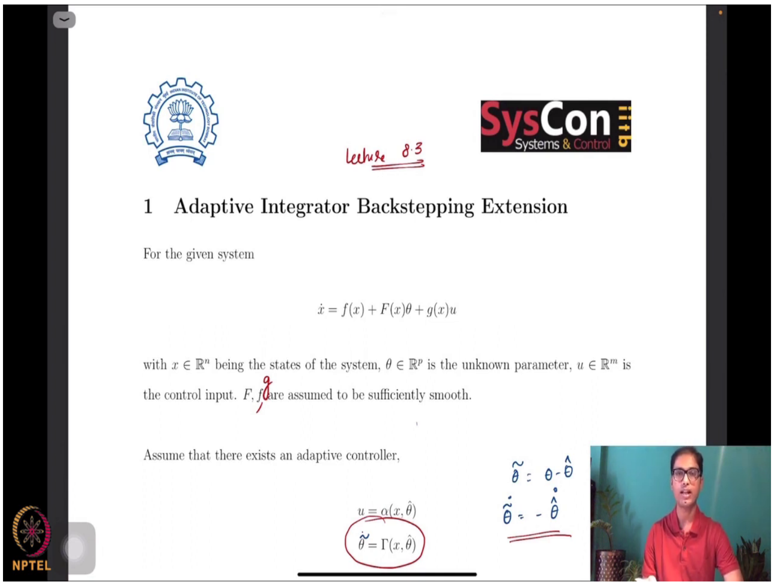
Nonlinear Adaptive Control
Professor Srikant Sukumar
Systems and Control
Indian Institute of Technology, Bombay
Week 8
Lecture No: 47
Extended Matching Design for Avoiding Overparametrisation

Hello everyone, welcome to yet another session of our NPTEL on Nonlinear and Adaptive Control. I am Srikant Sukumar from Systems and Control, IIT, Bombay. So, we are well into the 8th week of our course on Nonlinear and Adaptive Control.

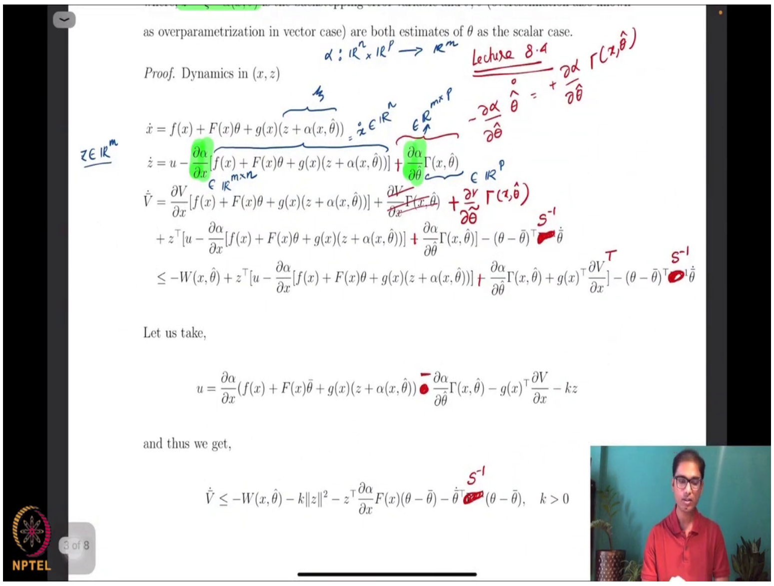
And I think it is evident to all of us that we are now in a very good position to be able to design control algorithms that drive autonomous systems such as the SpaceX satellite that you see in the background. We have looked at several different sorts of adaptive control designs by now. And I think all of you have gotten a very good hang of how we are going about these designs.

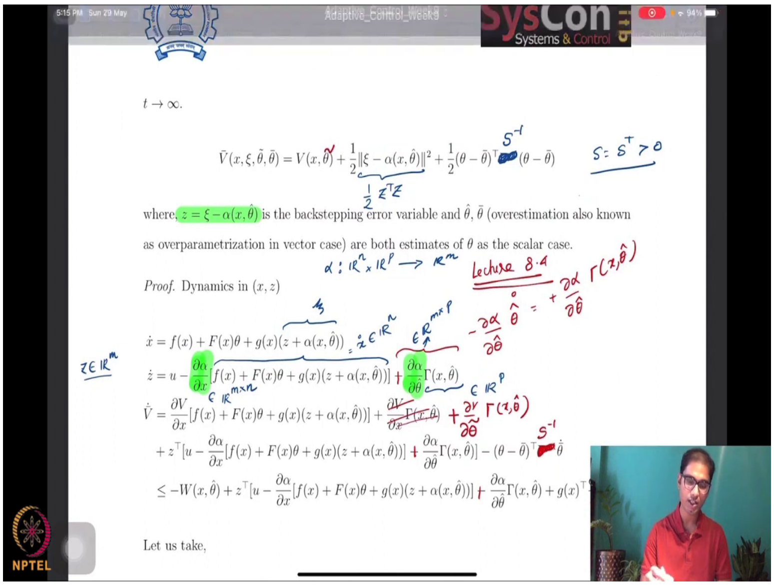
(Refer Slide Time: 01:08)



In the last couple of lectures, specifically, we were sort of looking at this vector version of the adaptive integrator backstepping design. And this is a rather powerful method, if you may, as you can see, and the idea here is that you can use the over parameterization in order to compensate for the fact that the unknowns are unmatched with the control dynamics, alright, so this is, you know, what I would say is a rather strong, you know, positive of this adaptive control method. So, we looked at this in some detail in the past few lectures. And, and so this is, of course, you know, something that is rather good.

(Refer Slide Time: 02:19)



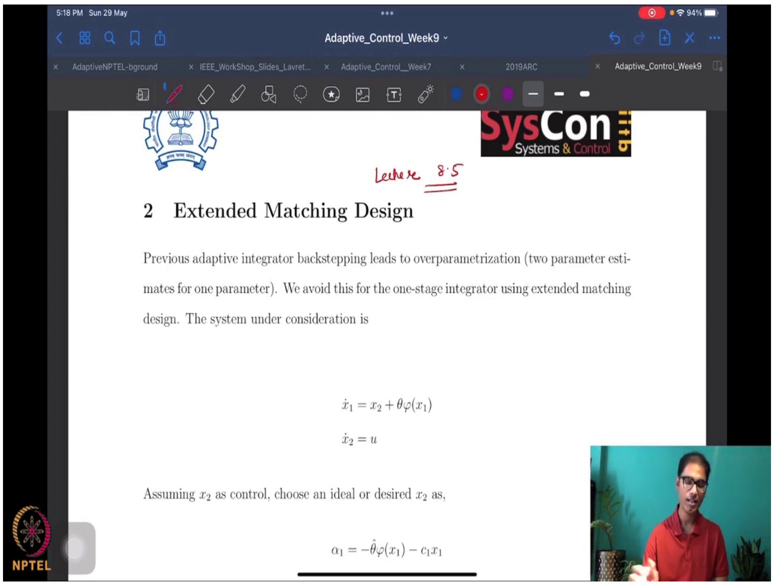


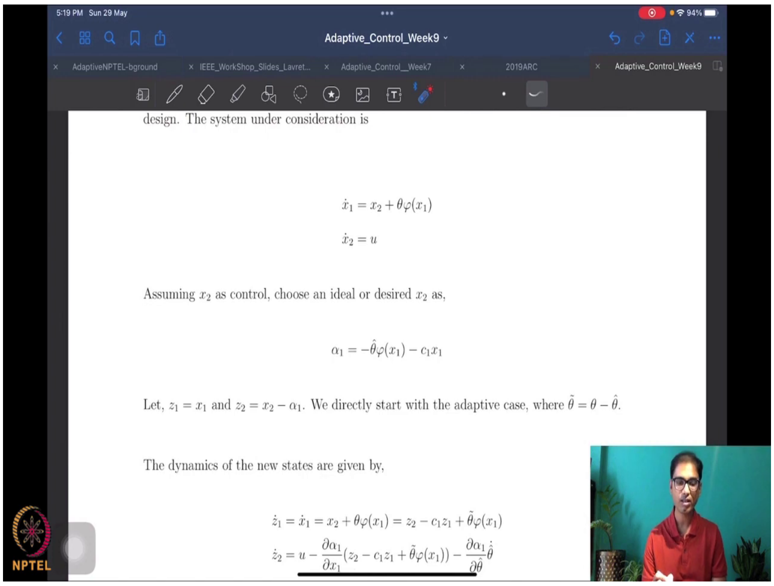
We also showed how to construct the adaptive laws. We also showed, most importantly, how the second level, Lyapunov function is, in fact, constructed. So, that is essentially was this guy. So, so yeah, I mean, I really hope that all of you got a very good feel for design of adaptive controls from vector systems. Yeah, we have looked a lot at the scalar case.

But of course, we also want to look at the vector cases carefully. The vector case, of course, is different only in the sense that the Lyapunov function now contains norms and norms squares, instead of the typical scalar square points. And, of course, we had to do a little bit of careful book keeping, in order to prove these nice properties.

So, that is really, you know, I mean, I would really say that the differences are rather minor. And it is not, you know, a significant novelty in terms of the design method or anything. In fact, at every stage, you can find a parallel with the scalar method.

(Refer Slide Time: 03:37)



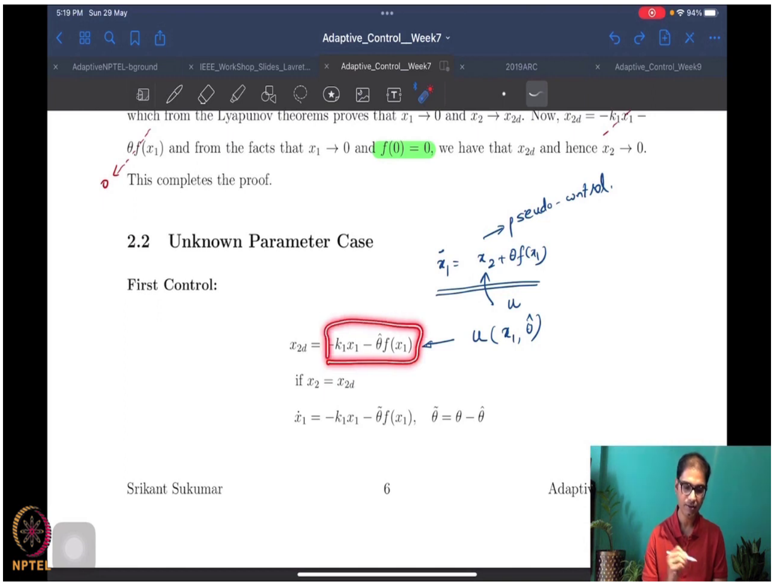


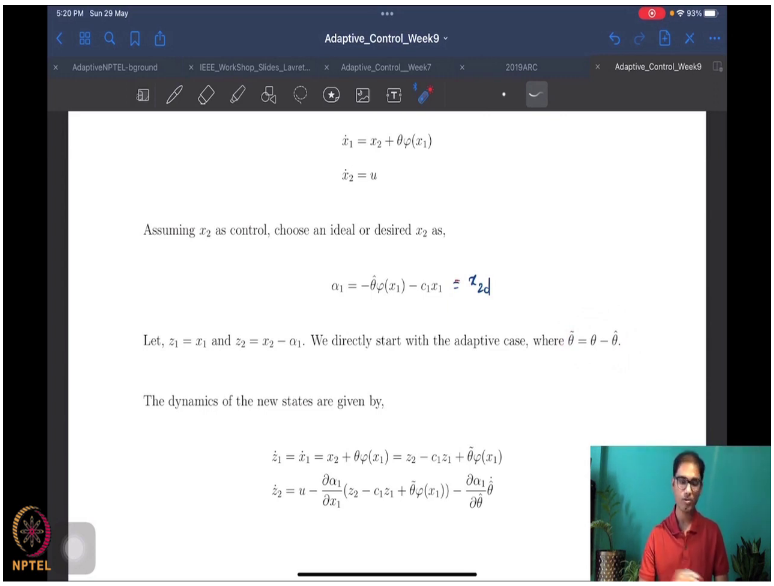
We, of course, pointed out the big issue, and that is the requirement for, in this case 2 parameters, two estimates are to identify the same parameter theta. And we certainly want to get rid of this issue. And this is where the extended matching design method comes in. So, I am going to mark the lecture here, we talked a little bit about the extended matching design and the fact that we are starting again with the scalar case system.

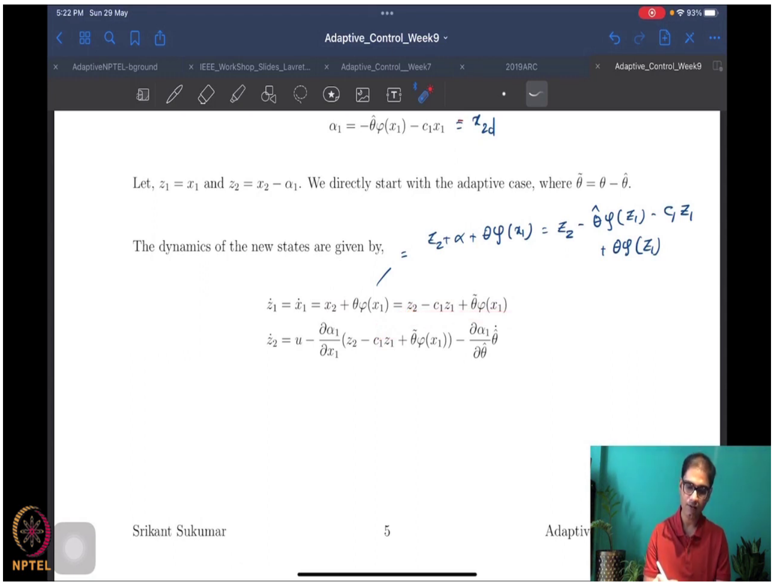
And just like before, but still I want to mark the lecture right here just for the purpose of, you know, making a new beginning on a new page. That is all. So, if you look at the system, like we said, it is still the single integrator system amalgamated with a control in the next stage, that is really the idea here.

And we of course, want to do an extended matching design, which means it but not it does not mean anything. It is just a name. The idea here is that we do not want to over parametrize. So, let us look at what is the desired ideal control. Then if you remember, alpha 1 that is the desired ideal control is not different from the week 7.

(Refer Slide Time: 05:04)







Because if you look at week 7. The unknown case if you see, the desired xd is still something like this. And that is essentially what we will have here to just different phi instead of an F, but otherwise is the same. We do not know the parameter, therefore we replace it with an estimate, multiplied by the phi, and then we cancel the… Sorry we introduce a good term in the x1, so this is actually equal to x2 desired. In our notation, this is actually equal to x2 desired in our notation.

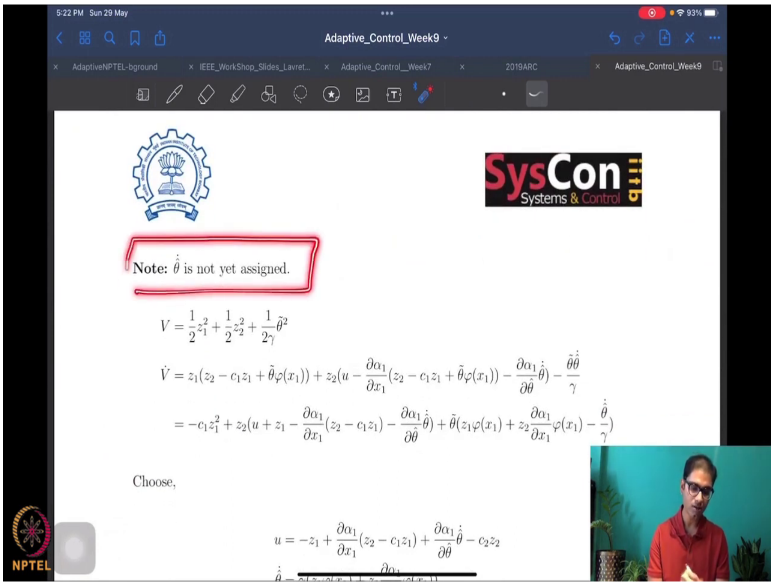
Now, we of course just redefine states we call the first state as x1, z1 equal to x1, and z2 is just the backstepping error again, why because x2 is not really the control. So, x2 cannot really be equal to x2 design. So, we do the next best thing we try to make x2 to chase the x2 design. And we, of course, define theta tilde as theta minus theta hat as we do in all our adaptive control problems.

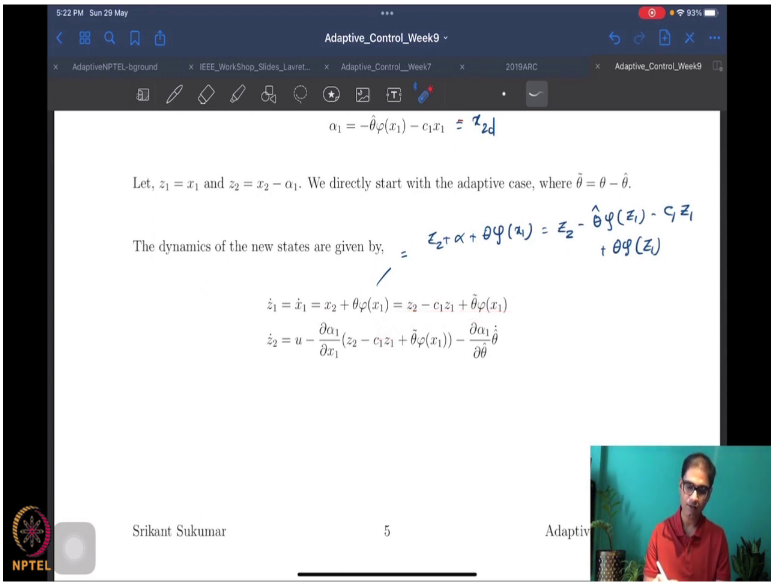
So, in the new states, we write the dynamics in the new states, of course, so the z1 is just x1. So, z1 dot is, just x1 dot, and that is x2 plus theta phi x1. And x2 can be written as z2 plus alpha 1. So, if I do that, I am going to carefully rewrite these things. So, this is I am going to write a few intermediate steps, this is equal to say, 2 plus alpha plus theta phi x1. And I am going to replace alpha as z2 minus theta cap, phi x1, which is the same as z1 minus c1 x1 is the same as z1. And then I have plus theta phi, z1.

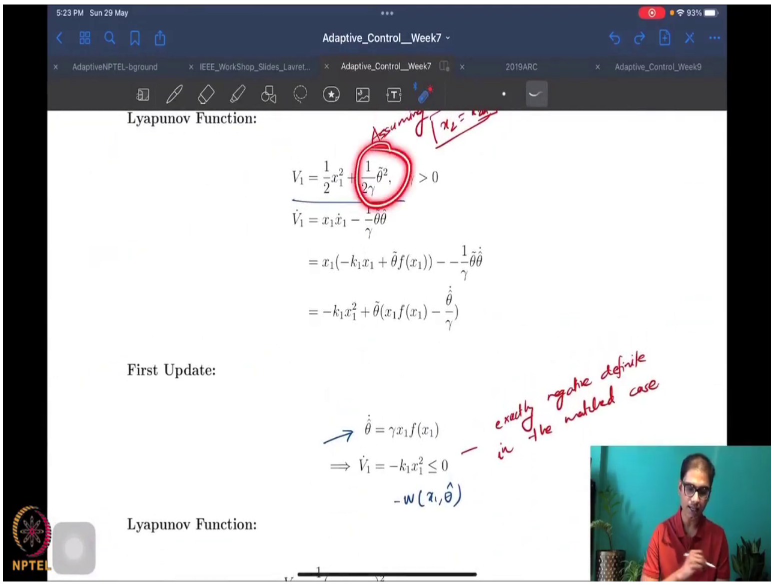
Now, these two of course, combined to give me theta tilde phi x1 or z1, it does not matter, then I have minus c1 z1, and z2 here that is it. And z2 dot is x2 dot minus alpha 1 dot, which is x2 dot is just the control. And alpha 1 dot has two pieces, because alpha 1 is a function of x1 or z1, and theta cap.

So, this is del alpha 1 del x1 times x1 dot, which is what we plug in from here. And then alpha 1 del theta cap, times theta cap dot. So, we have not yet specified what this theta cap dot is. So, we are not going to specify it now. Let us go on. So, we have the dynamics in the new variables, z1 and z2.

(Refer Slide Time: 08:07)





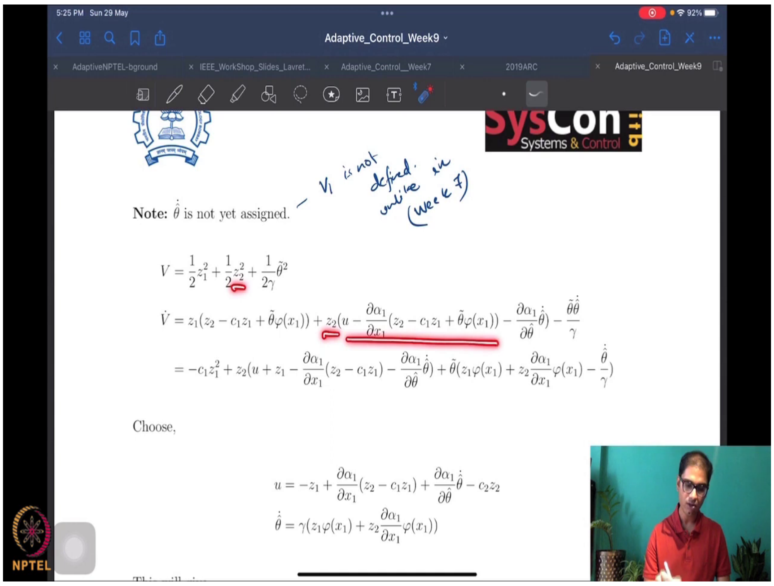


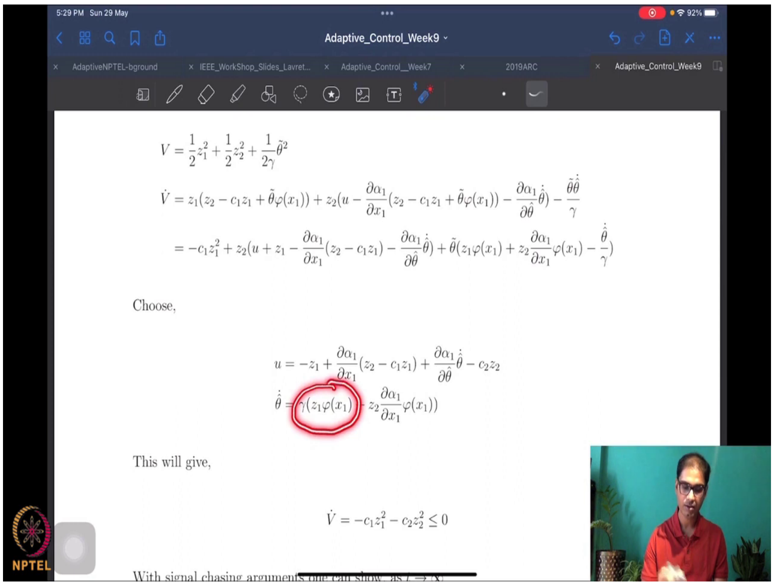
So, the important thing to note is that theta cap dot is, not yet assigned. Now, what we do is, seen the earlier version, what we were doing was we had, as soon as we wrote a z1 dot or an x1 dot in, the earlier version. We immediately defined a V1 with the theta tilde, and we came up with an update law with the theta cap.

Why we did this is because this is how we sort of understand backstepping. To handle the first state, first all aspects of the first state. So, what we did was we defined the Lyapunov candidate V1, which ensured that the first state was negative, ensured that the derivative Lyapunov function along the first state was negative definite, or semi-definite, at least, and then move on to the next state.

So, it was essentially completely inspired by how backstepping means that is you deal with the first state, then you go to the next and so on and so forth. But now, we sort of control each, you know, to handle everything for the first state. So, we do we do not define a theta cap dot we do not define a V1 at all.

(Refer Slide Time: 09:24)





So, the important thing is that theta cap dot is not assigned. V1 is not defined. Unlike in week 7, unlike in week 7, V1 is not defined. And because V1 is not defined, theta cap dot is, also not assigned. And what do we do instead? Like I said, we control our itch, and we define the combined Lyapunov function now, and what is that?

It is the same simple idea, I took the z1 squared, then I take the z2 squared, and I just have 1 theta tilde squared over 2 gamma. No longer a second estimate and all that, because I did not even create a first estimate, did not create a first V1. So, I just am… to end up with the, you know, final V and just 1 estimate, at least, that is my hope. And now I of course, nicely take the derivatives. And, you know, see if I can define a theta cap dot.

So, I have z1, z1 dot, which is this z2, and z2 dot, which is this is z2 dot is, just x2 dot minus alpha 1 dot, which is del alpha 1 del x1 x1 dot, which is the same as this. And del alpha 1 del theta cap dot. So, you see, there is already another theta cap dot here. But the good thing is at theta cap dot is in fact, known quantity, I mean, in the sense that I, as a user, I am going to specify it. So, even if it appears in my V dot, it is not a big deal.

Even if the theta cap dot appears in my V dot, this is not going to be a big concern at all. So, this is the idea, this is what sort of helps me. And then of course, we have the last term, which is the usual term theta tilde theta cap dot over gamma in V dot, great, now, I have the nice negative term typical in backstepping and then there is a term I mean I can club this term with the z2.

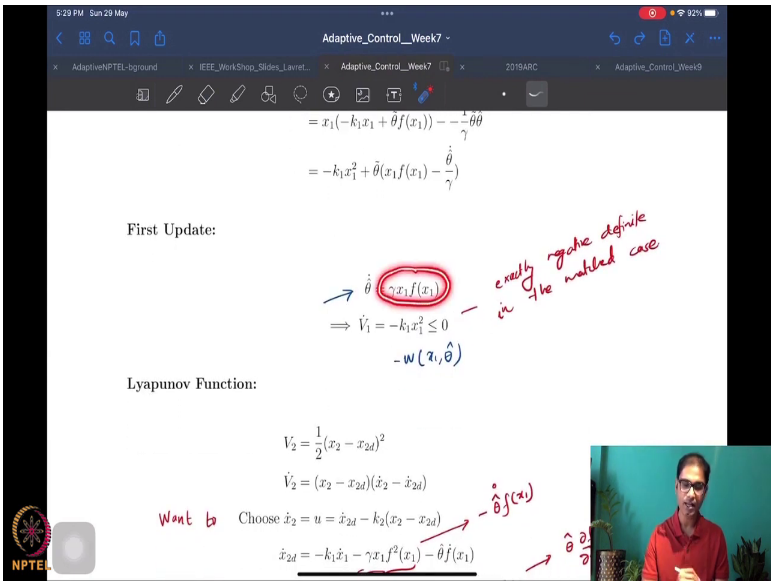
Which is u plus z1 from here. And then all the terms which do not contain a theta tilde. So, I take u of course z1 of course, then I have these two guys. Yeah, and then I have this guy, because remember, theta cap dot cannot contain theta tilde. And if it does, then you have a problem. Because theta tilde is unknown. And if your parameter update law, theta hat dot contains the unknown, then you have not decide an adaptive controller at all, you just designed a controller.

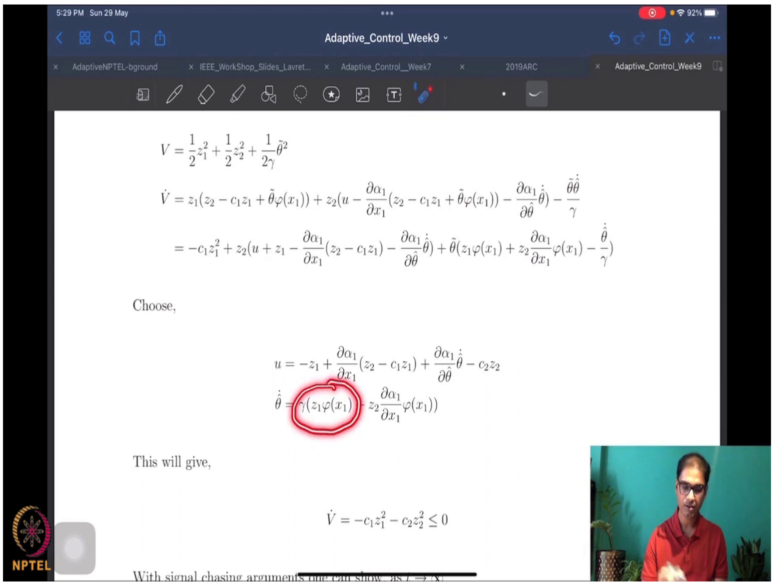
So, you take in the z2 term, all the quantities which do not have a theta tilde in them. So, this is the only thing that gets left out. Everything else remains. And then this guy comes from here. So, that is what is the term here. And then I clubbed the terms in theta tilde separately, so that is 1 term, then this term gets combined. So, that is this guy here. And then I also get a term here. Remember, I also get a term from the first state. And that appears here, the term from the second state appears here. And then of course, this term in the unknown quantity.

Now, something really nice and neat has happened. I mean, if you look at these terms, already you have a nice negative quadratic, in z1 in the first state, and in the second state, I have the control. So, I can introduce a quadratic by cancelling all these quantities, because none of it depends on the unknown. So, I cancel them, you know, comfortably, and I introduce a nice negative term. And the third term is an uncertain or, you know, I mean, it is not a sign definite term, I really do not know what kind of sign it will have.

So, I do not concern myself about trying to make it negative definite and things like that. So, all I will do is I will simply try to push it to 0. And this is what anyway is our goal in typical nonlinear control. Whichever terms, you see are not signed definite, you will usually try to push them to zero. So, that is the idea. So, that is what I do, I use my theta cap dot in order to cancel these terms. And that is what I get. Now, if you look at this very carefully this term.

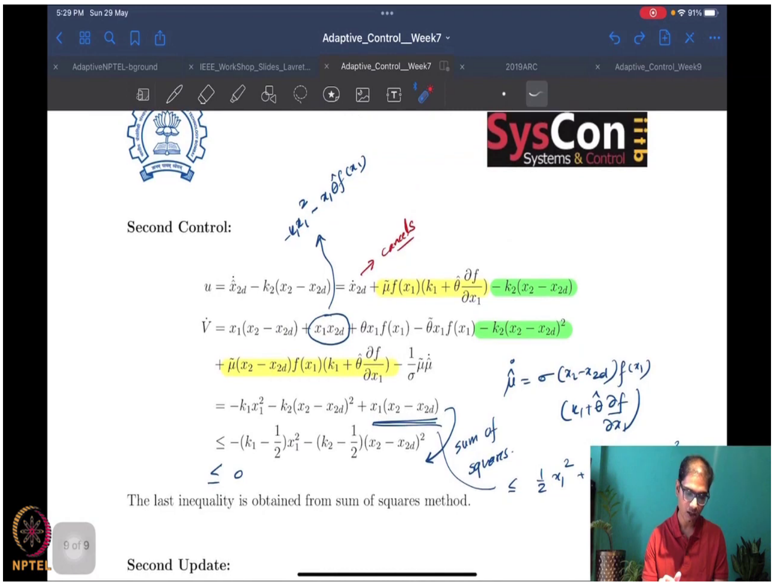
(Refer Slide Time: 14:19)

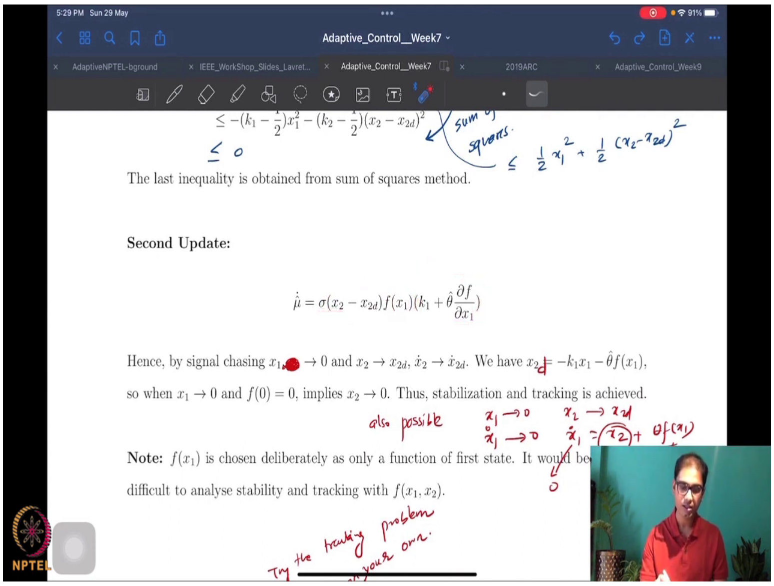


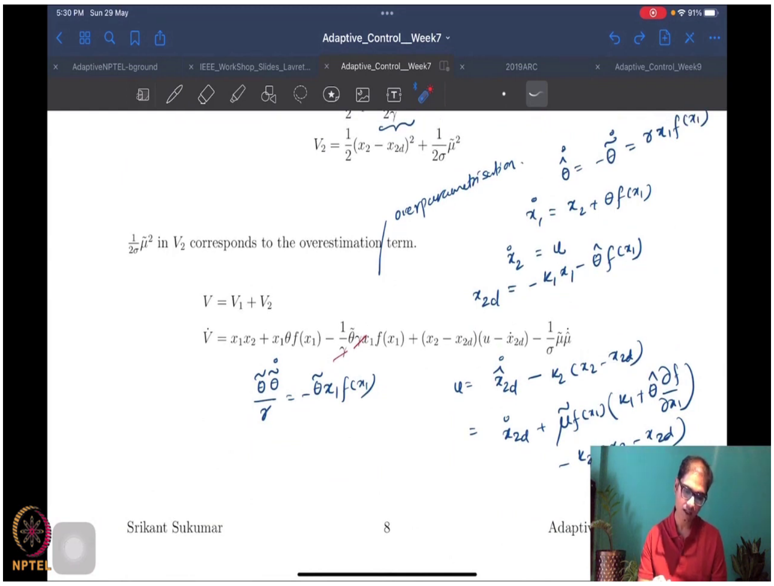


So, it has two terms in it earlier, you had two separate estimates. And the first estimate contained gamma x1, fx 1 as the first update law had gamma x1 fx 1 notice. And that is here, gamma x1 fx1 just in different notation.

(Refer Slide Time: 14:45)



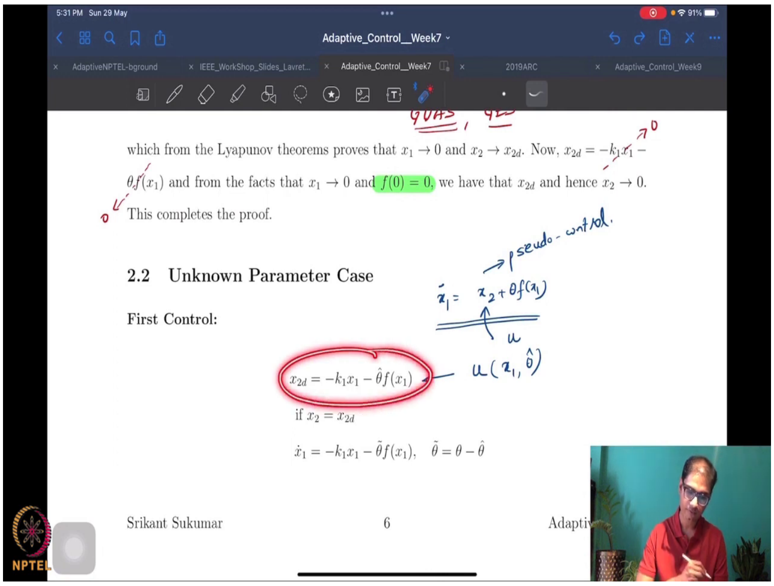


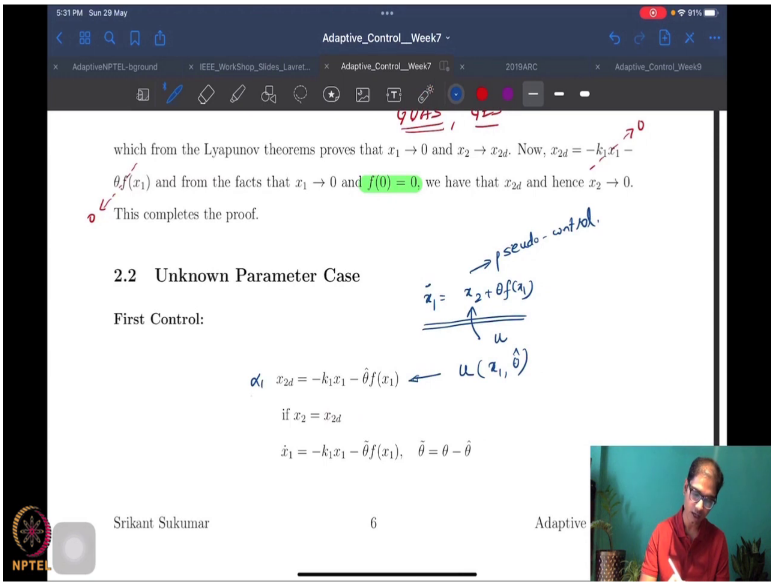


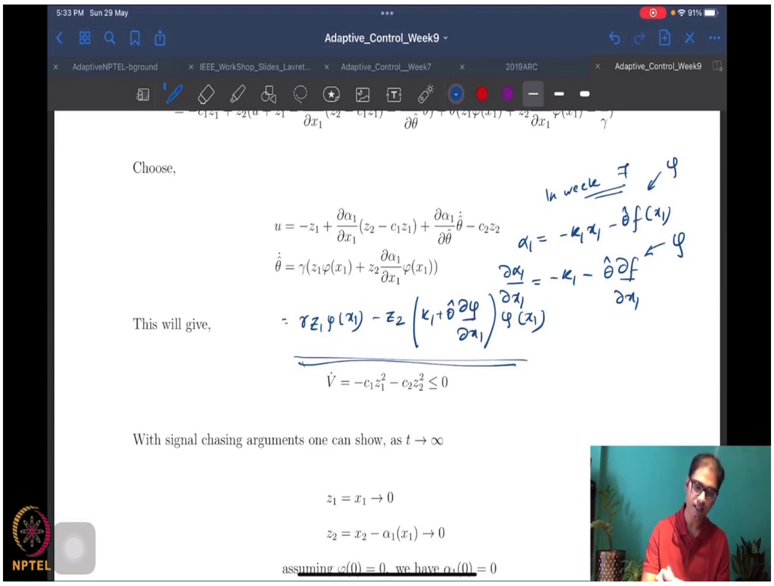
The second update law, that is the mu hat dot contained in something a little bit more complicated, which is the z term if you may. This is the z2 the way you have defined. So, it is a sigma z2 f times some this quantity times this quantity. So, if you look at this guy, you have z2 times a gamma times this quantity. It is actually the same quantity. It is just that there is, you know, there seemingly is an additional, you know, piece here, which is the k 1 term, which we do not have, which we do not seem to have here.

Which looks like we do not have this k1 term. Why did I am just trying to look at why we get the k1 term? That comes from the control here. And that comes in from? Yeah, I think that comes in from somehow this x2 desired dot term. That somehow comes in from the x2 desired dot term. And I, I believe that… I believe I look at it carefully, this term is exactly the same as what we have earlier. This term is exactly the same, because if I look at what was alpha 1 here, so let me look at what was alpha 1 from here. Let me try to evaluate in that sense.

(Refer Slide Time: 16:32)



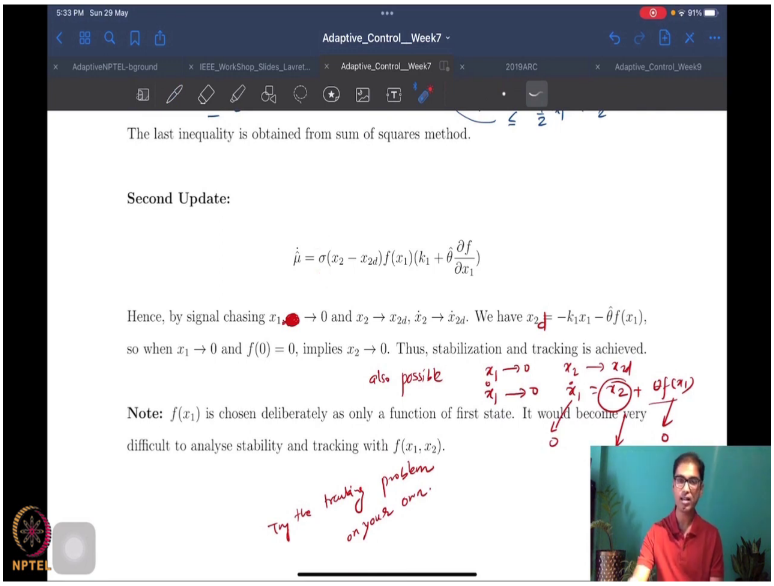


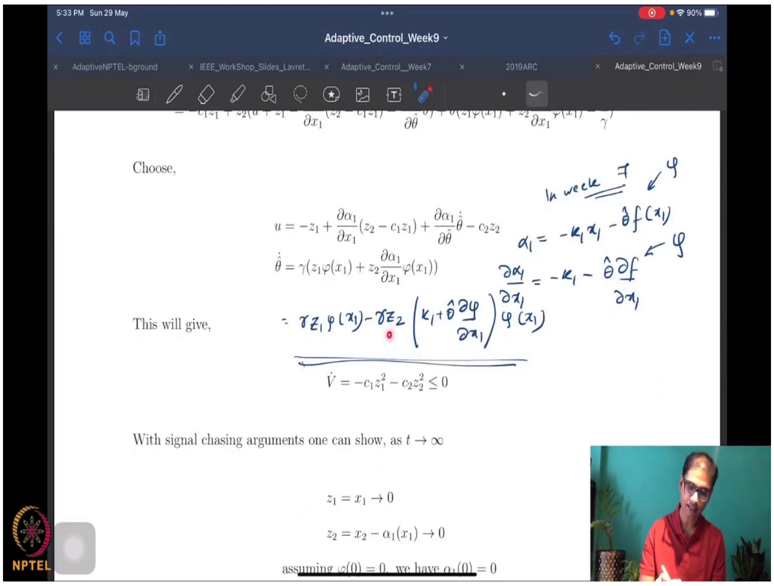


So, alpha 1 was this guy, Alpha 1 was exactly this guy. So, I can mark this as my alpha 1. So, write it as such. So, in the… in week 7 alpha 1 was actually minus k1 x1 minus theta hat fx1 minus theta hat, f x1. So, what I will do is I will try to evaluate, as we already have seen that the first term is identical. So, I want to evaluate the second term, to show that this is also identical to the second update law, we saw that the first term was identical to the first update law.

And what I want to do is to show that the second term is identical to the second update law. So, let me compute del alpha 1, del x1, and that is minus k1 minus theta cap, del f del x1, so this term becomes gamma z1 phi x1 plus or minus z2. So, our f is actually equal to phi, remember. So, I am going to write this as phi, here, use phi instead of an f, that's ok, minus z2 k1 plus theta cap, del phi, del x1 times phi x1. Yeah, that is what would be what you would get for the week 7, type control.

(Refer Slide Time: 18:27)





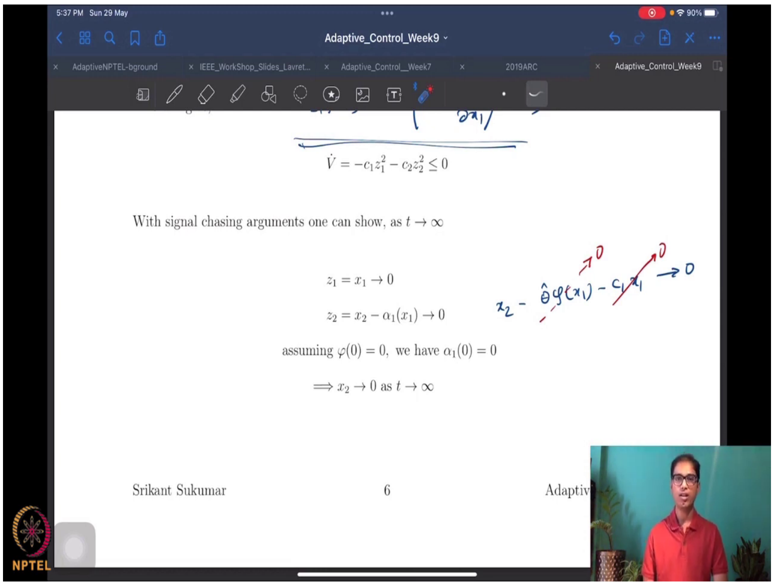
And if you look at the second update law, that was the mu hat dot, it is the same. It is sigma, some gain, which is gamma in our case. It is the same you have gamma, wait a second. I'll remove this then you have the same you have the gamma, then you have the z2, which is this guy, z2, and then you have exactly of phi fx1, which is the phi x1, then you have k 1 plus theta cap del phi del x1 k1 plus theta cap del phi del x1.

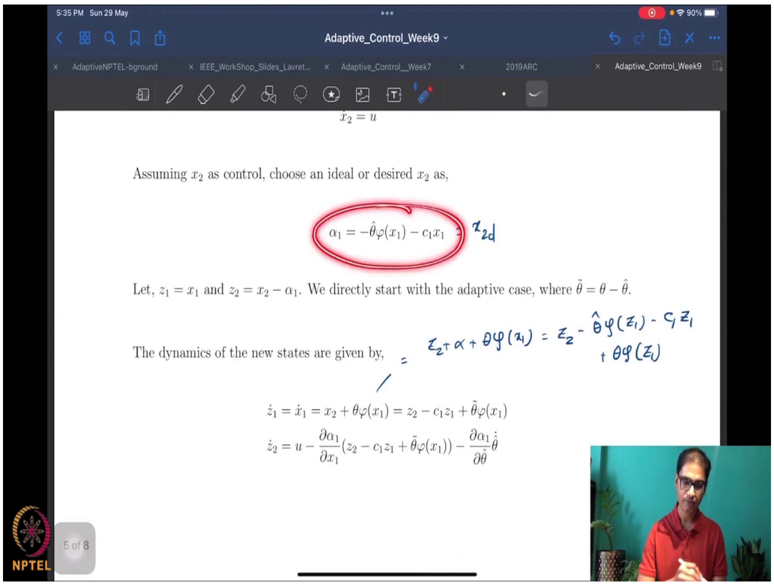
So, this is exactly the same term. So, let us not worry about the sign so much, but what you see is you have exactly the same term as the second update law. So, that is really the idea of what has happened by not choosing a V1 and a theta cap dot in the first step is that we have obtained the theta cap dot which is essentially just an addition of the two update laws. Then we have two parameter estimates and two update laws.

So, we have essentially obtained the same you know, two update laws added together for theta cap dot in this case. So, this is rather cool. Yeah, we did not have to, you know, actually designed two different estimates. And it almost seem like a posteriori. Once I look at this, it seems like it is almost just, you know, I mean, why did you even do the previous method is what you think.

So, once we actually have done this like have actually implemented this theta cap dot, and this u, what we have is that you get a nice negative term here and a nice negative term on the second piece, and this goes to 0. So, I have a nice negative definite negative semi definite V dot, just like you expect in all adaptive control.

(Refer Slide Time: 20:44)



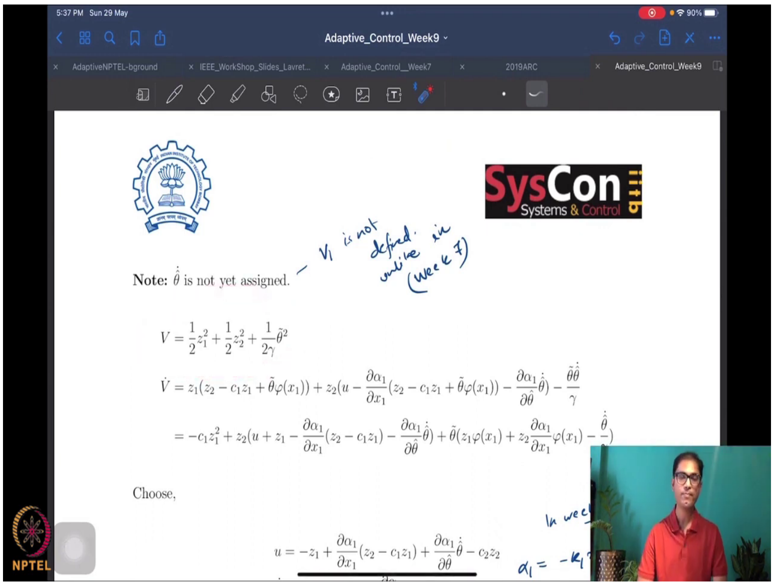


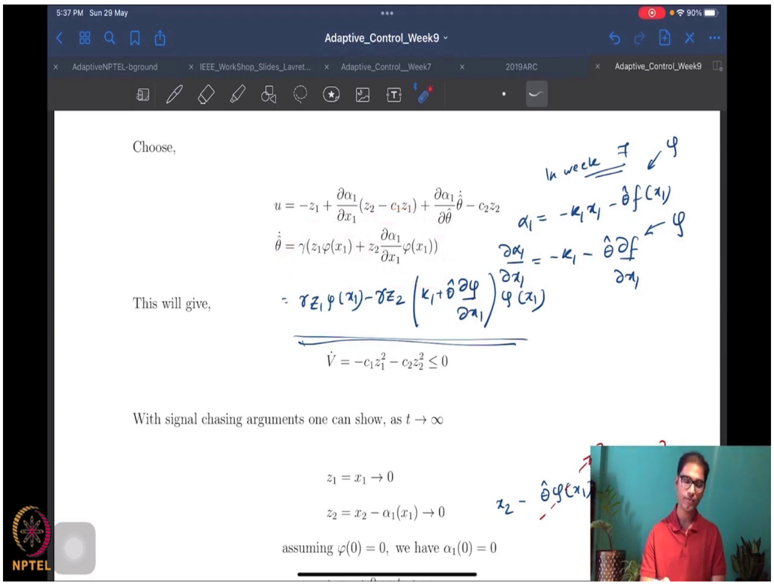
And from standard signal chasing arguments, what you can show is that z1, that is x1 goes to 0 and z2, that is the backstepping error actually goes to 0. And if you remember, what is alpha 1? Alpha 1 is again, this guy, I mean, this guy, theta phi x1 minus c1 x1. So, this is x2 minus theta hat, phi x1 minus c1 x1, let me see, yeah, goes to 0. Now, I already know that x1 is going to 0. Now, if I also assume that phi 0 is 0, that is, when x1 goes to 0, this quantity becomes 0, then this guy will also go to 0.

And therefore, you will be able to claim that x2 goes to 0, as t goes to infinity. And that is essentially what you want. You want all the states to go to 0? Yeah, again, we are looking at the stabilization problem, the tracking problem would have been no different. But so, in this case, we are concerned with the states x1 and x2 actually going to 0, and you obtain that, that is essentially what you wanted. And that is essentially what once you obtain.

So, this is sort of the extended matching method. And the idea is rather simple. You avoid, you know, designing the first level parameter estimate. And the first level candidate, Lyapunov function, and you simply go to the next step, and directly design a complete system Lyapunov candidate function.

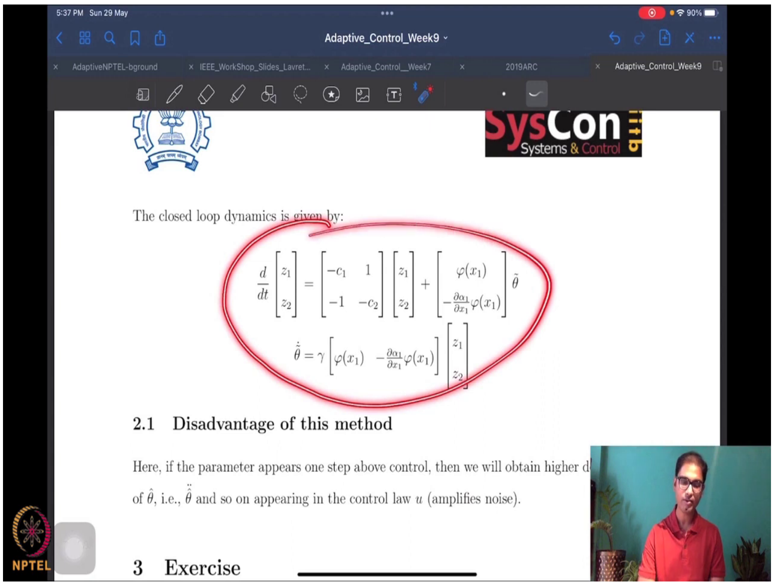
(Refer Slide Time: 22:41)

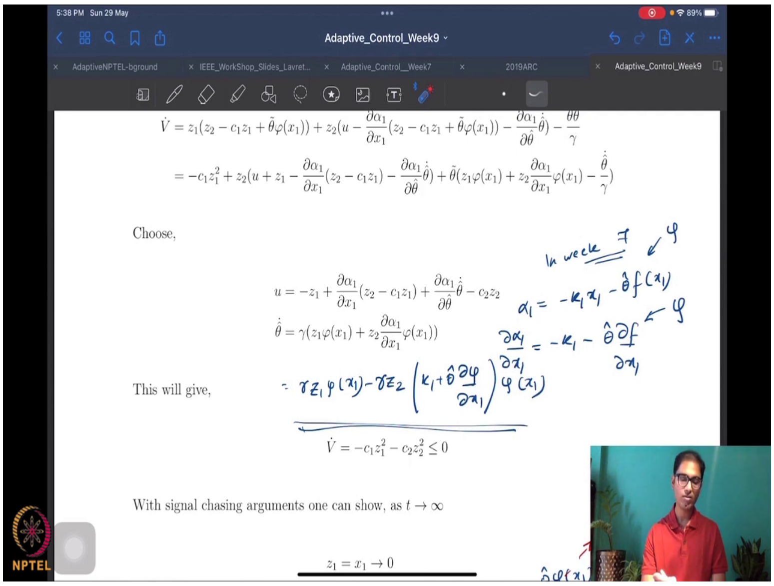




This actually helps you with, you know, removing one of the parameter estimates, and you also find that structurally, the parameter estimate that you get is ident.. I mean parameters update law that you get is essentially contains the sum of the two parameter update laws from last time.

(Refer Slide Time: 23:09)

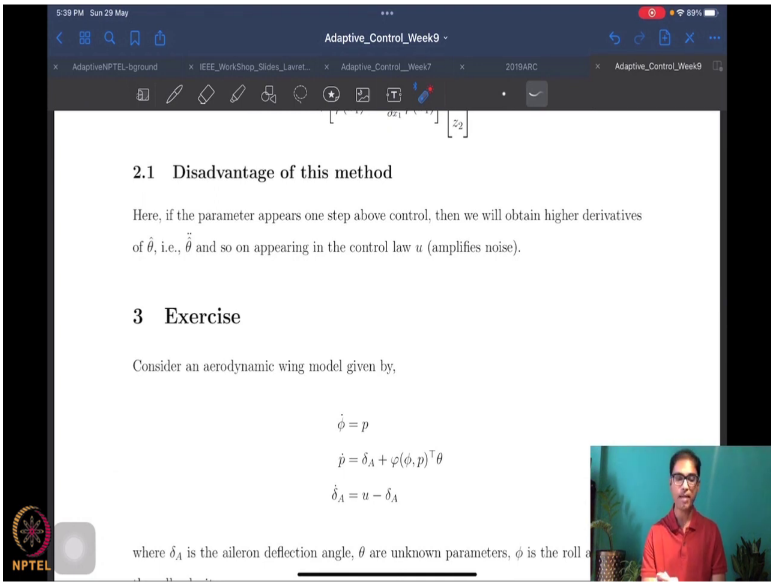


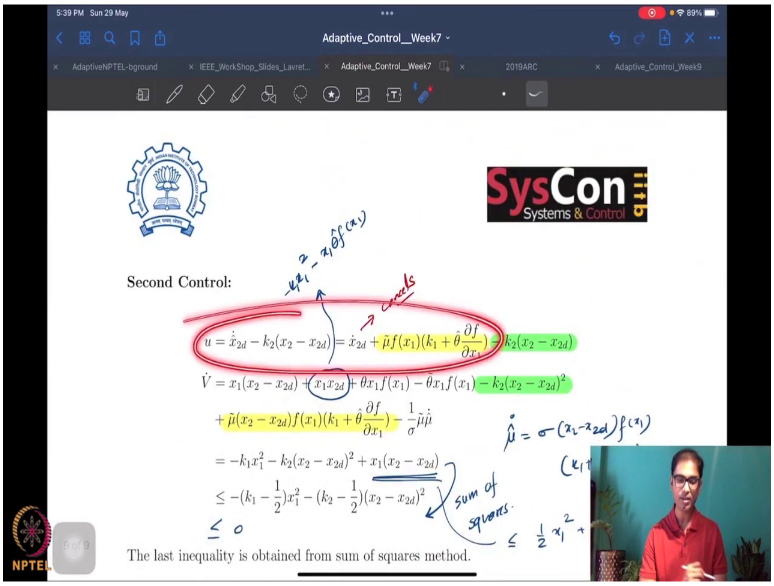


Now. so, that is sort of, you know, something cool. So, this is the sort of closed loop dynamics you would get, if you actually plug in everything and put everything in place. Yeah, there is the sort of closed loop dynamics you will get. Now, one of the issues that you have is that basically, you when you look at the control law, here the control law is not appearing.

So, if you look at the control law here, you can see that there is a theta cap dot which appears now this theta cap dot appears, because the unknown parameter is 1 step above the control. Now, it is not difficult to see just by looking at the pattern that if the control appears two steps below the unknown parameter, then a theta cap double dot will appear and so on and so forth.

(Refer Slide Time: 24:18)

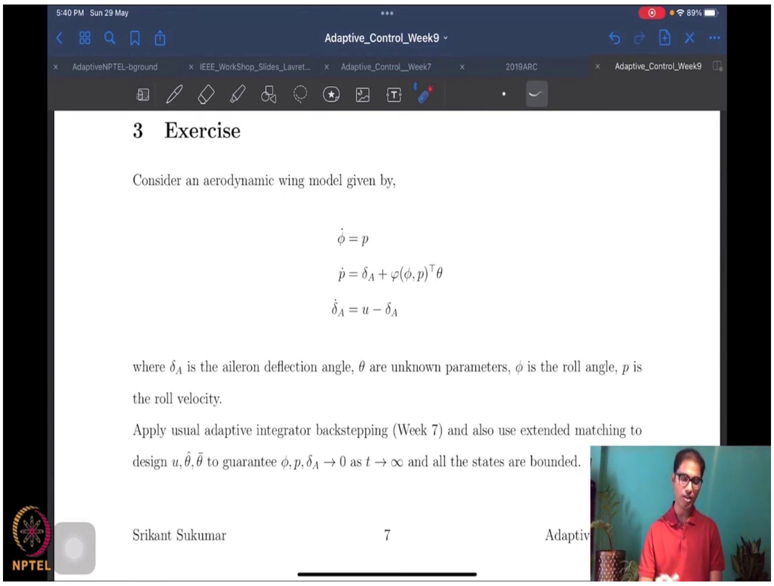




And usually, you know, I mean, it is not considered very healthy to have derivatives of quantities, and this can lead to amplification of noise. So, you notice that, this does not have happen in the week 7 method. In the week seven method, there is no theta cap dot that appears in the controller. And this is a good thing in the non-extended case, but in the extended case, this extended matching design, this is a problem.

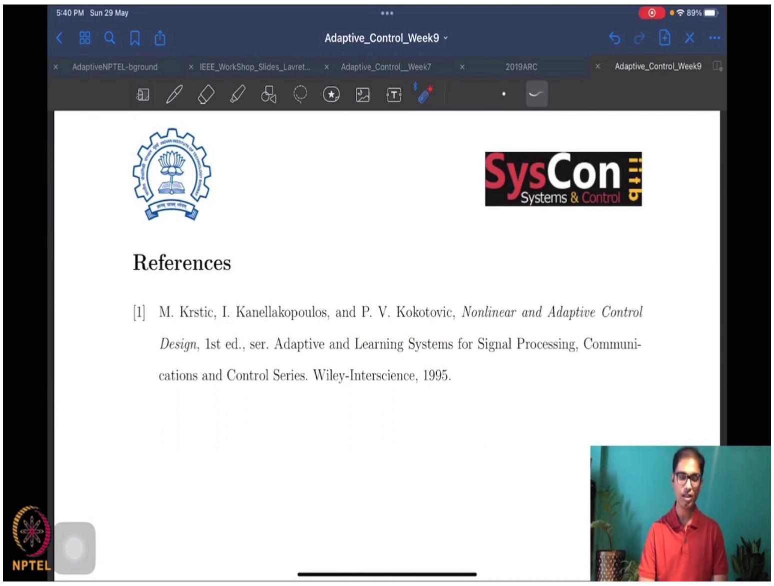
So, it is not exactly apples to apples, there is a slight difference, the control law, which sort of gets hidden in the closed loop dynamics contains the theta cap dot, which can lead to problems in implementation. And these as you go to control which is lower and lower below the unknown parameter in the dynamics, then the theta cap double dot, tripled hence and multiple derivatives and higher and higher derivatives of theta cap will start to appear.

(Refer Slide Time: 25:17)



So, I mean, keep this in mind, this is a sort of nice exercise. Anyway, I mean these, I mean, I recommend that you do this as an aerodynamic model for a wing. And you want to try to use the typical integrator backstepping, and also the extended matching method in order to do a stabilization of this model. And that is the idea.

(Refer Slide Time: 25:38)



So, this is the reference for this set of lectures, and very, very useful. So, I strongly recommend that you look at these notes. So, anyway, so what did we do this time, in this set of lectures today, is that we looked at the extended matching design method. And what the extended matching design method does is to sort of reduce the number of parameters that you are trying to estimate.

So, we are still have just one estimate for one parameter or one set of parameters. So, you do not have a larger number of estimates than the number of parameters. The problem we see that we end up with is that you find higher derivatives of the update law, which may or may not be convenient in implementation, because of noise issues, etc, etc, so keep this in mind.

So, that thing is free. So, we hate some issues in the control laws itself, yeah, but other than that, yes, we deduce the number of states in the controller and this is of course, a significant advantage when you are looking at implementations. Alright, great. So, this is where we will stop now. We will continue in the subsequent session. Thank you.