Transcriber's Name: Crescendo Transcription Pvt. Ltd.
Nonlinear Adaptive Control
Professor Srikant Sukumar
Systems and Control
Indian Institute of Technology, Bombay
Week-6
Lecture 32
Adaptive Control Design: First Order Scalar Systems

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 into the sixth week of our course. So, we are sort of at the almost at the halfway mark. And we have already covered quite a few different tools for analysing nonlinear systems, and also adaptive systems, and also parameter identification laws, and their convergence. So, starting this week, we are going to look at how to actually design adaptive controllers. So, this is sort of the plan. And of course, as always, we have this very nice motivational background of this SpaceX satellite orbiting the Earth. And the hope is that the algorithms that we design and develop will help drive systems such as these autonomous.

So, what we were, we had sort of started looking at last time was an introduction into the first adaptive control problem for this course. So, thank you, for your patience, of course. You would have probably hoped to see an adaptive controller, which is the topic of the course, so, which is the name of the course, sooner maybe, but, well, I mean, we of course had to cover quite a bit of preliminary material in order for us to be able to understand and design adaptive controllers. So, great, so we had just started with our scalar first order system. So, I am in fact going to mark this piece as lecture 6.1, just for our reference.

So, we had looked at sort of an introduction of what we want to typically do in adaptive control, we looked at this first order scalar system, where we have a constant unknown parameter appearing linearly in this system. So, x is a, x and u and f are all scalar value, and this theta star constant, which is unknown appears linearly in this system. And we have the control objective of the state x of t, tracking a smooth bounded trajectory. And we also stated the standard assumptions that are prevalent in all adaptive control designs, at least most adaptive control designs, and definitely the one that we are going to focus on in this course.

So, this is where we are. And this is where we will start among lecture 6.2, this is the second lecture of this week. So, the first thing that we do is design an error system. So, why do we do that? Why do we design an error system? We are almost always interested in driving things to 0. If you remember, when we were sort of talking about stability of nonlinear systems in the sense of Lyapunov, we eventually started assuming at all points that, the equilibrium is the 0, or the origin is the equilibrium. And that is why we, whenever we are given a tracking problem, for example, where a signal x has to track a signal r. We simply construct an error signal, which is x minus r. So that wherever this e goes to 0, we know that x goes to r.

So, this is essentially the aim of constructing a error signal. And once we have an error state, if you may, we write the error dynamics, just by taking a derivative of course, and substitute for x dot, from our original system dynamics, we get something of this form, that is in equation 2.8. So, here the first two pieces are essentially the same as what you have in the system dynamics. And you just

have a minus r dot corresponding to the reference trajectory that we are trying to follow. So, it should be sort of obvious to you that if you want, if your reference is a constant, that is you want to move to a constant value or to 0 value, then r dot is 0, and therefore, this piece is also sim.

So, whenever you are trying to track a constant reference, then it is typically this is called the if r of t is constant. It is called the regulation problem. The tracking problem is when r is a function of time, the regulation problem is when r is a constant. And if r is 0, if r equal to 0, it is called the stabilization problem. So, three kinds of problems. One is if r is a function of time, just like we have written here, then it is a tracking problem. If r is a constant value, then r dot is 0, and it is called the regulation problem. And if r of t is in fact 0, that is, you just want to go to the origin, just want the states to go to the origin, then it is called a stabilization problem. So, they are just nomenclature for you. So, anyway, so this is what are the error dynamics. So, this is the error dynamics, this is the system that we will work with.

So, the first step, in any adaptive control design is to design a controller, assuming that the parameter is known, so, this is called the known parameter control design. So, this is the first step, again, sort of this is a very standard sort of thing that mathematicians and applied mathematicians do, we have already talked about this, you try to solve the simpler problem first, so that you get some nice ideas to solve the more complicated problem. So, that is really what we are trying to do. The simpler problem is when this parameter theta star is just assumed to be known. And the question is, can you construct a controller u, so that e is asymptotically stabilized.

So, how would you design this feedback? So, one of the simplest ways to do this is to imagine or to sort of specify a target system. So, then the standard question that arises is what is a good target system? So, it should be obvious that any target system should be such that the, in that at least for the target system the error is converging to 0 as t goes to infinity, and also is asymptotically stable. So, you it is your target system should be nice, should do what you wanted to do. In our case, you want the variable e to be asymptotically stabilized to 0.

And so, therefore, you should have a target system which is asymptotically stabilizing to 0. And second, you should ensure that the target system has some similarity to the original system. If this is for example, if this is a first order system, just like it is, it would be sort of ridiculous to consider a second order target system, because I will never be able to compare this similar, so things like that. So, the target system should sort of match the original system, and it should have the nice properties of asymptotic stability that you desire from the original error system.

So, in this case, what do we choose? We choose our target system to be e dot is minus ke, why? We know the solution for this, it is a scalar system. So, I know that this is in fact exponentially stable. So, is a good it, it has good behaviour of e of t. The second thing is it should be close it should match. What do I mean by matching? I mean that by choosing some control here, I should be able to match this right hand side, with this right hand side. And, yes, it is in fact possible by choosing a control law of this kind. So, this is an important thing, remember.

So, I am going to repeat it, the target system is chosen to be e dot is minus ke, the first sort of principle of choosing a target system is that it should behave well, which in this case means e is asymptotically stabilized for the target system, which it does, in fact exponentially stable. The second is it should sort of match the original system, what is this match? It means that by choosing an appropriate controller, I should be able to match this right hand side with the target system right hand side. This is these two are important, and in this case indeed I can do that just by choosing u to be of this form.

So, the second term here cancels the term corresponding to the trajectory, the last term cancels the

dynamics. Notice that we have assumed theta star is known, otherwise this control is not implementable. So, if theta star was unknown, I cannot implement this control. So, this is not an adaptive controller, this is just a controller in the case of the known parameter. So, this just cancels the nonlinear dynamics, this cancels the trajectory component, and the first term helps with the matching. The first term is essentially what helps with the matching.

So, now that we have this, we know that this is exponentially stable, and we also can very very easily construct a Lyapunov function which is just the basic one-half e squared, that is the simplest possible choice. Which is in fact radially unbounded and all the nice jazz, which you look for in a Lyapunov candidate. And further, if you take V dot, so this is the closed loop system now, if you take V dot along this closed loop system, I get minus ke square. And therefore, V dot is also negative definite. So in fact, you get global asymptotic stability for e. In fact, you get global exponential stability, why do I get global exponential stability? Because V is just half e squared. So, it is bounded on both sides by this class k infinity function, which is half the e squared itself, in take half e squared itself.

And further, V dot is also, V dot, the minus V dot is also upper bounded by a class k infinity function of the same magnitude, they can use just ke squared here. Which is the same order of magnitude class k infinity function. Therefore, this is in fact globally exponentially stable equilibrium. So, e equal to 0 is globally exponentially stable. So, this is in fact, more than what we were looking for, but well in this case, we are able to do that. So, what have we been able to do? We have been able to design this controller for this known case, we have been able to design this controller, with of course, k has to be a positive constant, such that your error exponentially decays to 0.

And is asymptotically and in fact, exponentially stable equilibrium and 0 is an exponentially stable equilibrium of the error system. So, essentially everything that we want. Except, so, I mean so, for the known case we are I mean, it is a scalar system of course, and when it was not very hard work to be admit, to construct a stabilizing control as you would have imagined. So, with the known case, we are able to do all the good things. Excellent, and as we should, we are very happy about it, but not a big deal, still a scalar system, great.

Now, the next step now, is to design the controller for the unknown parameter case that is when theta star is unknown, what do we do? How do we construct an adaptive controller, and this follows a very very simple logic. And this logic is stated in the form of the certainty equivalence principle. So, the certainty equivalence principle says that, in order to design the controller for the unknown case, we retain the same controller structure as in the known case, and replace the actual values of the parameters by the estimates. So, the certainty equivalence principle. So, certainty means that when the parameter values are known, equivalence mean that the structure of the controller remains exactly the same.

The only thing you change is replace a true value like this that is the theta star, by its estimate, which is denoted by theta hat. So, this is very important. So, we always use the theta hat to denote estimate of theta star. So, the hat operator denotes the estimated value, and the tilde is the theta minus theta hat, or theta star minus theta hat is the parameter error. This is very standard notation that we will continue to use throughout this. So, please get used to this notation theta hat, or the hat notation always indicates an estimate. The tilde notation always indicates the parameter error, or the estimation error.

So, what do we do? We pick up the same control law as 210, and we replace this theta star by its estimate theta hat, because everything else remains the same. Notice that we are yet to prescribe how theta hat is calculated. So, this will come subsequently, but for now, all we do is we replace the

true value theta star by its estimate theta hat, and this is what the certainty equivalence principle dictates. So, now, the important thing to note is that because theta hat may not be equal to theta star, as would be natural because you do not know the value of theta star. So, you cannot possibly have an estimate which exactly matches the theta star.

So, when you write the error dynamics there is a small change, the first term in remains the same, but then there is an additional term which arises because of the parameter error, because this theta hat is not equal to theta star, an error term shows up here. And this is the theta tilde that we are talking about. So, one of the things that should have already come to your notice is that I have very carefully put in a time argument on the estimate. Although my original parameter was a constant, my estimate of the parameter is an evolving object, it is always evolving or changing over time.

Therefore, it is a function of time, it is not a constant. Because there is literally no logic to having a constant estimate for another constant, because if your estimate is wrong, you never get to sort of improve it, because you chose a constant estimate. So, if you chose an estimate which was off by 10 units, then it remains off by 10 units all the time, you can never improve the performance, you cannot expect to cancel the effect of the unknown parameter, if you did not choose a time vary estimate. So, there is no point in having a constant estimate for another constant. So, although our true value is assumed to be a constant through the entirety of this course, the estimate will always be a time dependent constant. Great.

So, now, we have a different closed loop system, which is this guy. Which contains the parameter error now. So, what do we do? We now do what is called Lyapunov redesign. And what is this? Which basically, we use a Lyapunov candidates in order to come up with an update law. This is a very standard method. So, what is the idea? Idea is use the Lyapunov candidate to derive theta hat dot update law. So, the expression for theta hat dot, that is the expression for evolution of the update is called the update law. Just the natural name. So, what is the idea? I first guess a Lyapunov function, I do not guess a theta hat dot, I do not start by guessing a theta hat dot, I start by guessing a Lyapunov candidate.

And what is the simplest Lyapunov candidate? I pick the original candidate, which was half e squared. For the known case if you know, if you remember, it was half e squared. So, I choose the original piece. But then I know that there is now another state, why is there another state? I introduced the parameter error, which appears in the dynamics as a state. And therefore, I add a term corresponding to that state, which is something like 1 over 2 gamma theta tilde square. Where gamma is just some positive constant. Excellent. So very, very simple choices. Standard, simple quadratic choice. Nothing too complicated. I chose whatever was already there, which is one half e square. And I added to it a quadratic term in the new state, which is the parameter error.

And that is with some positive gamma. So now for this system with theta tilde, and e as our states, this is in fact, positive definite. So, in fact, V is radially unbounded, not just positive definite, just quadratic sum of two quadratics in the two states. So, what do we do? Well, I mean, of course, gamma is what is called the adaptation gain, it will show up soon in the update law, you will understand why this called the adaptation again. So, what do we do? We take the derivative of V, and we substitute from the error dynamics here. So, what is the derivative? It is simply e e dot minus gamma theta tilde theta hat dot.

So, the question is, why did we get a minus sign? This is because of how we define theta tilde. So, theta tilde is equal to theta minus theta cap. And so, theta tilde dot is equal to minus theta hat dot, those theta star is a constant, so its derivative is of course 0. So, this is how you get the negative sign. So, all that we have done is taken the derivative of this V. And now for e dot, we substitute the dynamics. So, we do nothing for theta hat dot, because we have not even specified the dynamics for

theta hat dot. So, we do not do anything, but we plugin for e dot from the dynamics.

And once we do that, and you just expand it out, you will get the first term minus ke squared. Notice there is a nice negative term already. So, we like this term, we are not going to mess with this term, because k is positive. And so, this is a minus ke square, nice negative term. And then you get the term corresponding to the parameter error. So, this term would not have existed if the parameter was known. So, this is there only because there is a parameter estimation error, and that is simply theta tilde f times e. So, the first two terms are just coming from here. And the last term is copied as it is.

Now, behold this magic. What is this magic? Both these terms have theta tilde in it. So, what do I do? I take the theta tilde common. See, if there was no theta tilde in one of these terms, for example, there was no theta tilde here, it would have been a difficult challenge to specify theta hat dot, because this theta tilde is multiplying theta hat dot. So, in order to specify a theta hat dot, I would have had to have theta tilde in the reciprocal. And this creates all sorts of problems. First, theta tilde is not known, theta hat is known, but theta tilde is not known. Therefore, having theta tilde in any kind of update law does not make any sense. It is no longer feasible.

And secondly, you never want to design any controller or update law with states in the denominator. Because if they go close enough to 0, then things will blow up for sure. So, you do not usually have ever design update laws, or control laws with denominators containing the states. So, this is something that you should be very careful about. So, this is a rather nice coincidence if you may, although this is just a outcome of the structure of the candidate Lyapunov function, that both these terms have theta tilde, which can be nicely taken out. And then what we are left with is? These things.

Now, notice gamma is strictly positive, so can be divided now. So, not a big deal. So, what do we do? We do the best thing we can, because this is a mixed term, I do not even know what this is? I cannot have a theta tilde in theta hat dot, because like I said theta tilde is not implementable, theta tilde is not known. So, this is very important. Theta tilde is unknown. Many, many students make these mistakes. Theta tilde cannot appear in the control or update law. This is important. If you ever designed an adaptive controller, and your theta tilde or the parameter error appears in the update law, the control law, then you did it wrong. Because you cannot have an unknown. Because it is unimplementable, it is as simple as that.

So, what do we do? So, that we do the best thing we can, we simply cancelled this one, because it is not definite, indefinite quantity, when we do not know what to do with it, we simply make it, try to make it simple. And that is what we do by choosing this kind of a theta hat dot. And once we make this choice, this term goes away, and we are left with just this. Now, look at this carefully, I say that this is negative semi definite. Why do I do that? For the known case, if you notice, I had the same expression for V dot, although the V was different, I had the same expression for V dot, and I said it was negative definite. But here I say it is negative semi definite only.

Why? Simple, because the system now has two states not just e, like in the known case, but also theta tilde that is a parameter error. And I have said this quite a few times that any Lyapunov candidate or its derivative, if it does not contain all the states, it cannot be definite. For a function to be definite, it has to contain all the states, there are no two ways about it. Now, V dot does not contain all the states, but it is definitely non positive. Therefore, it is negative semi definite only. It is only negative semi definite. So, what do we know? We can all as use, as always apply the Lyapunov theorems, and what will I get?

I will get that the system that is the errors dynamics, or the e theta tilde dynamics is uniformly

stable at 0, 0 equilibrium. Whenever I talk of stability, it is important to specify the equilibrium. So, we only get uniform stability from the Lyapunov theorems. We cannot get anything more, why uniform? Because of course, your V and V dot do not depend on time explicitly, and therefore you have uniform stability. So, which is nice, which is a nice property, which means the straight trajectories are going to behave well, not start to blow up and stuff like that.

However, we cannot guarantee that they converge to the origin, because asymptotic stability has two pieces like it is stability and convergence. We got the stability, we do not have the convergence. So, in order to do, analyse the convergence, we need additional tools, and what is this additional tool? This is of course the Barbalat's Lemma and what is called signal chasing analysis. And this is what we will sort of start with next time.

So, great. So, what did we look at today? Is that we started to discuss an adaptive control design, for a first order scalar system. We first designed the controller for the known case. And then using a certainty, the certainty equivalence principle, we obtained a controller for the unknown parameter case. In order to derive an update law, we guessed the Lyapunov function first, or a candidate Lyapunov function first. And then took the derivative in order to guess an update law. And until today, we have been able to prove that the system is uniformly stable using the Lyapunov theorems, but we of course want to move in sort of claim more, which is what we are going to look at in the next session. So, this is where we stop today thank you for joining.