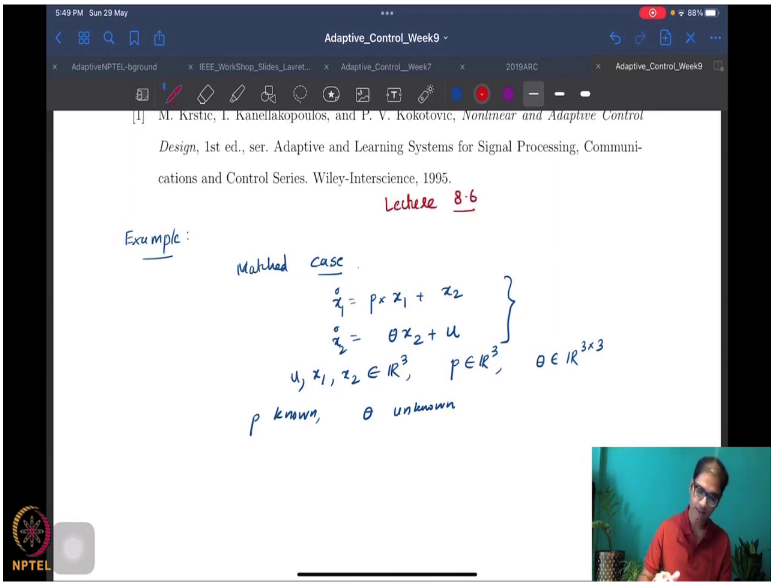
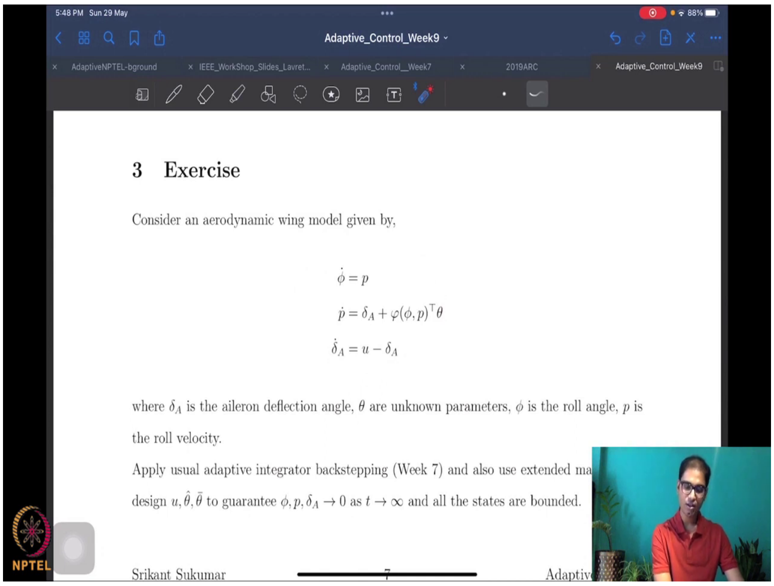
Nonlinear Adaptive Control
Professor Srikant Sukumar
Systems and Control
Indian Institute of Technology, Bombay
Week 8
Lecture No: 48
Adaptive Integrator Backstepping Method: An Example (Part 1)

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, in this week number 8, we have been looking at Adaptive Integrated Backstepping in more details in the last few lectures.

And so, we essentially saw the unmatched design and how to generalize it to the vector case. And we also saw how to do an extended matching design which allows us to prevent, over parameterization. We are well into the design and development of algorithms that help drive systems such as the SpaceX satellites orbiting the Earth that we see in the background.

(Refer Slide Time: 01:19)





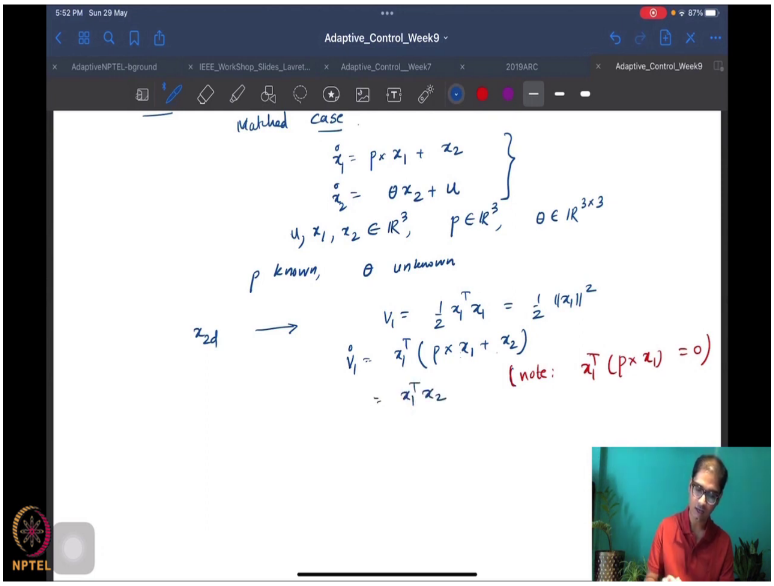
Now, we have already looked at several methods now. And what I wanted to do was today in this session was to look at an example, that is a vector example for both the matched and the unmatched case. And look at both the conventional design and also the extended matching design. So, let us see how we can do this?

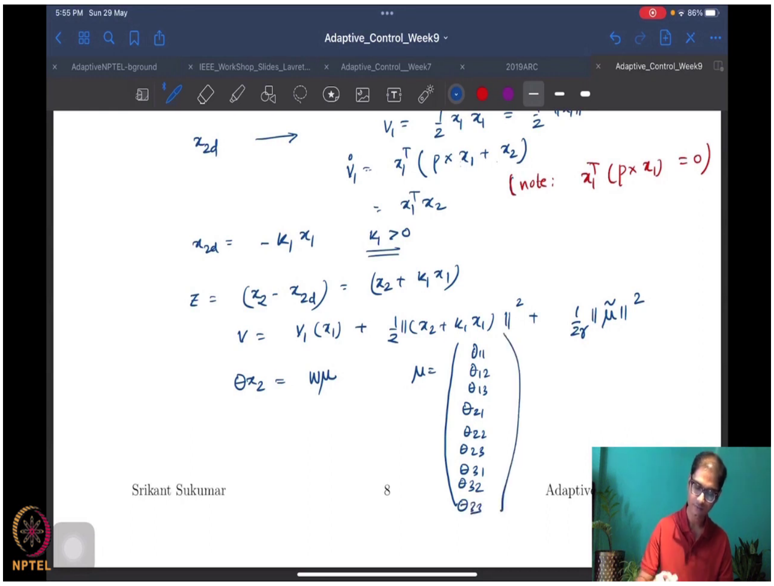
So, this is I mean, there was already an example that I asked all of you to a problem that I asked all of you to sort of try to solve, which is not a, which is also has vector parameters. So, of course, I wanted all of you to try this. But what I want to do is to look at a complete vector example with the states and control everything being vectors. So, and so that is what this is. So, this is sorry, just allow me a moment.

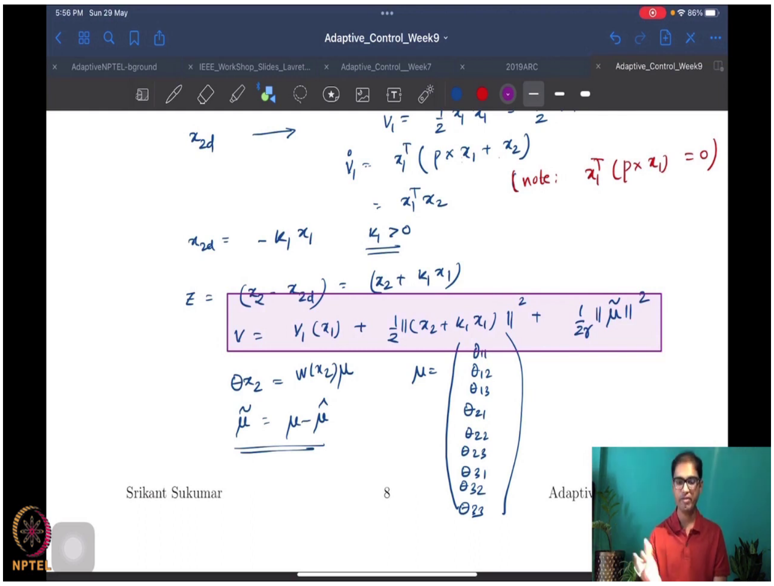
So, this is where we start. This is lecture 8.6. So, I think this was 8.5. So, we are at lecture 8.6. And this is the sort of system this is a concocted sort of system. So, I would not worry so much about what is the real relevance, but the aim here is to design a stabilizing adaptive controller with sort of parameters p and theta, we start with the matched case, where we assume that theta is the unknown, and theta in this case is a 3 by 3 vector x1 and x2 and the control are vectors in R3, p is also in R3. So therefore, you have a cross product here.

And theta is actually sorry, I apologize. Theta is actually a 3 by 3 matrix. I am sorry, so theta is actually a 3 by 3 matrix. So, let me just give it a thought first, that if this is the example that I want to consider. Let us sort of look at this example. Let us sort of look at this example and see how things go. So, let me try to do the match case, in the matched case, p is assumed to be known, theta is assumed to be unknown. So, let us see how the design goes.

(Refer Slide Time: 04:09)







So, I will start with the first stage and I want to prescribe x2d, but before I prescribe x2d, I want to prescribe a V1 as half x1 transpose x1 or the norm. This is the standard choice whenever vector states are involved. And we can also try a weighted version of this, but for us, this is okay. And I take V1 dot and this I used to actually obtain my x2 desired right because I assumed my x2 to be the control.

So, V1 dot comes out to be x1 transpose x1 dot, which is p cross x1 plus x2. Now, the interesting thing is that, so I am going to say note, x1 transpose p cross x1 is actually 0. Why? Because p cross x1 is orthogonal to x1 and dot product of orthogonal quantities is 0. Or you can also think of this as a scalar triple product. So, this also can be thought of as a dot b cross c and I can rearrange it to obtain again x1 cross x1, which will give me 0.

So, this quantity is 0, so this product does nothing. So, I am left with x1 transpose x2. So, a good choice of x2 desired still my usual nice choice, which is some minus k1 x1 where k1 is some like for now, I just take it as a positive scalar. Just a scalar quantity. Great. Now, what do I do? I define my backstepping variable as z as being equal to what my backstepping error, because x2 is my true state, and I wanted to follow x2 desired.

So, this just becomes x2 plus k1 x1. And now, what do I do? I define my V as V 1, which is I am going to say this V1 for now, V1 x1 plus half the norm of the backstepping error square x2 plus k1 x1 norm square I apologies. And then of course, because it is the unknown parameter case, I will always deal with the unknown parameter.

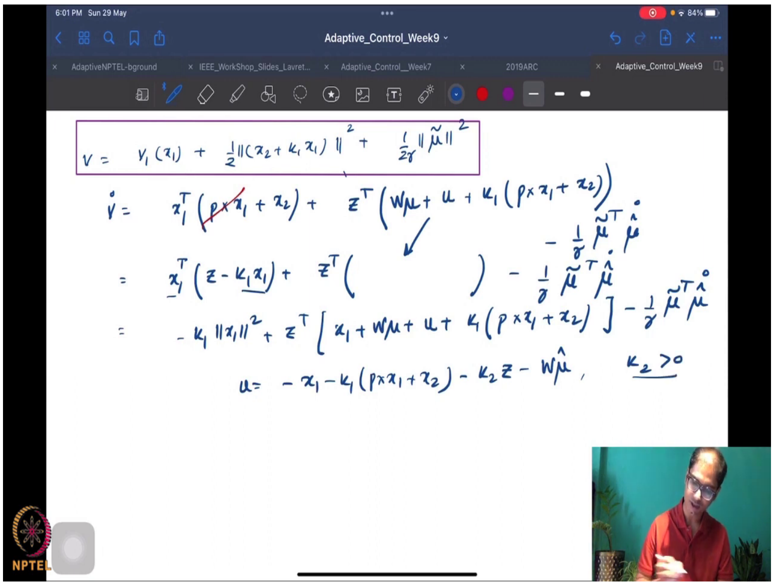
So, I have to add some term corresponding to that. I am going to add some term corresponding to that, and I am going to call it if you may, I am going to call it as half 1 over 2 gamma mu tilde norm squared. And I will explain what is mu tilde soon… Why do I need mu tilde? Let us look at this. So, here, the parameter is actually a matrix, and I do not we do not like to deal with matrix parameters.

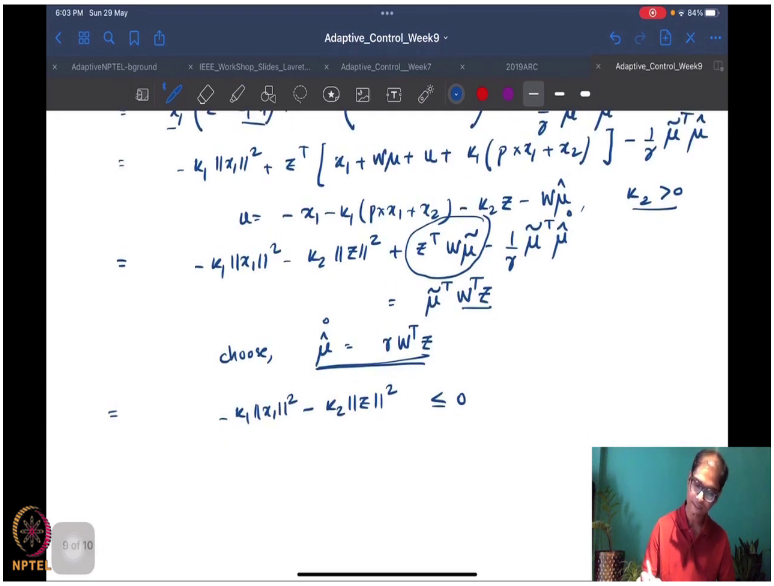
So, what do we say? We say that theta x2 is equal to something like a w times mu, you know that this left-hand side is linear in the elements of the matrices of the matrix theta. Therefore, I rearrange it. So, what is mu? mu is actually equal to theta 1 1, theta 1 2 theta 1 3. So, this are basically just the columns stacked. Theta 2 1, theta 2 2, theta 2 3, theta 3 1, theta 32 theta 3 3. So, that is what is your mu.

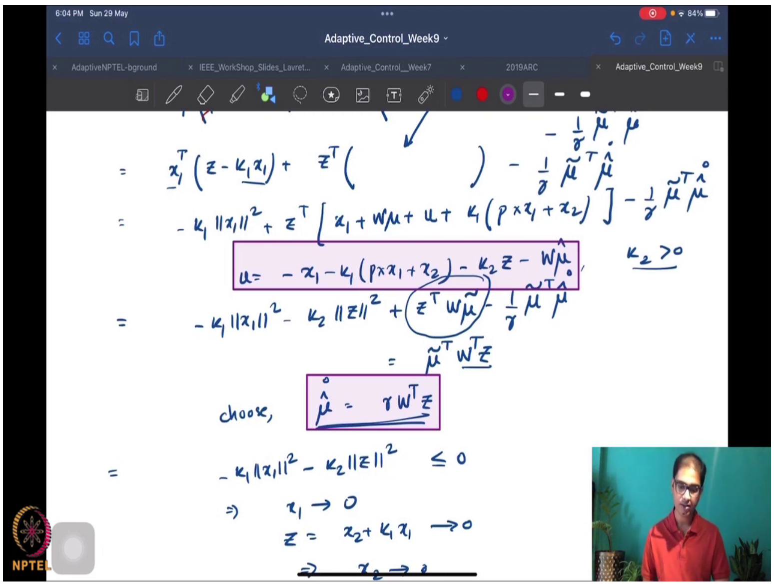
So, this is very simple, I have just, in fact, I can be careful and say that this is actually just w of x2 times my new parameter mu, which is nothing but the thetas, which was a matrix stacked in vector, column wise, and then mu tilde is just mu minus its estimated mu cap. So, this is just because I have a matrix parameters theta and I do not like to deal with matrix parameters.

Yes, I did it the, you know, model reference adaptive control case, but I do not want to do it. So, I am going to deal with the vector parameter mu. So, this is what is now my, you know, candidate Lyapunov function for the entire system. This is what is my candidate, Lyapunov function for the entire system. Great. So now, what do I do? I take derivatives and try to find the controller.

(Refer Slide Time: 10:15)


Adaptive_Control_Week9
$V = V_1(x_1) + \frac{1}{2}\|(x_2 + k_1 x_1)\|^2 + \frac{1}{2\gamma}\|\tilde{\mu}\|^2$
$\dot{V} = x_1^T(\rho^* x_1 + x_2) + z^T\left(W\mu + u + k_1(\rho^* x_1 + x_2)\right) - \frac{1}{\gamma}\tilde{\mu}^T\dot{\hat{\mu}}$
$= x_1^T(z - k_1 x_1) + z^T(\quad) - \frac{1}{\gamma}\tilde{\mu}^T\dot{\hat{\mu}}$
$= -k_1\|x_1\|^2 + z^T\left[x_1 + W\mu + u + k_1(\rho^* x_1 + x_2)\right] - \frac{1}{\gamma}\tilde{\mu}^T\dot{\hat{\mu}}$
$u = -x_1 - k_1(\rho^* x_1 + x_2) - k_2 z - W\hat{\mu}, \quad k_2 > 0$
NPTEL


Adaptive_Control_Week9
$= -k_1\|x_1\|^2 + z^T\left[x_1 + W\mu + u + k_1(\rho^* x_1 + x_2)\right] - \frac{1}{\gamma}\tilde{\mu}^T\dot{\hat{\mu}}$
$u = -x_1 - k_1(\rho^* x_1 + x_2) - k_2 z - W\hat{\mu}, \quad k_2 > 0$
$= -k_1\|x_1\|^2 - k_2\|z\|^2 + z^T W\tilde{\mu} - \frac{1}{\gamma}\tilde{\mu}^T\dot{\hat{\mu}}$
$= \tilde{\mu}^T W^T z$
choose, $\dot{\hat{\mu}} = \gamma W^T z$
$= -k_1\|x_1\|^2 - k_2\|z\|^2 \leq 0$
9 of 10
NPTEL


Adaptive_Control_Week9
$= x_1^T(z - k_1 x_1) + z^T(\quad) - \frac{1}{\gamma}\tilde{\mu}^T\dot{\hat{\mu}}$
$= -k_1\|x_1\|^2 + z^T\left[x_1 + W\mu + u + k_1(\rho^* x_1 + x_2)\right] - \frac{1}{\gamma}\tilde{\mu}^T\dot{\hat{\mu}}$
$\boxed{u = -x_1 - k_1(\rho^* x_1 + x_2) - k_2 z - W\hat{\mu}}, \quad k_2 > 0$
$= -k_1\|x_1\|^2 - k_2\|z\|^2 + z^T W\tilde{\mu} - \frac{1}{\gamma}\tilde{\mu}^T\dot{\hat{\mu}}$
$= \tilde{\mu}^T W^T z$
choose, $\boxed{\dot{\hat{\mu}} = \gamma W^T z}$
$= -k_1\|x_1\|^2 - k_2\|z\|^2 \leq 0$
$\Rightarrow x_1 \to 0$
$z = x_2 + k_1 x_1 \to 0$
$\Rightarrow x_2 \to 0$
NPTEL

So, let us do that. Let us do that. So, I am going to, let us say let us see if I can copy this whole thing. I believe I can copy and to move to the next page and simply paste it. I do not know what I copied somehow popping the whole thing, maybe I can still paste and delete it. That is fine. This is okay. So, what I will do is I will just erase this.

Great. So, now if I take my v dot here, what do I get? I will get V dot as V1 dot, which is 1 half x1 transpose x1. So, I will get x1 transpose x1 dot, which is again, p cross x1 plus x2 plus this guy, 1 half x2 plus k1 x1 or I am going to just write this as z, z transpose z dot, which is x2 dot, which is W times mu plus u is x2 dot plus k1 x1 dot, which is k1, p cross x1 plus x2 plus just 1 over gamma mu tilde transpose mu, mu hat dot with the negative sign.

Because mu tilde dot is just mu hat, great. So again, I know that this term amounts to nothing. So, I am going to write this as x1 transpose times x2 plus half z transpose this whole mess. Well, actually, you know what, I am going to make my life simpler. And I am going to write x1 transpose x2 in terms of z.... So, this is actually z minus k1 x1 plus half z transpose, this whole thing gets here and minus 1 over gamma mu tilde transpose mu cap dot.

Now, you can see that I have a minus k1 norm x1 squared from here, and sorry, I do not have a half here and forth, that is a mistake and plus z transpose times i get a x1 plus W mu plus C u plus k1 times p cross x1 plus x2 minus 1 over Gamma mu tilde transpose mu hat dot. Again, just copying terms and moving this z x1 to this term, I get this. Now, I am in a good position to design my control, this is an unknown, so I will just replace it with estimate.

So, my control will be just minus x1 minus k1 p cross x1 plus x2 minus a good term which is k2 z minus W mu hat and here k2 is of course some positive quantity. So, what I have done? what have I done? I have cancelled this have cancelled this introduce the nice negative term and try to cancel it with my estimate. So, with this, my V will be minus k1 norm x1 squared minus k2 norm z squared because of this guy.

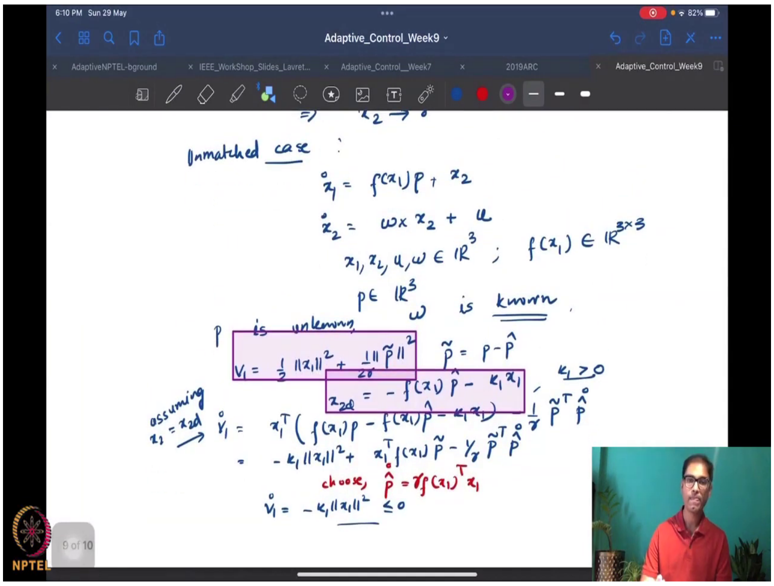
And I we will be left with plus z transpose W mu tilde because I have a W mu here minus W mu hat from here I have a W mu tilde and a minus 1 over gamma mu tilde transpose mu hat dot. Now, if you look at these 2 terms together I can take transpose here. So, this is equal to mu tilde transpose W transpose z because these are all scalar quantities I can take as many transposes as I wish, no problem.

So, using that I choose mu hat dot to be so that I cancel this guy. So, this will be gamma W transpose z, so if I do that, I end up with V as minus k1 norm x1 square minus k2 norm z squared, which is less than or equal to 0. And from here I can prove that x1 goes to 0, z equal to x2 plus a1 x1 goes to 0. And this implies that x2 goes to 0. So, this is what we want, we want, well, I mean, we did the stabilization problem.

But again, the tracking problem is exactly the same. No problem. So, we this is what we want, this is what we get, we get a nice update law here, we get this nice update law here. And similarly, we get a control law here. So, we get an update on the controller just like we want and this is a standard adaptive controller. And you can see that dealing with the vector case did not significantly complicate things or anything, it was quite okay, it was quite okay.

So anyway, so let us, so this is how you do the design for the matched case that is where the unknown is theta. And p is in fact known. And p is in fact known. Great. Now, let us look at an example for an unmatched case. So, I will try to construct an example.

(Refer Slide Time: 18:14)



So, let us look at the unmatched case and see what happens. So, what I want to do is to have a similar system. So, here I will have not p cross x1 because that will not be interesting enough. So, I will do something like x1 dot is definitely something like f x1 times a p plus x2 and x2 dot is omega cross x2 plus a control. Where again, x1 x2 u omega in R3 f of x1 belongs to R3 by 3. That is a 3 by 3 matrix and p of course belongs to R3, p is unknown.

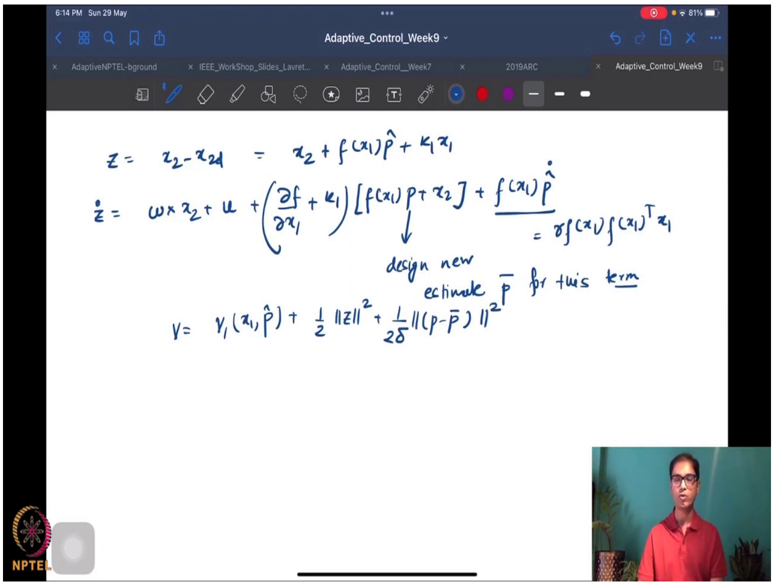
Omega is known. So, this is the system. So, now this becomes an unmatched case. Because I have unknowns here. So, let us try to do this with the standard integrator backstepping type design. So, what would I do? I am directly dealing with the unknown case, of course, so, we will see what we will do and so, I will take my… So, I will first take my first set of dynamics, I will want to do V1 as 1 half norm x1 squared plus 1 over 2 gamma p tilde squared, where p tilde is p minus p cap and x2 is assumed to be the control.

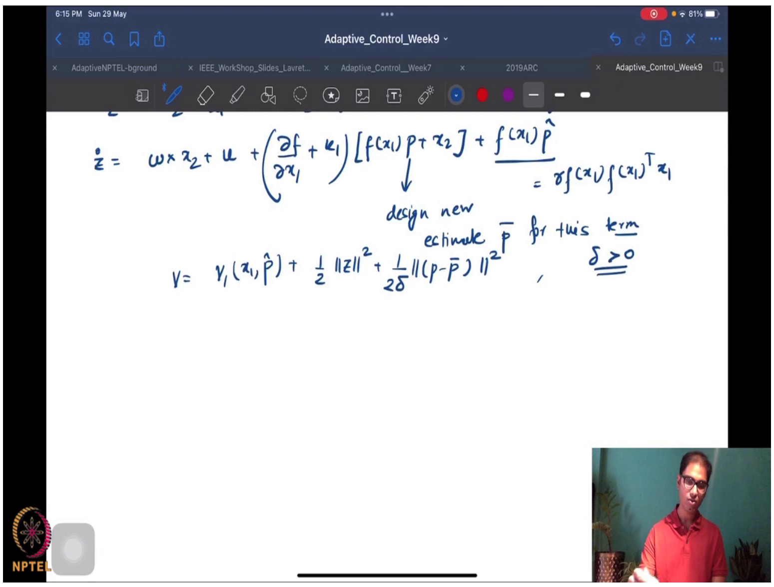
So, x2 desired is of course minus f x1 p cap minus k1 x1 for some positive k1. So, with this what would I get? I will get V1 dot as x1 transpose x1 dot I mean this is again assume this V1 dot is assuming x2 is x2 desired you will get x1 transpose x1 dot which is f x1 p minus f x1 p cap minus k1 x1 plus 1 over gamma I apologize this should have been a norm also because p is in R3 and so, this will be minus 1 over gamma p tilde transpose p cap dot, and this is just minus k1 norm x1 squared plus x1 transpose f x1 p tilde minus 1 by gamma p tilde dot p cap dot I am simply reproducing this step here by just combining terms carefully.

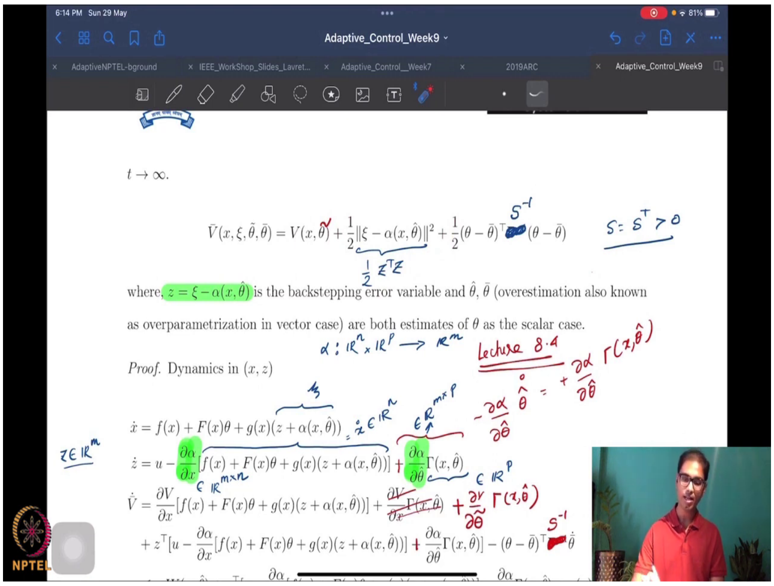
So, now, if I choose p cap dot as f x1 transpose x1 times gamma then I have V1 dot is minus k1 norm x1 squared which is negative semi definite and I am good this is essentially what I want it in our standard adaptive integrator backstepping design so, I have a first estimate first Lyapunov candidate, I have a first estimate I have first Lyapunov candidate.

I have a first estimate I have first Lyapunov candidate and we are good. And we also have an x2 desired of course, we also have our x2 desired right here we also have an x2 desired right here, great. So now, what do we want to do? We want to, of course create our backstepping variable. So, we want to create a backstepping variable. And how do we do that? It is pretty straightforward.

(Refer Slide Time: 24:17)







We design our z which is equal to x2 minus x2 desired. I am going to explicitly write this up and what is x2 desired? x2 desired is this guy minus f x1 p cap minus k1 x1. So, this is plus f x1 p cap plus k1 x1. This is what is my z.

And so, so what happens now, if I actually compute my z dot, which is what I will do that compute my z dot, I get x2 dot, which is omega cross x2 plus u plus this which is partial of f with respect to x1 plus k1 times x1 dot and x1 dot is what x1 dot is f x1 times p plus x2 and then I get f x1 p cap dot p cap dot is already specified, so p cap dot is this gamma f x1.

So, this is actually equal to f x1 times gamma f x1 transpose x1, so this is what is this term. And now, what do we do? We sort of try to of course, we want to try to specify the control, we want to try to specify the control. But now we are little bit of a soup why? This is what happens in this adaptive integrated term, there is another unknown appearing here.

And because we have already come up with a p hat dot we cannot replace use p hat again and the control, so what do we do? We use a new estimate p bar. So, we designed the control as minus but anyway we of course, I will just not design the control right now. Design new estimate p bar for this term. Because we already have a p hat.

And we already have an update law for p hat. So, we cannot use this earlier estimate p hat, so, we need to create a new estimate. And that is what we call p bar. Now, once we know this, we define our complete Lyapunov function as V1 which is a function of your x1 and p hat and add to it are backstepping error terms which is standard and then a term in the new parameter error and then a term in the new parameter error that is p minus p bar squared.

So, this is what we have been doing, I mean, if you look at this, how we did not the extended design but this guy. So, if you look at this, what was it, it was the original V, then the term in the square term in the backstepping error, then a quadratic in the new parameter error. That is exactly what I did I just instead of having a matrix, instead of having a matrix S inverse, have taken just a delta, just as some delta positive.

And with this, my plan is to design the p bar dot. So, that is what we do, great. So, what I will do is I will continue with this in the subsequent lecture. So, we look at it very carefully. So, what did we do today was we have already seen all the methods for design of parameter update laws for matched and unmatched control via adaptive integrator backstepping. And of course, the extended matching design.

What I wanted to do was to take up an example. So, that is what we did for the match design, we took our vector example and we showed how to design a control and then estimation law or update law for the parameter. And similarly, we are now doing the same for a vector example for the unmatched case and we are first trying to do the design using the adaptive integrator backstepping.

And subsequently we will use the extended matchings. So, this is so that we get some exposure to vector system adaptive control design. So, that is the idea. Excellent, so this is where we will stop and we will continue with this example in the next session. Thank You.