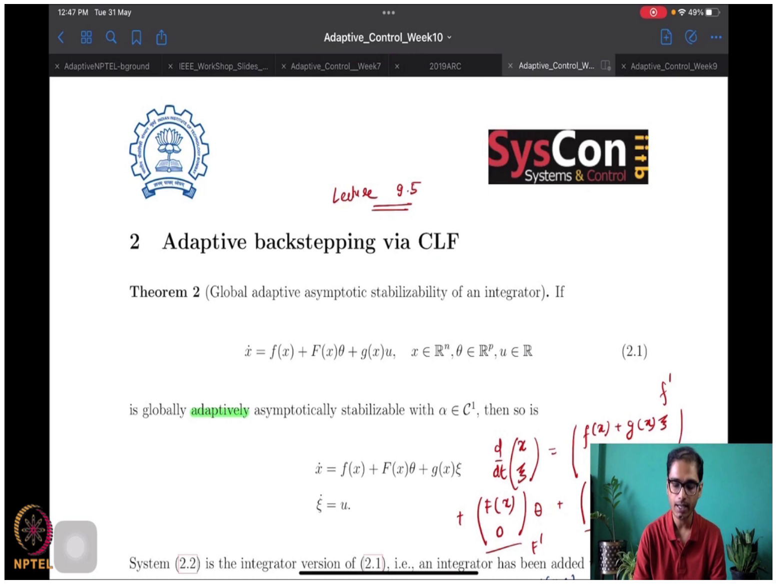
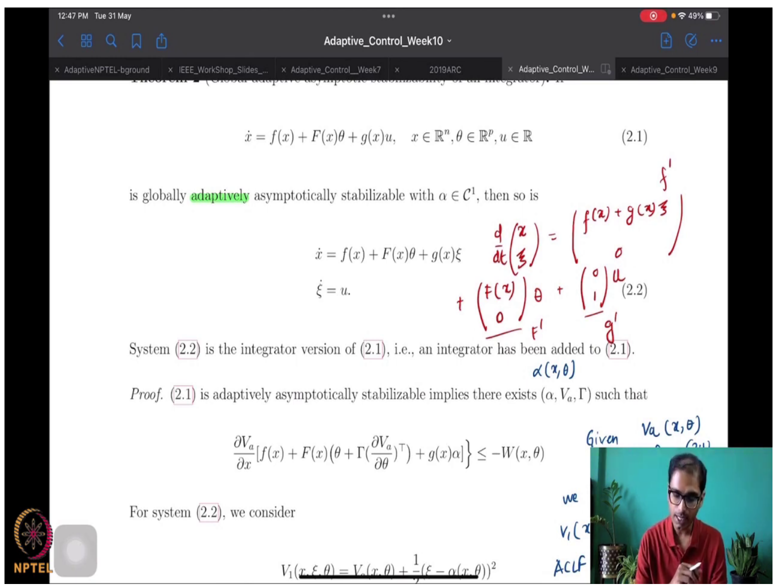
Nonlinear Adaptive Control
Professor Srikant Sukumar
Systems and Control
Indian Institute of Technology, Bombay
Week 9
Lecture No: 54
Adaptive Backstepping via CLF: An Example

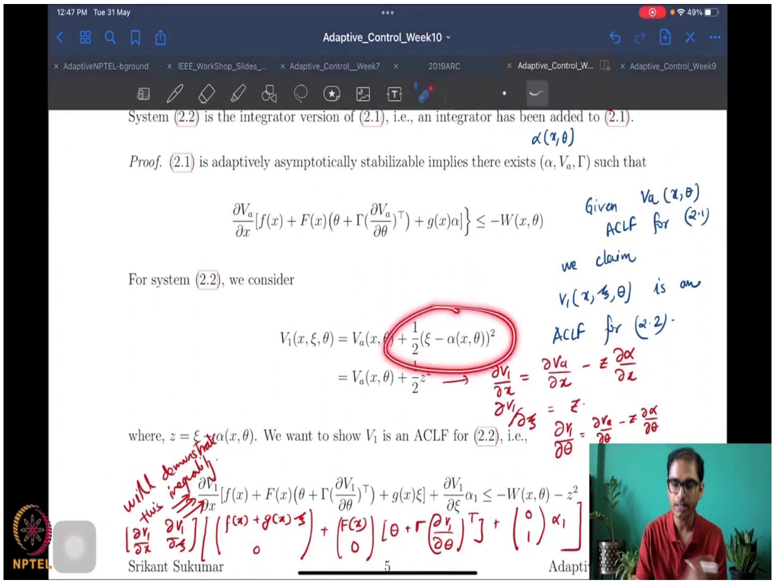
(Refer Slide Time: 00:17)

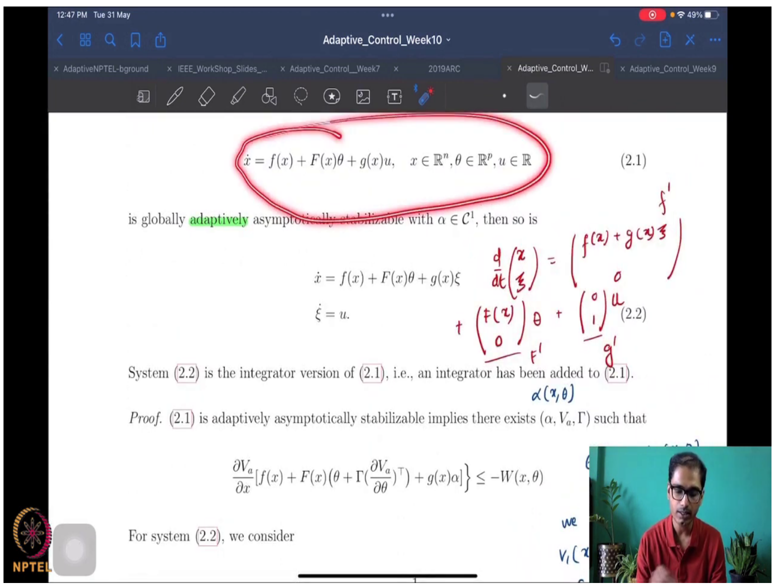
Hello, everyone. Welcome to yet another session of our NPTEL on Nonlinear and Adaptive Control. I am Srikant Sukumar from Systems and Control, IIT Bombay. So, we are pretty much at the end of week 9 of our lectures. And we have hopefully done a pretty good job of designing and analyzing algorithms that will drive autonomous systems such as the SpaceX satellite orbiting the earth, that we see in our background.

(Refer Slide Time: 00:55)







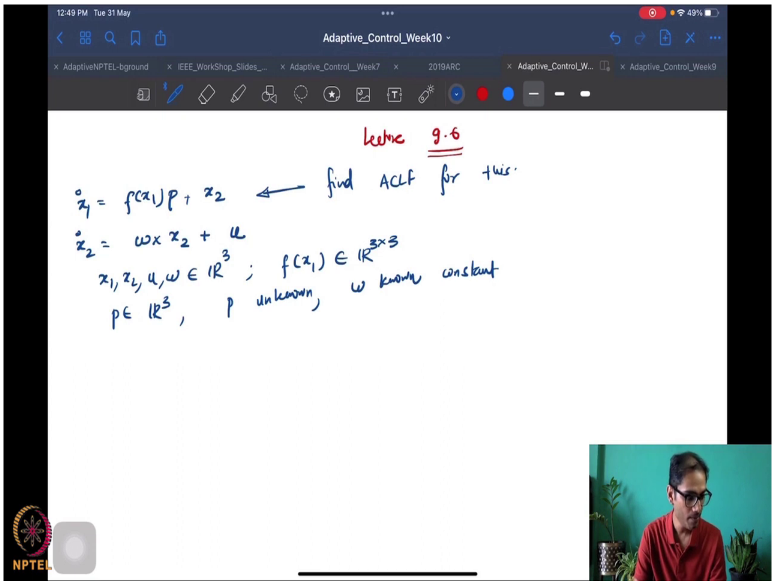


So, what we did last time was essentially the tuning function method for designing adaptive backstepping laws. So, we essentially saw that having an ACLF for a system of the form 2.1 was implied immediately that we could construct an ACLF for system of the form 2.2, which is essentially an added integrator layer.

Now, we of course had particular methods of doing that. We, we essentially took the ACLF of system 2.1 and added to it, a backstepping error term. We added to it, a backstepping error term. And that was how we constructed a new control law and of course we got an adaptation law and so on so forth.

Now, of course in order to be able to do all this, we need to have an ACLF for the original system also. And how one can come up with an ACLF for this system 2.1 again goes back to the same idea as adaptive integrator backstepping type methods. So, what I want to do is to sort of do a slight modification of this as an example.

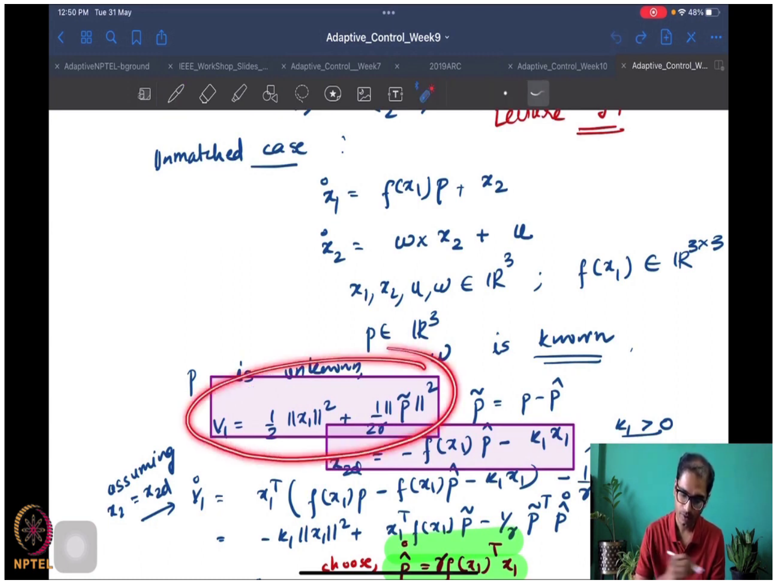
(Refer Slide Time: 02:20)



If you remember we had considered this particular example for adaptive integrator backstepping, and also extended matching. But I want to try to at least use the ACLF based backstepping method on this problem. Now, remember, until now that for the ACLF method and the ACLF backstepping method, the control was always assumed to be a real number. Again, xi was also a real number because it had the same dimension as the control. But here our control and this x 2 which is a xi state is in R3. So, we are still going to try to do this, try to use the tuning function ACLF method. And hopefully it works. Let us give it a shot.

So, this is the beginning of lecture 9.6. We are getting a little bit adventurous here. We did the theory on scalar control or a single input system but we are actually trying to solve a multiple input order vector control problem. So, in this case, we know that p is unknown and omega is a known constant. Now, the first question is what is an ACLF for this system. So, the step 1 is find ACLF for this. How do we do that? What exactly was an ACLF? An ACLF essentially guaranteed that this system was adaptively asymptotically stabilizable. Now, so what I would do is I would take notification from how we solved the unmatched case using the integrator backstepping method.

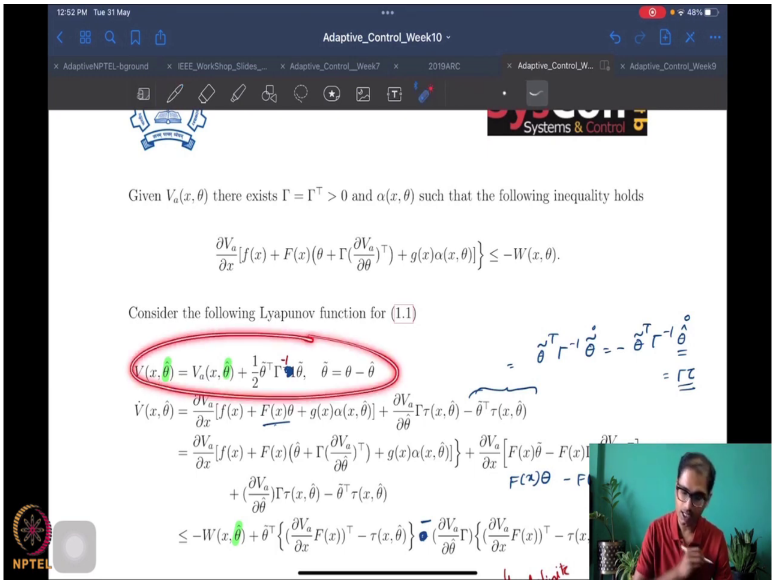
(Refer Slide Time: 04:20)

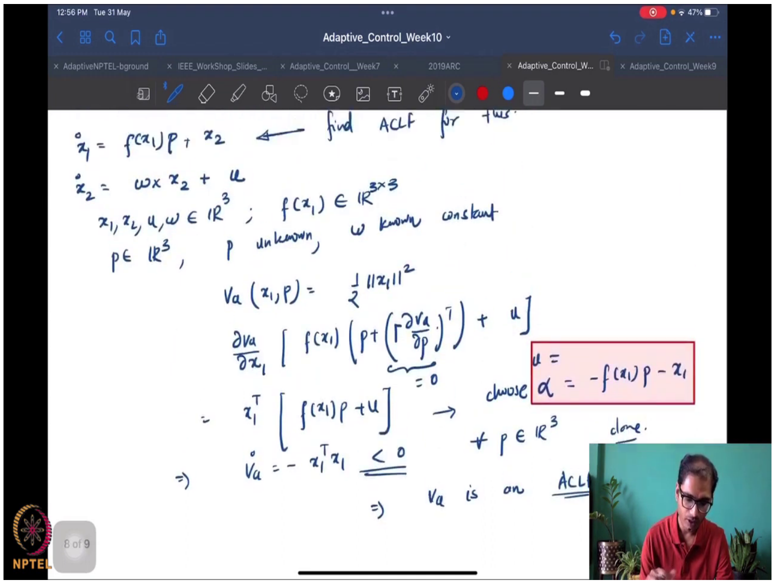


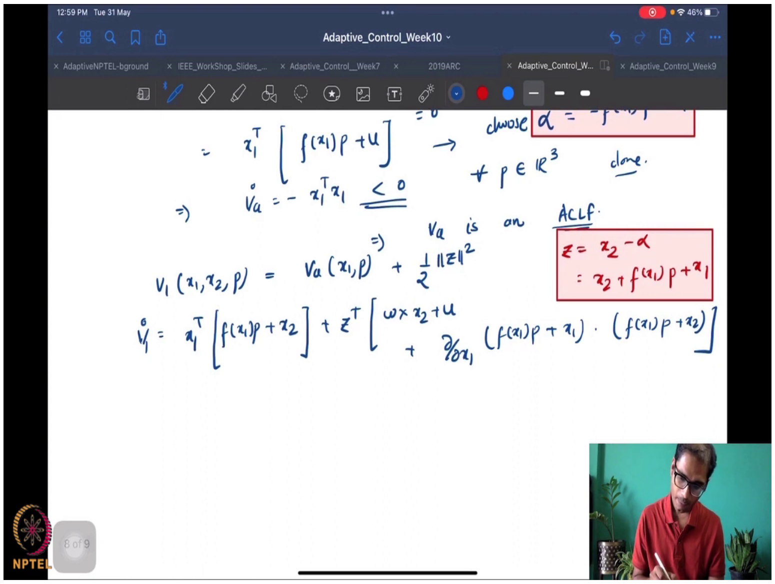
Because in the integrator backstepping method, we of course had an Lyapunov candidate function, and that is what we will try to use. And that function was this V 1 function. And that was this V 1 function because we had an x 2 desired and of course, we had, whatever, we had this, which was this, which is this alpha, if you may, in the tuning function setting, and with that you can actually show that now you have some nice properties.

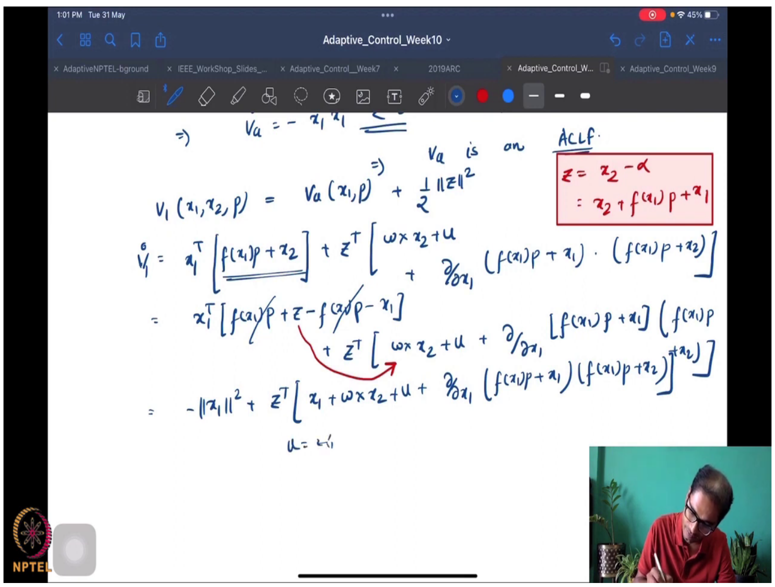
So, what we propose is that our V a, assuming again, I mean that that everything is known, so p is known, so I am going to call this V a x 1 comma p is equal to one half, let us see is this correct, is this correct, is the question here. One half x 1 squared is that what we are saying here. So, we have to be careful to figure out what is going to be the appropriate ACLF. So, one term is of course one half norm x 1 squared, this is one piece. Now, will it have any dependence on theta, is what I am wondering.

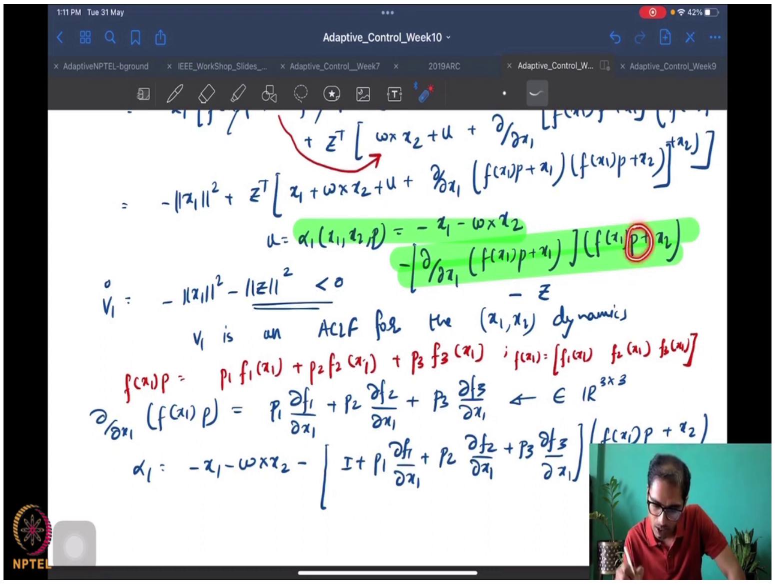
(Refer Slide Time: 06:11)

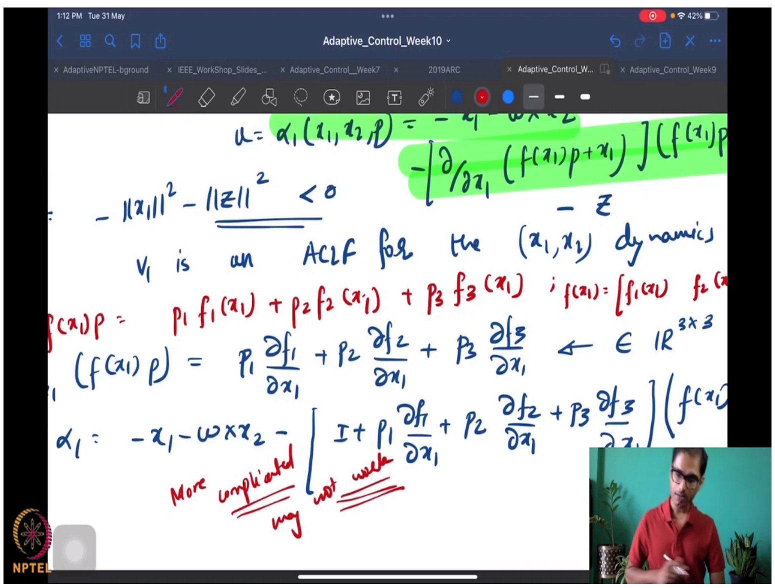


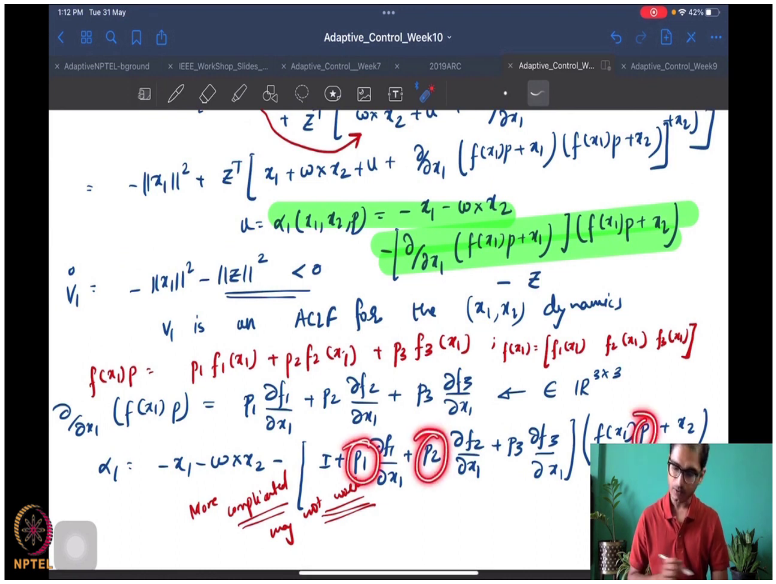



Adaptive_Control_Week10
= x1^T [f(x1) p + u] → choose α = -f(x1)p
⇒ V̇a = - x1^T x1 ≤ 0   ∀ p ∈ R^3   done.
⇒ Va is an ACLF.
z = x2 - α
= x2 + f(x1) p + x1
V1 (x1, x2, p) = Va (x1, p) + 1/2 ||z||^2
V̇1 = x1^T [f(x1)p + x2] + z^T [ω×x2 + u + ∂/∂x1 (f(x1)p + x1) · (f(x1)p + x2)]


Adaptive_Control_Week10
⇒ Va is an ACLF.
z = x2 - α
= x2 + f(x1) p + x1
V1 (x1, x2, p) = Va (x1, p) + 1/2 ||z||^2
V̇1 = x1^T [f(x1)p + x2] + z^T [ω×x2 + u + ∂/∂x1 (f(x1)p + x1) · (f(x1)p + x2)]
= x1^T [f(x1)p + z - f(x1)p - x1] + z^T [ω×x2 + u + ∂/∂x1 [f(x1)p + x1] (f(x1)p + x2)]
= -||x1||^2 + z^T [x1 + ω×x2 + u + ∂/∂x1 (f(x1)p + x1)(f(x1)p + x2)]
u =


Adaptive_Control_Week10
+ z^T [ωx x_2 + u + ∂/∂x_1 [f(x_1)p + x_1] (f(x_1)p + x_2)]
= −||x_1||^2 + z^T [x_1 + ωx x_2 + u + ∂/∂x_1 (f(x_1)p + x_1) (f(x_1)p + x_2)]
u = α_1(x_1, x_2, p) = − x_1 − ωx x_2 − [∂/∂x_1 (f(x_1)p + x_1)] (f(x_1)p + x_2) − z
V̇_1 = − ||x_1||^2 − ||z||^2 < 0
V_1 is an ACLF for the (x_1, x_2) dynamics
f(x_1)p = p_1 f_1(x_1) + p_2 f_2(x_1) + p_3 f_3(x_1) ; f(x_1) = [f_1(x_1) f_2(x_1) f_3(x_1)]
∂/∂x_1 (f(x_1) p) = p_1 ∂f_1/∂x_1 + p_2 ∂f_2/∂x_1 + p_3 ∂f_3/∂x_1 ← ∈ IR^{3×3}
α_1 = − x_1 − ωx x_2 − [I + p_1 ∂f_1/∂x_1 + p_2 ∂f_2/∂x_1 + p_3 ∂f_3/∂x_1] (f(x_1)p + x_2)
NPTEL


Adaptive_Control_Week10
u = α_1(x_1, x_2, p) = − x_1 − ωx x_2 − [∂/∂x_1 (f(x_1)p + x_1)] (f(x_1)p
− z
− ||x_1||^2 − ||z||^2 < 0
V_1 is an ACLF for the (x_1, x_2) dynamics
f(x_1)p = p_1 f_1(x_1) + p_2 f_2(x_1) + p_3 f_3(x_1) ; f(x_1) = [f_1(x_1) f_2(
(f(x_1) p) = p_1 ∂f_1/∂x_1 + p_2 ∂f_2/∂x_1 + p_3 ∂f_3/∂x_1 ← ∈ IR^{3×3}
α_1 = − x_1 − ωx x_2 − [I + p_1 ∂f_1/∂x_1 + p_2 ∂f_2/∂x_1 + p_3 ∂f_3/∂x_1] (f(x_1
More complicated way not weak
NPTEL



So, what we had done to begin with was, and then in order to prove, and then in order to prove that we have adaptive asymptotic stabilization, we add it to this V theta tilde term. So, now the, the question that we have is if this term is in fact, this, this V 1 is in fact ACLF or not. So, let me just think about this a little bit carefully. I believe this is fine. So, this V 1 is an ACLF, this V a is an ACLF. Why do I say this is because if I take V a dot, that is del V a del x 1 times f x 1 plus gamma, sorry, times p plus del V a del p times gamma, sorry, gamma del V a del p transpose, because that is the additional term.

So, the modified system is gamma del V a del theta transpose. So, if I take this here, plus whatever, x 2 is the control, so I am going to call it, say, u, then this term is equal to 0 because this term does not depend on, this V a does not depend on theta at all, or p at all, so here we will have x 1 transpose times f x 1 p plus u. So, if I choose u as minus f x 1 p minus x 1, or basically I will call this u equal to alpha, so you are choosing alpha, as this guy, so I am going to choose this alpha as this guy.

I apologize. Give me a second. If I choose my alpha as this guy, then I get V a dot is equal to x 1 transpose, minus x 1 transpose x 1, which is negative definite for all p. So, done. So, this is in fact, this V a is in fact an ACLF. Implies V a is an ACLF. So, V a is an ACLF for the first state. So, remember, I replace the first state with the control here, I replace the first state with the control here because that is how you would find an ACLF for the first system.

Now, the question is, how do I find the ACLF for the second system? You know the method. Now, V 1 is x 1 x 2 p, and this is simply V a x 1 p plus the backstepping error term. Now, in this case, I cannot just do a square. So, I will take a norm squared. So, backstepping error is, of course, just z equal to x 2 minus alpha, which is x 2 plus f x 1 p plus x 1. So, this is just the standard backstepping error term.

And so now once I know that, so this is just norm of z square. So, this is what is my ACLF for the new system. So, for corresponding to this, how do I find the control? Just take the derivative. I mean, so this is how we did it. So, V 1 dot is basically V a dot which is x 1 transpose x 1 dot. So, x 1 transpose x 1 dot is just f x 1 p plus x 2. And plus z transpose z dot, which is basically x 2 dot which is omega cross x 2 plus u plus del f del x 1.

So, this is del f del x 1. This is going to be more complicated, I think, because f x 1 is, is a matrix, f x 1 is a matrix. So, this, has to be written in a nicer way, this has to be written in a nicer way. But this is, let us see, this is a function of one state. So, I have to sort of write this in a smarter way. So, I, I think I will not get into the detail that much for this. I am trying to wonder if I should get into the detail here. I will just call this del del x 1 f x 1 p plus x 1 times x 1 dot which is f x 1 p plus x 2. So, that is what is the expression. Now, what do I know about this guy is that it has x 2. So, I can write this as z plus alpha.

So, this is x 1 transpose z plus alpha which is z minus, so this is f x 1 p plus z minus f x 1 p minus x 1 plus z transpose, this whole quantity. Again, I am writing this just as such times x 1 dot. I am writing this just as the whole thing. And this is basically just going to cancel out these quantities. And I am going to left be left with minus norm x 1 squared plus z transpose.

So, this guy will move here. So, sorry, so this quantity will come inside this bracket. So, this is x 1 plus omega cross x 2 plus u plus del del x 1 f x 1 p plus x 1 times, this is just, again, f x 1 p plus x 2. And that is it. So, now, I control will, u defined as alpha, I apologize, as alpha 1 of x 1 x 2 p will be minus x 1 minus omega cross x 2 minus del del x 1 f x 1 p plus x 1 times f x 1 p plus x 2, and minus a nice negative term.

So, this gives me V 1 dot is minus norm x 1 squared minus norm z squared, which is negative definite, and of course we have obtained our ACLF. So, V 1 is an ACLF for the entire system, for the x 1 x 2 dynamics. In the process, what is our feedback? It is this guy. The feedback law is this.

Now, there is a little bit of complication here in the sense that, again, this is because we have vector control and all that stuff. If you look at this f x 1. So, this f x 1, if you remember was a matrix. That is why I have to think carefully before taking a Jacobian of this. That is the whole point.

So, I have to, because I have to take the partial of a matrix with respect to x 1. So, actually what you would have to do you have to write this f x 1 as a vector and multiply it as such. So, so essentially what would happen is that, I mean what I would have to do as a separate sort of problem is write f x 1 p as p 1 f 1 x 1 plus, this is a summation, so p is in say R, well p is R3, so it is not so bad.

So, plus p 2, f 2, x 2, sorry f 2 x 1 plus p 3 f 3 x 1, where f x 1 is being written as three columns. f x 1 is being written as f 1 x 1 f 2 x 1 and f 3 x 1, basically the three columns. The three columns. And p 1, p 2, p 3 are the three parameters. And now if I take a partial of this, so I am sorry, so del del x 1 of f x 1 p is going to be p 1 del f 1 del x 1 plus p 2 del f 2 del x 1 plus p 3 del f 3 del x 1.

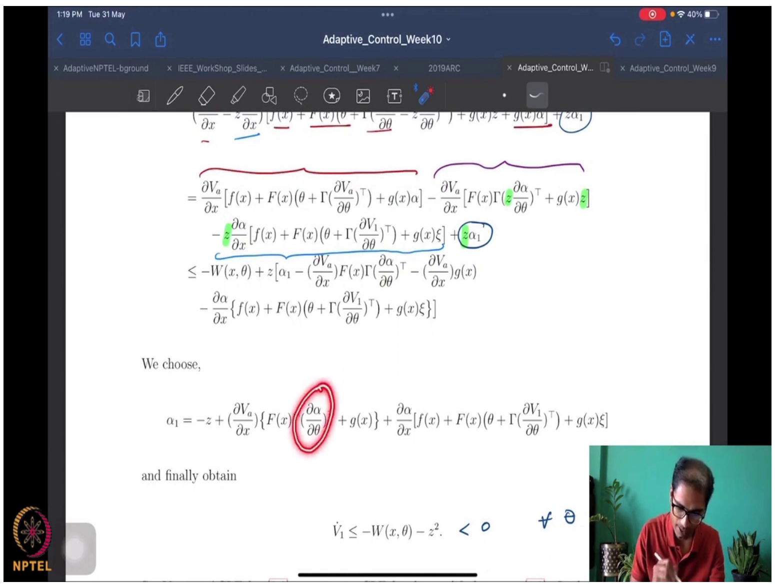
So, this is what it is going to be? So, that is what is del del x 1 of f x 1 p and that gets substituted inside here. It is better that I keep it like this. So, basically my alpha comes out to be, alpha 1 comes out to be minus x 1 minus omega cross x 2 minus, so the partial of x 1 with respect to x 1 is essentially the identity matrix. So, we will have identity plus p1 del f 1 del x 1 plus p 2 del f 2 del x 1 plus p 3 del f 3 del x 1.

And this is multiplied by f x 1 p plus x 2, and this is multiplied by f x 1 p plus x 2. So, this is what you have for your feedback. Now, again one has to be a little bit careful here, yeah, one has to be a little bit careful here and see if even this works out. Why I am wondering if this will work out is because now this guy looks quadratic in your unknown, because there is an unknown here, there is an unknown here.

So, somehow if I multiply this with that it looks like I will get a quadratic in my unknown. I will start to get a quadratic in my unknown. And this might put me in a soup, this, not this might, this will certainly put me in a soup, this will certainly put me in a soup. So, I think this sort of a setup is more complicated and may not work, may not be able to figure out.

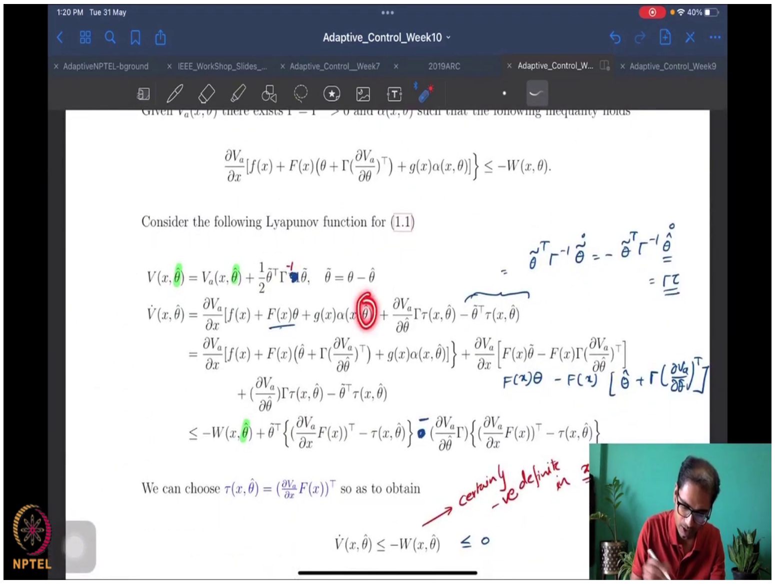
Because see, we always require linear parameterization. So, now what is happening is that with this kind of a control law, if I replace, for the unknown case I replace the p with the p hat, then there is a quadratic here in terms of the p. So, this may be more complicated, but let me actually go back and look at the expression that is here.

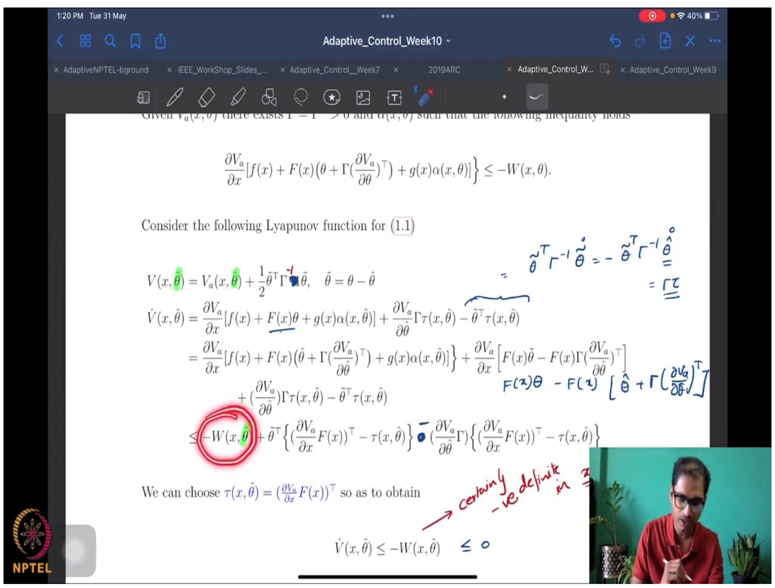
(Refer Slide Time: 22:29)

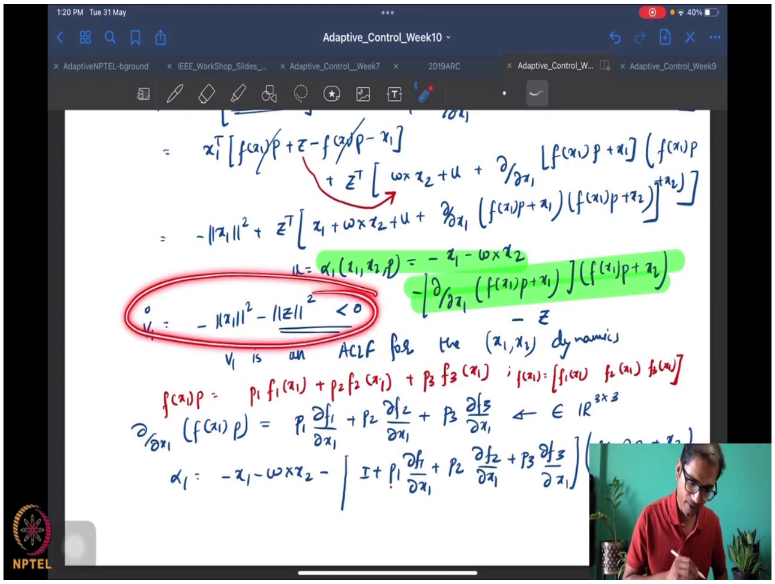


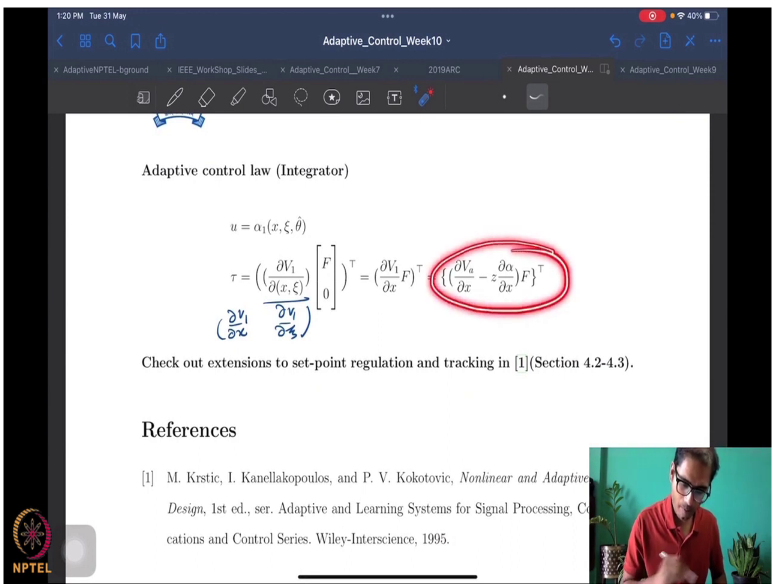
If you look at how you move from alpha to, how the alpha one gets constructed here, we can see clearly that alpha is a function of both x and theta. And so, in this alpha 1 expression, here you could have a dependence on theta, here you could have a dependence on theta, and here you already have a theta. So, the idea is these two can also combine to give you a quadratic. I guess this does not matter that therefore that the alpha 1 contains quadratic in theta. This is not of our concern.

(Refer Slide Time: 23:09)





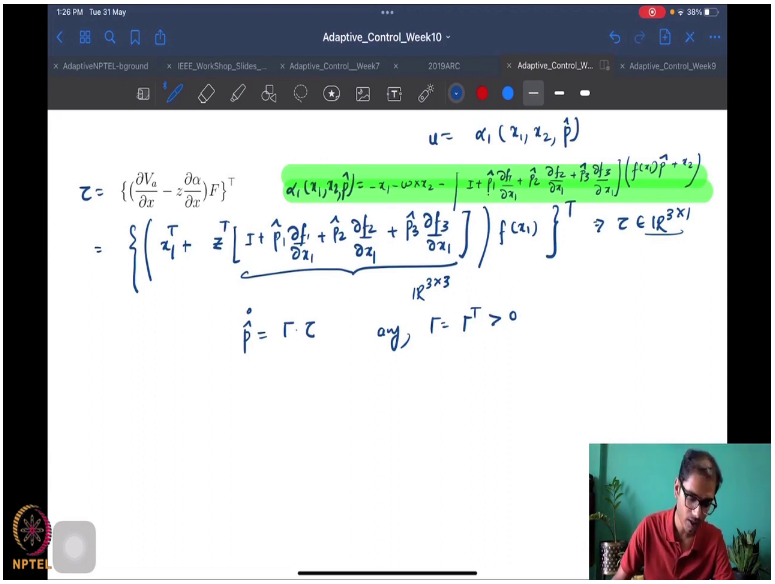


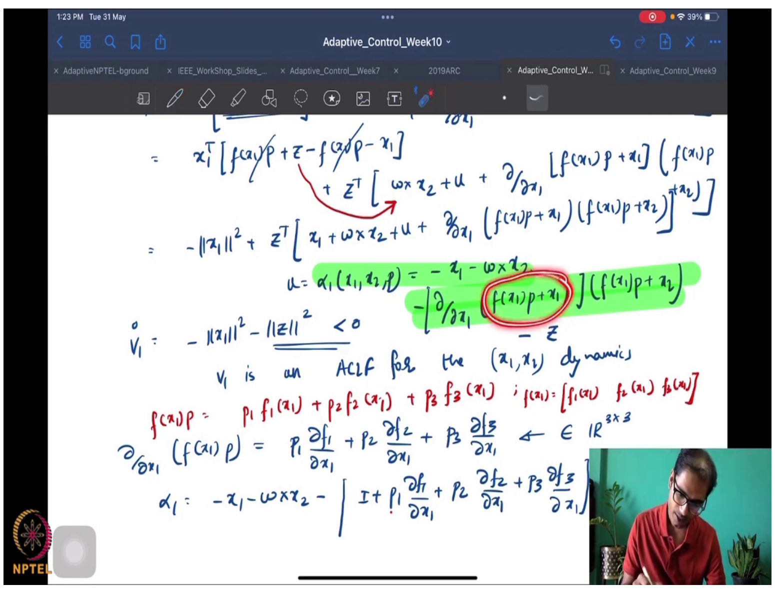


What will happen when we design the adaptive law is that we just replace the theta with the theta hat and we will still get a nice negative definite term in the x and theta hat. So, all we are doing is replacing the theta with the theta hat. So, I guess this is absolutely okay. This is going to be fine. I will not say this is complicated. This is okay, this is the alpha 1. For whatever it is worth, this is, in fact, the alpha 1.

And we have our V dot like we desire, which is an ACLF. And now that we have an ACLF, the expression for the tau is this guy. So, I just have to compute this, which is now del V a del x. So, I am going to actually try to copy this, so I am going to just copy this. So, this is copied. And so, this is the expression that I am trying to compute for the tuning function because I already, so my feedback is of course, alpha 1 x 1 x 2 and a p hat, so I replace p by p hat everywhere.

(Refer Slide Time: 24:16)





So, if you want, I can even reproduce that expression here. This is going to be, yeah, this whole mess. And all I am doing is adding the hats to these quantities and this is my alpha 1 x 1 x 2 p hat. And this is going to be my tau. And so, what is my tau? My V a was simply half norm x 1 squared.

So, del V a del x is basically just x 1 transpose minus z del alpha del x, which is the first feedback. So, alpha was this guy. So, I will again have something like, this expression. So, alpha was essentially this. So, partial of, so alpha was essentially this guy with a negative sign. So, the partial of this is simply going to be this entire thing plus an identity with a negative sign.

So, this is going to be plus identity plus p 1 del f 1 del x 1 plus p 2 del f 2 del x 1 plus p 3 del f 3 del x 1. And this whole thing multiplied by f. In our case what is f? f is the what multiplies the unknown. So, that is just f x 1. Multiplied by f x 1 and the whole transpose of this kind. So, let me see if we are cons, consistent in the dimensions.

So, if you are consistent in the dimensions here. So, this is my control law and this is my tau. I am just trying to verify if my dimensions are correct. So, x 1 transpose is a 3 by, 1 by 3 vector, z is a 3 by 1 vector. I am trying to see if we got this correct. So, this is a 3 by 3 vector. So, this is all 3 by 3. So, I am trying to see if this is correct. I think this should be a z transpose.

So, this is 3 by 3. So, this is 1 by 3 times 3 by 3, so this is fine, 1 by 3. This is 3 by 3, so this is 3 by 1. So, this will be here, so implies tau belongs to R3 by 1. And that should be fine, I think this is correct, this should be z transpose. I think this is fine. So, my update law will be p hat dot will be equal to the gamma matrix times the tau. Gamma is just symmetric positive definite matrix.

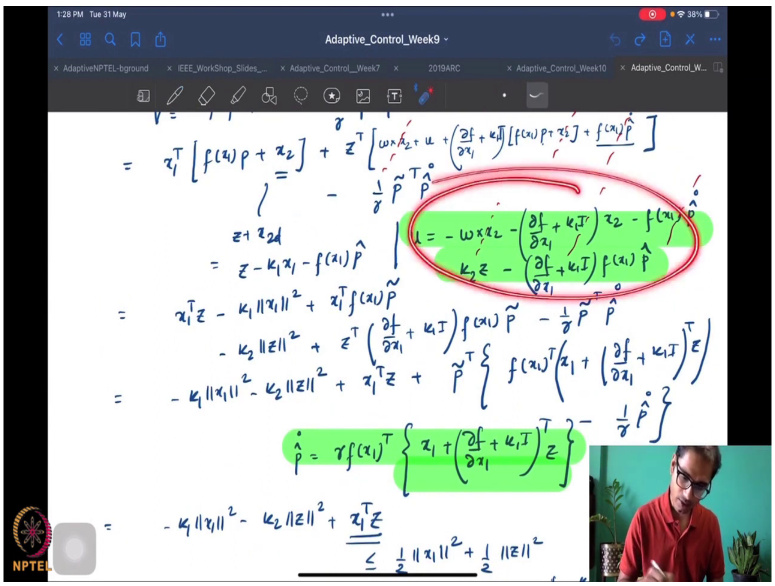
So, gamma can be anything in this case. So, any, so any gamma is okay, because if you remember, the gamma actually came from this expression. And here because del V a del p is 0, any gamma would be completely okay here. So, this is what would be your update law. And of course, this is not a p. I will have to replace the p by the hats, again. Correct. I will have to replace the p by the hats because of course, they are unknown.

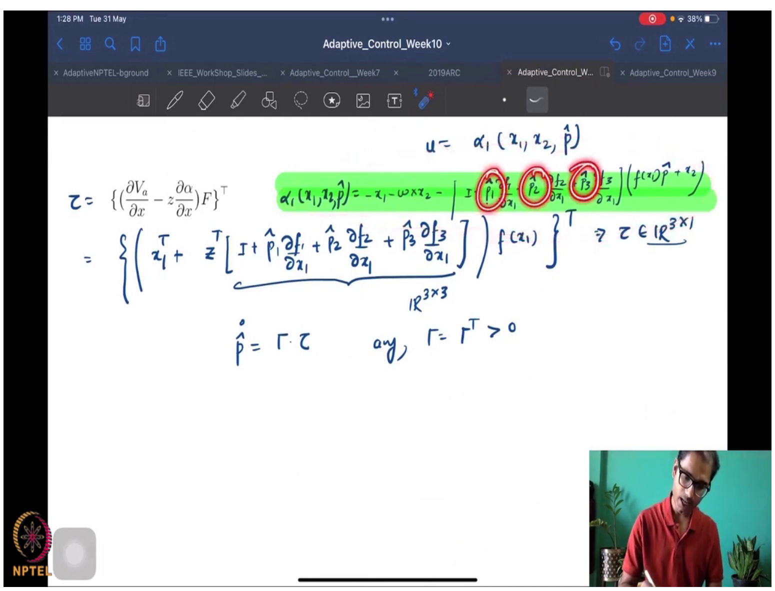
So, this is what you have. This is essentially what you have. You, you do not have this kind of an expression. And all the, all the taus get replaced by the hat variables, and, and that is of course coming from this. You can see that there is a hat quantity here, hat quantity here. So, also, the tau is essentially coming from this guy.

Absolutely, because the V a contains only the hat quantities here, and therefore the tau will also contain only the hat quantities because you replace the thetas by the theta hats here. So, I will actually mark it here. So, when you do this, x theta hat, therefore you are. Correct. This is absolutely right.

So, you can see that, well the interesting thing is that if you look at the control law here and you try to compare with, say, unmatched case control law and all that here, this is the controller that you get in the, I mean actually, let us try to compare it with the extended matching case.

(Refer Slide Time: 31:23)





So, this is the extended matching case where the control law is something like this. In fact, looks rather simple. And our controller looks rather complicated. In fact, it has quadratic terms in p hat, a rather complicated controller. And you also see the adaptation law which also looks relatively simpler maybe.

And our adaptation law looks like this, with some gamma tau, where tau is this guy with the hat terms. Again, looks complicated. I mean, slightly more complicated, of course. In this case, the adaptation law did not contain the p hats on the right. Here, we do have the p hats on the right also. But the cool thing is the control law does not contain any p hat dots whereas the control law does contain p hat dots.

And there is only one update law, there is only one update law, one parameter estimate per parameter. So, this is the idea. So, yes, the outcome looks rather complicated. In fact, I was also baffled because I saw some quadratics. But that is not a problem. We are obtaining a feedback law and an update law which is devoid of p hat dots and of course, devoid of over parameterization.

So, I hope that this example helped you to understand how to apply the tuning function method. Please do not get worried like we did get worried. And that is good. So, we understand how we can get worried because of, because we start seeing some quadratics. But the idea is that everything is nice, no problem, because of how the derivations go.

So, you start to see p hats appearing in the update laws also, and quadratics in p hat appearing in the feedback law. But this is not a problem. We can, we can, we have already proved asymptotic, adaptive asymptotic stability of, with the tuning functions. So, I really hope that you got an idea.

And also, we did not really get any serious issues because of the vector control as opposed to the scalar control that we consider in a theory. So, this is where we end our week 9. So, I really hope that you have learned a decent bit about adaptive control tuning functions and integrator backstepping and so on. We will continue with more interesting material in the next week. Thank you.