double dot is a q_d double dot minus a q double dot, which is a q d double dot remains and minus a q dot is b sine q minus tau, which is f q minus tau. So the computed torque control tau is obviously f q plus, this is u term k_de dot plus K_pe, yields the

following closed loop error dynamics. a e double dot k_de dot plus K_pe equal to 0,which is stable and achieve trajectory tracking. This is my closed loop error dynamics. Just like here we said that you select u put it here this u then this becomes closed loop error dynamics. So that is putting this u as a PD controller.

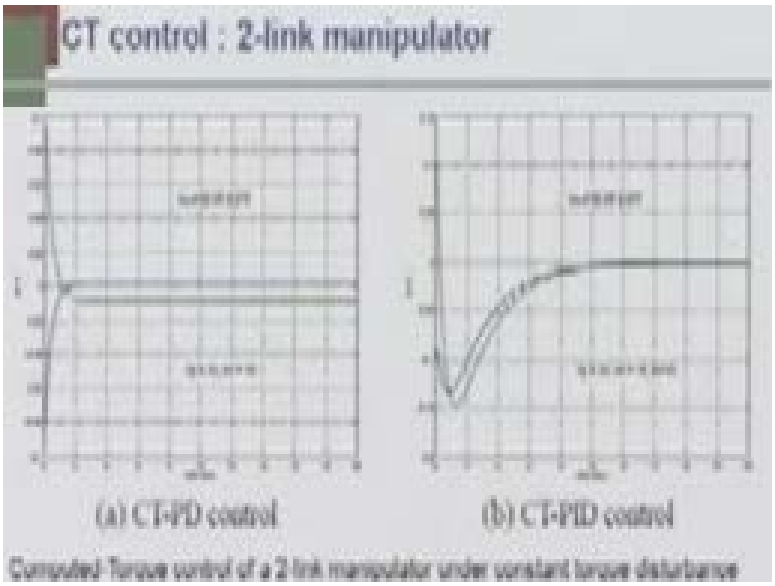
(Refer Slide Time: 19:14)



For single link manipulator, the dynamics is a stable dynamics. The computed torque control, if I take a 2-link manipulator link 1 and link 2, actually this can be written as in the usual form - M q q dot double dot V q q dot. In this, the q dot term, this is actually V_m q dot is multiplied with V together is the total Coriolis force, the gravity force equal to torque, where if I define alpha, this quantity where a1 is the link length 1 and a_2 link length 2, q_1 is the angle of link length 1 with the horizontal axis and q_2 is the link angle of link 2 aligned with link 1.

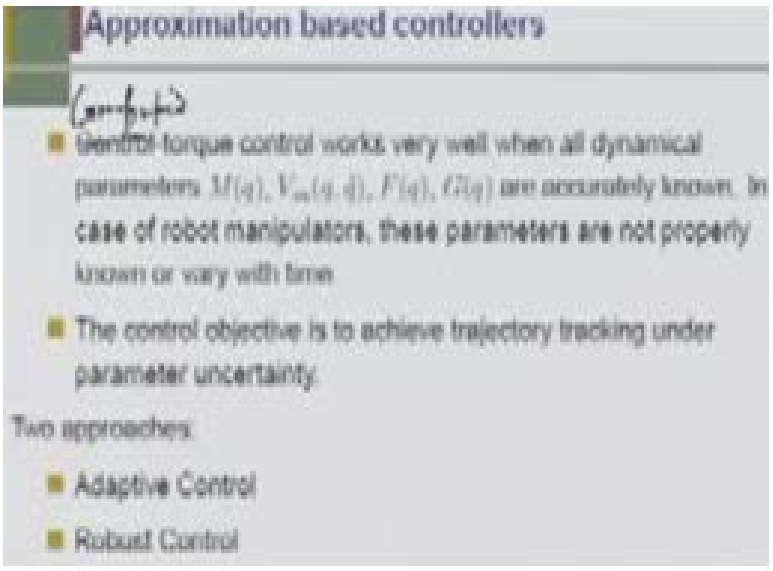
Then, if I define alpha beta eta as well as e1 in this particular manner, the dynamics can be written in this form, so this is your M matrix (Refer Slide Time: 20:42), this is your V matrix, Coriolis force and this is your gravity force. This quantity is tau_1, tau_2; the tau_1 is actuated here and tau_2 is actuated here.

(Refer Slide Time: 20:55)



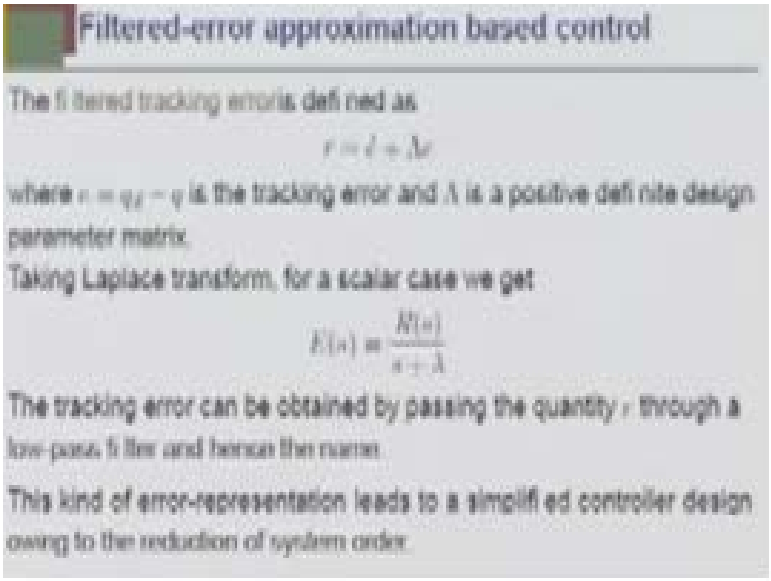
You see that if you do the simulation if you give some small disturbance, then you see that the PD controller has some steady state error. It does not exactly go to 0 whereas the PID controller even if any disturbance it goes to 0. That is, the steady state error is eliminated in case of PID controller; in case of PD controller it is not eliminated, which is expected.

(Refer Slide Time: 21:37)



Now, computed torque control works very well when all dynamical parameters M q V q F m F G are accurately known. In case of robot manipulators, these parameters are not properly known or vary with time. The control objective is to achieve trajectory tracking under parametric uncertainties. Two approaches that are normally followed are adaptive control and robust control.

(Refer Slide Time: 22:13)

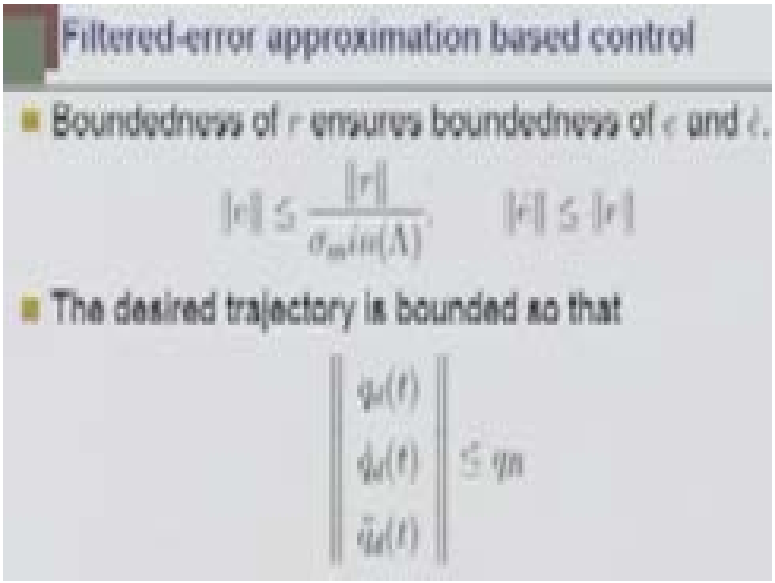


What is adaptive control? I will first introduce what is filtered error approximation based control. The filtered tracking error defined as r, this is the filtered tracking error, which is e dot the derivative of the error plus lambda error. So, lambda is a matrix, r is a vector and e is error.

So, e is q_d minus q, you know that q is a vector. If I have a 2-link manipulator, q dimension is 2; if I have a n link manipulator, dimension of q is n. The error is a vector; r is also vector; this is a matrix, obviously. If e is 2 dimensional, the matrix is 2 by 2, e is n dimensional, matrix is n by n. So, taking Laplace transform for a scalar case, if it is a scalar thing, then we can write out E s is R(s) upon s plus lambda. The tracking error can be obtained by passing the quantity r through a low pass filter, you can easily see this is a low pass filter, hence the name, this is filtered error tracking. This kind of error

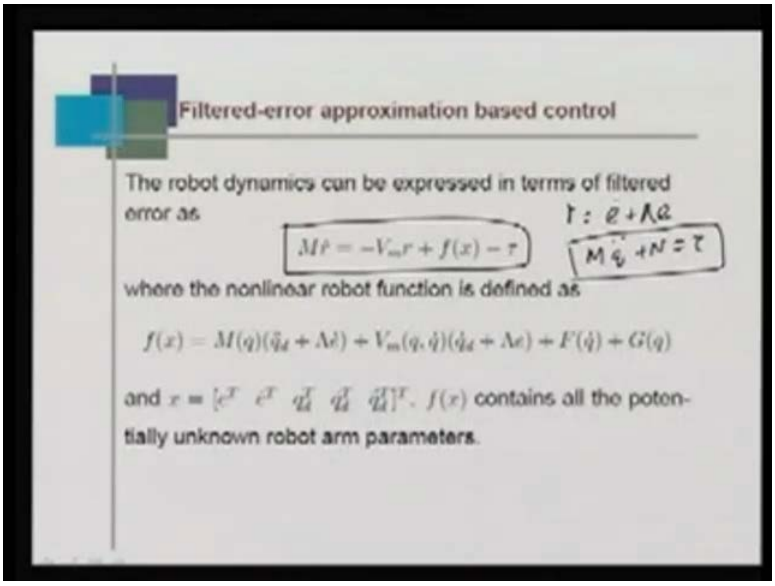
representation leads to a simplified controller design owing to the reduction of the system order.

(Refer Slide Time: 23:44)



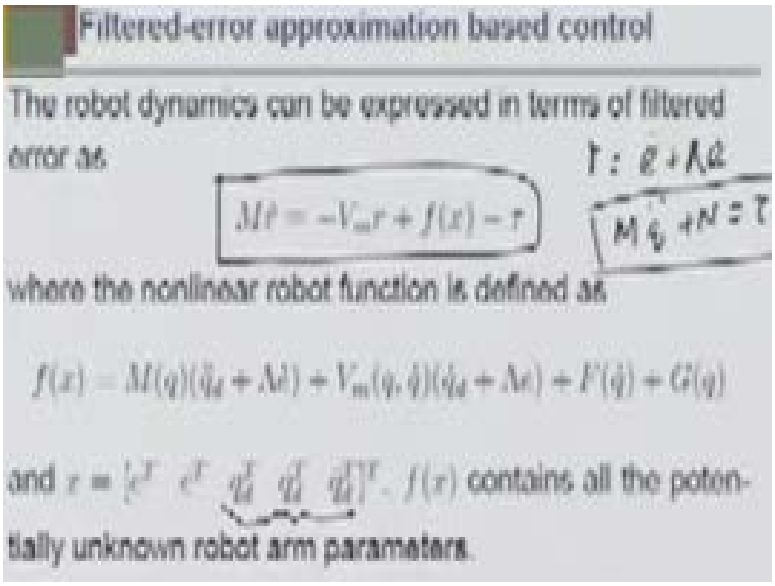
Boundedness of r ensures boundedness of e and e dot. This is just a certain idea, that if I know that r is bounded which is r is e dot, as we saw lambda e, this is bounded. Then, it ensures that e is bounded as well as e dot is bounded. So, boundedness is defined by this: e bounded is less than r bounded by sigma m, this quantity and the boundedness of e dot is less than boundedness of r. The desired trajectory is bounded so that q_d, q_d dot, q_d double dot, this is less than q_B so always we define a boundedness of the desired trajectory.

(Refer Slide Time: 24:50)



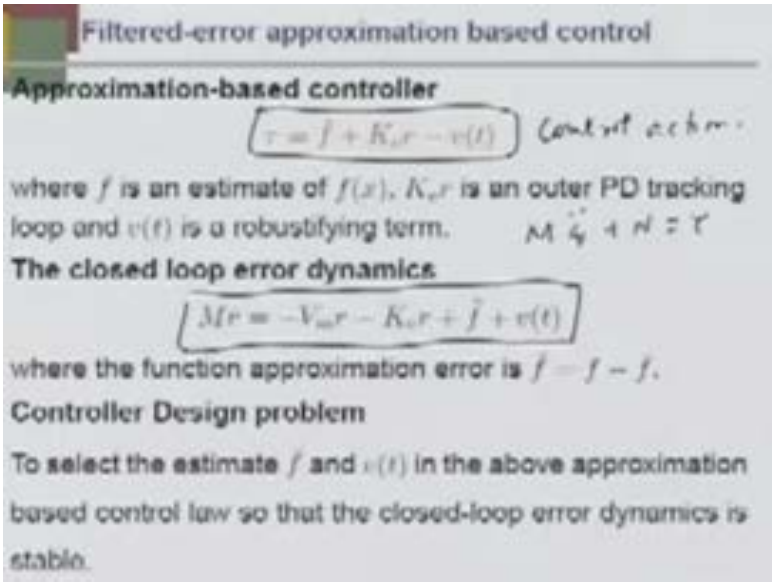
The robot dynamics can be expressed in terms of filtered error. Once I define r as e dot plus lambda e given that the robot dynamics M q double dot plus N is tau. This can be written as M r dot equal to minus V_mr plus f(x) minus tau, where the non-linear robot function is defined as f(x) is M(q), q_d double dot plus lambda e dot plus V_m q q dot q_d dot plus lambda e plus F q dot plus G q. We can derive this equation. If I define r and given the robot dynamics, this is the filtered error dynamics where this x here f(x) x is e transpose e dot transpose q_d transpose q_d dot transpose q_d double dot transpose. So, this state vector x in all known quantities because I know from the motion of the robot, I know the position and velocity.

(Refer Slide Time: 26:39)



I know what e and e dot are and the desired trajectories are already given, x is known this quantity is actually known. f(x) - it is not an unknown quantity, provided all these quantities M(q) V_mF and G are known. So, f(x) contains all the potentially unknown robot parameters, arm parameters - that is how we have separated - that in this f x, we have all these unknown parameters MV_m F and G.

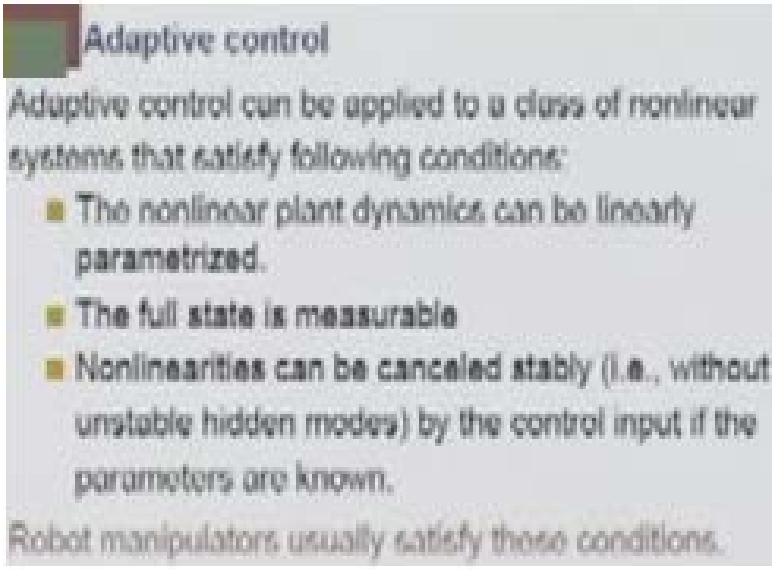
(Refer Slide Time: 27:20)



Approximation based controller: The previous one was M r dot is minus V_m r plus f(x) minus tau. So in that, tau is f hat plus K_v r minus v(t), where f hat is an estimate of f x, K_v r is an outer P D tracking loop and v(t) is robust term, If I assume this as my control action and replace this action in my term which is M q double dot N equal to tau.

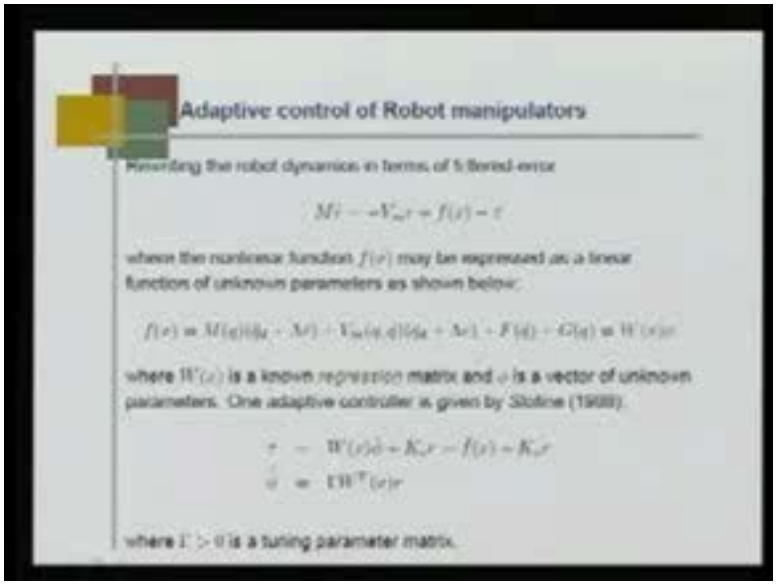
If I put this here, that is in this tau f hat plus K_vr minus v(t), then we eventually reach this closed loop error dynamics, which is M r dot minus V_mr minus K_vr plus f tilde plus v(t), where the function approximation error f tilde is f minus f hat. Controller design problem is to select the estimate f hat and this v(t) in the above approximation based control law, so that the closed loop error dynamics is stable.

(Refer Slide Time: 28:48)



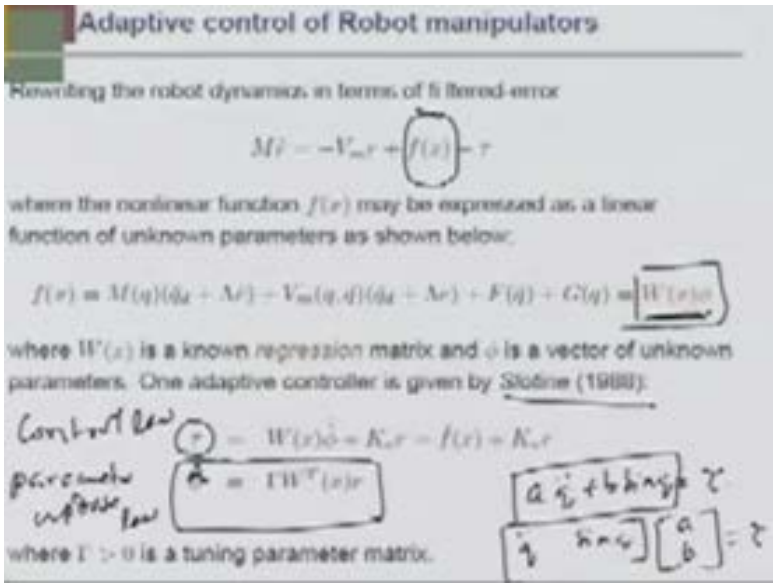
Adaptive control can be applied to a class of non-linear systems that satisfy the following conditions: The non-linear- plant dynamics can be linearly parameterized, the full state is measurable; non-linearity can be cancelled stably. That is, without unstable hidden modes by the control input if the parameters are known, robot manipulator usually satisfies these conditions.

(Refer Slide Time: 29:19)



Adaptive control of robot manipulators rewriting the robot dynamics in terms of filtered error which is M r dot is minus V_mr plus f(x) minus tau. This is the actual robot dynamics in terms of filtered error, where the non-linear function f(x) may be expressed as a linear function of unknown parameters, as shown below. We have already shown this f(x) and this can be written, if I can approximate this in terms of a regression matrix and an unknown parameter vector file.

(Refer Slide Time: 30:09)

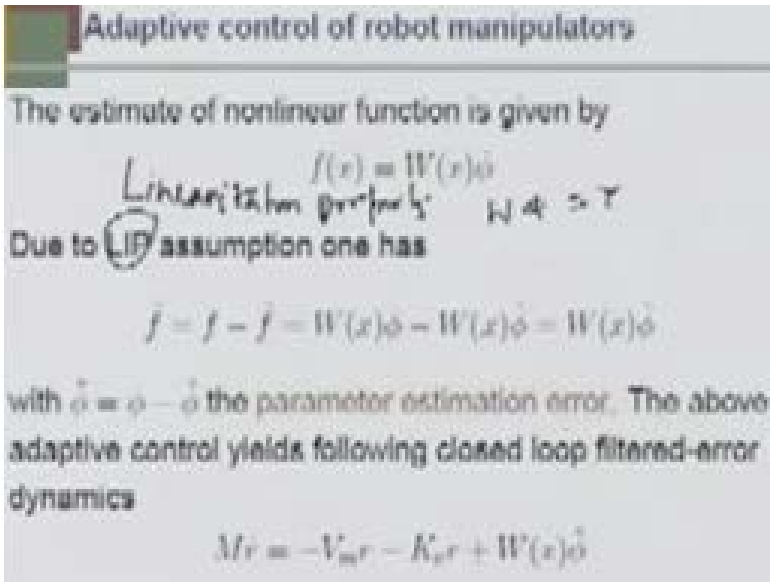


This is very important; I can show you what this W(x) phi means. If I write a single link manipulator which is a q double dot plus b sine q is tau. This particular theme can be written in W(x) file, you see that q double dot sine q. This is my regression matrix and these are my parameters a and b is tau. This is the meaning of W(x) and phi.

This f(x), which we are talking about, is a non-linear function. This, in case of robot manipulator can be represented in terms of a regression matrix W(x) and these are all known quantities provided if we, because q q dot and normally it is in this W x q double dot will not be there, is not, in this x example it is q double dot. But, normally it is all quantities -this regression matrix would have all the quantities. Those are all measurable state vectors and we can compute them. So, where W(x) is known as the regression matrix and tau is a vector of unknown parameters and in this case, unknown parameters are a and b. One adaptive controller which Slotine designed in 1998 is that control action tau is given by W(x) phi hat, the estimated parameter plus $K_v r$, very simple. This is kind of a P D kind of controller and this is our regression matrix into phi hat. This quantity is exactly F hat x plus $K_v r$. You can easily see that, if I do that here tau is exactly F x plus $K_v r$ when this system is stable. But, when it is F hat, then the system stability is for we have to again have a law for phi hat, because we do not know what phi is. So, phi hat dot is gamma which is a positive definite matrix W transpose x into r. This is the parameter update law.

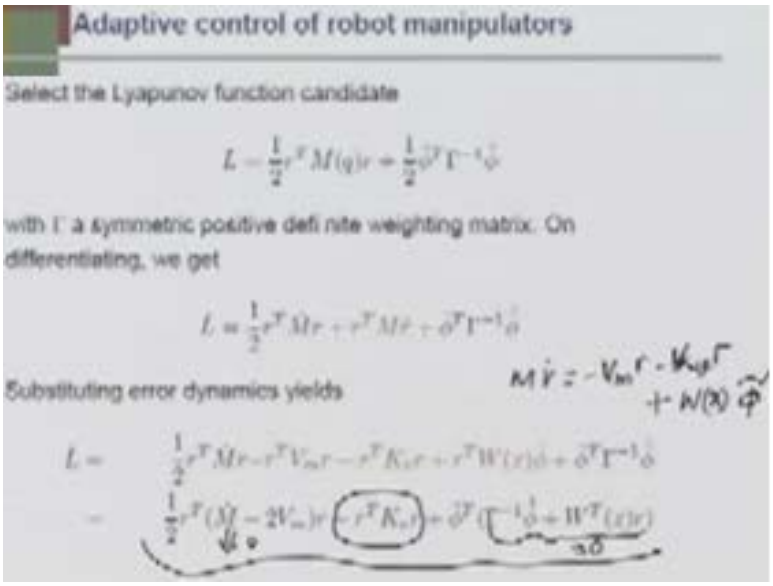
So, phi hat dot, this is parameter update law and this is control law. You will always find adaptive control means, we will have a control law and then this control law consists of a term called unknown parameter phi hat and that unknown parameter is computed using parameter update law, which is phi hat dot is Gamma W transpose x r.

(Refer Slide Time: 33:57)



The estimate of non-linear function is given by f hat x is W x phi hat due to linearization property, this is linearization. Linearization property, as I you told you earlier that robot dynamics can be written as a regression vector W into phi is tau. This is robot manipulator, any robot manipulator can be written in terms of phi is the vector of unknown parameter, W is the regression matrix. So, f tilde is of course W(x) phi tilde, phi tilde is phi minus phi hat, the parameter estimation error. The above adaptive control yields the following closed loop filtered error dynamics M r dot minus V_mr minus K_vr W (x) phi tilde. If I have this quantity phi tilde is 0 that means if I know the parameters, then this is stable dynamics, because M is a positive definite matrix. So r dot is V_m plus K_vr. Now select the Lyapunov function. But the point is that we have this quantity, so select the Lyapunov function candidate. Actually whether the system is stable or not, when this quantity is not there, we can do Lyapunov function analysis again to show that, this is stable. We will take the total thing and we will show that the system is stable by giving the parameter adaptive control below.

(Refer Slide Time: 35:56)

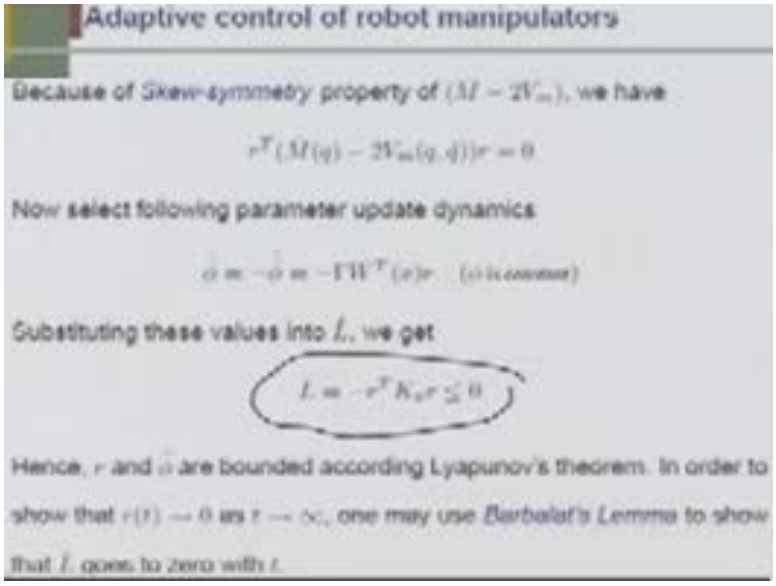


This is the Lyapunov function which is r transpose Mr plus half phi tilde transpose gamma inverse phi tilde. If I take the rate derivative of this Lyapunov function L dot, I get this quantity r transpose M dot r plus r transpose M r dot and then phi tilde transpose gamma inverse phi tilde dot. This is direct. Substituting the error dynamics r dot and what is error dynamics that is M r dot is minus V_mr minus K_vr plus f tilde, which is here Minus V_m r minus K_vr W x tilde, that is same actually; W x phi tilde, this is same as F tilde. If you look at now, this is my r dot. If I replace r dot here, M inverse also comes there and putting this quantity r dot, we get L dot is this quantity.

You know that, robot dynamics has a skew symmetric property. That means M dot minus 2 V_m is skew symmetric and hence r transpose this quantity into r, this goes to 0 due to skew symmetric and then L dot is minus r transpose K_vr. This is negative definite. This L dot can be made negative definite if I say this is 0.

From here, I get the parameter update law phi tilde dot is minus equal to minus gamma W transpose x r and phi tilde dot is minus phi hat dot. That is how we got…

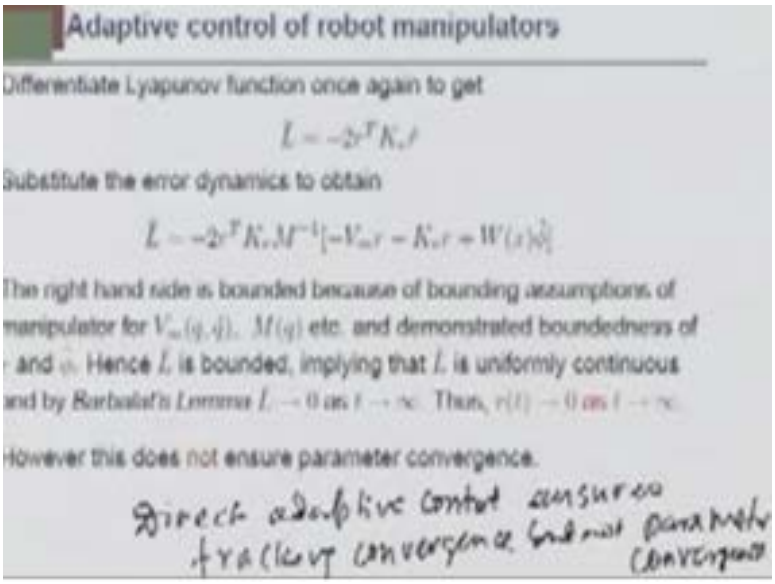
(Refer Slide Time: 38:29)



Because of skew symmetric property, this quantity is 0. Selecting the parameter update law phi tilde dot which I said minus phi hat dot is minus gamma W transpose r. obviously; my parameter update law was phi hat dot is gamma W transpose r.

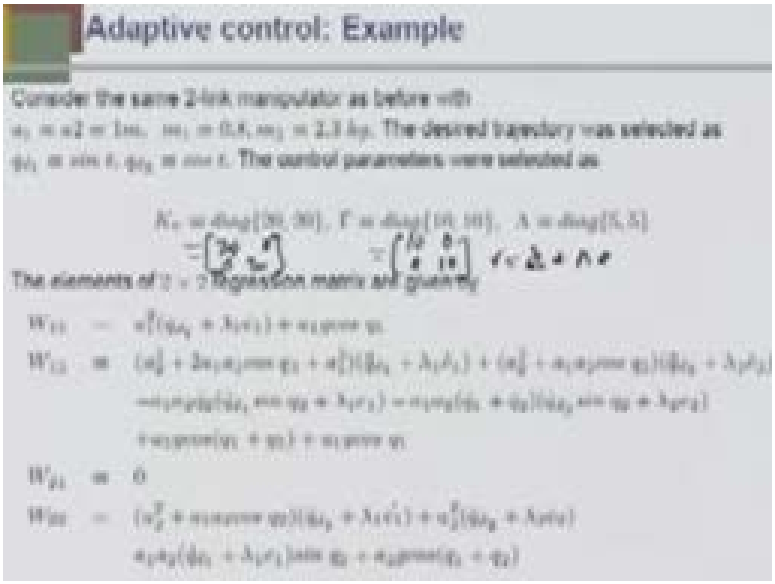
If this is the parameter update law and this is my control action, we proved that system is Lyapunov stable. So, L dot is a negative term. Hence r and phi tilde are bounded according to Lyapunovs theorem. In order to show that r t tends to 0 as t tends to infinity, one may use Barbalat's Lemma to show that L dot goes to 0 with t. So, with that r goes to 0, that means obviously e and e dot goes to 0. Differential at Lyapunov function once again, you get L double dot is 2 r transpose K_v r dot.

(Refer Slide Time: 39:33)



Substitute the error dynamics to obtain this quantity, the right hand side is bounded because of the bounding assumptions of the manipulator and demonstrated the boundedness of r and phi tilde. Hence, L double dot is bounded implying that L dot is uniformly continuous. By Barbalat's Lemma, L dot tends to 0 as t tends to infinity thus r t tends to 0 as t tends to infinity. However, this does not ensure parameter convergence. The direct adaptive control, as we have been saying that in direct adaptive control, the objective is not to, parameter update law will give you the exact parameter, but always the tracking error will go to 0. What we finally said that direct adaptive controller ensures tracking convergence but not parameter convergence.

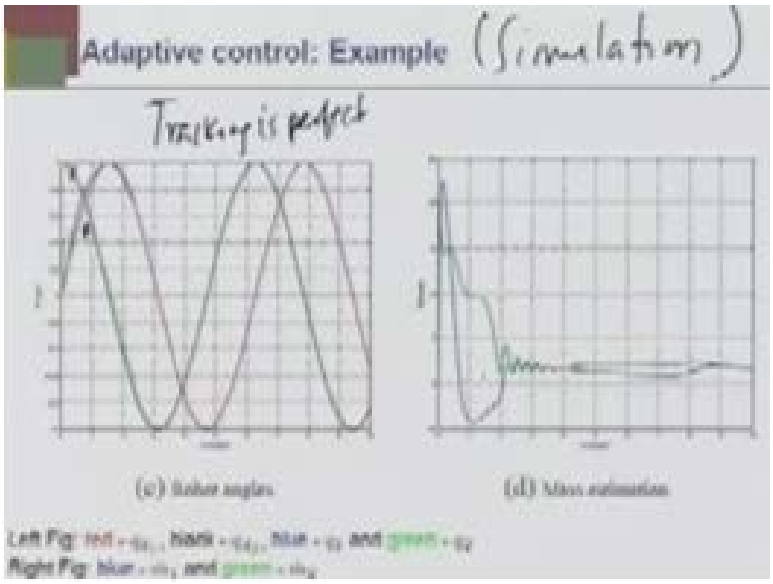
(Refer Slide Time: 41:04)



We will take an example of adaptive control; consider the same second link manipulator that we showed earlier. The parameters are taken as follows: The desired trajectory was selected as the first desired trajectory for link 1 as sine t and desired trajectory for a link 2 as cos t.

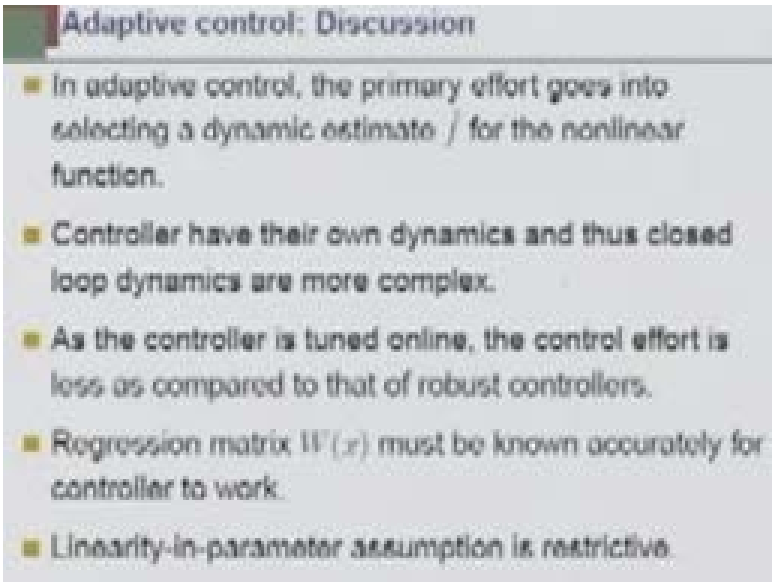
The P D parameters K_v is 20 diagonal 20, that means this is 20 0 0 20 and the parameter update law requires the vector matrix gamma, which is again 10 0 0 10 and this is a filtered tracking error which is e dot plus gamma e, so where gamma is diagonal phi. The elements of 2 by 2 regression matrix are given as W_{11} is given this, W_{12} this quantity, W_{21} you will find it to be 0, W_{22} is given by this quantity. For a 2-link manipulator, this regression matrix is quite involved.

(Refer Slide Time: 42:27)



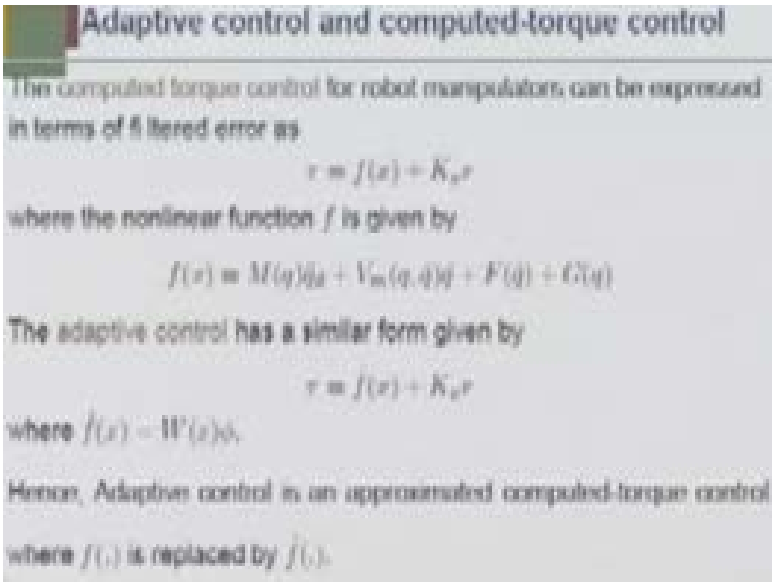
If you implement this in simulation - this is actually simulation not experiment - the red is the desired q_d 1 and blue is the actual q 1. In the beginning red and blue don't match; after sometime the tracking is perfect, very perfect. Similarly, the desired trajectory in link 2 is q_d 2 which is black and the green is the actual one. In the beginning, there is a lot of discrepancy which is obvious, but as time progresses green and black converge and there is no tracking error. Tracking is perfect or accurate and that the estimation of mass which is m1 and m2, these two things we assumed to be not known, the actual mass is point 8 and 2 point 3, but here you see they do not -point 8 and 2 point 3 do not converge and actually they have become same here. The estimated quantities do not converge with the actual quantities but tracking is perfect. That is the objective of adaptive control.

(Refer Slide Time: 44:31)



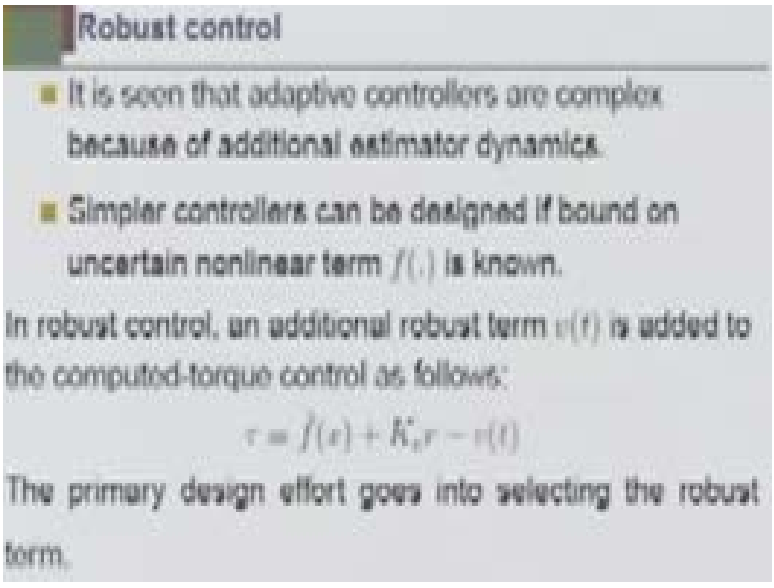
Finally, going back to the discussion on adaptive control; in adaptive control, the primary effort goes into selecting a dynamic estimate f hat for the non-linear function. Controller has its own dynamics and thus closed loop dynamics are more complex. As a controller is tuned online, the control effort is less as compared to as that of robust controllers. Regression matrix W x must be known accurately for controller to work, linearization property assumption is restrictive, because this type of controller that we talked about is only valid for robot manipulator, because that follows linearization property and this linearization property is not valid for all kind of non-linear system. That is why, whatever we will be discussing- in the following lectures they are only meant for robot manipulators, because there is a certain advantage but in general, it is not true.

(Refer Slide Time: 45:29)



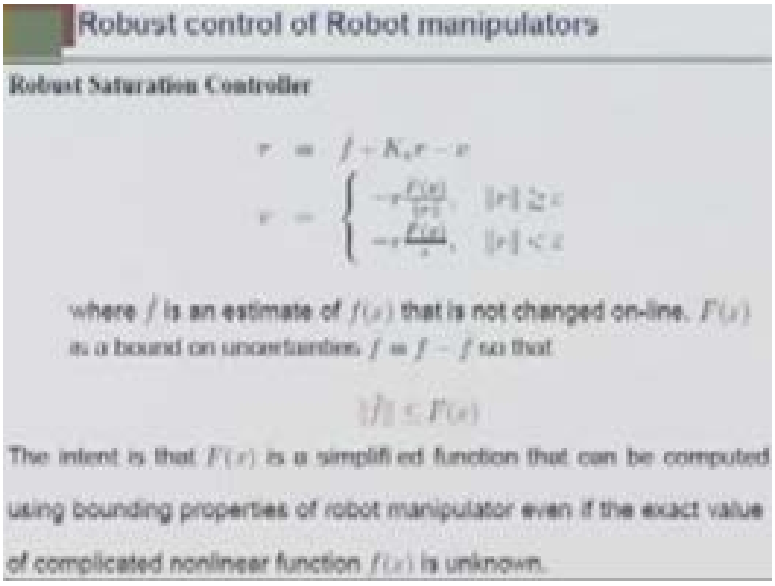
The computed torque control robot manipulator can be expressed in terms of filtered tracking error as f(x) K_vr where f(x) is this quantity and adaptive control is f (x) hat K_vr. This is my computed torque control and this is my adaptive control, where f(x) is replaced by f hat. Adaptive control is an approximated computed torque control where the actual f is replaced by the estimated f and for that, we have a parameter update law that is given by phi hat dot is gamma W transpose into r.

(Refer Slide Time: 46:09)



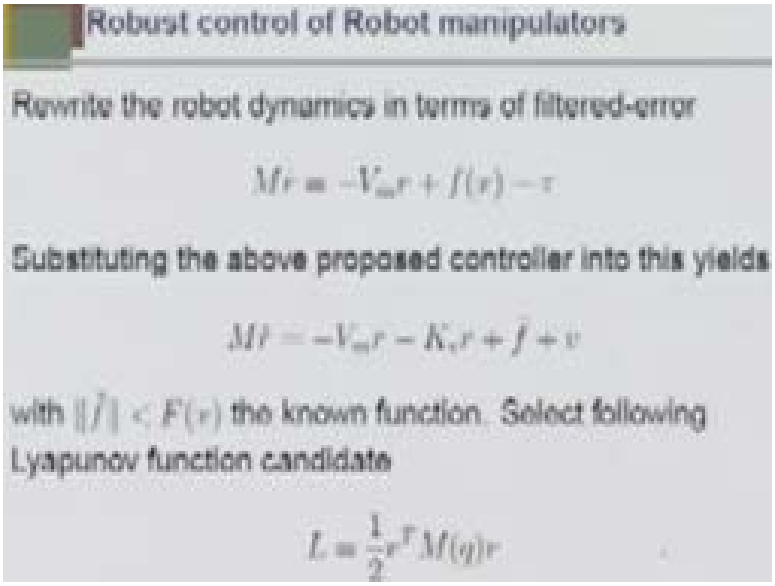
It is seen that adaptive controllers are complex because of additional estimator dynamics. Simpler controllers can be designed if bound on uncertain non-linear term f is known. In robust control, an additional robust term v(t) is added to the computed torque control as follows. This is the connection, this was our adaptive controller and then we found out. But here, we add a robust term minus v t. Primary design effort goes into selecting this robust term v t, so that the system is stable.

(Refer Slide Time: 47:03)



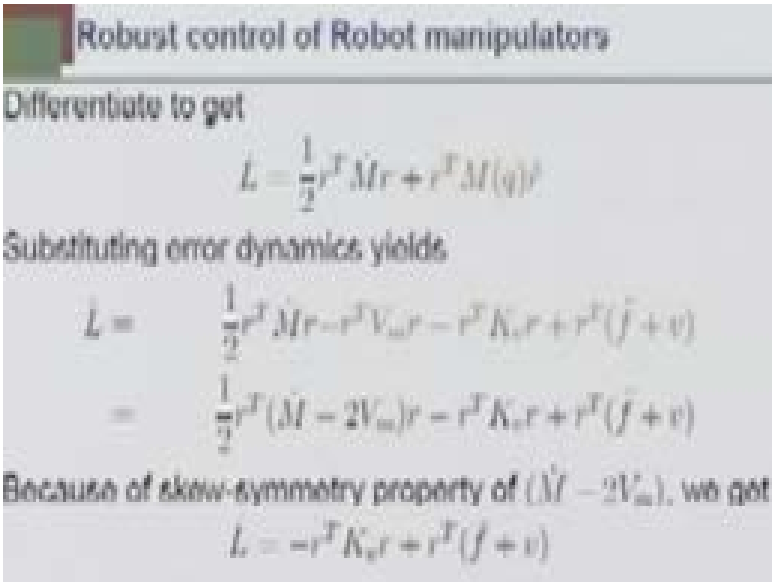
This is our controller action, this is my v and f is an estimate of f(x) that is not changed online. F(x) is a bound on certain uncertainties. If I know this bound on uncertainties -the intent is that F(x) is a simplified function that can be computed using bounding properties of robot manipulator even if the exact value of complicated non-linear function f(x) is unknown.

(Refer Slide Time: 47:32)



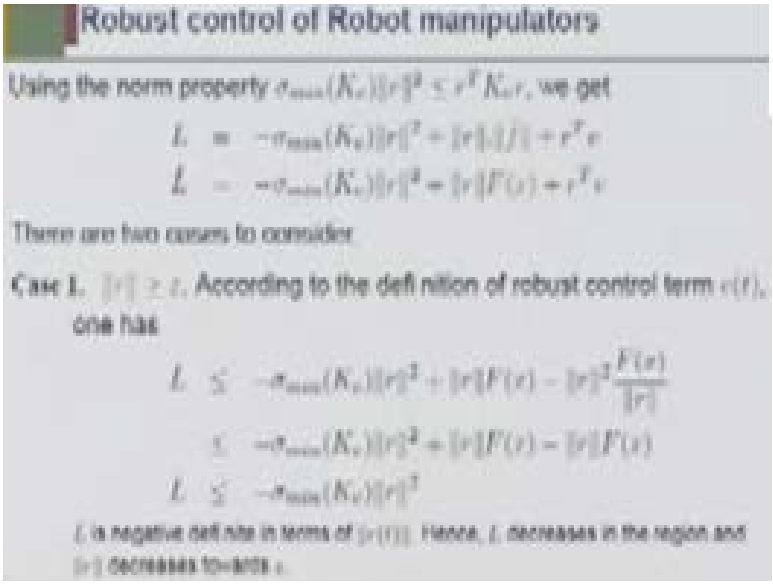
Rewrite the robot dynamics in terms of filtered error, we have already shown that and substituting the above proposed controller into this, we get this closed loop error dynamics and taking the Lyapunov function is half r transpose M(q) r.

(Refer Slide Time: 47:51)



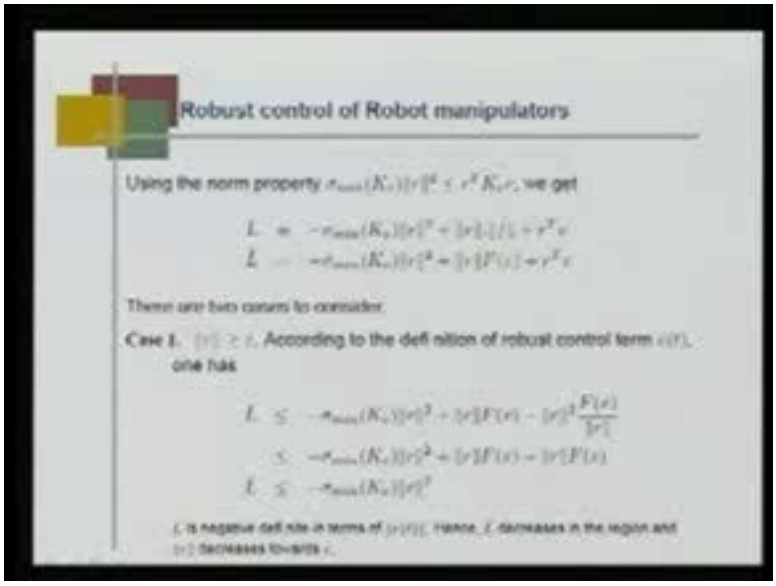
When differentiating, you will get this quantity because of skew symmetric property goes to 0. From here, we can find out what is V?

(Refer Slide Time: 48:06)



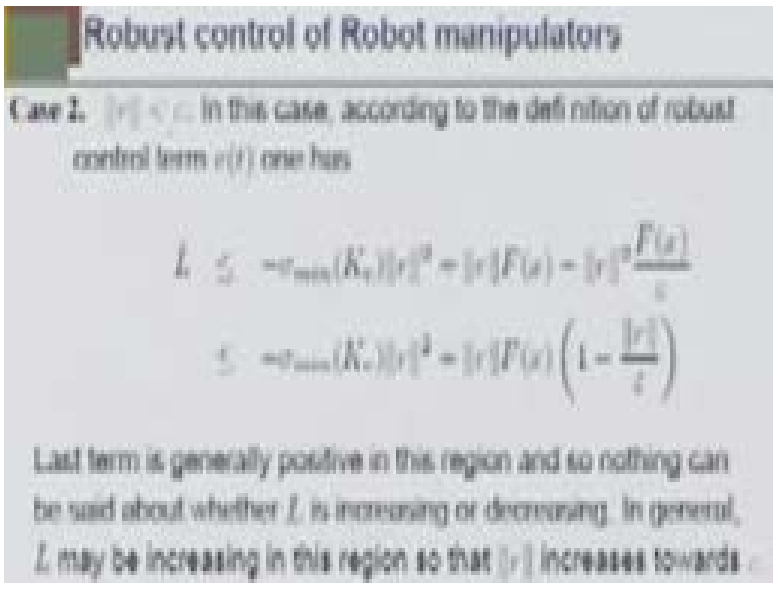
Using the norm property, we get L dot is this quantity and this quantity. There are two cases to consider. I am just going little faster because I have to go to the neural network based controller. There are two cases to consider, the boundedness r is greater than equal to epsilon.

(Refer Slide Time: 48:06)



According to the definition, robust control term v t 1 has L dot, is this quantity which is less than this quantity, this is same this is same and this is replaced by simply r f(x). This cancels out, so this L dot is only this quantity which is negative definite

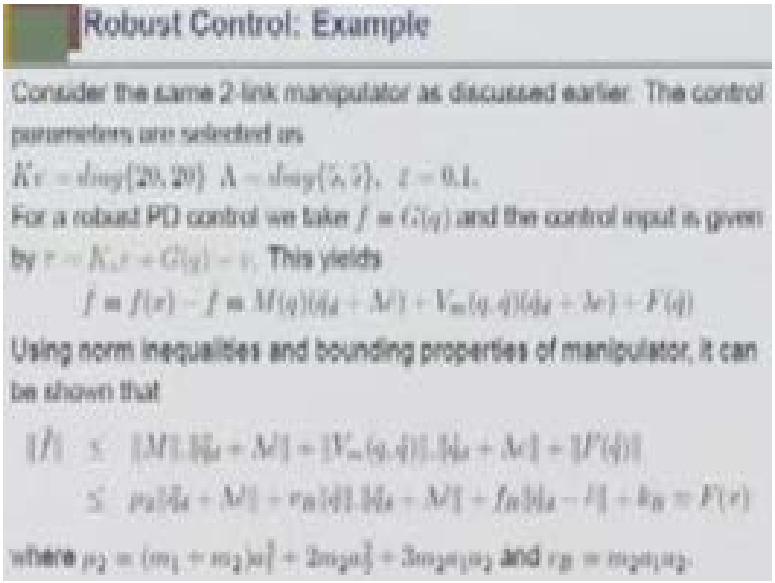
(Refer Slide Time: 48:59)



Case 2: When it is less than epsilon, in this case according to definition of robust control term v t 1 has again this quantity, which is this quantity plus r f x 1 minus r bounded upon

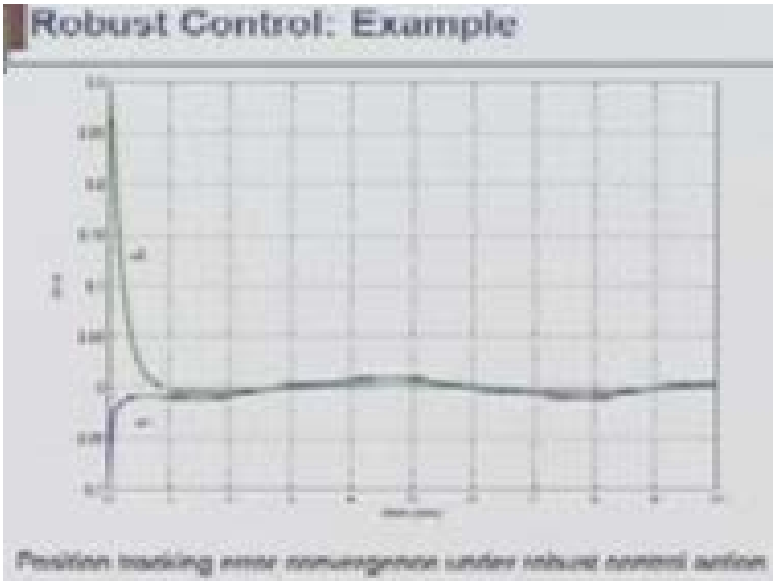
epsilon. Last term is generally positive, this quantity and this region and nothing can be said about whether L is increasing or decreasing. In general L may be increasing in this region so that r increases towards epsilon.

(Refer Slide Time:49:35)



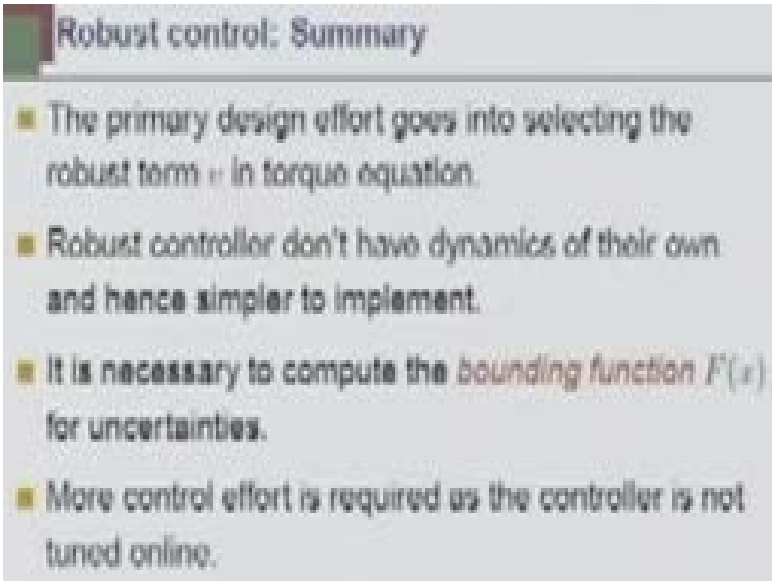
Robust controller example: Consider the same 2-link manipulator as discussed earlier. The control parameters are selected as with epsilon as point 1; for a robust PD controller we take f equal to G q; control input is given as this quantity. If you implement it, given an initial trajectory disturbance finally, this converges to 0.

(Refer Slide Time: 49:58)



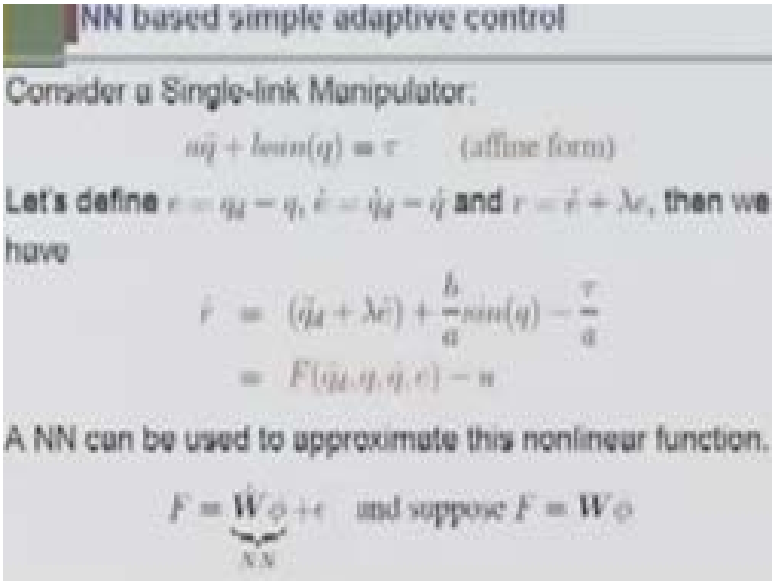
Position tracking error convergence under robust control action.

(Refer Slide Time: 50:11)



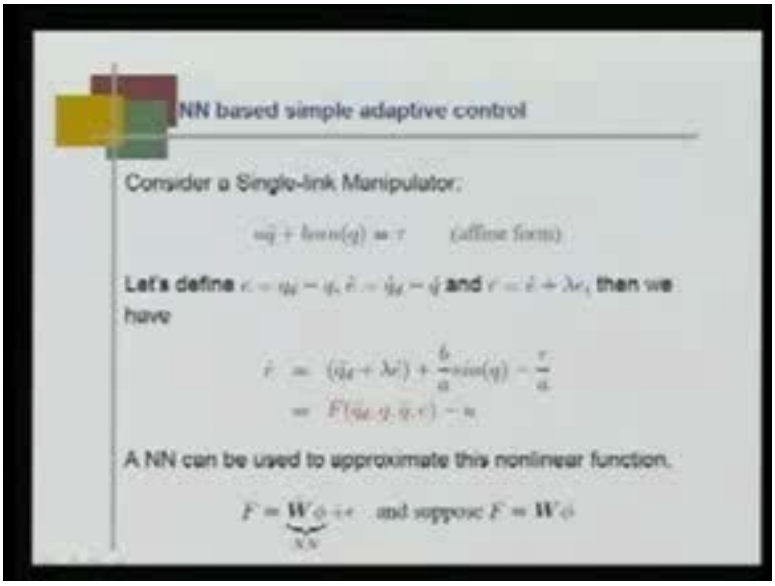
The primary design effort goes into selecting the robust term v in torque equation, robust controllers do not have dynamics of their own and hence simpler to implement. It is necessary to compute the bonding function F(x) for uncertainties. More control effort is required as the controller is not tuned online.

(Refer Slide Time: 50:30)



This is the final part of this lecture. We will be talking about neural network based adaptive controller, so that how we can do the same direct adaptive control using neural network. Take a simple example a q double dot plus b sine q is tau, which is a single link manipulator defined as usual e is q_d minus q, e dot is q_d dot minus q dot and the tracking filtered tracking error r is e dot plus lambda e

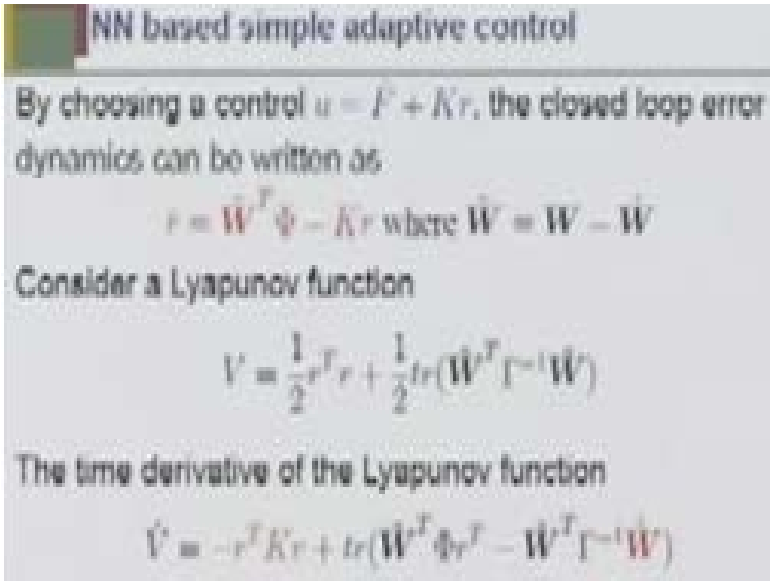
(Refer Slide Time: 50:30)



We have r dot is q_d double dot plus lambda e dot plus b upon a sin q minus tau upon a which is my regression matrix F minus u. So the N N can be used to approximate this

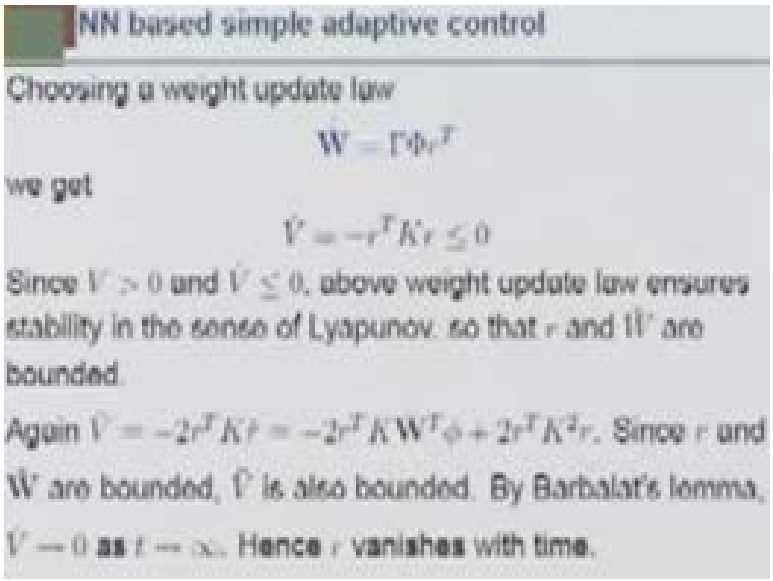
non-linear function this F can be written as W hat phi plus epsilon and suppose F is W phi.

(Refer Slide Time: 51:44)



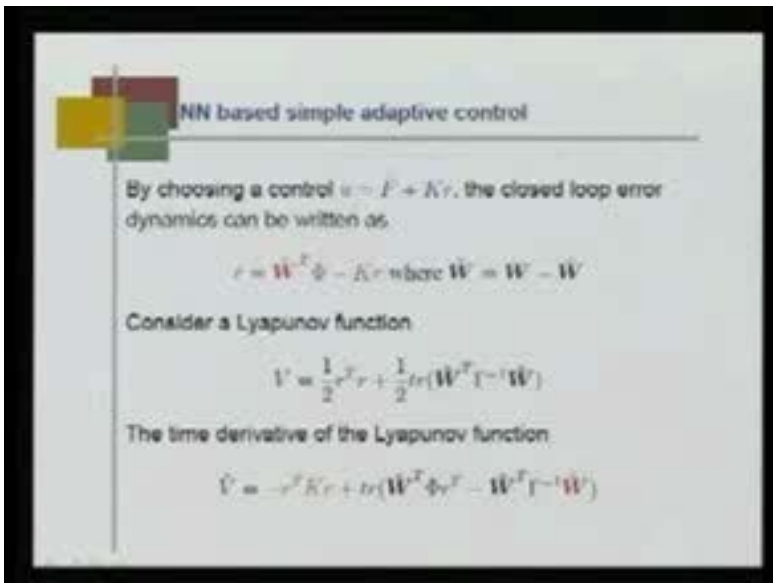
By choosing a control law, u is F hat plus Kr the closed loop error dynamics can be written as r dot is W tilde transpose phi minus Kr. What I am saying here is that this non-linear function is represented by a neural network. That is very simple, because W is the weight of the neural network and phi is the input to the neural network. The closed loop error dynamics can be written as r dot is W tilde transpose phi minus Kr. If I now take Lyapunov function V in this form, which is r transpose r W tilde transpose gamma inverse W tilde, time derivative of Lyapunov function is - minus r transpose K rand then this is trace of this quantity. We can make V dot is minus r transpose Kr, provided this term is m- with 0.

(Refer Slide Time: 53:10)



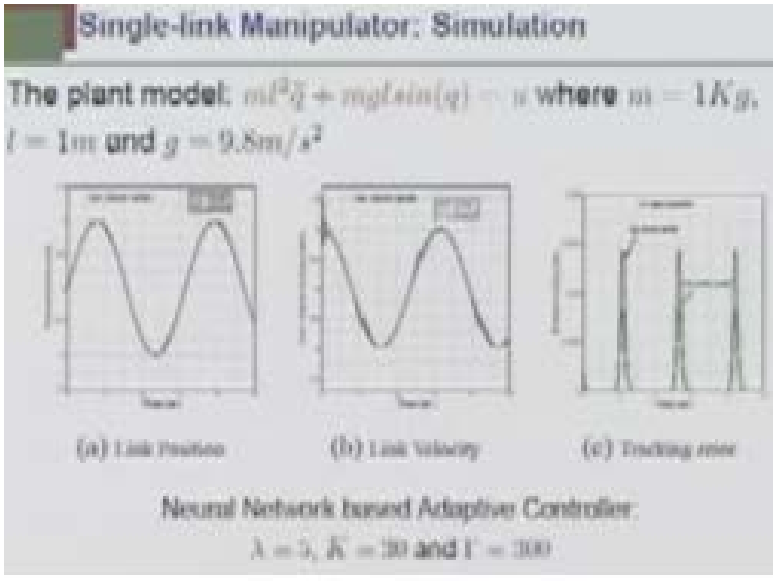
That can be achieved W tilde dot is gamma phi r transpose which you can easily see.

(Refer Slide Time: 53:17)



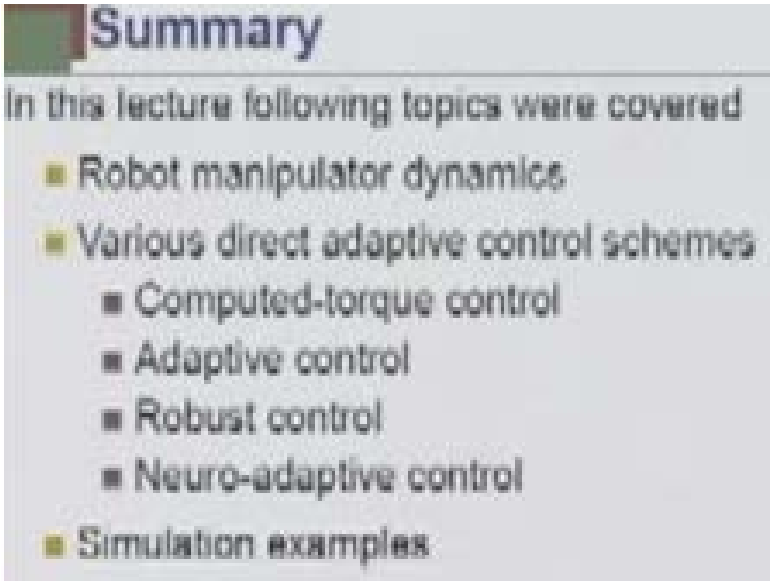
I can take this gamma W tilde transpose phi r transpose is minus gamma inverse W tilde dot. So, W tilde dot is gamma phi r transpose, which is this one. This is the weight update law for neural networks. Using that we saw V dot is negative definite V double dot is this quantity and since r and W tilde are bounded, V double dot is also bounded by Barbalat's lemma .V dot tends to 0 as t tends to 0; hence r vanishes with time. As r vanishes with time, so is the derivative of error.

(Refer Slide Time: 54:08)



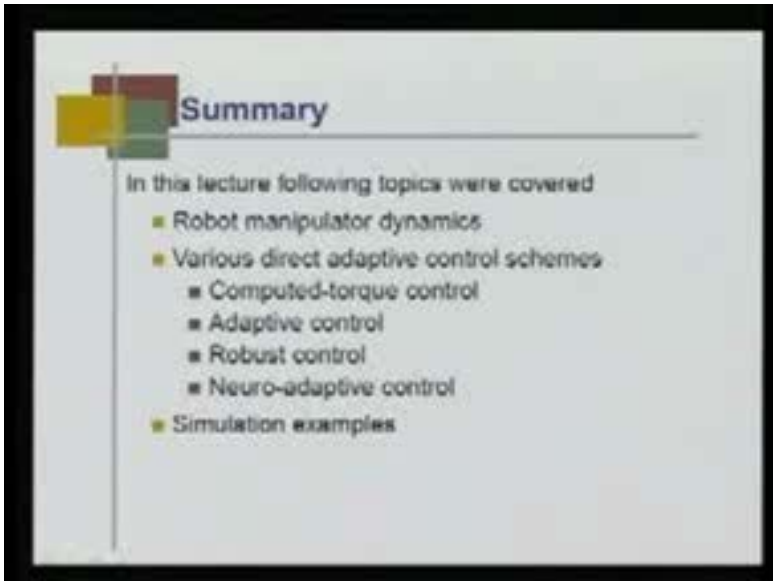
If we now do the simulation taking a neural network based adaptive controller, where the neural network represents this quantity, W hat phi. Taking the single link manipulator, we see that actual and desired very closely following and the velocity actual and desired also following and tracking error is very small, point 0 0 means N to the power minus 3 order. Taking these parameters, we got these simulation results. This is control input in the link and this is the function approximation, this is a PD controller. Finally the summary in this lecture. Following topics were covered.

(Refer Slide Time: 55:05)



Robot manipulator dynamics, various direct adaptive control schemes, computed torque control adaptive control, robust control, neuro-adaptive control, simulation examples

(Refer Slide Time: 55:18)


Summary
In this lecture following topics were covered
Robot manipulator dynamics
Various direct adaptive control schemes
Computed-torque control
Adaptive control
Robust control
Neuro-adaptive control
Simulation examples