Nonlinear & Adaptive Control
Dr. Shubhendu Bhasin

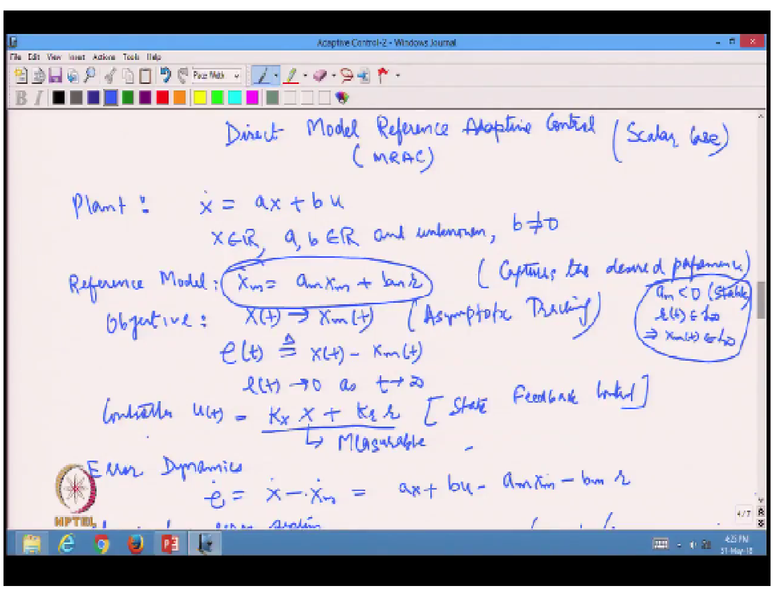
Department of Electrical Engineering
Indian Institute of Technology - Delhi

Lecture – 04
Model Reference Adaptive Control Part - 2

Welcome everyone to lecture 4 of this course nonlinear and adaptive control. In the last lecture, we had introduced model reference adaptive control. Specifically, we had done the direct model reference adaptive control. So we had done 2 flavours of adaptive control, direct adaptive control and indirect adaptive control and I had mentioned before in direct adaptive control, there is an online parameter estimator which directly estimates the controller parameters which are then used to update the controller.

In indirect adaptive control, the online parameter estimator estimates the parameters of the, of the system or the plant and that is used to then compute the controller parameters through an algebraic relation and then that is used to update the controllers, controller. So these are the 2 main flavours in adaptive control.

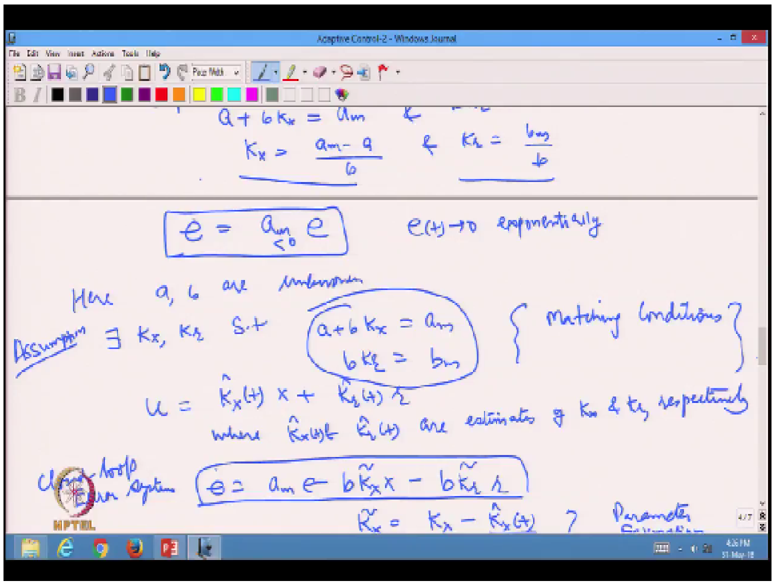
(Refer Slide Time: 01:21)



So in the last class, we had covered the direct model reference adaptive control case. So if we do a quick recap, the objective of model reference adaptive controller is for, is to design a controller such that the plant state tracks the state of the reference model. The reference model can be chosen by the user.

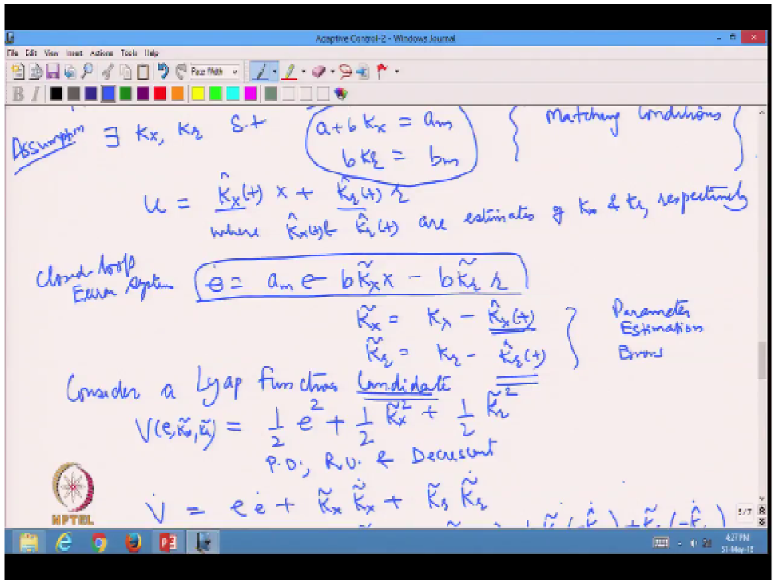
It captures the desired performance, desired behaviour of that you expect from the plant. So and of course, you would want to select a reference model which is stable and input which is bounded. So the task is to design a controller U of t such that x of t asymptotically tracks xm of t.

(Refer Slide Time: 02:13)



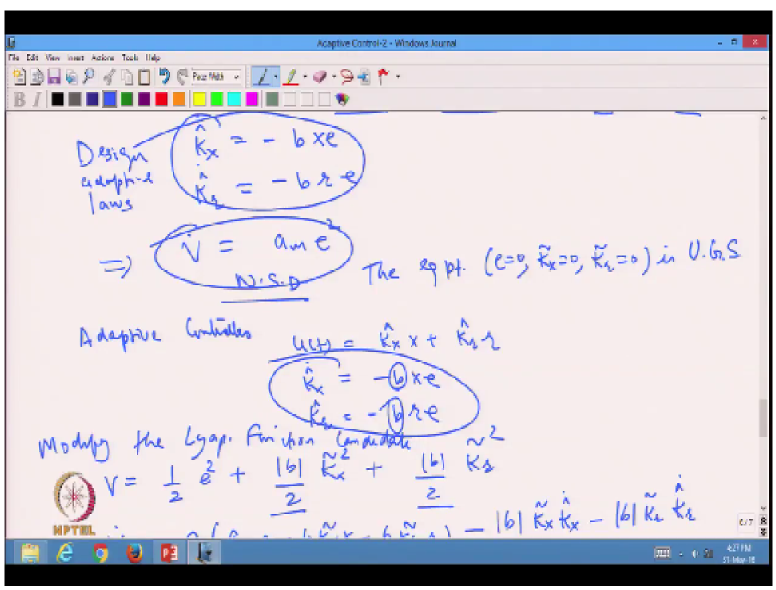
The last time, we had designed a direct adaptive controller and we found that the adaptive update laws. So first thing is that the matching conditions have to be satisfied. For the scalar case, we said that these condition are always satisfied but because the plant parameters a and b are unknown, we cannot find kx and kr from the matching conditions.

(Refer Slide Time: 02:40)



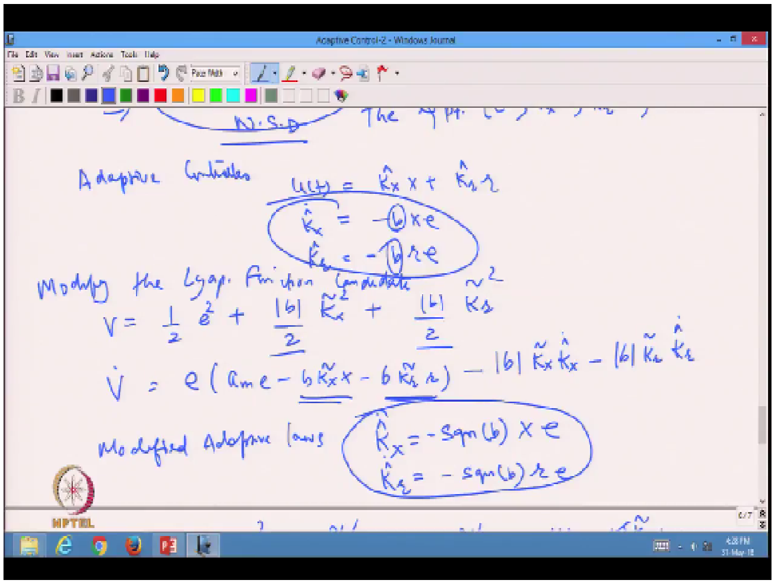
But what we can at least do is in the controller, we can assume that we had estimates of these kx and kr, kx hat and kr hat. Then the task of course remains to find out the adaptive laws for kx hat and kr hat. So these adaptive laws are found from the Lyapunov stability analysis which is not just a tool to analyse stability, it is also useful for designing controllers as we saw in the last class. So we had chosen Lyapunov function candidate and then designed the adaptive laws like this.

(Refer Slide Time: 03:21)



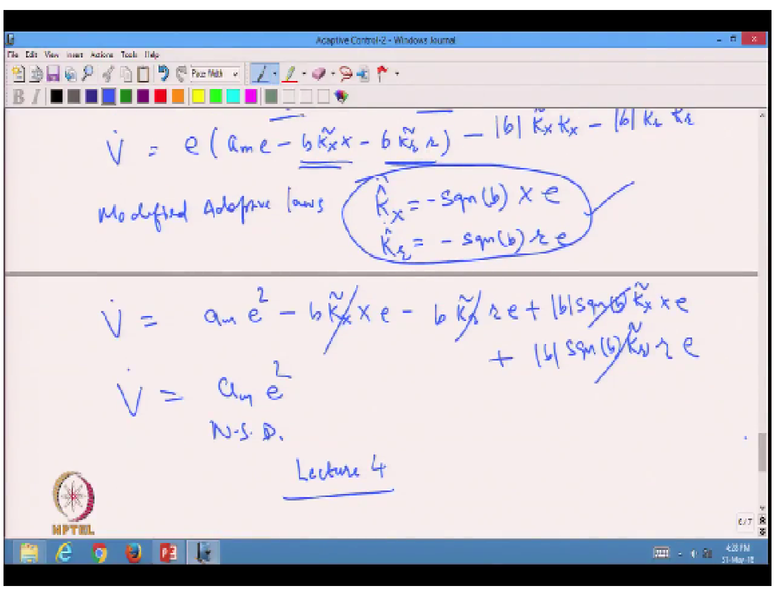
And we found here that these laws are dependent on the unknown plant parameter b. So if the plant parameter b is known and a is unknown, this design would still work because we are able to prove that v dot is negative semi-definite which means that the equilibrium point is uniformly globally stable. Of course, the objective that e goes to 0, we still need to prove but at least the stability is proved.

(Refer Slide Time: 04:07)



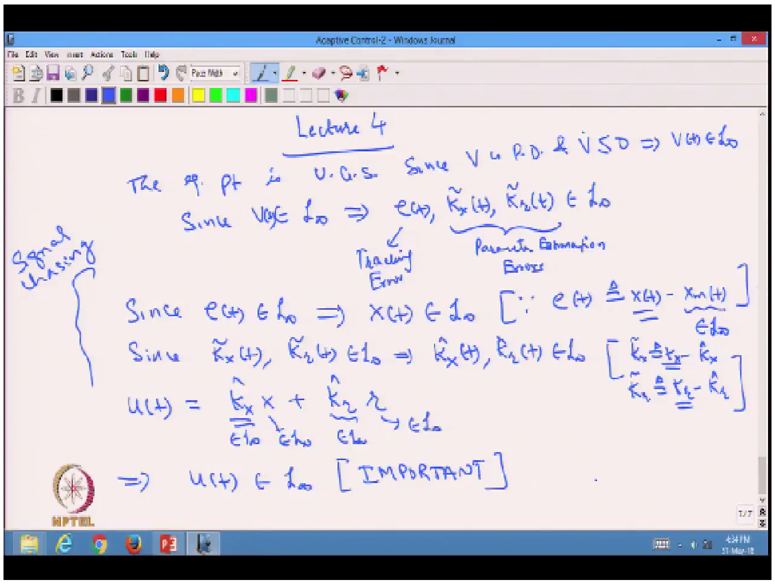
In the case where the plant parameter b is unknown but the sign of b is known, we can still move forward by choosing a Lyapunov function candidate which is again positive definite radially unbounded and decrescent. So the way we do it is we chose the absolute value of b in the term involving the parameter estimation errors kx~ and kr~.

(Refer Slide Time: 04:30)



And then we smartly chose the adaptive update laws like this and include the sign of b in these update laws and then we see that why we included the absolute value of b in the Lyapunov function candidates because when you take the product of mode of b with sign of b, you actually get b that then cancels with these terms in the Lyapunov derivative. So eventually what we get is V dot=ame square which is the same conclusion that we had, when we had assumed that b was known, b could be used in the adaptive laws. Alright, so what more can we say about this system?

(Refer Slide Time: 05:22)



So we have already proved that this is the equilibrium point is uniformly globally stable. What we can further say is, since Lyapunov function V is positive definite and V dot is negative semi-definite which can be represented as <= 0, what we can say is that V of t is in fact bounded, right. So if V is positive, if some positive definite function and its derivative is <= 0, then we can of course say that this function is always going to be bounded.

So we say that V of t belongs to L infinity and since V belongs, V of t belongs to L infinity, we can conclude that e of t, kx~ of t and kr~ of t belong to L infinity. So why is that so? That is very straightforward, if you look at the Lyapunov function candidate, if we say that V if bounded, then of course e, kx~ and kr~, all have to be bounded. There cannot be a case where you can have an unbounded error stage here.

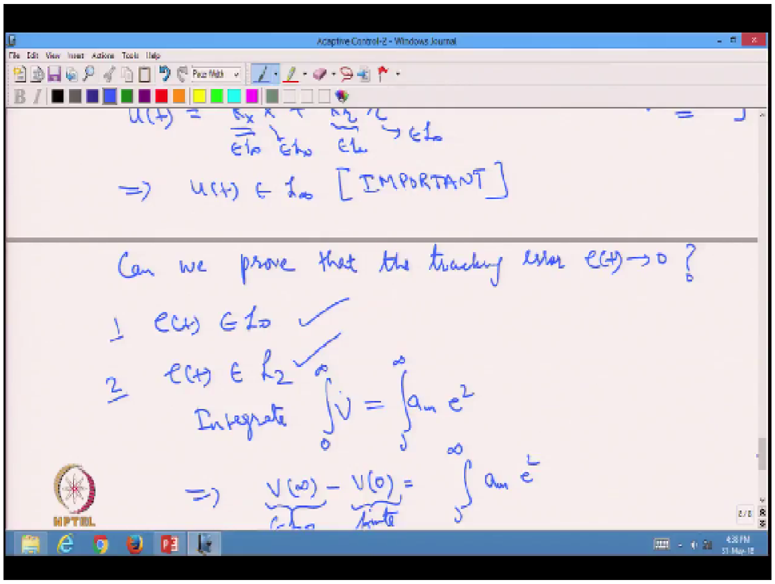
So we can conclude that since V is bounded, all the error states, they track in, so this is called as a tracking error, where tracking error is bounded and these are called as a parameter estimation errors. The parameter estimation errors are also proved to be bounded, so since what we can further say is that since e of t is bounded that means that x of t is also bounded, right. So this is because e of t we have defined as x of t-xm of t and since xm of t we have already said is the bounded state of the reference model, and since e of t is bounded, so x of t is also going to be bounded.

So what we are saying is that the state of the plant is bounded for all time. So it belongs to L infinity. What more can we say? So since kx~ and kr~ are bounded, we can say that kx hat and kr hat are bounded. This is because we have defined kx~ to be kx-kx hat and kr~ to be kr-kr hat. Since kx and kr are some constants and kx~ and kr~, we have already proved to be bounded so as a result, we can say that the estimates kx hat and kr hat are also bounded, okay.

So this is, all this is called as signal chasing. So it is important that we chase each and every signal in this analysis and prove that it is bounded. We would not like to have any signal in our system which is unbounded. It is also very critical to look at the control input and prove that it is bounded. So control input, the expression for that is given by kx hat x+kr hat r, so since kx hat we just proved that this is bounded. x, we have already also proved to be bounded.

kr hat we have proved to be bounded and r is a bounded reference signal, so if you take this altogether, we can say that the controller is also bounded. So this is very important. We must make sure that the control input that we are using is bounded. So it is very critical that once you do the Lyapunov analysis, you also do the signal chasing through each and every signal in the system is bounded. This is required for internal stability and also that the controller u of t is also bounded. Okay, so what more can we say about the error states?

(Refer Slide Time: 10:51)



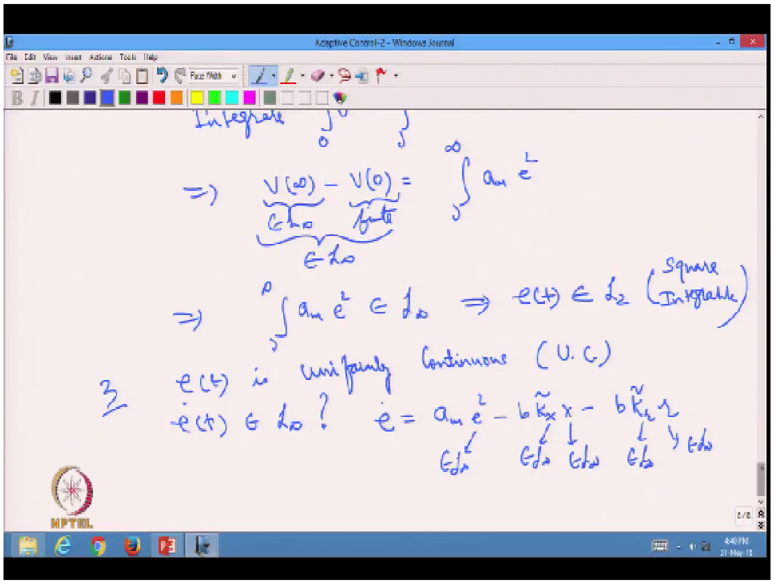
So remember that we had started with the objective that, so we go back and let us look at what the objective, original objective was? The original objective was the tracking error to go to 0 as t goes to infinity. So we have only proved thus far that e of t is bounded, okay. So can we prove that the tracking error e of t goes to 0? So remember we had discussed the Barbalat's Lemma in lecture number 2 which was around the preliminary.

So the condition for the Barbalat's Lemma is if there is a signal which is both L2 and L infinity and it is also uniformly continuous, then we can say that as t goes to infinity, the signal goes to 0. So can we try and prove that in the case of the tracking error, so we have already proved that e of t is bounded, so that is true. The next thing is to prove that e of t is L2.

So for that we can integrate the Lyapunov derivative equations which is V dot=ame square. So we integrate that from say 0 to infinity both sides. And what we get is V infinity-, so V of, so this is a lose notation. What I mean is V of t which tends to infinity-V of 0=to infinity ame square, so since we have been able to prove that V is always bounded, so this term is always going to be bounded.

So V as, the value of V as t goes to infinity is always going to be some finite quantity. This we can also show that V eventually converges to a limit. So this is some finite quantity, V of 0 is V, when you substitute the initial conditions of the error state and the parameter estimation errors, so the, so this would be some finite quantity which means that the left hand side is bounded.

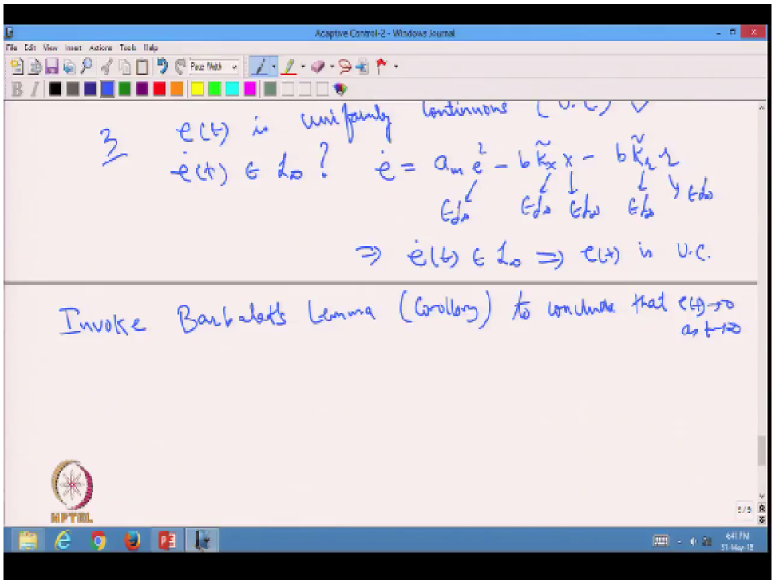
(Refer Slide Time: 13:50)



Which just implies that the right hand side should also be bounded. So what we can say is that this implies that e of t is L2 or square integrable, okay. So condition number 2 is also satisfied. Now if we can only prove that e of t is uniformly continuous. So in short we will be writing UC. So if e of t is uniformly continuous, then we can invoke the Barbalat's Lemma. So as I had mentioned before, it is sufficient if we are able to show that e of t, e dot of t is bounded.

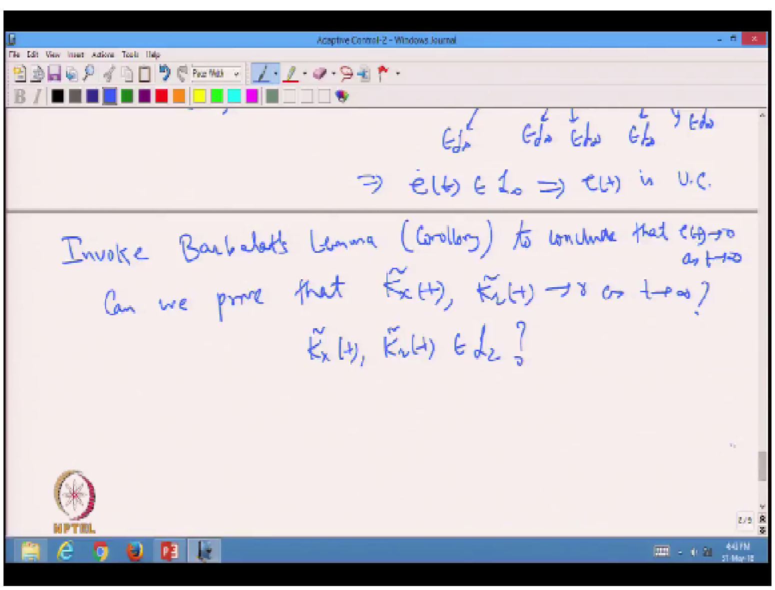
So is that true? So for that we look at the close loop error system which is given by, we have to go all the way back. So which is given by this equation. So e dot=ame square-bkx~x-bkr~r. So let us look at all the terms in this equation. So e we have already proved to be bounded, b is some constant, am is some constant, kx~ is bounded, x is bounded, kr~ is bounded and r is also bounded. So all the terms in this equation are in fact bounded.

(Refer Slide Time: 15:53)



Which means that e dot is also bounded. And since e dot is bounded, we can conclude that e of t is uniformly continuous. So the third condition is also true. So now we can invoke the Barbalat's Lemma, in fact we invoke the corollary error using 1 2 and 3 to say that e of t goes to 0 as to goes to infinity. So good. We have been able to achieve our objective. We have been able to prove that the tracking errors in fact converges to 0 as t goes to infinity. Can we say the same thing about the parameter estimation errors? That is the next important questions.

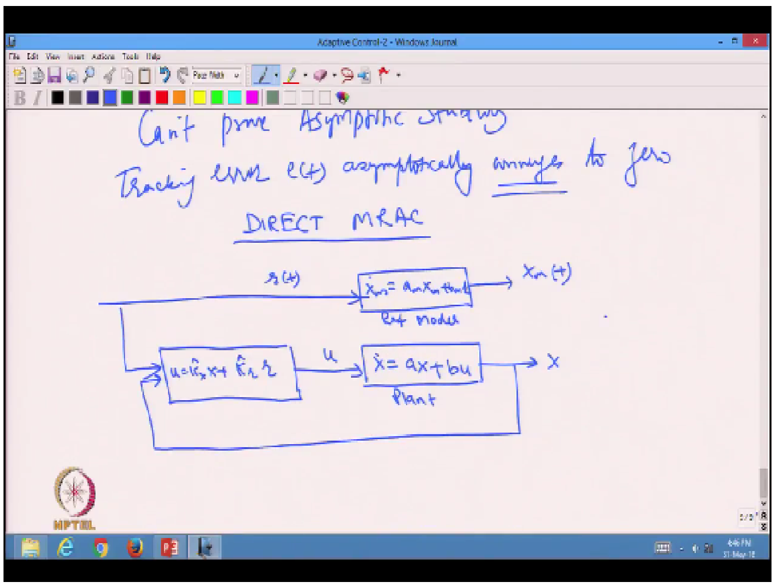
(Refer Slide Time: 17:00)



Can we prove that Kx~ and kr~ go to 0 as k goes to infinity? So it turns out that it is not straightforward to prove that the parameter estimation errors converge to 0 because although we have been able to prove that they are bounded, it is not apparent how to prove that they are L2 and let us see if we can prove that they are uniformly continuous. So we will have to look at kx~ dot.

So it seems that we would be able to prove that they are uniformly continuous because if we look at these questions for kx hat and kr hat dot on the right hand side, all the terms are bounded. So although we could satisfy condition 1 and 3 for Barbalat's Lemma, but it is not possible to show that kx~ and kr~ are L2. So this is what prevents us from proving that the Barbalat's lemma cannot be invoked. So we cannot prove that kx~ and kr~ in fact go to 0.

(Refer Slide Time: 18:33)

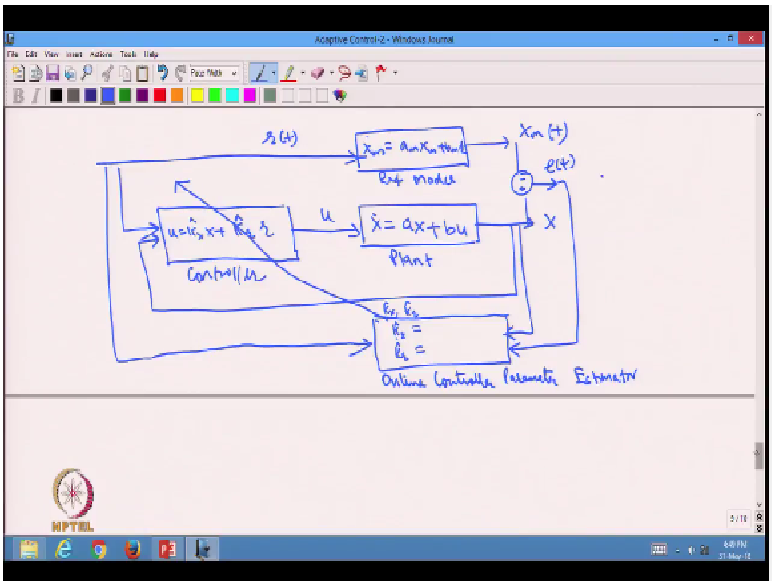


Okay, so what can we conclude about the entire closed loop system which consists of the states e, kx~ and kr~. So we have been able to prove that e goes to 0 but all the other errors, states are only bounded. So we still cannot say that the equilibrium point or the system as in particularly stable. So we cannot prove asymptotic stability because for that, kx~ and kr~ also have to go to 0 as k goes to infinity.

Since we are not able to prove that, we cannot say that the entire system will asymptotically stabile. What we can say is that the tracking error is asymptotically convergent to 0. So the tracking error e of t asymptotically converges to 0. So we prove convergence of the tracking error but not the stability, asymptotic stability of the equilibrium point. Okay, so the, this is the Direct MRAC case.

So if you were to draw the block diagram for this case, so the plant is considered as x dot=ax+bu, so it has an input u and an output x, okay. So there is a reference model. So this is the plant, a reference model is the desired behaviour that I would like the plant to follow. That is given by amxm+bmr. So the input is r of t and the output in this case is the state of the model xm of t. The controller that we use here is a straight feedback controller where the gains are time varying. So u is given by kx hat x+kr hat r which means that it is taking the feedback from the plant and it is also using that signal, okay.

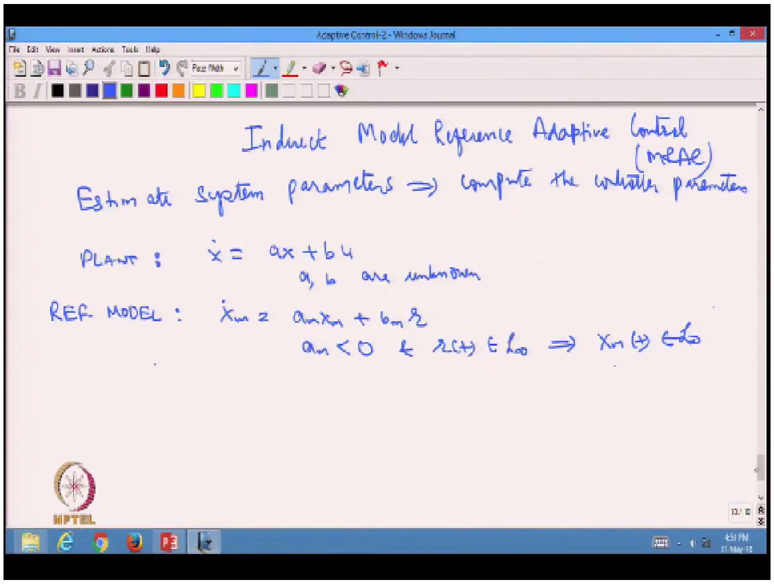
(Refer Slide Time: 21:38)



And kx hat, so we have already derived the expression for kx hat dot and kr hat dot. So the input to this block is signal x. So we have to, of course look at the update laws for kx hat dot and kr hat dot to find out what the input to this block will be. So these update laws are designed like this. So xe re are the input. So this is x, r we can take from here, let us just to this. So this is r of t and then we also have the error between the plant state and the state of the reference model.

So we have this as e of t. So this is also given as the input to this. So this is v adaptive law for the online controller parameter estimator. The output of this is used to update, so the output is kx hat and kr hat. So when, whenever you draw this kind of an arrow which cuts across a block, it just goes to show that you are trying to adjust the parameters. So in this case, the online parameter estimator updates the parameters of the controller, okay. So that is why this arrow which cuts across the controller block, okay. So this is the controller, okay. So this is an example of Direct MRAC case.

(Refer Slide Time: 24:05)

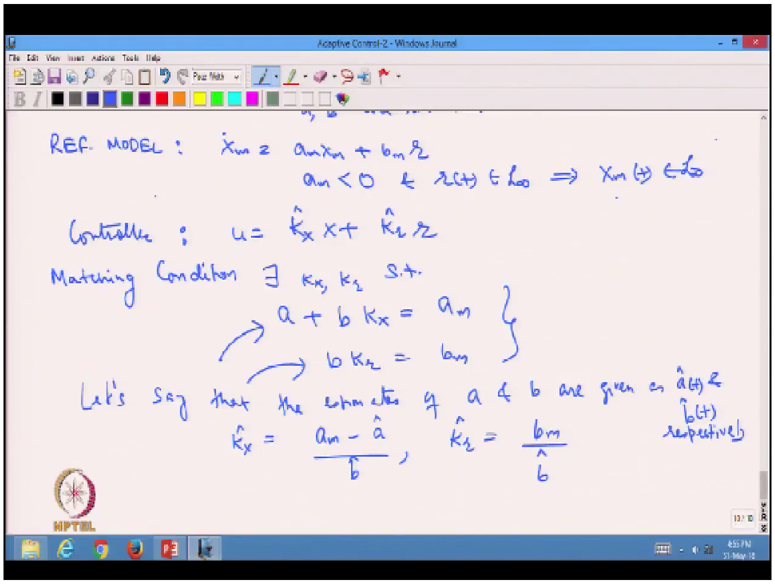


Now we move on to the other case which is the indirect model reference adaptive control. I had mentioned before that indirect model reference adaptive control first estimates the, so it has an online estimator for estimating the system parameters and those are then used to estimate the, to compute the plant parameter, the controller parameters, okay. So let us see how the indirect model reference adaptive control can be designed. First you estimate system parameters that you use to compute.

So you do not compute the controller parameters. Rather you use the estimates on the system parameters to compute the controller parameters. So let us see how this can be done? So let us again consider the plant model to be x dot=ax+bu where let us consider for now that a and b are unknown. The reference model is considered to be xm dot=amxm+bmr. So this is based on some desired performance.

Based on that you come up with a reference model. So a primary requirement is that am<0 and r of t is a bounded input signal which that xm of t is bounded. So this is known a priary.

(Refer Slide Time: 26:27)

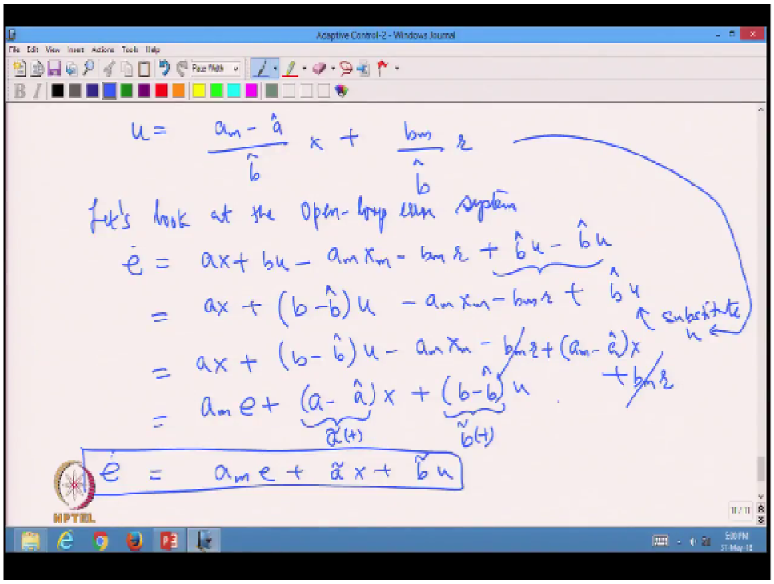


Now if you recall, we had designed the controller as, we designed the controller structure to be similar as that of direct MRAC, kx hat*x+kr hat*r and the matching conditions in that case if you recall or that there exist ideal kx and kr such that a+bkx=am and bkr=bm. So these 2 matching conditions should hold for the MRAC design to be valid, right. So this was what we had come up with in the direct case.

In the indirect case also if you use this as the controller, we will come up with the same matching conditions as what we had achieved for the direct case. So here what we can further do is if a and b, since a and b are not known, let just assume that we know, we, we, we can use the estimates of a and b. So let us say that the estimates of a and b are given as a hat and, sorry, a hat of t and b hat of t respectively, okay.

So the estimates for a is a hat of t and the estimate for b is b hat of t. So using that we can, we can find out the expression for kx hat and kr hat. So since in this equation we are going to substitute for the estimates of a and b, so we can find out the estimate of kx and kr. So kx hat is given as am-a hat/b hat and kr hat is given as bm/b hat.

(Refer Slide Time: 29:40)



So if we can somehow find out a way to estimate the system parameters a hat and b hat, so similar to the direct case, here if you could come up with an online parameter estimator such that a hat and b hat can be computed, then we can use that to compute the controller parameters kx hat and kr hat which is the indirect approach. So now the task remains to come up with the update laws for the system, system parameters a hat b hat, okay.

So then the controller u is going to be am-a hat/b hat*x+bm/b hat*r. So this is we have just substituted for kx hat and kr hat that we have used in the previous equation. So let us substitute this u or in the open loop error system. So substituting u of t in the open loop error system, what we get is e dot as, so okay, let us not substitute it right now. Let us just look at the open loop error system.

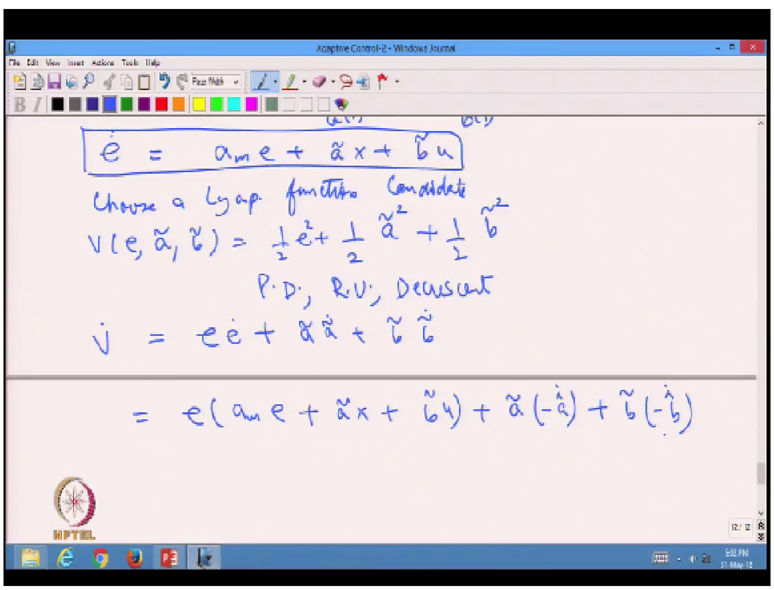
So let us look at the open loop error system, so e dot is ax+bu-amxm-bmr, okay. So just for convenience we add and subtract b hat u. So does not change anything, we have just added and subtracted the same term but it will make our life easier. So we will see how? So now what we can do is we can combine the terms a; bu-b hat u. So these 2 terms can be combined.

Then we have b hat u. So substitute for u here. Substitute u in this term. So what we get is, by the way this term can be written as b-b hat*u, okay. So if, when we substitute for u here, what we end up with is ax+b-b hat*u-amxm-bmr+am-a hat x+bmr. So this bmr and this -bmr be cancelled out and what we are left with is ame+a-a hat x+b-b hat of u. So let us note this term as a~ of t and this term as b~ of t.

So a~ of t is simply the difference of a and a hat of t and b~ of t is simply the difference of b and b hat of t, okay. So the close loop error system comes out to be ame+a~x+b~u, right. So this is what our closed loop error system. Again from here from this equation, we can see that if the parameter estimation errors are = 0 or go to 0, then we are left with just e dot=ame which is exponentially stable.

The task is to look at the complete stability analysis of this error system and improve that, if possible prove that all the error states are bounded and the tracking error goes to 0. So let us look at, of course ideally you would also like to prove that the parameter estimation errors converge to 0. Let us see if it is possible in this case. We saw that in the direct case, it was not possible. We could only prove that the parameter estimation errors are bounded. In this case, let us see what we can do?

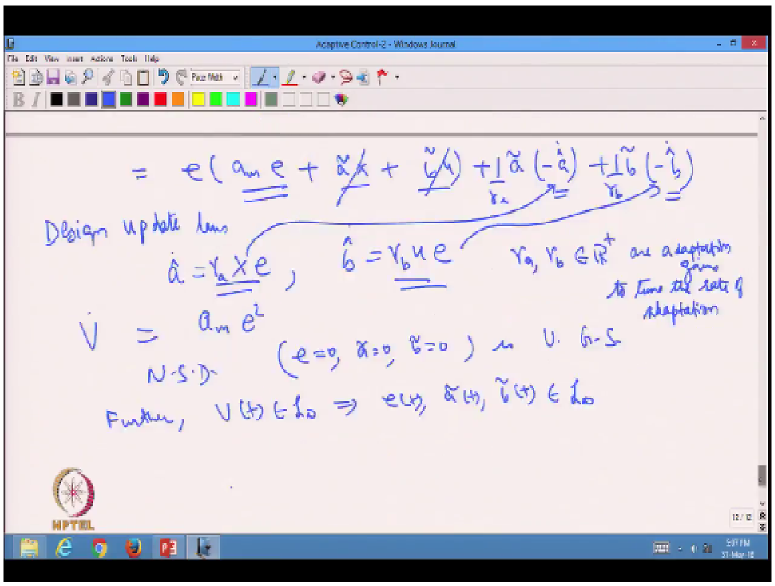
(Refer Slide Time: 35:10)



So let us choose Lyapunov function candidate v of e a~ b~ as 1/2e square+1/2, so in this case, we have the tracking error state e, the parameter estimation error states a~ and b~, so let us have the sum of errors of those terms as well and let us see if this works out. This is positive definite, radially unbounded and decrescent, so it is an ideal Lyapunov function candidate. Let us see if it is indeed a Lyapunov function.

So for that we have to take the derivative v dot. So v dot is given as ee dot+a~a~ dot+b~b~ dot that is given as ame+a~x+b~u+a~-a hat dot, because we have defined a~ to be a-a hat and when we differentiate that, then a dot=0 because a is a constant and we are left with -a hat dot. Generally, for b~ dot, we will have the derivative as -b hat dot.

(Refer Slide Time: 36:54)



Alright, so now the task is to design the update laws and the update laws a hat dot and b hat dot for a hat and b hat, have to be designed in such a way that the Lyapunov derivative is at least negative semi-definite. Because that is the least that we need to prove Lyapunov stability, okay. So the first term here in the Lyapunov derivative, this results in negative term whereas these 2 terms are sign indefinite.

We do not know their signs and you would ideally want them to cancel using our update laws. So, let us design our update law for a hat dot to cancel this term in the analysis. So a hat dot= say if we choose to be = xe and b hat dot=ue, then we have, we see here that these 2 terms cancel out and what we are left with is simply ame square, okay. So which is negative semi-definite which means that since v is positive definite, radially unbounded, decrescent.

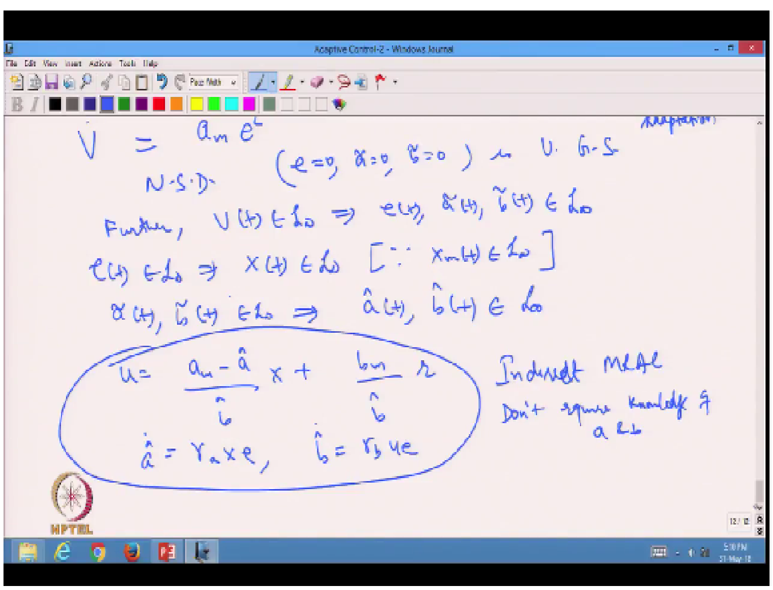
And v dot is negative semi-definite, we can invoke one of the theorems that we had discussed before to show that the equilibrium point. So in this case the equilibrium point would be e of 0, a~ 0 and b~ 0, so we, this is uniformly globally stable. Further we can state that v of t is bounded. This is because v is positive definite and v dot is <= 0. So v is bounded which means that the errors e of t, a~ of t, b~ of t are also bounded.

So one more thing I wanted to tell you in these update laws. We would always like to have some, some design degree of freedom, so we can always try and include an adaptive gain here in the update laws. So let us have this as gamma a and this as gamma b where gamma a and gamma b are some positive constants. We can call as the adaptation gains. So you can tune these gains to alter the rate of convergence.

So to tune the rate of convergence, tune the rate of adaptation to be more precise. So but if we add these 2 here, they will go into these terms and then we will not be able to cancel this term here. So we would have to modify the Lyapunov function candidate slightly by saying there is, in the denominator, we have gamma a and in this term also, the denominator we have gamma b and since these are positive, the Lyapunov function candidate still is positive definite, radially unbounded and decrescent stays our problem there but this helps us cancel this, these 2 terms.

So here we will have 1/gamma a and here we will have 1/gamma b, okay.

(Refer Slide Time: 41:25)

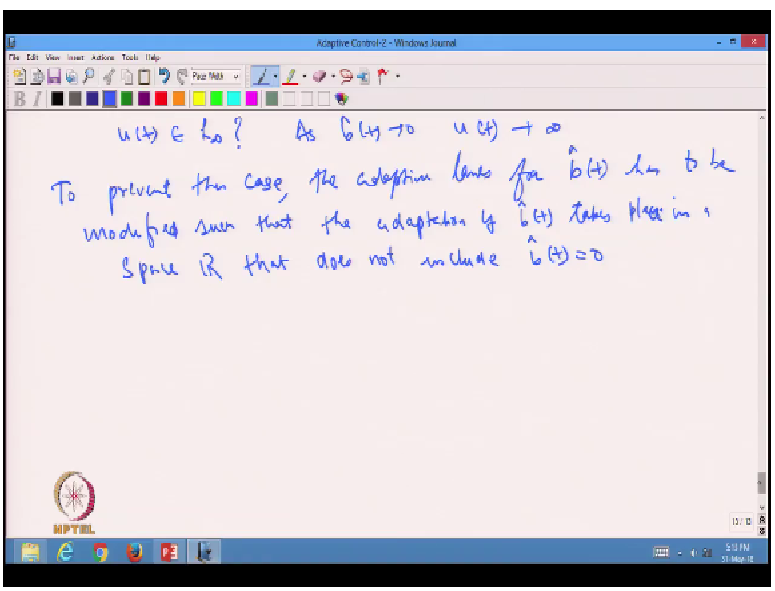


So let us move on with signal chasing. So we want to chase each and every signal in our system and prove that it is bounded. We have already proved that all the error states are bounded. So since e of t is bounded, what we can prove is that x of t is bounded. This is because xm of t is bounded. Also since a~ of t and b~ of t are bounded, we can say that a hat of t and b hat of t which are the parameter estimates, they are also bounded, okay.

So one interesting feature of this, of this controller, so let us look at this controller again. So here we have u as k, let us just, so here u is am-a hat/b hat*x+bm/b hat*r, right. So one interesting feature here is that, no I just forgot about the update laws which are given as gamma axe, b hat dot given as gamma bue. So these adaptation gains can also be included for the direct model reference adaptive control case I forgot to mention that in the previous lecture.

But similar to this case, you could also include these adaptation gains in the direct case as well, okay. So looking at the indirect MRAC, so this is the indirect model reference adaptive controller. The interesting thing to note here is that we do not require knowledge of a and b. So we do not require plant knowledge, the knowledge of the plant parameters. So this is different from the direct MRAC case where we had assumed at least that the sign of the term b is known.

(Refer Slide Time: 44:25)



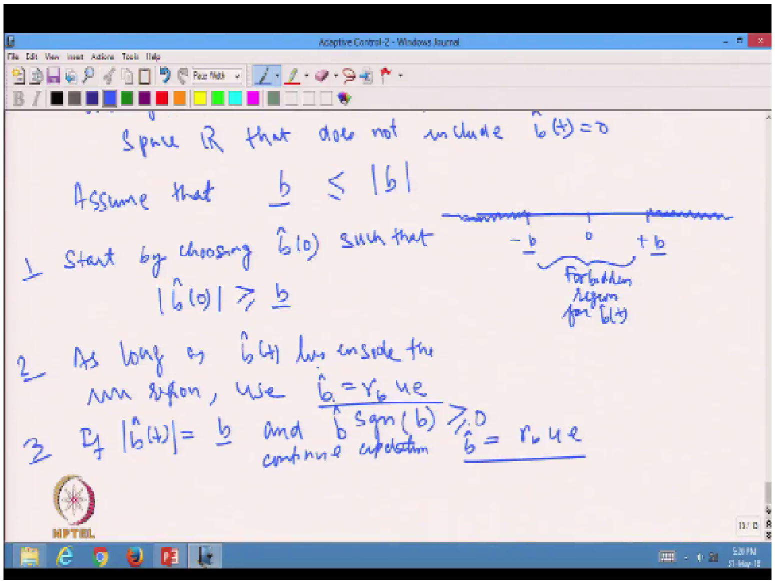
But what we have not proved here still is that, is u bounded. So that will give us some interesting insights into this indirect MRAC case. So all term individually in u are bounded. For example, a hat is bounded, b hat is bounded, r is bounded and x is bounded. So all these terms are bounded but b hat here appears in the denominator, that is a critical part. So even though b hat is bounded, b hat can in fact be very close to 0.

It can, in fact converge to 0 and what happens when b hat becomes very close to 0? Then u of t blows up. So as b hat of t goes to 0, u of t goes to infinity. So the controller just blows up. So this is not a desirable situation to be in. This is not a very good design. We do not want to have any terms in the denominators which can potentially go to 0. So we will have to modify our design to make sure that b hat is updated in such way that it is not near 0.

So updation of b hat in this case, in our design here is not in our hands because it is, it just follows some differential equation. So it might happen that for some values of e and u, b hat in fact becomes = 0. So we want to avoid that case here. So to prevent this case, the adaptive law for b hat of t has to be modified such that the adaptation of b hat of t takes place in a space R of real numbers that does not include 0.

So we have to somehow project b hat on to a space of real numbers because b is simply a scalar. So b hat will take real value. So we want to project b hat on to a space of real number which does not include 0. So that is a way to make sure that b hat never comes near 0. So how do we do that?

(Refer Slide Time: 47:40)



So I will just first tell you the basic idea and then the actual details will follow. So for this we will have to assume that we know the lower bound of absolute value of b. So which means that for the actual parameter b, suppose this is 0 and this is +b bar and this is -b lower bar. So I know that the actual system parameter b lies in this region, okay. So in this wiggly region, I know that the parameter b lies, okay.

So I can always, so if I know the lower bound b, b bar for the mod of b, I can always create this space here where this is the forbidden region for b hat. So we would always like b hat to be in these wiggly region because that is where b lies, okay. So how do we make sure that the b hat is updated in such a way that it always stays in these wiggly regions? So the first thing that we can do is we can at least start in that region.

So start in the wiggly region by choosing b hat of 0, so b hat, the updated law for b hat is in our hand. So we can choose the initial condition as well. So we can choose b hat of 0 such that the initial value b hat of 0 is >= b lower bar. So we want to start b hat to be in these wiggly regions, right. So at least in the first time and strength, we want that it starts in the wiggly region and then let us see how, how it gets updated?

So then we have, we have 3 cases. So case 1 is rather, let me tell you the main idea here. So, as long as b hat, the value of b hat lies within these inside of these wiggly regions, not at the boundary but at these, inside of these wiggly regions, we can continue to update b hat with the update law that we have designed. So as long as b hat of t lies inside the wiggly region, we can use the same update law that we had designed earlier because b hat is away from 0.

So we do not really care. However, we have to be careful when b hat tries to come near 0. So because this is the updation of b hat occurs continuously, so b hat of course cannot jump from, from this wiggly region into this forbidden region. It has to continuously come towards, towards the forbidden region. For that, it has to cross the boundary which is the value b lower bar. So if b hat of t=, if the absolute value of b hat of t=b lower bar.

And, suppose b hat is at the boundary, so it is either at this point or it is at this point. Now at the, at these boundary points, we check the derivative of b hat. So if the derivative of b hat is, suppose b hat comes at this point, if it is at the boundary, so at this point the value of b hat=b lower bar and then we calculate the derivative of b hat which is b hat dot.

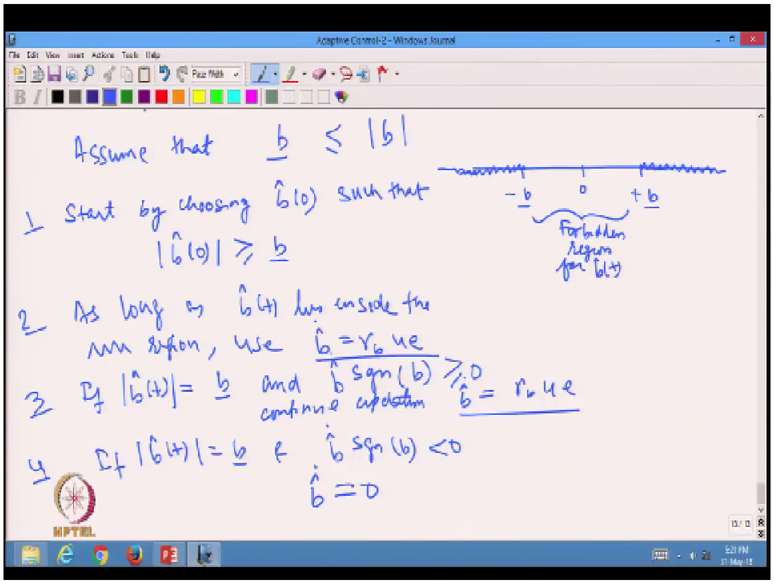
And if the derivative is >0 which means that b hat is increasing, so in the next instant b hat will move inside the wiggly region which is the safe region. Suppose b hat is at this point which means that we are talking about a negative b. So for that b hat can come to the boundary at this region which means that the b hat, the value of b hat becomes = -b lower bar.

So at this point, we again check the derivative of b hat and if b hat dot comes out to be <0, which means that b hat is decreasing, it means that b hat is going to move into the wiggly region in the next time instant. So together we can say that if b hat dot signum of b is >= 0, this just illustrates the cases that I mentioned, that the boundary and whether the derivative is increasing or decreasing.

So in both the cases, you know, we can actually combine the 2 cases using this. So if this is true, it means that we are still safe because b hat, although it is at the boundary, in the next time instant, it is again going to move inside the wiggly region which is the safe region. So we still continue to update b hat like we have been doing. So continue the updation which means that b hat dot=sigma b, or gamma bue. So we use the same update law that we had designed earlier.

Because b hat is still not in danger, okay.

(Refer Slide Time: 54:20)



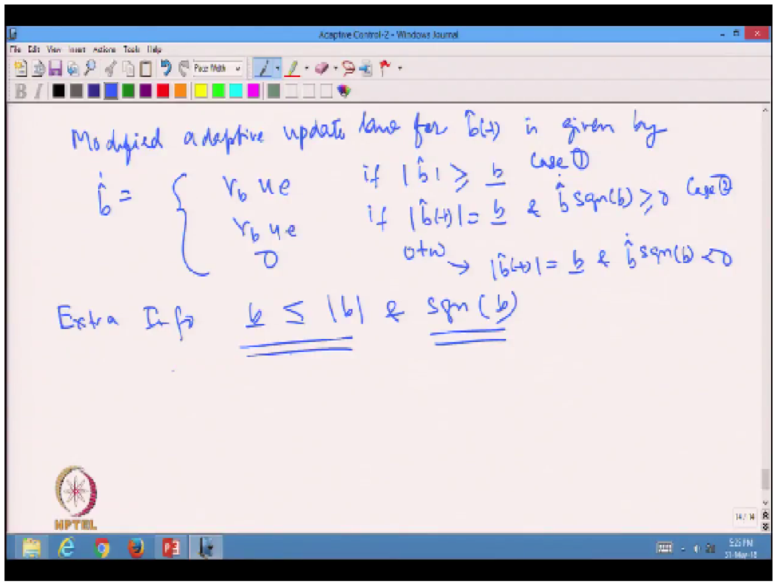
So then we move to the case where b hat will be moved to the last case where b hat of t is at the boundary and b hat dot signum of b<0, then b hat would try to enter the forbidden region which is unsafe region because that region is near 0 and it can lead to disastrous circumstances for the controller, it can even blow up. So we do not want b hat to enter inside this region. So we immediately will, you know, would like to stop the updation on b, b hat.

So we say that b hat dot=0, which means b hat just stays at, at b lower bar, okay for that time instant and we stop the updation. But for the next time instant, we do not know where b hat would be. Because for the next time instant, b hat in fact might move inside the region. So the same process continues. So what we have to do is, we have to, we have to check for, for where b hat is.

If it is inside the wiggly region, then there is no problem. If it is at the boundary, then we check the derivative and, and combine with the sign of b. If it is >= 0, then we continue to use the same adaptive update law. If b hat dot signum of b<0, then of course we know that in the next time instant, it is going to move into the forbidden region and so we stop the updation there.

But we have to keep doing this. If we stop the updation at one time instant, that does not mean that for all time after that, b hat dot will be equal to, you know, is going to get stuck at b lower bar. For the next time instant, again we have to check this condition and see where b hat lies. Okay.

(Refer Slide Time: 56:23)



So, then what we can conclude is that the modified adaptive update law for b hat is given by b hat dot to gamma bue if b hat if >= b lower bar, okay. It is gamma bue again. If b hat of t=b lower bar which means it is at the boundary but the slope is pointing towards the wiggly region which means that b hat dot signum of b is >= 0, so that is also a safe region.

Otherwise, b hat dot=0 which means that the value of b hat stays, stays constant. So this case is, the otherwise case is when b hat of t=b lower bar and b hat dot signum of b<0, okay. So we have done 3 cases here. One is that b hat lies inside this wiggly region, wiggly region. Then we have the case 1. If b hat of t lies on the boundary of the wiggly region and the derivative is pointing inside the wiggly region, then we continue the updation as we have derived using the Lyapunov analysis.

However, if b hat lies on the boundary of the wiggly region and its derivative is pointing away from the wiggly region, that means that b hat may, will enter into the forbidden region and we should immediately stop the updation of b hat. So at that time b hat dot=0. So this is a method to prevent b hat from getting very close to 0. However, as we can see here the extra information that we have used is we have used b lower bar for the absolute value of b and we have used the signum of b.

So this was the only information that we had used in the direct case. So in this case in fact it turns out that we need some more information to make sure that the design is valid, okay. So there is one more thing that we need to discuss about the indirect case which is now that we have modified the update law for b hat, how does that change the Lyapunov analysis? Do we need to modify the Lyapunov analysis?

Because eventually we, we want to prove that the equilibrium point is stable and if possible, then the tracking error goes to 0. So with this modified update law, how can we prove that? That will be the topic that we take up first thing in the next lecture and then we will move on to other things. Thank you.