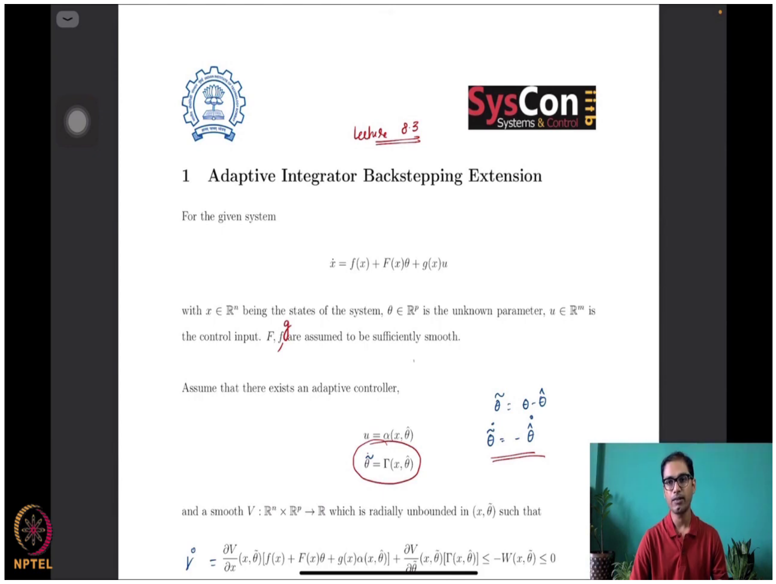
Nonlinear Adaptive Control
Professor Srikant Sukumar
Systems and Control
Indian Institute of Technology, Bombay
Week 9
Lecture No: 49
Adaptive Integrator Backstepping Method: An Example (Part 2)

(Refer Slide Time: 00:18)

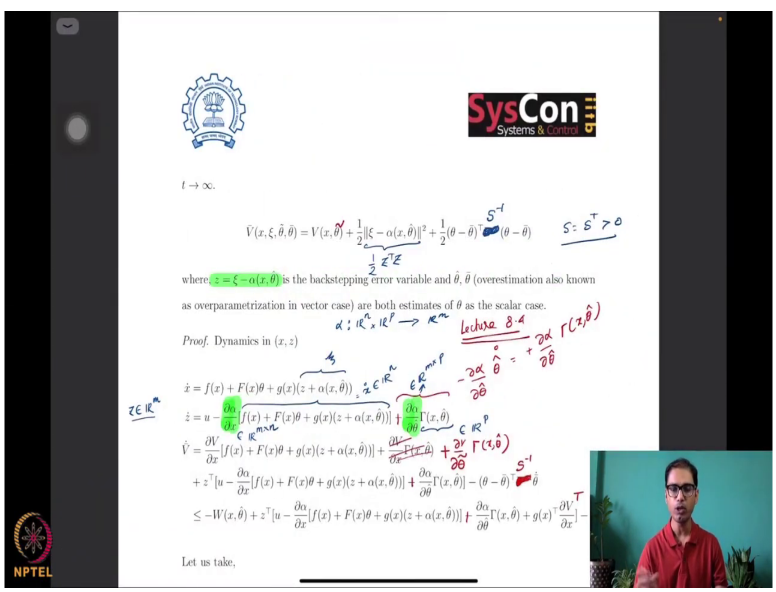
Hello, everyone. Welcome to another week of our NPTEL on Nonlinear and Adaptive Control. For all of those who have been following this lecture series, we are entering week number 9 in the course, and all this time, we have always been motivated by our desire to develop algorithms that will drive uncertain autonomous systems, very much like what we see in the background, which is a SpaceX satellite orbiting the earth. Now, I would like to welcome you again to week number 9 of this NPTEL. I am Srikant Sukumar from Systems and Control, IIT Bombay.

(Refer Slide Time: 01:11)



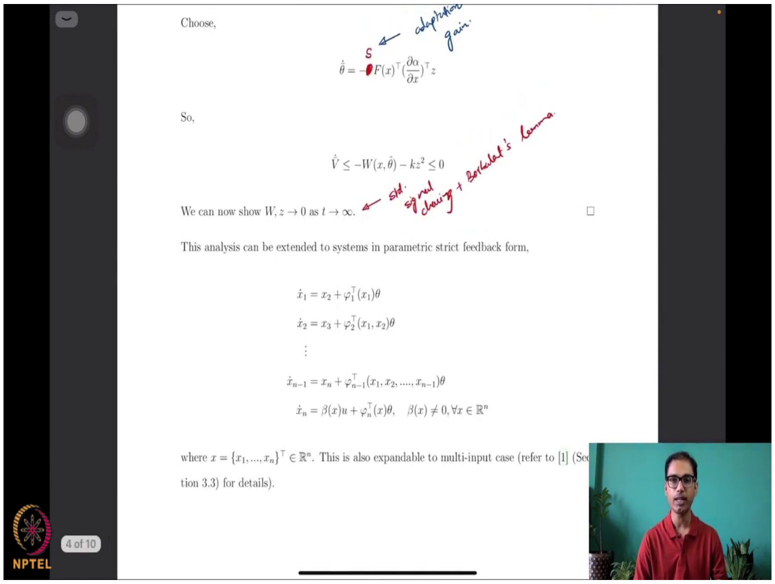
So, just to give you a quick recap as to what we were doing just in the preceding set of lectures, what we had looked at to begin with, starting lecture 8.3, of course, before that we did several other interesting topics like model reference adaptive control, but in, starting lecture 8.3 we move through talking about the extension of the adaptive integrator backstepping to the vector case.

(Refer Slide Time: 01:48)



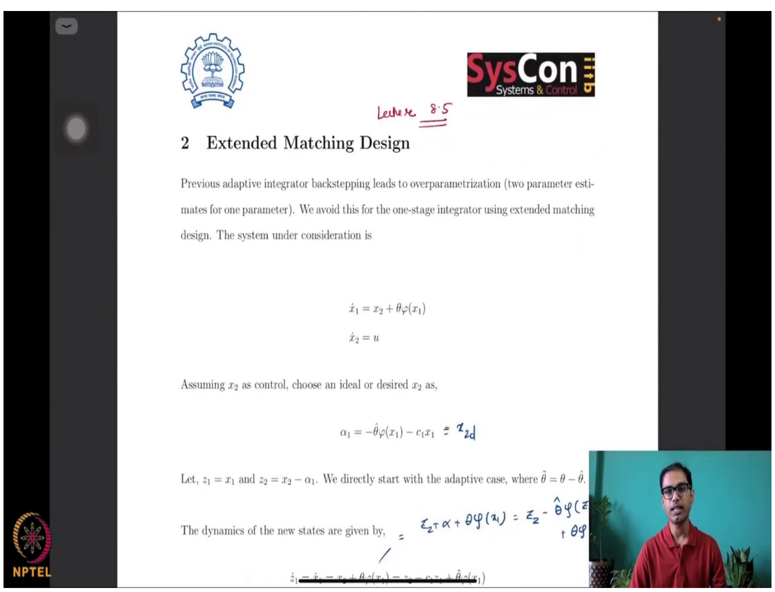
So, this is what we were looking at. And we saw how things do not change significantly in the vector case, except for a little bit of bookkeeping that is involved.

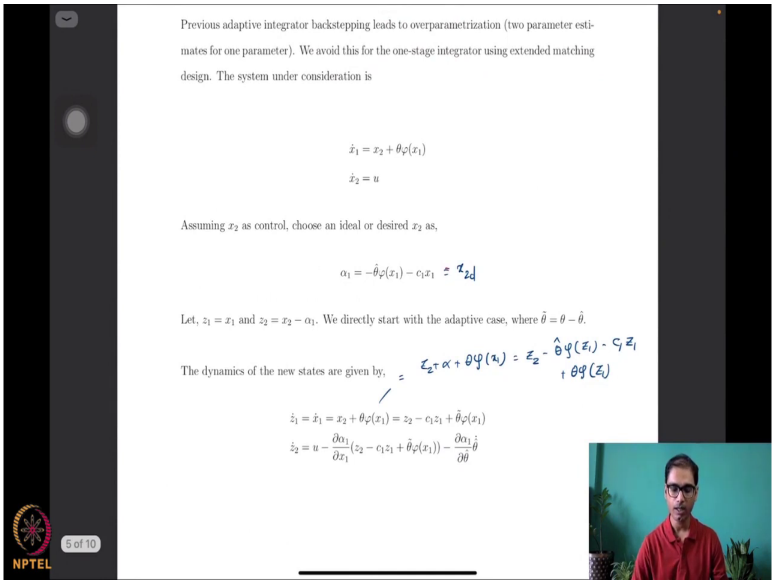
(Refer Slide Time: 01:59)



And we were able to design nice adaptation and feedback laws so as to stabilize this system. We were of course looking at the unmatched parameter case and we did realize that having two estimates per parameter was a rather significant constraint for real implementation.

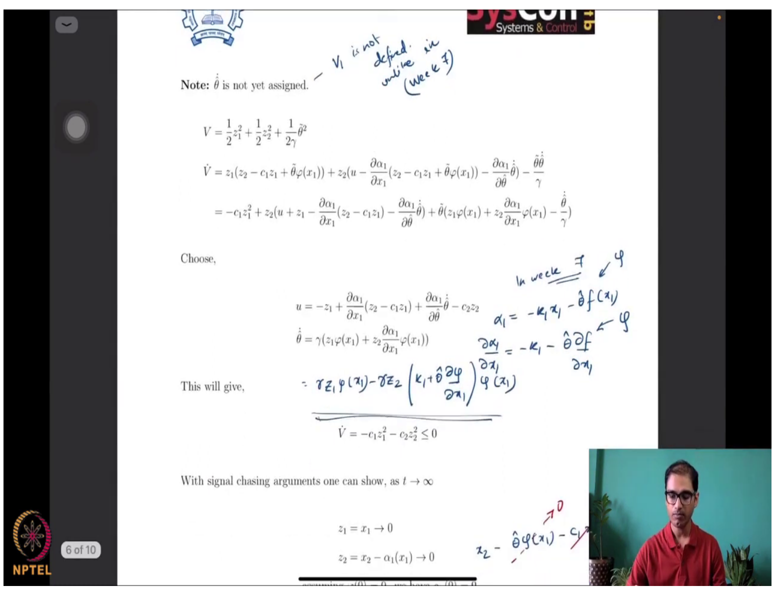
(Refer Slide Time: 02:20)

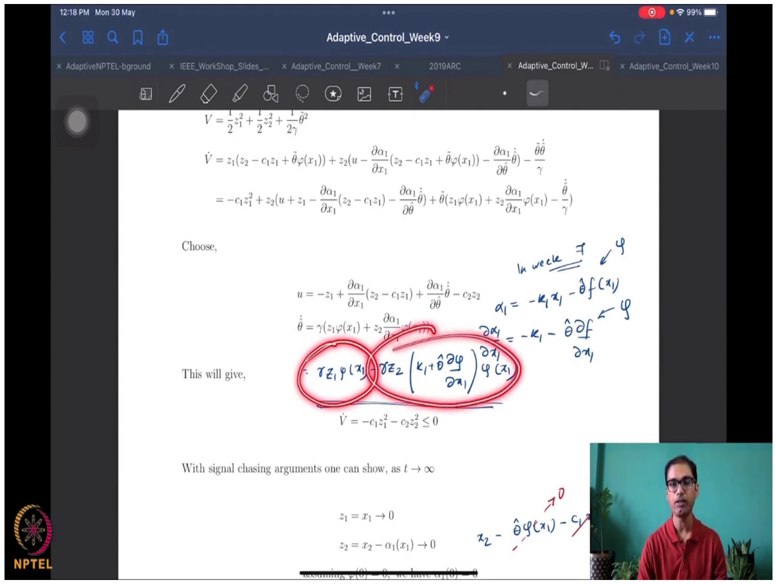


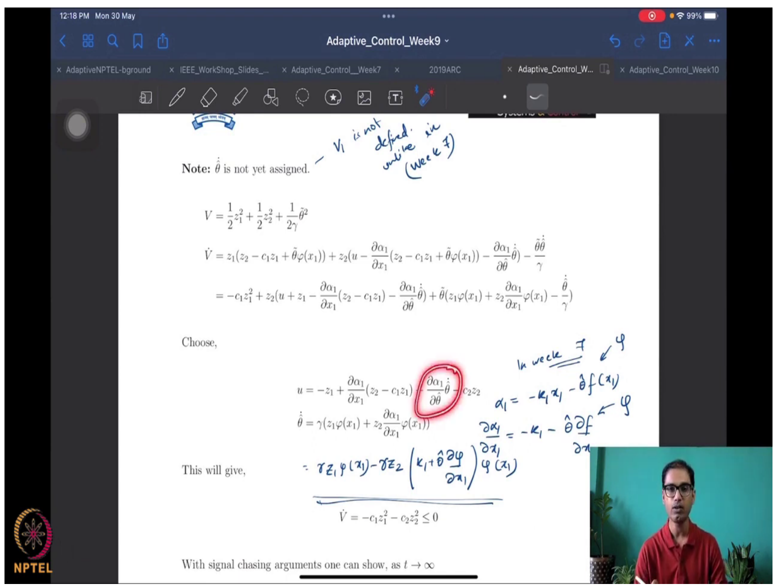


And so, we started to look at the extended matching design. So, what was different in the extended matching design, if you all remember, was that we do not define or declare the update law in the first stage itself.

(Refer Slide Time: 02:37)





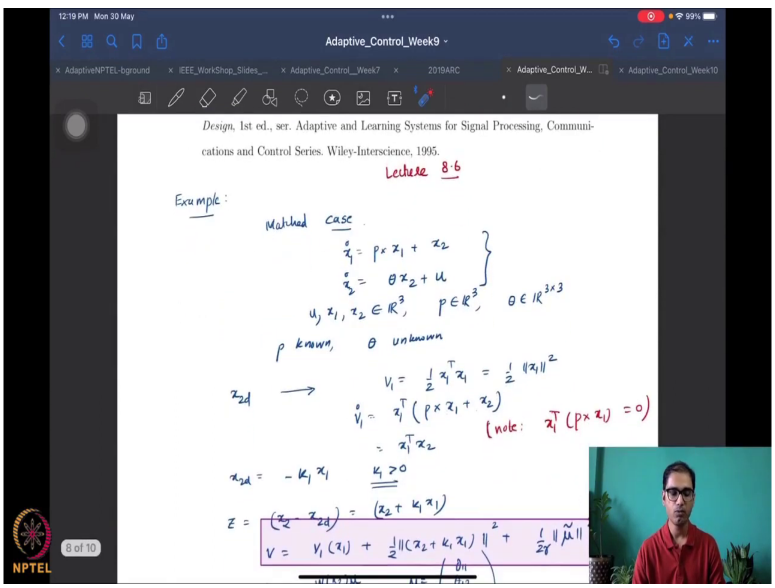


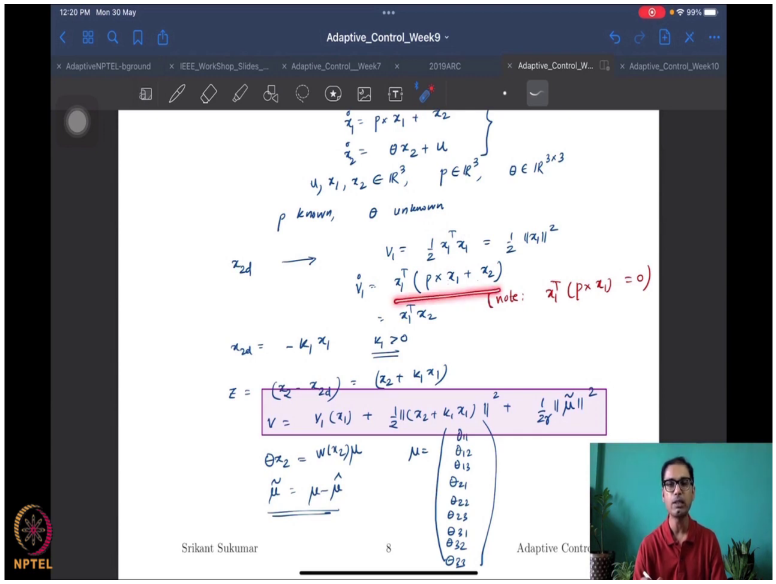
But we rather wait, and when we define the complete candidate Lyapunov function V, that is when we actually define the update law theta hat dot. And this way, we have only one parameter estimate for an unknown parameter. And so of course, we go back to the, what is expected, that if we just have one estimate per parameter. And what we realized also was that this estimate that you get is in fact the sum of our previous estimations. So, we had two estimates earlier. I apologize. We had two estimates earlier, and this turns out to be the sum of those two update laws. So, that is also interesting to see.

Now, one of the issues that we did point out was that although the estimates was nice, there is only one estimate per parameter which turns out to be the sum of the previous estimates, which is all nice, but what is hidden and not very evident is that the feedback law contains the theta hat dot.

So, there is, and we understood, that there is one derivative per level of unmatchedness. So, I am sorry for using a cooked-up word, but the control appears exactly one level below the unknown parameter, and therefore you have a theta hat dot. If the control appeared two levels below, there would be a theta hat double dot and so on and so forth. And having these subs, like successive derivatives of theta hat is not very healthy for any control implementation. So, this is one of the issues of the extended matching design. But in any case, it did help us alleviate the earlier issue of having two parameter estimates per parameter.

(Refer Slide Time: 04:47)



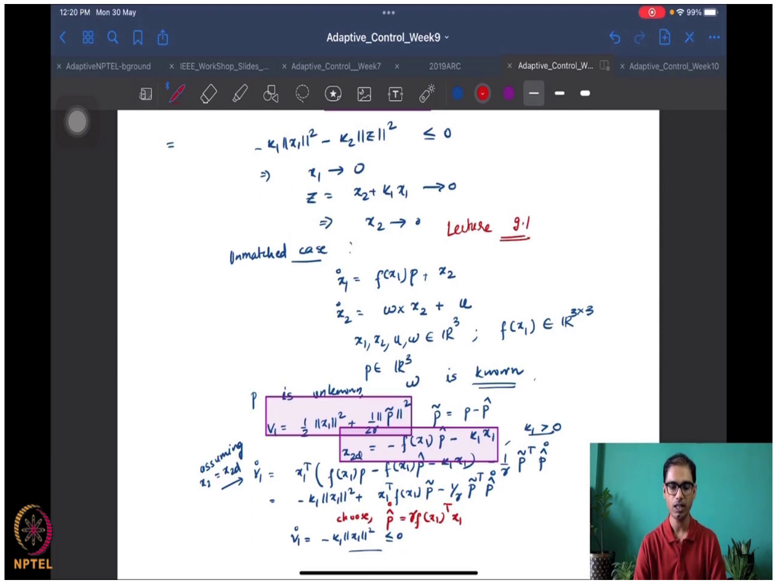


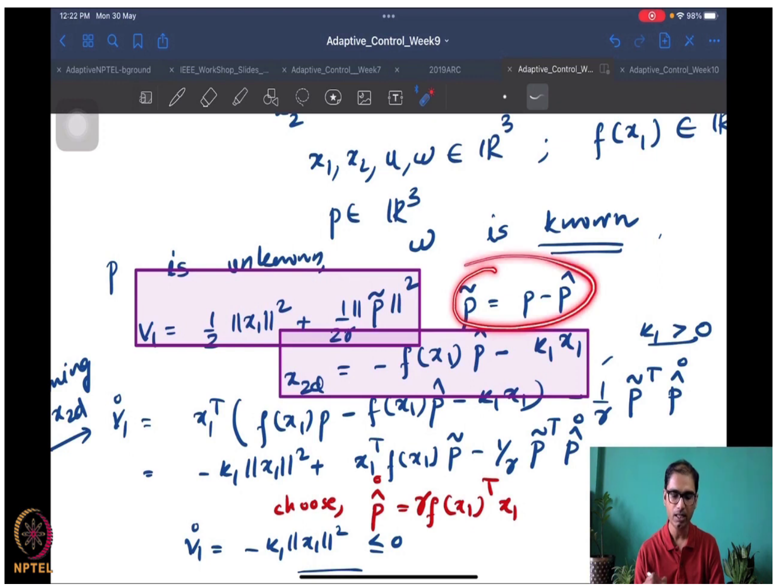
So, then what we sort of wanted to or started to look at in the subsequent lecture was an example because I was, I mean, all of us do want to know how these things do get applied. So, this was sort of a cooked up example. We first did the matched case meaning that the uncertainty was in the same dynamics as the control.

And, and this was the vector case because a lot of us may not have had a practice to deal with the vector case. And we saw that it is, things are not significantly different. I mean it is just that you use norms instead of using squares of scalars, you use norm squares. And that is pretty much the only difference.

You have to be careful about taking transposes and careful of the sequence of things. You cannot flip, move around things just like you would do in the scalar case. But other than that it was rather straightforward. So, the matched case is what we had completed. And then we had started looking at the unmatched case.

(Refer Slide Time: 05:40)





And that is where we will, of course, begin again. So, I am going to sort of start marking this again. I am going to mark it again here, simply because I want to discuss the unmatched case quickly again. I am going to mark the lecture here. And this is the first lecture of the 9th week. So, we are well into our course in Adaptive Control. So, great.

So, the unmatched case dynamics, again, these are cooked up dynamics but you will later on seeing your assignments and homeworks and exams, and I mean all sorts of things that similar dynamics appear in several places, several real systems. So, it is not also far-fetched after all. So, we have again two, the two systems, x 1 dot is f x 1 p plus x 2 and x 2 dot is omega cross x 2 plus u, where x 1, x 2 and, u and omega are all in R3, they are all vectors in R3 and f x 1 is a 3 by 3 matrix. p is also in R3, and in this case, because we are considering the unmatched case, p is assumed to be unknown while omega is assumed to be known.

So, it was sort of flipped off the matched case because in that, you had something unknown here and everything was known here. So, how do we go about this? We are first doing the standard adaptive integrator backstepping design where you have two estimates per parameter. So, how do we do that?

We start off assuming that p is unknown. We do not look at the known case and unknown case because I think all of you are now well exposed to the steps. So, it is not very difficult to now skip some of these steps. So, we start with the unknown p case, and we just look at this piece of dynamics in the adaptive integrator backstepping.

And the first thing we do is we define a candidate Lyapunov function. The best choice is take a norm square for the first state, and you of course take a p tilde square term 1 over 2 gamma p delta square term, because this is essentially, this is the parameter error here. This is the parameter error.

And now what, we declare what is our desired value of x 2 because we think of x 2 as the control so that is the x 2 desired. How we declare it is we just take f x 1 p hat because we cannot use a p, p being unknown, so we just use the estimate instead. So, it is minus f x 1 p hat, and we introduce a good term.

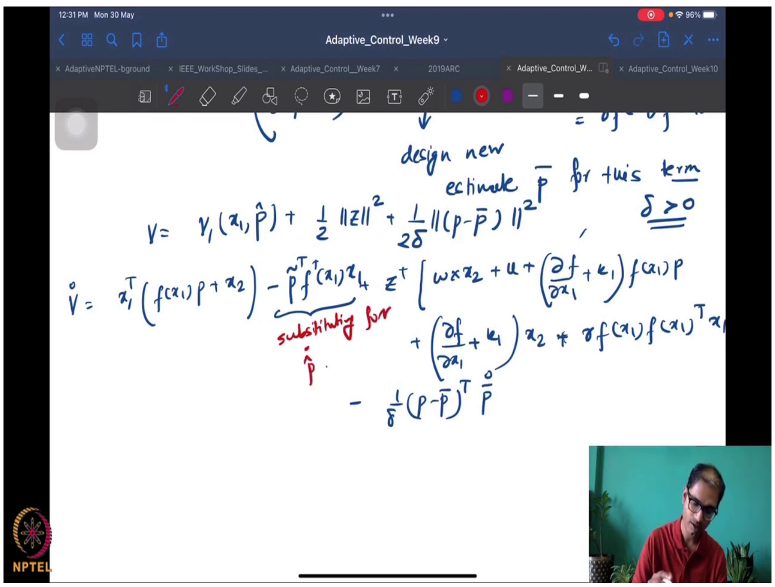
Now, we can, we sort of continue the Lyapunov analysis assuming that x 2 is in fact equal to x2 desired. So, that is how we do it, that is how we implemented the adaptive integrator backstepping. So, let us diligently take V 1 dot. V 1 dot is just x 1 transpose times x 1 dot. which is now f x 1 p from here, plus x 2 desired which is minus f x 1 p hat minus K 1 x 1.

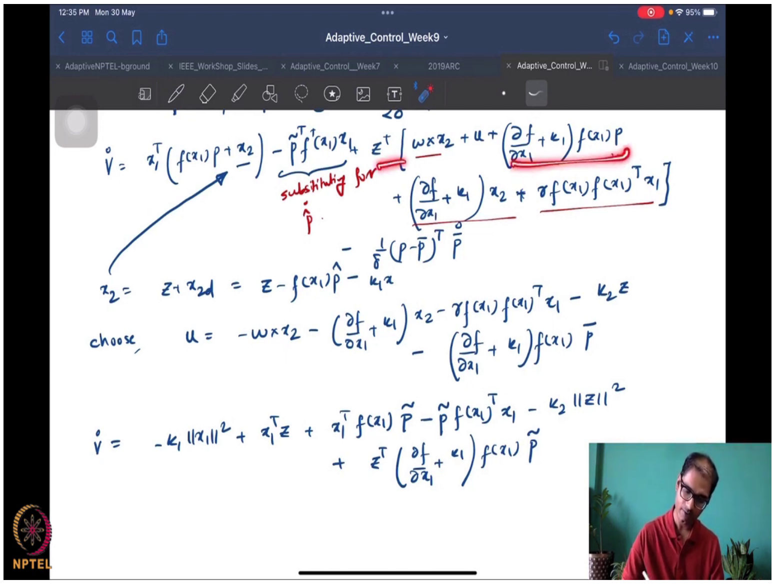
And then you have this, from the second piece, you just have minus 1 over gamma p tilde transposed p hat dot. So, let us see if I can make this smaller. So, this is the second piece is from minus 1 over gamma p tilde transpose p hat dot. This is from the fact that p tilde dot is minus V hat dot. So, that is just being applied here.

Now, you can see that I get a nice term here, minus K 1 norm x 1 square, and here I get f x 1 p tilde so I get x 1 transpose f x 1 p tilde and minus 1 over gamma p tilde transpose p hat dot. Now you can see that there is a p tilde here, there is a p tilde transpose here. I know that this entire thing is in fact a scalar quantity because V 1 is a scalar therefore each term is a scalar. So, I can take a transpose and nothing changes.

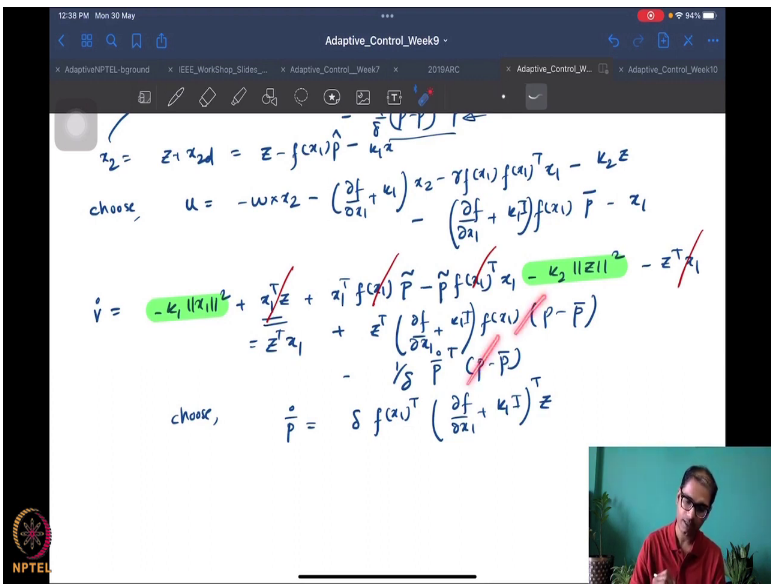
So, I get p tilde transpose, f transpose x 1 and I can take p tilde transpose common, and I will just implement p hat dot as gamma f transpose x 1. And once I do that, once I implement this parameter update law, p hat dot, I will get my V 1 dot as minus K 1 x 1 norm square which is negative semi definite. So, great. So, we have obtained the first parameter update law, and the first Lyapunov, candidate Lyapunov.

(Refer Slide Time: 11:06)





Adaptive_Control_Week9
$\dot{V} = x_1^T\left(f(x_1)p + x_2\right) - \underbrace{\tilde{p}^T f^T(x_1)x_1}_{\text{substituting for } \hat{p}} + z^T\left[\omega \times x_2 + u + \left(\frac{\partial f}{\partial x_1} + k_1\right)f(x_1)p + \left(\frac{\partial f}{\partial x_1} + k_1\right)x_2 + \gamma f(x_1)f(x_1)^T x_1\right] - \frac{1}{\delta}(p-\bar{p})^T\dot{\bar{p}}$
$x_2 = z + x_{2d} = z - f(x_1)\hat{p} - k_1 x$
choose, $u = -\omega \times x_2 - \left(\frac{\partial f}{\partial x_1} + k_1\right)x_2 - \gamma f(x_1)f(x_1)^T x_1 - k_2 z - \left(\frac{\partial f}{\partial x_1} + k_1\right)f(x_1)\bar{p}$
$\dot{V} = -k_1\|x_1\|^2 + x_1^T z + x_1^T f(x_1)\tilde{p} - \tilde{p}f(x_1)^T x_1 - k_2\|z\|^2 + z^T\left(\frac{\partial f}{\partial x_1} + k_1\right)f(x_1)\tilde{p}$



Adaptive_Control_Week9
$x_2 = z + x_{2d} = z - f(x_1)\hat{p} - k_1 x$
choose, $u = -\omega \times x_2 - \left(\frac{\partial f}{\partial x_1} + k_1\right)x_2 - \gamma f(x_1)f(x_1)^T x_1 - k_2 z - \left(\frac{\partial f}{\partial x_1} + k_1 I\right)f(x_1)\bar{p} - x_1$
$\dot{V} = -k_1\|x_1\|^2 + \underbrace{\cancel{x_1^T z}}_{= z^T x_1} + \cancel{x_1^T f(x_1)\tilde{p} - \tilde{p}f(x_1)^T x_1} - k_2\|z\|^2 - \cancel{z^T x_1} + z^T\left(\frac{\partial f}{\partial x_1} + k_1 I\right)f(x_1)(p-\bar{p}) - \frac{1}{\delta}\dot{\bar{p}}^T(p-\bar{p})$
choose, $\dot{\bar{p}} = \delta f(x_1)^T\left(\frac{\partial f}{\partial x_1} + k_1 I\right)^T z$

$$\dot V = -k_1\|x_1\| + \ldots$$

$$= z^T x_1 + z^T\left(\frac{\partial f}{\partial x_1} + k_1 I\right) f(x_1)(p-\bar p) - \frac{1}{\delta}\dot{\bar p}^T (p - \bar p)$$

choose,

$$\dot{\bar p} = \delta f(x_1)^T \left(\frac{\partial f}{\partial x_1} + k_1 I\right)^T z$$

$$\dot V = -k_1\|x_1\|^2 - k_2\|z\|^2 \le 0$$

$$x_1, z \to 0 \quad \text{as} \quad t \to \infty$$

$$z = x_2 + f(x_1)\hat p + k_1 x_1$$

$z \to 0$, $x_2 \to 0$, $f(x_1)\hat p \to 0$, $k_1 x_1 \to 0$

as $t \to \infty$.

$t \to \infty$.

$$\bar V(x, \xi, \tilde\theta, \bar\theta) = V(x, \tilde\theta) + \underbrace{\frac{1}{2}\|\xi - \alpha(x, \hat\theta)\|^2}_{\frac{1}{2}z^T z} + \frac{1}{2}(\theta - \bar\theta)^T S^{-1}(\theta - \bar\theta) \qquad S = S^T > 0$$

where, $z = \xi - \alpha(x, \hat\theta)$ is the backstepping error variable and $\hat\theta$, $\bar\theta$ (overestimation also known as overparametrization in vector case) are both estimates of θ as the scalar case.

$\alpha : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$

Lecture 8.4

$$-\frac{\partial\alpha}{\partial\hat\theta}\dot{\hat\theta} = +\frac{\partial\alpha}{\partial\hat\theta}\Gamma(x, \hat\theta)$$

Proof. Dynamics in (x, z)

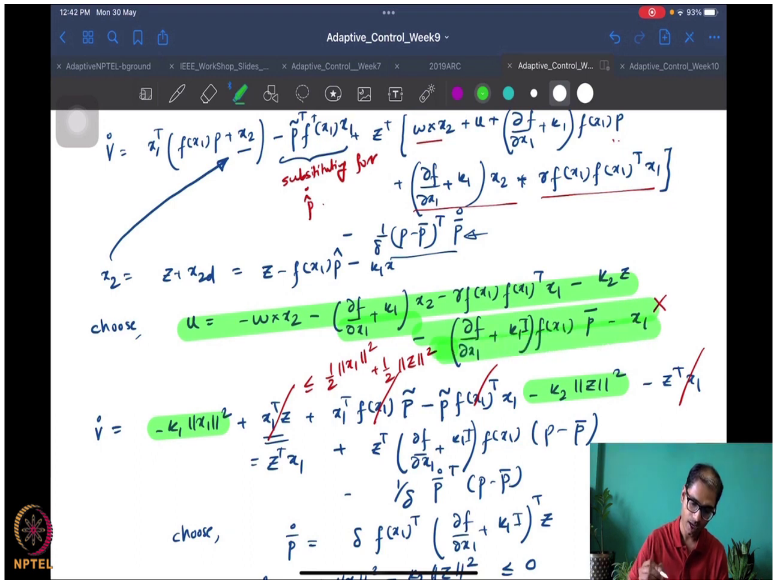
$$\dot x = f(x) + F(x)\theta + g(x)(\overbrace{z + \alpha(x, \hat\theta)}^{\xi}) = \dot x \in \mathbb{R}^n$$

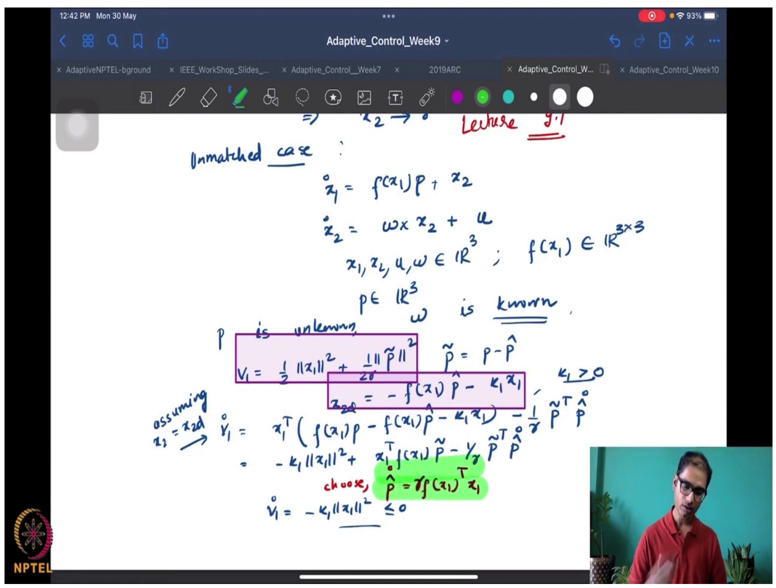
$$\in \mathbb{R}^m \quad \dot z = u - \underbrace{\frac{\partial\alpha}{\partial x}}_{\in \mathbb{R}^{m\times n}}[f(x) + F(x)\theta + g(x)(z + \alpha(x, \hat\theta))] + \underbrace{\frac{\partial\alpha}{\partial\hat\theta}}_{\in \mathbb{R}^{m\times p}}\underbrace{\Gamma(x, \hat\theta)}_{\in \mathbb{R}^p}$$

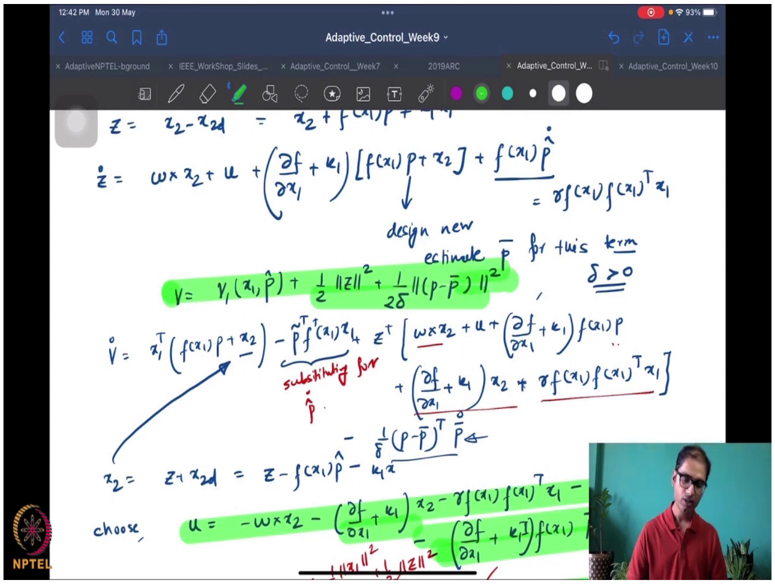
$$\dot{\bar V} = \frac{\partial V}{\partial x}[f(x) + F(x)\theta + g(x)(z + \alpha(x, \hat\theta))] + \cancel{\frac{\partial V}{\partial x}\Gamma(x, \hat\theta)} + \frac{\partial V}{\partial\tilde\theta}\Gamma(x, \hat\theta)$$

$$+ z^\top[u - \frac{\partial\alpha}{\partial x}[f(x) + F(x)\theta + g(x)(z + \alpha(x, \hat\theta))] + \frac{\partial\alpha}{\partial\hat\theta}\Gamma(x, \hat\theta)] - (\theta - \bar\theta)^\top S^{-1}\dot{\bar\theta}$$

$$\le -W(x, \hat\theta) + z^\top[u - \frac{\partial\alpha}{\partial x}[f(x) + F(x)\theta + g(x)(z + \alpha(x, \hat\theta))] + \frac{\partial\alpha}{\partial\hat\theta}\Gamma(x, \hat\theta) + g(x)$$


Adaptive_Control_Week9
substituting for
choose,
choose,


Adaptive_Control_Week9
Lecture 9.1
Unmatched case :
p ∈ IR³
ω is known
p is unknown
k₁ > 0
assuming
choose,



Now, what is the next step? Now, we know that x 2 is not actually equal to x 2 desired so the next step is about creating a backstepping error variable, that is this guy, and that is just x 2 minus x 2 desired, which is exactly this. So, we know that x 2 desired also brings in the parameter again. So, we are going to compute x z dot, the dynamics of the backstepping error variable first.

So, what is z dot? It is x 2 dot which is these two terms. And then I have an x 2 desired dot. So, what is this x 2 desired dot? x 2 desired dot is just this guy. x 2 desired dot is just this guy, which is del f del x 1 times x 1 dot and K 1 x 1 dot. So, I have combined the two del f del x 1 x 1 dot plus K 1 x 1 dot, I have combined into these terms because x 1 dot is just these guys.

And then I have a f x 1 p hat dot, that is this. And I already know what is p hat dot because I have already designed it. So, I substitute for it, and this term comes out to be gamma f x 1 f x 1 transpose x 1. Great. So, now I know what has happened. I have already defined the p hat dot but the unknown quantity p appears again, the unknown quantity p appears again.

And this is what is going to create a problem when I design the controller. I cannot use a p hat again. I need to create a new estimate and, we call it p bar. And what is the idea of the adaptive integrator backstepping, is that you take the earlier candidate function, add to it a norm squared term in the backstepping error and a norm squared term in the new parameter error, the new parameter error, that is p minus p bar square.

So, you can see that this is exactly, I even showed it to you last time, this is exactly the new function V. Let us see it is, not here. So, this is the extended design but in the original design, this is the new one so earlier V then a norm squared term in the backstepping error then a norm squared term in the new parameter error. Here, there is a weighing matrix.

Instead, I have just used a, instead of S, a weighing matrix, I have just used a scalar. I have just used a scalar delta. So, that is okay. You can choose a matrix or keep a scalar, your call. Having a matrix, of course, gives you more handle on the adaptation game. So, that is more general. So, once I have this V, I am going to diligently take derivatives again.

And then using that, I will try to give an update law p hat dot, that is the idea. Because everything else is more or less chosen, we will of course choose some things here. Let us go forward and see what we need. So, V dot is first V 1 dot. So, V 1 dot is, let us see, it is x 1 transpose x 1 dot, which is f x 1 p plus x 2 plus, we had a 1 over gamma, in fact, there will be a negative sign, minus 1 over gamma tilde transpose p hat dot, that is your V 1 dot.

And this is just copied from here. The answer is just copied from here. I mean of course here you have plugged in x 2 desired but I am not plugging in x 2 desired because x 2 is not actually equal to x 2 desired. I just keep the x 2 as it is. Everything else is just this. So, I have this. And then I have a z transpose z dot, which I can plug from here, omega cross x 2 plus u plus, I will write this as del f del x 1 plus k 1 f x 1 p plus del f del x 1 plus k 1 x 2 plus gamma f x 1 f x 1 transpose x 1. This is just z transpose z dot. And then I have minus 1 over delta p minus p bar transpose p bar dot. So, here again for p hat dot, I can substitute this quantity, gamma f transpose x 1. So, that is what I do, I will erase this. And this becomes p tilde transpose f transpose x 1.

So, here of course, I have substituted, substituting for p hat dot. After substituting for p hat dot, this is what you will get. So, now, I use the usual tricks, what I will do is, I will write x 2 as z plus x 2 desired in terms of the backstepping error variable, and that is z minus f x 1 p cap minus K 1 x 1. So, this goes in here. So, what will happen? I will get V dot as equal to minus K 1 norm x 1 square. This is from this term and this term together.

Then I will get plus x 1 transpose z, that is from this term and this term. And then I will finally get plus x 1 transpose f x 1 p tilde. And that is from this term and this term. And here, I had minus p tilde f x 1 transpose x 1. That is just rewriting this term. And now, let us see. What I am also going to do before I, I am going to actually move this downwards, because what I am going to do is I am also going to choose my control.

How do I choose my control, I just choose my control to cancel whatever I can and I will do my best and introduce a good term. So, I will choose my control as minus omega cross x 2 minus del f del x 1 plus K 1 x 2 minus gamma f x 1 f x 1 transpose x 1. So, that takes care of this term, this term and this term. And I am left with this guy, which I cannot actually cancel completely but I can always introduce the estimate.

First, I introduce a good term minus K 2 z, which gives me the good term and then I will introduce minus del f del x 1 plus K 1 f x 1 and p bar. So, because I cannot introduce the p hat or the p, which is not available, I introduce the new estimate which is p bar, which is p bar. So, basically, I am going to cancel this, introduce a good term, and get something like this here.

So, once I do this, once I have this, I will get my V dot in a more simplified form. I get a minus K 2 norm z squared, that is because of this guy combining with this guy, and I will get a plus z transpose del f del x 1 plus k 1 f x1 p tilde. And that is from this guy combining with this guy. So, sorry, so this is not p tilde because p tilde is p minus p hat. This is actually the p minus p bar.

So, what has happened now? I have a, I have a couple of nice terms. This is nice and this is nice. So, I get a couple of nice terms. And then what? I also have some cancellations because this term and this term is the same. By the way I missed writing one term which is this last term. So, this is of course there. Minus 1 over delta. I am going to write it as p bar dot transpose p minus p bar.

I have simply taken this and taken a transpose. This is this is just transpose. I can do that because these are all scalars, I can keep taking transposes. This is so that the p minus p bar appears on the right hand side in both of them because they are vectors, so I cannot reorder things the way I want. Anyway, so I have these two cancellations, these two are good terms. I think I, I missed something. I am sorry. I missed something. I missed something. I also need to use my control to cancel this guy. So, I can do that because this is actually equal to z transpose x 1. And so, if I just introduce a minus x 1, so then z transpose minus x 1 is minus z transpose x, and that gets cancelled here.

So, I will also have a minus z transpose x 1 from the control, that will of course get cancelled too. So, what I am left with is, I mean of course I now make a choice of my p bar dot as delta times f x 1 transpose del f del x 1 plus K 1. Let us see, I should have an appropriate dimension, del f del x 1 plus K 1. I am going to say I because this has to be the appropriate dimension, transpose times z.

Suppose I make this choice, I know that these two will cancel out also. So, I will be left with V dot as minus K 1 norm x 1 square minus K 2 norm z square, which I know is less than equal to 0. So, by the way, I did not even need to cancel this out. I did not need to cancel this out with this additional term in the control because you see that this is a mixed term in these two terms.

So, just by choosing a large enough gain K 1 and K 2, I could have dominated this term also. So, that was the other choice. I did not necessarily have to cancel this with this choice of control here. I could have removed this. And if you remember, I could have done a sum of squares to write this guy. So, I am going to, maybe I will do that in red, remove this term and then write this as less than equal to half norm x 1 squared plus half norm z square.

And, and these get, of course, combined with this guy and this guy respectively, and just by choosing a K 1 and K 2 large enough, I can dominate this mixed term. So, this term is not essential. So, this term is not essential in the control law. So, if you want, you can ignore adding this term. Just by choosing large enough gains K 1 and K 2, you will be fine.

So, once you have this V dot as negative K 1 norm x 1, x 1 squared and K 2 norm z squared, you are done. You have a negative semi-definite V dot and you sort of have, you, you can actually prove that, I mean just like in standard backstepping you can prove that x 1 and z go to 0 as t goes to infinity. And I know that z is nothing but x 2 minus x 2 desired. So, it is, z is x 2 plus f x 1 p hat plus K 1 x 1 plus K 1 x 1.

So, now, you know that this guy is going to 0 here. This guy is going to 0. If f 0 is 0, then this piece is also going to 0. So, the only way, only thing that remains possible is that x 2 also goes to 0 as t goes to infinity. This is the only possible choice. So, so that is it. So, essentially, you have done your design. You designed a, just like you want, you designed two controllers.

So, you, so you have designed one control. Control is just one. You have designed two parameter estimates. You designed two parameter estimates. One is the p bar dot and the other one is the mu hat dot, sorry, other one is the p hat dot. And you also have the combined Lyapunov function choice, which is this guy.

So, this is what happens in adaptive integrator backstepping. You have one combined Lyapunov function, one control law and two parameter estimates corresponding to each parameter. So, this is what you would expect. And this is what we had promised. So, great, so this is the adaptive integrator backstepping, the classical one.

What we want to do next and we will do it in the subsequent lecture is basically, the extended matching design, which sort of helps us to remove this additional parameter estimate. So, that is really the idea. So, what did we look at today? We continued our problem on adaptive integrator backstepping for this vector concocted system.

We, I hope all of us did get a fair idea of how to work with vector systems. And I really hope all of you get the message that it is not significantly different from the scalar case. And we, of course, designed this adaptive integrator backstepping based controller. So, we have two parameter estimates and one feedback loop.

But now of course we want to move towards removing this additional parameter estimate for the same system. So, we will in fact go ahead in the subsequent session and start with an extended matching design for the same system. So, I really hope that this sort of example is giving you a good exposure into how to do this design.

And I really hope that all of you will be at this stage able to pick up problems from your own fields, own, your systems, autonomous cars, suspension systems, satellite systems, electrical system biological systems and start working on some adaptive control designs for peace. So, this is where we stop now. And we will continue. Thanks.