Advanced Control System Design
Prof. Radhakant Padhi
Department of Aerospace Engineering
Indian Institute of Science, Bangalore

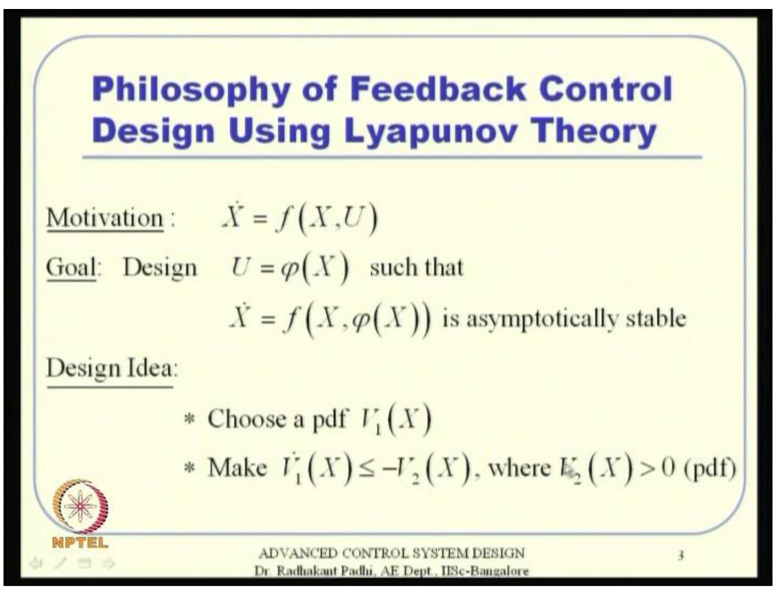
Lecture No. # 39
Integrator Back-Stepping; Linear Quadratic (LQ) Observer

We are almost at the end of this course, we are just at a lecture number 39 (Refer Slide Time: 00:19). Towards the end I thought, we will study some of this modern control technique, especially for the non-linear (( )) philosophy. One of that (( )) to be integrated integrator back stepping in something on that we are going to study today actually.

Towards the end of this lecture, I will also talk about linear quadratic observers, which are essentially leads towards Kalman filters and all that actually. And that this two are this, not related topics just put together, because because of like time is sufficient to discuss both the things today actually.

Alright, so philosophy of non-linear control designs using Lyapunov theory is something like this they (( )) this back stepping design uses Lyapunov theory extensively. So, what is the generic philosophy, suppose somebody wants to use a Lyapunov theory for in a control design, then what is the idea there actually?

(Refer Slide Time: 01:14)

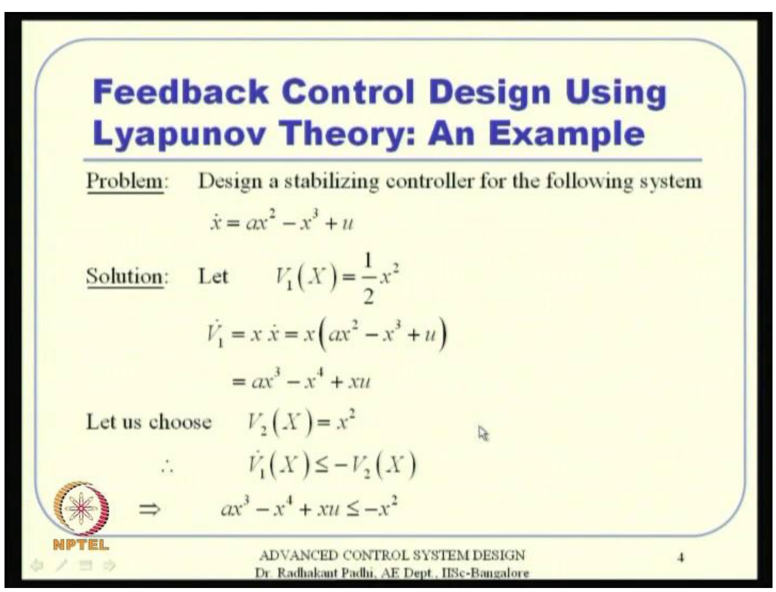


Something is the idea is something like this. Let us, say you have a non-linear system X dot is f of X U, where you want to design a control U in those straight feedback form, which is like phi of X. So, U has to be as a function of X, phi of X such that, this once you put it back into the system dynamics that is X dot is f of X and then, phi of X that system (( )) (( )) system dynamics should be asymptotically stable. That is that is the ultimate objective for Lyapunov (( )) control theory actually.

So, what is the idea there I mean how do you (( )) (( )) (( )) happens actually? So, the idea here is, first you choose a positive definite function V 1 let say; and (( )) of V 1 dot is less than equal to negative of V 2, where V 2 a positive definite function. And remember these are all like sufficiency conditions and all that. So, you will not be operating in a optimal way, but you will be operating on a conservative way basically, that is ok.

So, what do what you essentially do is, we select a Lyapunov function V 1 of X and make sure that V 1 dot of X which is once you take V 1 dot it is related to the system dynamically anyway. So, this V 1 dot of X should be less than equal to minus V 2 of X will be to itself is a positive definite function. So, negative of V 2 is negative definite function. So, we want to make sure that V 1 dot is less than equal to another negative definite function that way. So, that is the whole whole idea design idea basically.

(Refer Slide Time: 02:47)

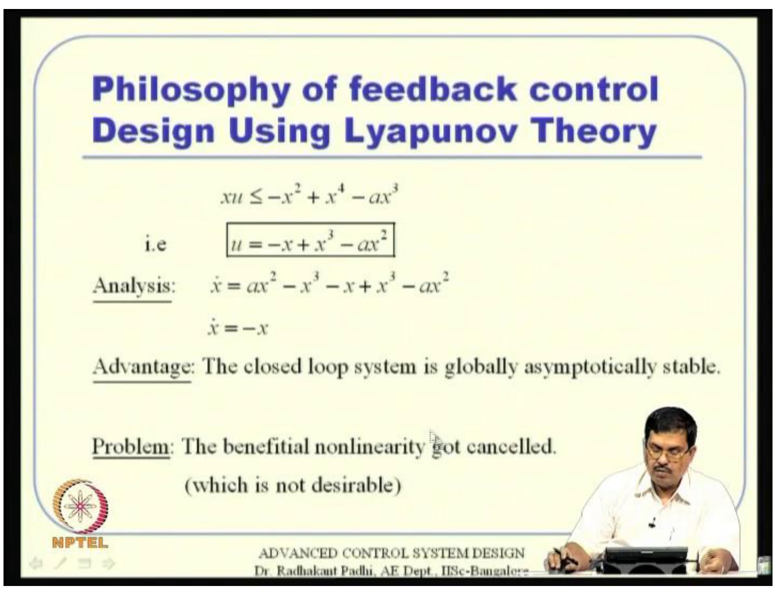


So, let us see a small example if you are talking in lot more formal theory and think like that. So, this is a simple scalar problem, where x dot is a x square minus x minus x cube plus u. And we want to design a straight feedback control u, so that the closed loop system is asymptotically stable, that is the objective actually.

So, we have to select V 1 first. So, let say V 1 will say will naturally select quadratic function half of x square. So, V 1 dot happens to be x times x dot. Obviously, because del del V 1 by del x is x times x dot; and x dot is this from dynamic what you see here, so we put it back here. So, essentially (( )) V 1 dot is something like this (Refer Slide Time: 03:30), now this V 1 dot has to be less than equal to negative of a positive definite function.

So, we have to select another V 2 like that. So, let us say select V 2 is x square basically let say, just you can select any positive definite function for say, this is a just one of those selections. And we want to make sure that, this in equalities valid actually. So, V 1 dot is less than equal to negative of V 2. So, if you put V 1 dot in this expression (Refer Slide Time: 03:56), so this expression is less than equal to negative of x square.

(Refer Slide Time: 04:01)



So, then from where you want to solve for a controller that means this this x u is less than equal to this this expression (Refer Slide Time: 04:07), and equal to sign will also is evaluate think. So, will take equal to actually and then make sure that, u is equal to that ok.

So, we cancel out x in both side, 1 x will go from both side and then, we have the u is in minus x plus x cube minus a x square. So, is it control good or bad I mean naturally, it is good; because we have make sure that V 1 dot is is negative definite function anyway. But, still just cross check is about it, is just put it back this controller, (( )) the system dynamics and then, try to analyse what is going on there.

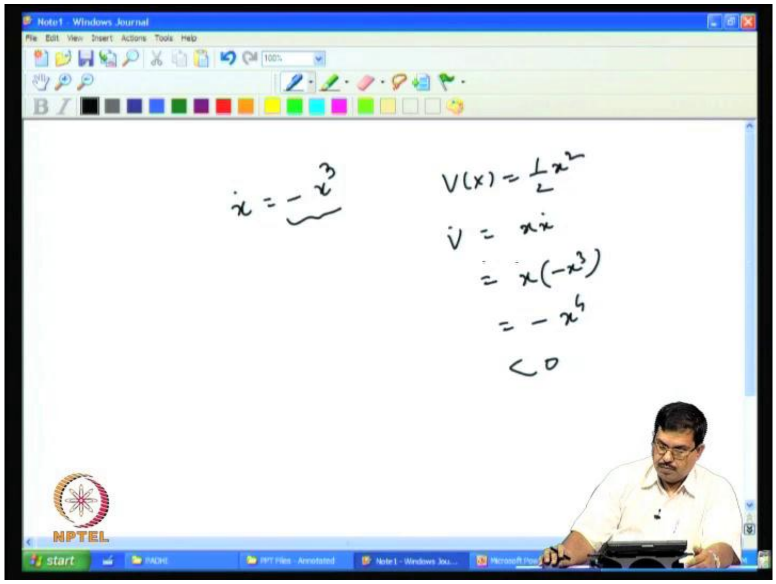
So, if you put this this system dynamics is a x square minus x cube plus u. So, plus u whatever u we are getting, we will put it back. So, a x square minus x cube plus u, u is that portion actually; this portion is is like plus u ok (Refer Slide Time: 04:55). So, that that is what do you do.

But, then once you put it back, these to these to are cancelled out (Refer Slide Time: 05:04), and plus x cube minus x cube also cancels out and you have left out is x dot is minus x actually. Then x dot is minus x is certainly a linear system dynamics actually; and obviously, because the Eigen values minus 1 obviously, the system will asymptotically stable. In fact, it is global asymptotically stable.

So, that is because this linear system dynamics ultimately, so it does not depend on cell condition that strictly straight away from Eigen values you can say, it is globally asymptotically stable. So, that is the advantage here actually. So, you design a controller it is a non-linear controller in such a way that, the closed loop system becomes globally asymptotically very stable. But, what is the problem there, it is also problem it is a remember that, this V 1 in V 2 (( )) Lyapunov theory these are like sufficiency condition and all that.

So, once you use that, the design can be actually conservative. And here, it is actually conservative design, primarily because this let us assume that; if you go back to this (Refer Slide Time: 06:08), this negative x cube, x dot is minus x cube is actually stabilising term that is that is also like easy to see. For example, is any (( )) power of like a scalar function if I take any (( )) power of X is a stabilising thing with negative sign.

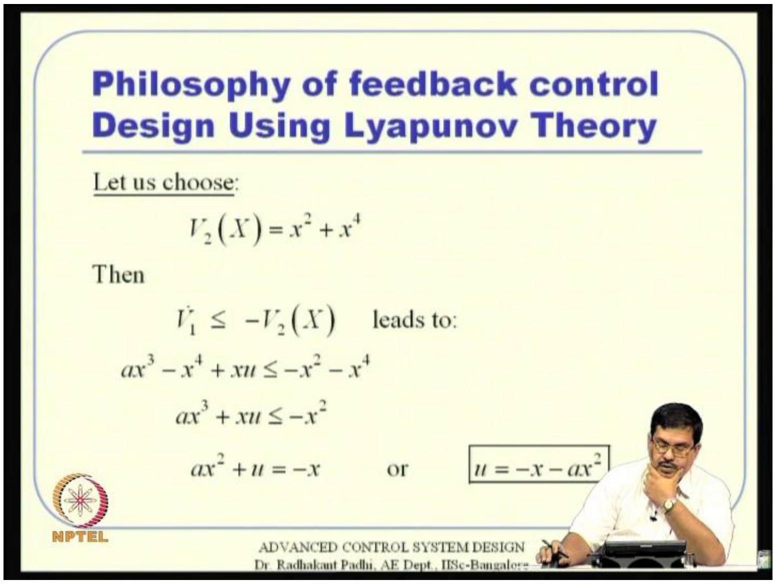
(Refer Slide Time: 06:26)



For example, let say x dot equal to minus x cube that that one if you are talking, then corresponding to that let us say Lyapunov function if you if you say it just half x square, then V dot happens to be x x dot this is nothing but, x times minus x cube. So, it is negative of x (( )). So, this is actually negative definite function.

So, any (( )) power of x with a negative sign here will be actually stabilizing term. So, but that, what what happens here is, unnecessarily we have actually cancelled that out. So, this controller actually tries to cancel out the good part of it also basically. So, we do not want that normally. So, what we do, how do you about that? So, let us be, because this this small problem we should be able to try analyse this little bit more, just by looking at equations and all.

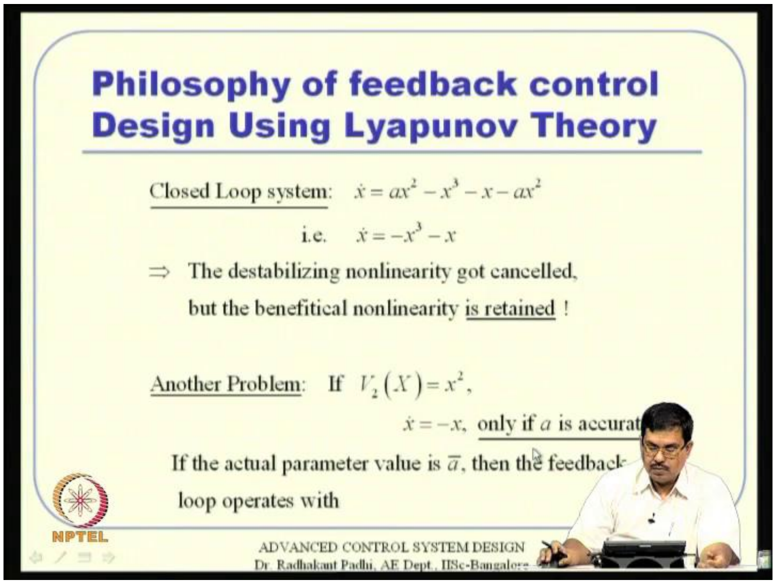
(Refer Slide Time: 07:15)



So, let us see we select something like V 2 I mean probably our V 2 selection was wrong. So, will will revise this V 2 selection and (( )) let us consider V 2 as x square plus x 4 th (( )), this is also a positive definite function by the way. Now, if you go back to this inequality again that, V 1 dot in a negative less than I mean negative less than equal to negative of V 2 X.

Then the same analysis will tell that, u is instead like that (Refer Slide Time: 07:41), again you put it back; and then plus a minus x 4 th n minus x 4 th will will get cancel out from both side, and you will be left out with that term actually. And here will be able to I mean cancel 1 x and all that actually, so anyway. So, that is that is what will happen here actually, so u will be like this. Again you go back and try to analyse what is a closed loop system.

(Refer Slide Time: 08:04)

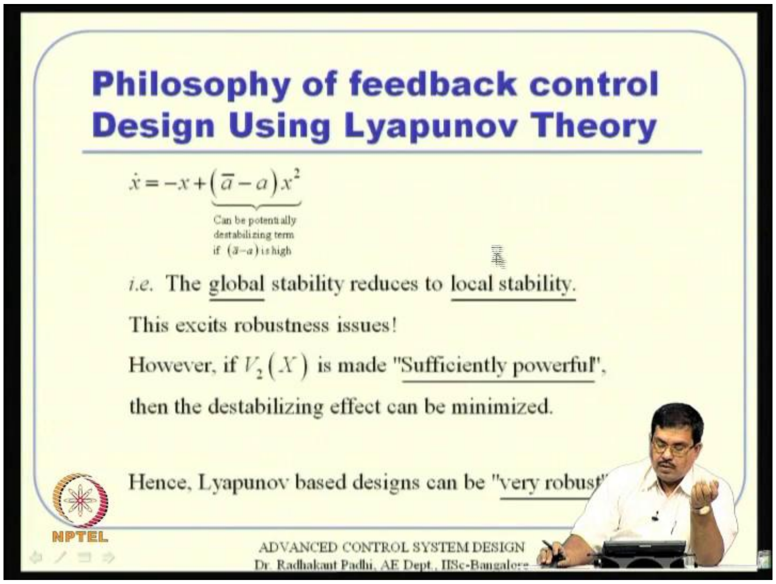


Then, x dot is a is like this now, because that is that is my u; in this case, this is my u now negative of u. Now, such this is plus u, because this this is our system dynamic actually. So, way to go back to this system dynamic and then, tell this is plus u here. So, you put it this u, whatever u we are getting here I will put it back; and then try to see that, x dot is instead minus x cube minus x.

So, we are able to preserve this this beneficial nonlinearity by selecting a different way, so just just the message here. So, the design is not just unique design. So, it subject to different different selection of V 2, V 1 and all; you will end up with different control actually. But, all of them will be will be able to do the job, and all of them will have some degree of robustness also; because, these are all sufficiency conditions and all, robustness also comes into picture.

Now, let us see how it is say we suppose we go back to that that initial selection. Let us, say V 2 whatever we selected initially is x square. So, will go back to that selection and with that selection, our closed loop system becomes something like x dot equal to minus x; but this x dot equal to minus x, only if the parameter a is accurately (( )) remember that, this this entire thing, this this process of cancellation and all that. What we did here, this this a x square and a x square, what about this a x square and this a x square got cancelled out, assuming that the way is known perfectly. But, if a is not known perfectly then, this a will different from what is use in the controller that a will be different. And hence, you will not be able to cancel out.

(Refer Slide Time: 09:57)

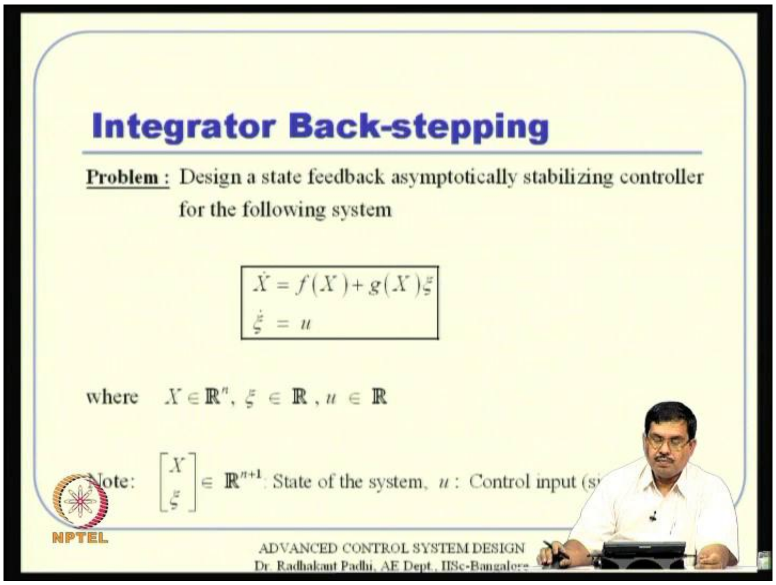


So, in that situation we will have additional term subject to this, this is not no more x dot is minus x, this it will be x dot equal to minus x plus an additional term with with x square actually. So, in this x square is certainly stabilizing term for I mean depending on this, how powerful is this a bar minus a is actually that means how much inaccuracy is there. (( )) whatever is inaccuracy, now because it is no more x, so it is minus x, this certainly this this global stabilities gone; and we can only conclude local stability, because globally this is no more valid actually, one is stabilizing term, other is (( )) stabilizing; so, obviously global thing no more actually.

However, if you make V 2 sufficiently powerful that means V 2 is large and large and thing like that; then obviously, the control will also go and take get more and more (( )) I mean powerful basically. So, essentially what you are telling, even in the presence of certainty like this, the design can be robust actually, because this term will not be that prominent at all actually.

If your V 2 is very powerful then, V 1 dot is minus of V 2 X any way. So, because that term is being very high quantity in thing like that, this will this will lead to V 2 robust control actually. So, that is the whole idea, why this this Lyapunov theory based control are typically robust actually, alright. So, with that (( )) in the background let us study about this this integrator back-stepping method. And then this is generalization of this concept, what is discussed actually.

(Refer Slide Time: 11:33)

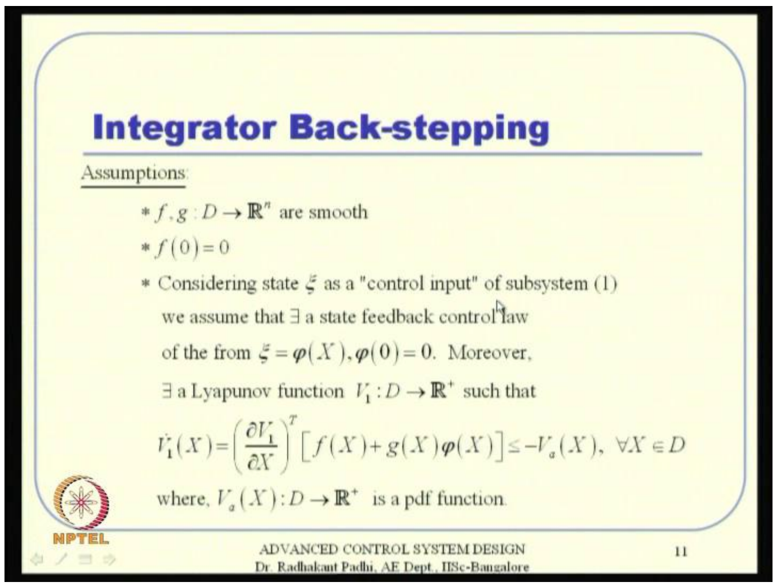


So for here, we need this system dynamics in a typical form and for that form will start with a simplified form of that version. And this is I mean towards end of this this topic we will see this is like what is call (( )) feedback system and all that actually. This is just one subset of that. So, what you are telling here is, let let X be n dimensional (( )), let xi be also state, where xi dot simply u. So, X dot is f of X plus g of X times xi, and xi dot is simply u actually; xi is scalar remember, xi and u both are scalars actually.

So, if the system dynamics is given in this form, then I can think of using I mean this back-stepping idea and all that. So, remember that, these both together are the state vectors actually that means X and xi together will define the states of the system. So, obviously the states of the system will be n plus 1 actually. And where is this this (( )) open think like that probably, if you see this any system dynamics in control affine form already and you want to incorporate let us say actually dynamics also then, this is the original control variable; but, this part will be actual dynamics you want to see the total system that way, then it will it will (( )) something like this actually.

So, these are not very unrealistic problems in that sense basically. So, here here we are talking about single input, the control is scalar; and the xi state is also scalar any way. So, that is the form that you are talking.

(Refer Slide Time: 13:08)



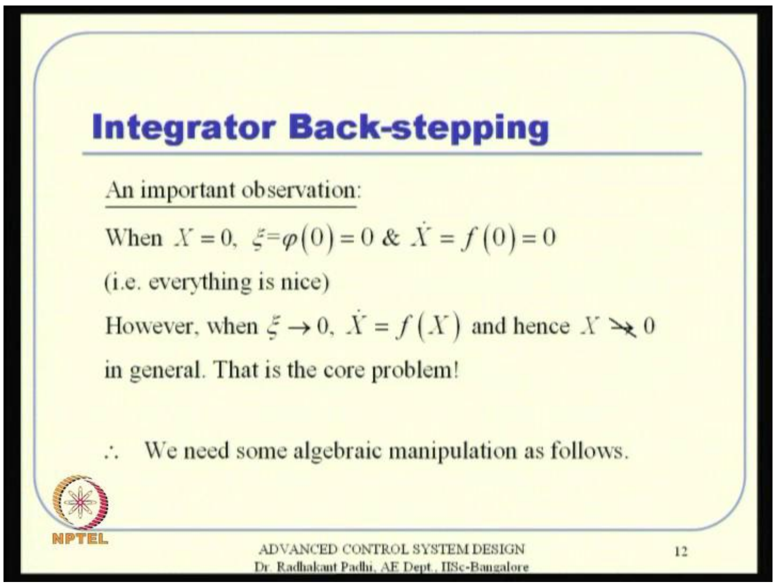
So, what are the assumption, assumption is first is f and g are both smooth I mean the derivative also continuous; f of 0 is 0 whatever f of 0 we are talking, here is should be 0. And if you consider this xi as a control input to this particulars state first part, first part of the equation is xi (( )) as a control variable.

So, if you if you consider that, then we assume that **(( ))** feedback control already in the form of xi equal to phi of X, where phi of 0 is 0. So, that that means assuming this is control variable, we have already within controller in appropriately I mean; so, what is that means? xi is already design as a function of X, where phi of 0 is 0. Because, it is a stabilizing controller, it also is a Lyapunov function with that.

So, the (( )) Lyapunov function V 1 such that V 1 dot which appears to be like this (Refer Slide Time: 14:04), it will be del V 1 by del X into X dot, X dot is like that anyway. So, that is less than equal to certain negative V a of X, where V a V a is a first definite function. That means, all that you are telling is first part let us consider is a as a separate problem, for which xi is control variable; and that this xi is actually already design as a straight feedback form with all good behaviour basically.

So, but again problem we need to we need to design this xi also basically that way. And as I told, this back-stepping design is kind of very popular for for incorporating actual dynamics into the system and all that actually, so anyway. So, this is the assumption here. Now, what we want to do? Now, we want to design u, we can think of that as input the X dot probably. So, that the overall system remains a stable actually. So, that is the that is the problem here actually.

(Refer Slide Time: 15:04)



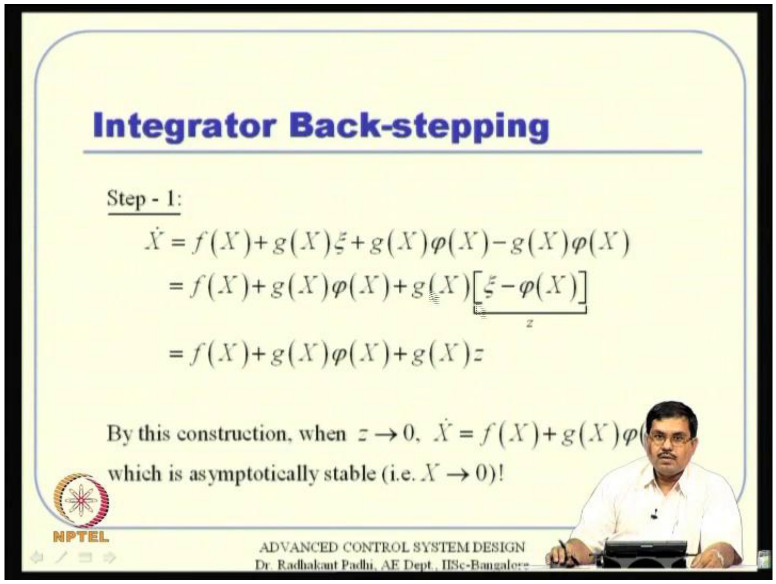
So, let us see how do you do that, but observation let say one important observation before you move further. Let us consider, what happens when X is already equal to 0, X is gone to let say X equal to 0 as already achieved then what happens. Now, remember phi of X by definition phi of 0 is a 0 anyway, that is what, that is how this xi is already designed. So, when X equal to 0, then then phi of 0 is 0 that means xi is also 0. So, this part is 0 anyway. And then, f of 0 is 0 by assumption.

So, X dot become 0. So, no (( )), so when X goes to 0, X dot is already 0. So, that is that is fine actually. So, the system will go to equilibrium condition and it will stay there. Then that is more important actually, that is not a problem. But, then the problem happens in another way round; like what happens when X goes to 0 nothing not a big problem actually.

But, what if this xi goes to 0, when xi goes to 0 this this this system dynamics, the first part of the system dynamics is almost becomes (( )) homogeneous dynamics actually, xi can go to 0. So, this part is not there, but X dot is a f of X that that system is no guarantee that, X will go to 0 that system dynamic is homogeneous dynamic. They I mean the only in the closed loops sense, X will go to 0 that that is how the guaranties already there.

That we are already design a feedback control of that for that behaviour, but for the homogeneous system you know (( )) actually that means, when xi goes to 0 we are not very sure that whether X will go to 0 or not actually. That is not a (( )), that is not a allowed actually, because our our total aim, the final aim is to make sure that, X goes to 0 also (( )). So, how do you do that? So, that that is the core problem actually; so for that, we need to do some algebraic manipulation as follows.

(Refer Slide Time: 16:57)



So, let see what we do there. So, for this this particular system dynamics first part of it, it contain only this part f of X plus g of X times xi. So, now let us add and subtracts this two terms actually. So, this will add g of X times phi of X, and subtract g of X times phi of X. when need do this operation, let us let us consider this this part of it, we will skip it, remember this phi of X is the same phi of X that we are talking actually here, xi is already design in the form of phi of X actually.

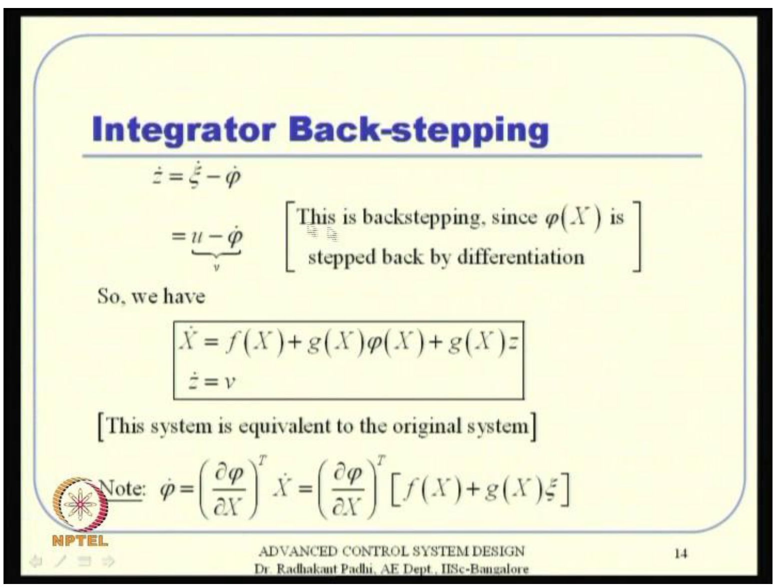
So, it is just an algebraic manipulation sort of it actually. So, what what you are doing here? So, we are keeping this part of it here, then this this part and that part will combine and then, tell this is this is what will happen actually. So, g of X times phi of X plus g of X times zeta minus phi of X; obviously, we are not changing anything here I mean a system dynamics we are just I mean playing around with little algebraic actually, without altering the system dynamic.

Now, with this change or (( )) this xi minus phi of X let me define as something like a new variable z actually. So, I am just defining that for for (( )) simplicity (( )). Once I do this, this definitions z then the system dynamics will nothing but (( )), f of X plus g of X times phi of X plus g of X times z for of X plus g of X time z actually.

Now, what actually, earlier when zeta goes to I mean xi going to 0, then we had a problem. But, what if the same problem is there or not? When you see this system dynamics in this way that means when when z goes to 0, this system is no more like only f of X; this system is f of X plus g of X times phi of X. And phi of X is design will appropriately already basically. That means when z goes to 0, this system dynamics is no more and purely open loop system dynamics, actually closed loop system dynamics. And hence, this asymptotically stable anyway, so X will go to 0.

So, the problem that we had that, when xi goes to 0, X will not go to 0 in general; that does not happen, when z goes to 0. When z goes to 0, X is gone to I mean supposed to go to 0 is basically. So, that problem will overcome that way basically. Now, how do you do further algebra, because we have to put it into this form, that form has to be there actually anywhere.

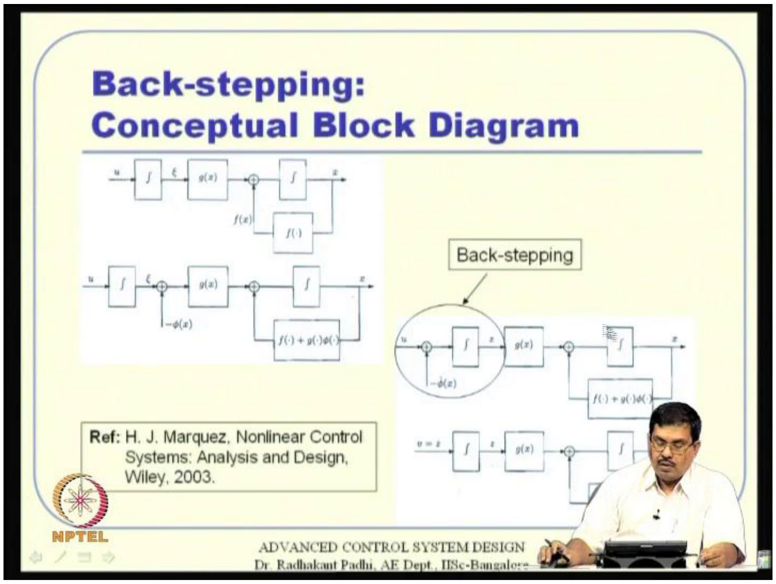
(Refer Slide Time: 19:23)



So, let us analyse this this z actually, z is by definition the remains as phi of X. So, we put it z dot is xi dot minus phi dot. So, if xi dot is nothing but u, so the z dot is nothing but, u minus phi dot actually. So, we have this system dynamics like this modified system dynamics in this form, where z dot is nothing but, v; v is this artificial control variable that I am interpreting that, u minus phi dot I am defining as v. And interpreting that z dot is nothing but, v actually.

So, if you see this system dynamics, now it appears in the similar form that we started with actually. So, this this form is nothing but, something plus g of X times z and then the next one is z dot is v. So, this system is equivalent to the original system actually. So, only thing and only problem is instead of I mean is phi is use in the form of phi dot actually. So, phi dot can be computable and that means del phi by del X into and X dot is available any way. So, that is that is the way, you compute phi dot.

(Refer Slide Time: 20:22)



So, what is going on in a block diagram sense, this is original system dynamics, what we started with u and then integrated goes to; see, if you see this u, if integrated zeta that is a xi (Refer Slide Time: 20:34). So, that is what, it happens here, if u integrate will get xi then, it multiplied by g of (( )) I will get g of times xi and then, this feedback operates that way actually. So, what you see here is a f of X plus g of X times xi. So, that is that is X dot.

So, that how it happens, but what are doing here is, first of all we are putting a g times phi here in this loop; and for that for that reason, you have to introduce negative phi of X phi of X here, but we do not stop here, this phi of this phi of X really not introduced here, it is introduced once one block before that; and because of that, we have to introduce phi dot instead negative of phi dot. And instead of phi of X, it is introduces negative phi dot X actually. So, phi to phi dot is actually (( )) actually.

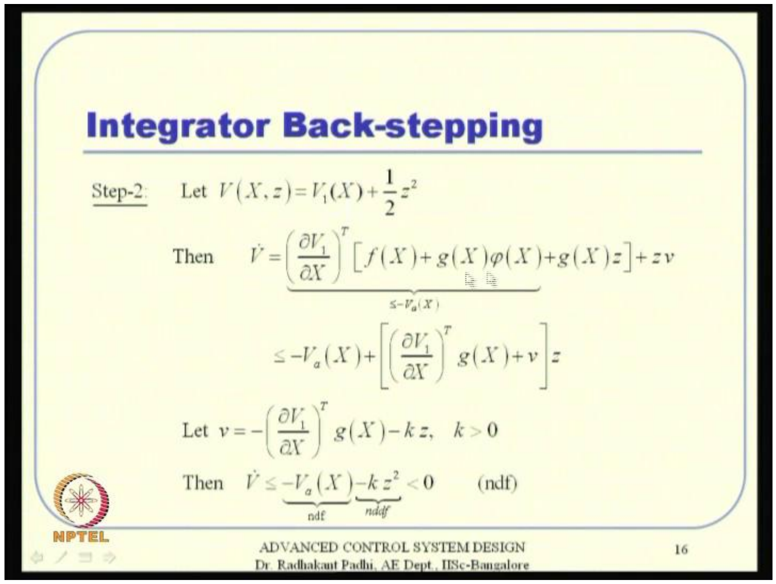
So, whatever is integrated that, that signal flows forward; whatever is differentiated, the signal flows backward. That is why it is called back-stepping actually, we are just stepping back in the block diagram one time and then, interpret I mean this this interpreting that u minus phi dot sort of thing, because of because this algebra basically. This entire thing is borrowed from from this book (Refer Slide Time: 21:49), and if anybody interested you can they can see this book also, this is a good book I think alright.

So, this integrator back-stepping, now we have to I mean see that, how do you design this V of X and thing like that. Now, we are talking about this system dynamics (( )). So, how do we do that? So, let us start with this V of X, now we have to define a Lyapunov function inclusive of z.

So, will define something like V 1 of X plus half z square sort of thing. So, V dot is nothing but, del V 1 by del X times X dot, and X dot is all that; because X dot is, is what it what we derived actually. So, the entire expression will put it back, once you put the entire expression this plus this z time's z dot. So, this this term will give us z times z dot, and z dot is nothing but v. So, this z dot is v that is what we will put here actually. So, that is how we will end up here.

So, but this part of the system I mean this part of the algebra is nothing but, negative of V a of X; that is what it is all are assuming that is already already designed that way. So, this is available to us actually right by the by the way we started with, we will start ok that this xi is already designed with associated Lyapunov function we satisfy this this inequality. So, (( )) is available to us. So, we use that ok.

(Refer Slide Time: 23:17)



So, this nothing but, less than equal to V a of X. So, we keep this part is less than equal to V a of X. Now, if you see this part this one and that one to get that sort of thing and that is that is what we will combine actually. So, then once you combine this two part, this this first part and this this X this term and then, you have z time's v actually.

So, when we combine this two term will will end up with this negative V a of X here, and that is already available to us anyway plus this additional term actually. Remember, z is common to both actually, so I can I can take out z actually. So, that is what happens here. Now, the question here is how do you measure, how do you design v such that, this entire thing is negative definite that that is the one point actually.

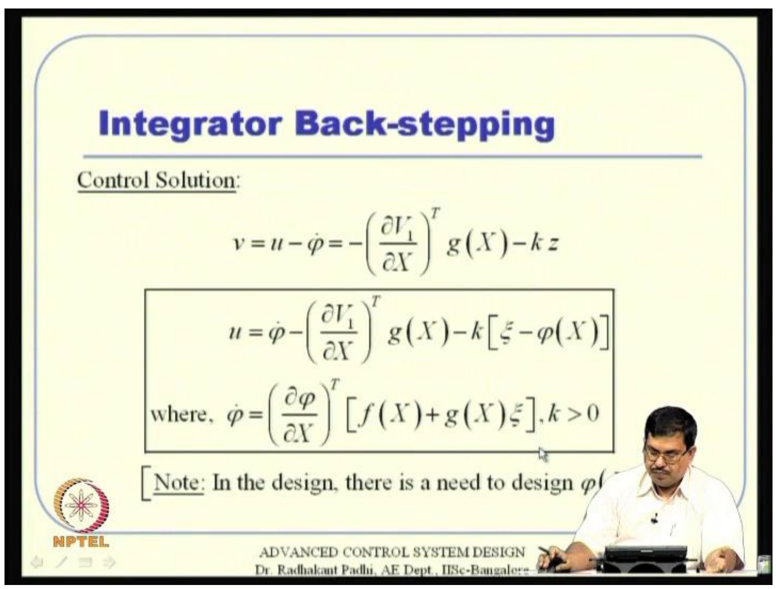
So, when you see something like this, the very natural the idea that comes to mind is let me design v this way. So, I will cancel out this term and whatever is left out, it will generate something like a negative k time's z square, where k is positive. So, naturally I will selective v, which negative of this quantity minus k z; so that, when minus k z is left out here, it will generate a minus k z square term actually.

So, that is the whole idea. So, now V dot turns out to be less than equal to a negative definite function that it is already available minus k z square which is which is also negative kind of semi definite and all that actually. Because, if you if you see the total things, so if you see the total things like total system dynamic the entirely Lyapunov function must contain both X as well as z actually right, then only the system dynamics we can conclude something actually; if any part is left out will have a negative semi definite sort of idea there.

So, when you see this this entire function, the first part assures that, it is negative definite for the first part of the system dynamic. In the second part, will assures that, it is also negative kind of semi definite for the second part; combine the (( )) together sort of thing, they will have a negative definite function for the for the (( )) total system actually. So, that is the way it will have a negative definiteness actually.

So, whole idea what is that actually? Now, the whole idea is v turns out two like this, and v is an also known to us already; v by definition is u minus phi dot, phi dot is anywhere available. So, using all that, I will be able to extract u basically ok.

(Refer Slide Time: 26:00)



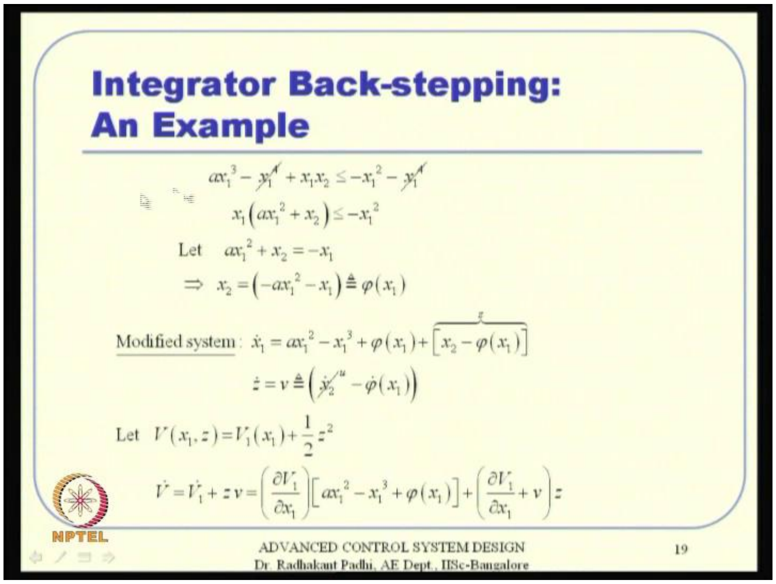
So, v is already u minus phi dot. So, and phi dot mean that expression is available to us any way, because v is already, v is designed like that. So, that is equal to that and phi dot is (( )) there. So, u equal to phi dot minus there and phi dot is like this already there actually. So, that is how we we design the control u basically.

So, but also note that, in this entire exercise first one should, first design this phi of X, because without that, it will not proceed further, but that is anyway there actually. For for this method to start, the phi of X should be should be design actually first. What you are assuming here that, there exist a feedback law for xi equal to phi of X with phi of 0 is 0. So, that is to be design first. Then we design this u and all that. So, let us see a simple example again, like x 1 dot is the same example probably what you go back, instead of u we will put x 2; and then x 2 dot equal to u, that is how we will make it as a (( )) system. So, that it is compatible with the theory that we discussed. So, ultimate objective here is u has to be designed. So, that over all system becomes asymptotically stable.

Now, we have already designed x 2 before and let us repeat the exercise one more time. (( )) So, you will consider a x 2 control variable for the first equation and then, we will I mean either you (( )) and put this entire like identify which one is what and just (( )) into the formula get the answer; or you can stop from the beginning and get the answer like that actually.

So, you have this V 1 of X is like that. First of all, you have to find the phi, for finding that we will take V 1 as half x 1 square. Then V 1 dot is like that, and this will be less than equal to some V a of x 1. And (( )) that previous example that, we started with the same example we will take x 1 square plus x 1 4 th, which will not cancel out the good non-linearity part of it; we will keep it that way.

(Refer Slide Time: 28:02)

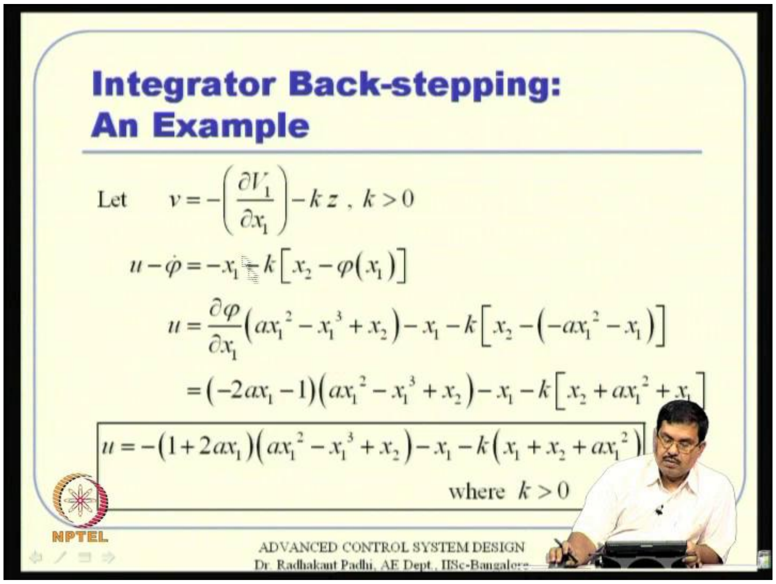


And hence, we will cancel out all the thing and design this this x 2, this is earlier we design it as u, but this time it is x 2 basically. So, this turns out to be phi of x 1. So, that is how we got phi of x 1 basically. Now, will go back and the modified system dynamics we have to write. So, we will add and subtracts and things like that.

So, will will write this x 1 dot in this form and where this z dot is v; and v is defined as u minus phi dot. So, that way we will define actually. So, we have to take this V as V 1 of x 1 plus half z square again, that is a following the same theory that we just discussed actually here. So, will put that the same idea back in here, (( )) is V 1 of x 1 plus half z square. So, V dot is nothing but, V 1 dot plus z time's v, z time's z dot, z dot is v anyway.

So, that is just what it is. This entire algebra gives us like this. And then will be locate this v has to be I mean this algebra whatever we see here, v has to be minus del V 1 by del x 1 minus k z square. So, that is I mean minus k z basically.

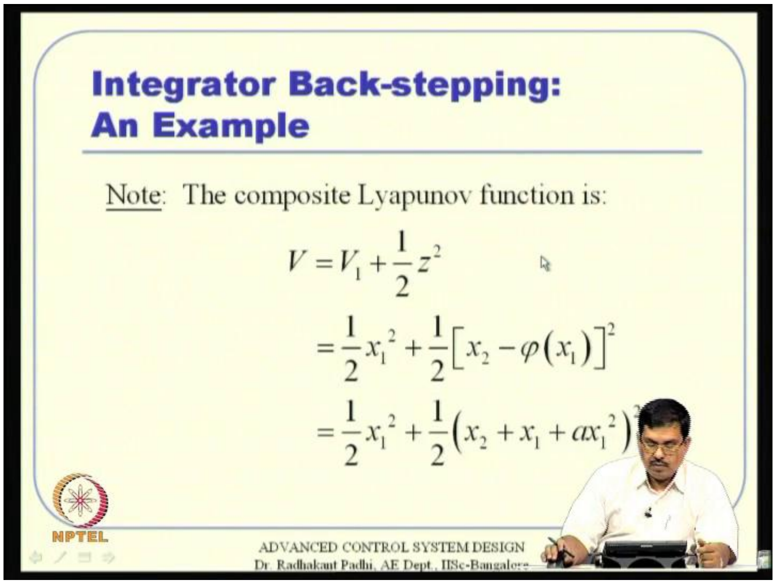
(Refer Slide Time: 29:09)



So, that is what it is. So, if you put it back and try to solve for (( )) control v is nothing but, u minus phi dot. So, u equal to phi dot minus x 1 minus k, about phi dot is a nothing but, del phi by del x 1 into x 1 dot; x 1 dot is like that minus x 1, so minus x 1 minus k times (( )), so minus k times (( )), phi of x 1 is that way anyway.

So, you now you expand all the thing, algebra and then get a control (( )) that form actually. So, this k becomes gain value, which is positive number which needs to be tune for your good performance basically. That is how that is that is (( )) example actually.

(Refer Slide Time: 29:50)

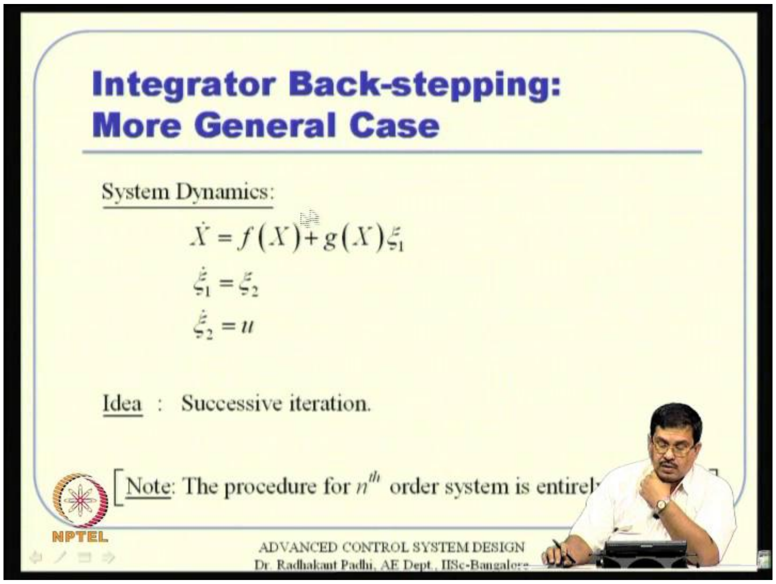


So, what is the associate Lyapunov function for this? The compositely Lyapunov function turns out to be V 1 plus half z square. So, V 1 is obviously already there and z square is nothing but, z is by definition u minus phi phi dot sort of thing right, so that that (( )) sorry. So, z is nothing but this what is that, that definition of this one extra minus phi of x 1 that is that is, so you put x 2 minus phi of x 1 that is that part of it.

So, this is a compositely Lyapunov function for the total system actually. First part is only for the first equation; when you put the second one, it becomes Lyapunov function for that total system actually. Now, more general case this is all about like, we have this this type of system dynamics to be started with.

This is the foundation is to anywhere, so we started with where X is n dimensional state, and one more state is the xi dot that is equal to u. So, that is how we start with. What about more general case actually. So, when you take more general case and all, so this all happens to be not that (( )) (( )).

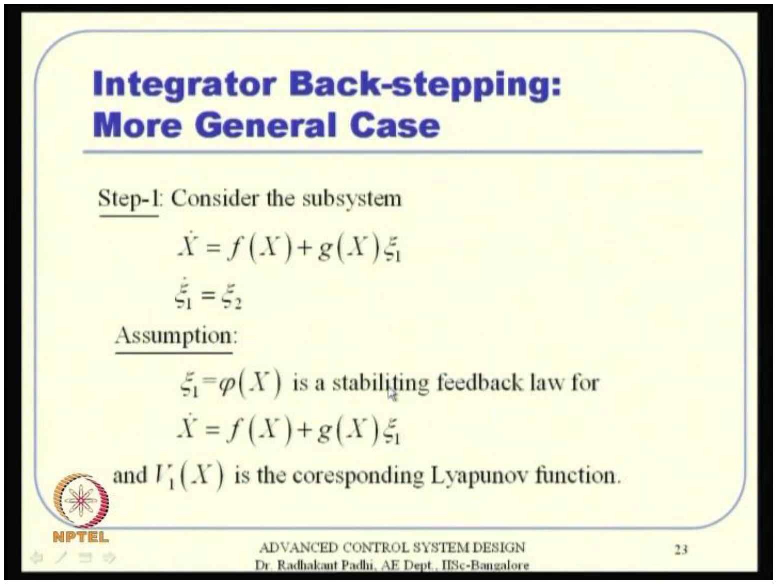
(Refer Slide Time: 31:05)



Let us, consider this one as a natural extension actually. We have xi 1 here, where xi 1 dot is xi 2 and then xi dot is u. So, obviously the idea is successive iteration actually; like if you consider only this system first, the first two equations first, once you have done with that, will get xi 2. And then you consider this first sub system is one and then you add one more xi 2 dot is here ok.

So, successively have to we have to design two times actually. And this it is the system dynamic is analogous that means let say, this is (( )) sequence does not stop here; but it is like xi 2 dot is xi 3, xi 3 dot is xi 4, and xi 4 dot is xi 5 like that. And then somewhere let us say xi 10 dot is u like that. So, you have to repeat this exercise like ten times actually.

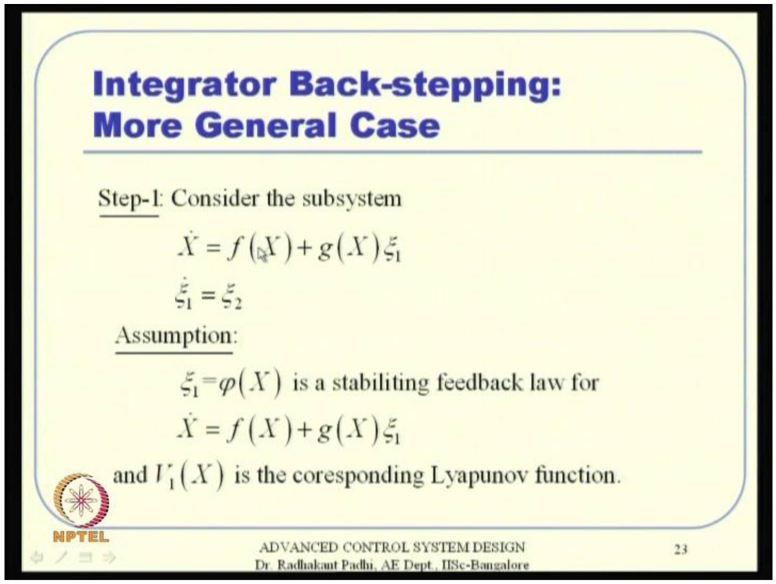
(Refer Slide Time: 32:00)



So, we concentrate only on this. So, where we talk about this xi 2 dot is u and then, let us see what happens actually. So, first will consider only first two equations. For the first equations we have just down everything, this first is like xi 1 and for the xi 1 phi of X like exists already. And we have to design something like xi 2 considering that this equation together.

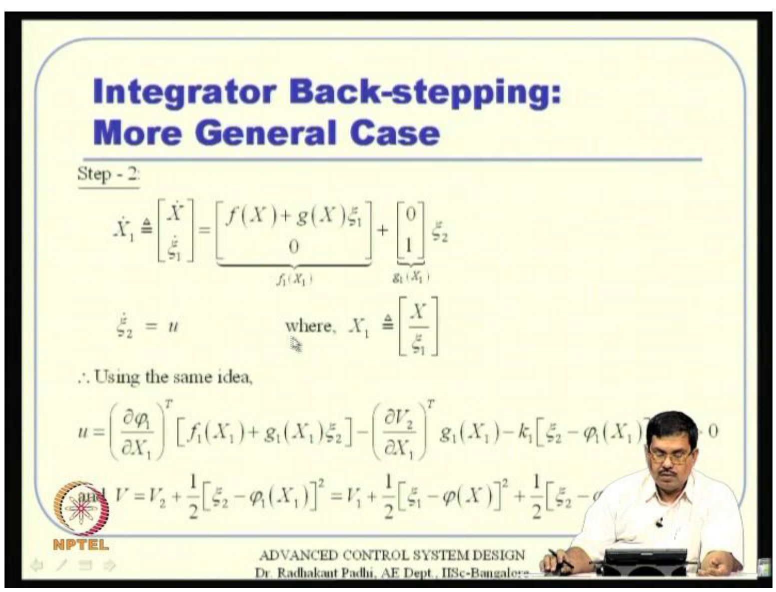
So, we will proceed the exactly same thing that we just discussed and will tell this this V 1 like the corresponding Lyapunov function is available, (( )) one can be done, the way we just discussed actually. So, then will take by the time by the way that, just I mean, so this this part of it, so we just do that. So, so the analogous I mean the then the inference we can directly write xi 2 instead of (( )) about all the derivation one more time.

(Refer Slide Time: 32:49)



So, let us write xi 2 and xi 2 has to be in this form, del phi by del X into a X dot minus this this term minus that term and all that actually. So, that is available and also we have a composite Lyapunov function for this to for this system, which is like V 1 plus this one. That also we just discussed. Now, what about extending to one more actually; so, if we want to extend this step, this is about step 1 sort of thing that we just derived.

(Refer Slide Time: 33:16)



Now, where step 2 we have to consider this this this one more (( )) into the picture basically; so, will tell this this type is first first it is like that, first subsystem will like this. And then we are an adding one more xi 2 dot equal to u. So, you consider this part of it something like f 1 of X 1, where X 1 is X and zeta 1 I mean xi 1 together. And then tell this a xi 2 term here, so I will put (( )). So, X 1 dot is this expression, so f 1 of X 1 plus g 1 of X 1 time's xi 2. So, X 1 by definition like that.

So, this expression what is what is here the first part of it, I will just change it the definition is X 1 instead of X. So, it is X 1 dot is whatever that is in the first equation, then xi 2 dot (( )). Then again we know what how to do how to do it. So, will do this algebra again and then tell u now becomes like that actually.

So, entire (( )) expression what we have operated here, instead of X we will have to operate it based on (( )) dynamics actually. And V 2 V is to be V 2 plus xi 2 minus phi 1 of X 1. So, that that is to be carefully noted actually and this phi 1 happens to be all that by the way this enter thing this entire thing is phi 1.

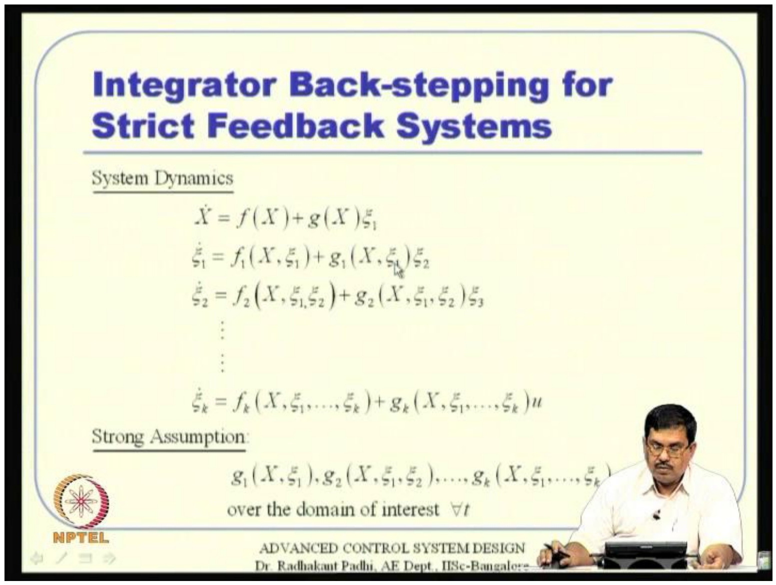
So, (( )) algebra is a necessity in this approach anyway. So, (( )) have to (( )) should not get I mean impression with algebra and all that. What essentially its its lot more (( )) keeping rather than two much difficulty actually. If you once you understand one or two steps, then is just extension of that same idea basically.

So, we have this and then, put it back here and then tell using the same idea will get a u. And the composite Lyapunov function for the entire system that means this system together X dot xi 1 dot and xi 2 dot (( )) to be like that. And as I told you before this this this is actually not very unrealistic, because if you see this xi 1 has control (( )) to the original plan; but this second this will like a second order actual dynamics.

First order actual dynamics is what we (( )) I mean we just discussed in this first part of the thing, first order actual dynamics that will happen that way. If you have a second order actual dynamics that will I mean the two equations that will happen that way actually.

So, this is this will be augmented to the system dynamic. We want to design robust a etcetera input and all that. So, then this is procedure, one of the procedure that is available to you is back-stepping actually. So, will go that and then, this is about that. And then we talk about what is best the generalization that you can have this this approach actually? The best generalization turns out to be in this strict feedback system and all those thing that we discuss are subsets of that anyway.

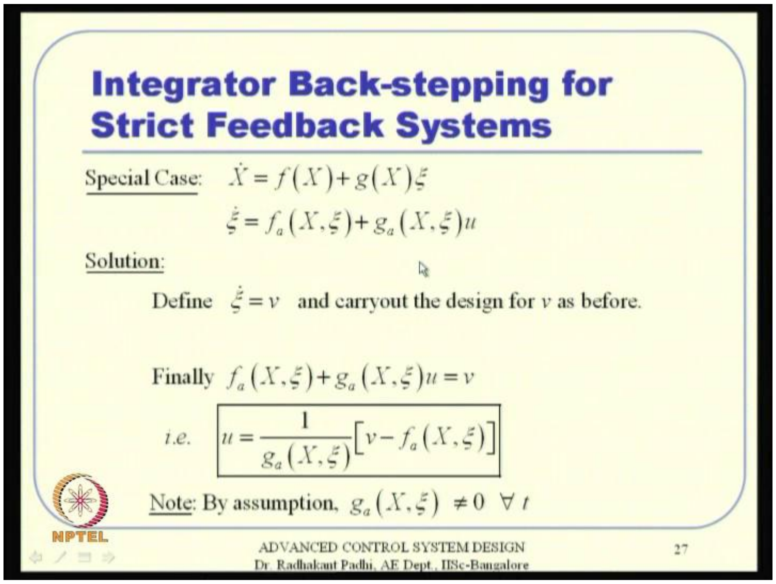
(Refer Slide Time: 36:09)



So, strict feedback system is something like this, we have X dot is like that, zeta 1 sorry xi 1 dot is like this, where xi 2 is an input; xi 2 dot is like this, so xi 3 is input like that it will continue to some arbitrarily or arbitrary order k basically. So, if it happens like that and there is a very strong assumption here by the way. What it tells you is? All this function that, you looking at not the first one, starting from g 1 g 2 up to g k, all this functions are certainly not equal to 0, over the entire domain of interest for all time. That is actually very strong assumption, but with that assumption, we should be able to solve it is actually.

So, what is the first the whole idea here, I mean we will not worry so much about this this functional dependence and all. Let us consider only this first I mean first two terms sort of thing.

(Refer Slide Time: 37:12)



Then this, whatever is whatever is a appearing here 1 plus g 1 time xi 2, I will simply define it is some sort something like a new variable sort of thing. So, let us consider only that part of it. So, X dot X dot is f plus g time xi, where xi dot is f a plus xi time's u, so that way. And what you are defining is, this xi dot is v and you carryout all the algebra, because (( )) v know how to do it. And ultimately once you have done with v, you can abstract u from that actually, this is by definition v. So, this is equal to v.

So, then if I solve it for u this simply algebraic equation anyway. So, I will simply (( )) and solve for u basically, once I solve for u this is this expression that I will end up get actually. And I can do that, because by assumptions g a is not equal to 0 anyway. So, that is that is anyway there actually.

So, with that I think will I mean without this this topic (( )) sufficient (( )) anybody to work with this this back-stepping idea and all that. So, you can see, one of the idea's that we use for I mean this analysis to (( )) no theory, before you have seen how it is useful for for (( )) adaptive design. Now, this class we will see, how it is useful in back-stepping design directly actually. And ideas, various ideas (( )) (( )) in the literature, where you can think of let say use that in in observer setting also like how do you design observer based on Lyapunov theory.

So, it is very generic tool, where you can think of exploiting that for for design purpose as well actually. So, with that exposure I think I will proceed to a different topic now. And then, we will rest of the time we will talk about that actually alright.

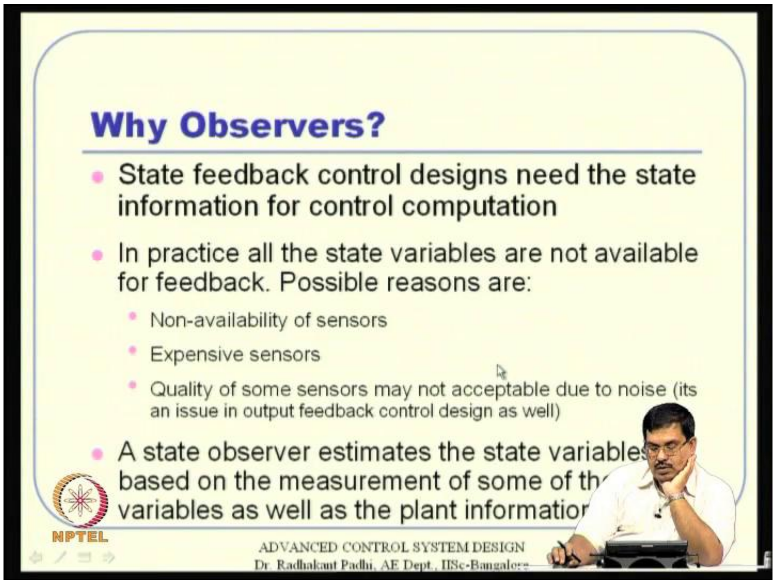
So, this this topic is linear quadratic observer (Refer Slide Time: 39:03). So, this is what will see before in that I mean sometime back is this (( )) type of observer, this is is valid for single input system and all. The moment it was multiple input and all, there is lot ambiguity confusion and all that, how do you do that. There are suggestion and all that, but then it is not very unique way of doing things.

So, this this particular observer design is it does not is not restricted to that. In other words, you can have multiple input system for the control design; and hence, for the (( )) system we have multiple output also. So, the (( )) placement type of observer is valid for only singular output system, whereas this L Q observer is valid for multiple output system directly basically.

We can have a number of measurements many basically. That is the advantage that you talk about LQ observer. Second this bigger advantage is this gives us a platform to develop towards (( )) filter which is even more elegant actually. Having said that, you can also note that this L Q observer is actually nothing but, a (( )) filter. So, a special (( )) filter sort of thing you can think about.

And then hence, even if you have this noise input in the system, then also it it is take a it is take a (( )) filtering it out actually. So, we will see that little bit (( )) of that, because I thought without this (( )) of L Q observer and little bit of (( )) filter at least, now control theory course is complete actually. Thus, most of the times where we have to use some problem of observer or filter like that actually.

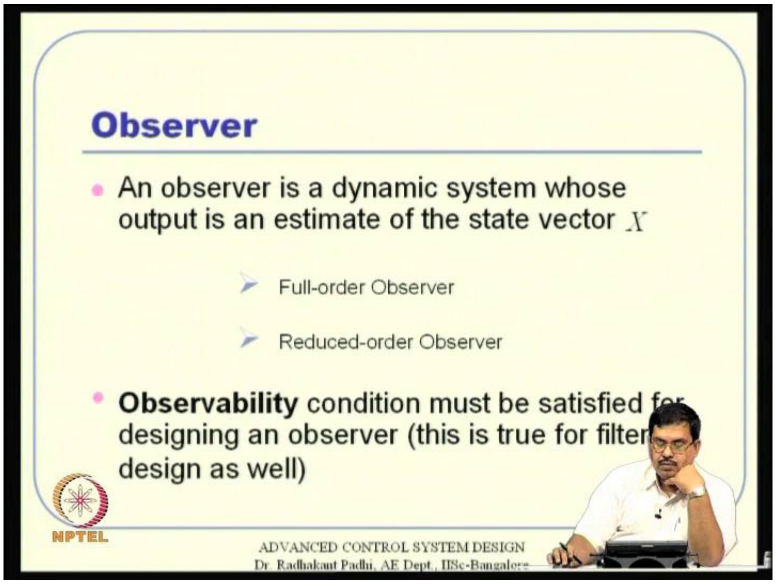
(Refer Slide Time: 40:41)



So, let us see what is going on here. So, why observers, we have seen this before that, the state feedback control designs need the state information for control computation. And in practice, all states may not be available and possible reasons can include let us say: sensors are not available, sensors can be very expensive, sensor and quality of the sensors may not be acceptable due to the noise content in that also. So, you probably if given a choice, we will not want to use that that sensor actually.

So, what do you what is the alternative? Let us say, we use limited a sensor which are of good quality anyway. And then, rest of the states we want to kind of have an estimate of that, based on those measurements in a sequential manner actually. (( )) will take measurements multiple time and from them from that information, we want to observe or we want to estimate what what should be my state and think like that actually.

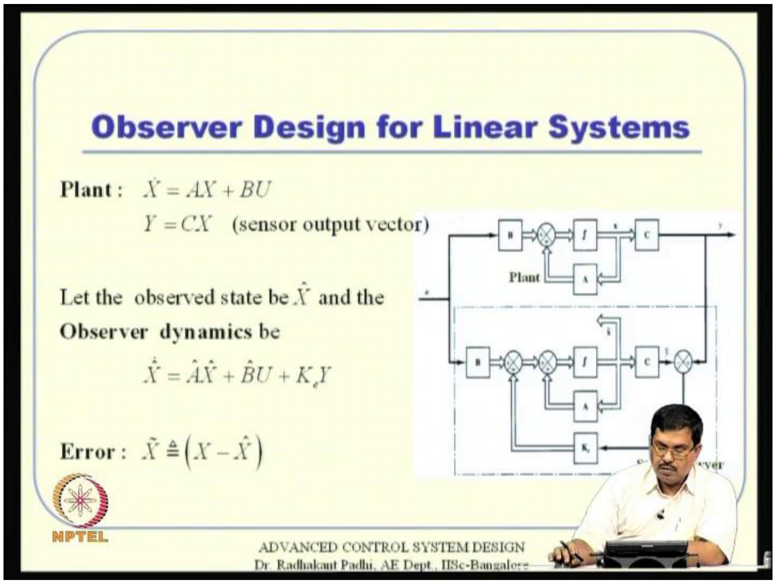
(Refer Slide Time: 41:34)



So, what is an observer? An observer is nothing but, dynamical system whose output I mean output of the observer is an estimate of the state vector X actually. So, that is what you are primarily (( )) estimate of the state vector X. Obviously, observer can be full order (( )) sort of thing like that, (( )) topics further discussion thing like that; we will not talk about that, we will simply confine our full order observer you can (( )) actually.

Now, also remember that before talking an observer, the observability condition is a must actually and this also true for filter design as well, is the system is not having observability property, then we cannot design an (( )) ok. So, that means we easily need extra additional sensors, some information that you cannot capture using limited sensor and all that actually alright. So, observability condition must be there and we are interested in full order observer and we are a interested in linear quadratic observer that is the objective here actually.

(Refer Slide Time: 42:37)

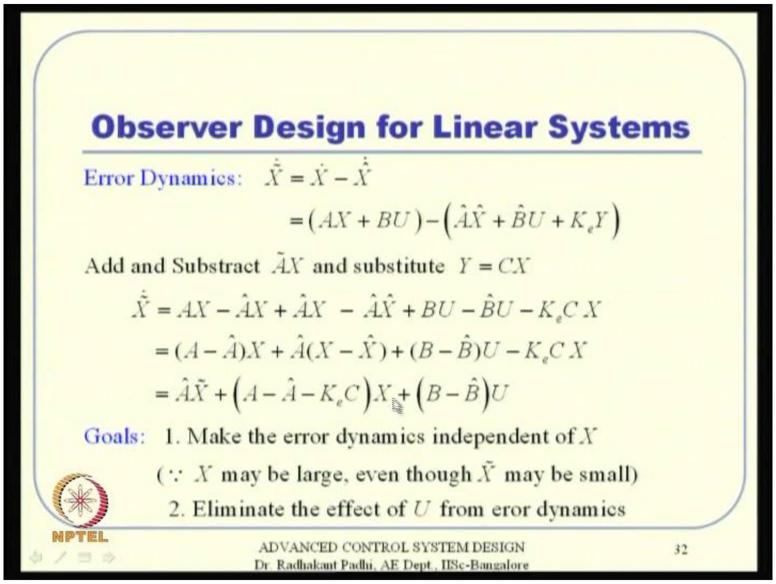


So, let us see what is what we are interested in here. So, let us plant is X dot (( )) all I am talking about linear system again (( )) actually. So, it (( )) X dot is A X plus B U and Y equal to C X, so this is nothing but, the sensor output vector. Sensor, whatever sensors you are getting that, you are writing as a linear equation, Y equal to C X.

And the let the observed state be X hat; and the observer dynamics that is what you have to we want to have an estimate; so, that is that let denote that is X hat and X hat dot is nothing but, that actually; that is almost same thing as that A X plus B U plus k e Y, but this plus K e Y is an additional term that you are interested to put in this. So, that is how will will inject sensor information into the dynamics actually.

So, what is our objective, X hat should go to (( )) as soon as possible. So, when I define this error X (( )) as X minus X hat, that means X (( )) should go to 0 basically that is thing. And remember, when you are talking about observer, we are not talking about any any noise in the system; (( )) neither process noise is there in the system dynamics nor sensor noise is there. So, in that setting (( )) talking that, only we are telling that all states I mean not available from the sensors from that we will try to estimate something actually.

(Refer Slide Time: 44:02)



So, when you define error like that, so obviously the error dynamics is also associated with that. So, X tilde dot is X dot minus X (( )) dot; the X dot is A X plus B U, X (( )) dot is that, I just put it that. And then, do this simplified algebra, you can do this add and abstract this A tilde X and and thing like that, it will end up with something like this.

Now, we have some thing whenever we have something like this obviously, we do not want to see I mean given a choice, if it is possible we do not want to see this this error dynamic is a function of original (( )) original control. So, you want to see the error dynamics as a function of error itself primarily.

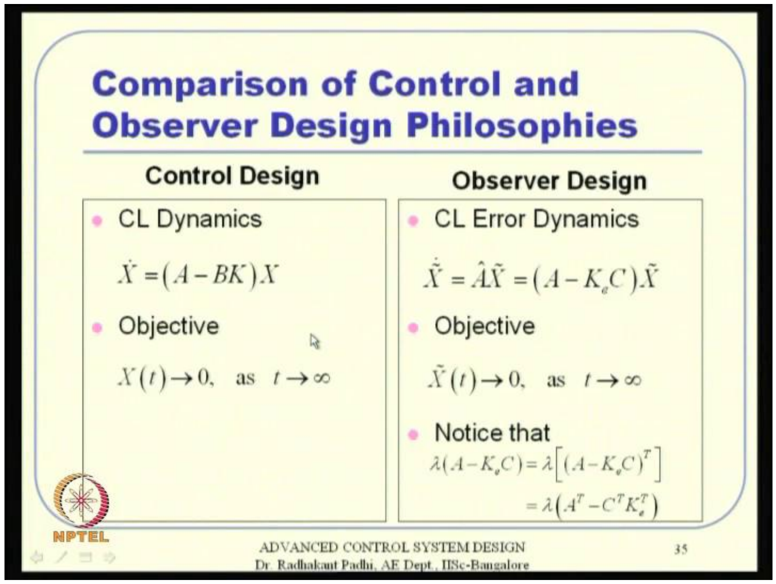
So, if you if you can do that that is the (( )) actually and because, because this coefficient are available, probably we can do that actually. Because, we are now selected what (( )), so far we have not told, what is A hat and what is B hat. So, this will actually give us platform to actually tell what is A hat, what B hat, because will will make sure that, the coefficient go to 0 that we will get B equal to B hat and A hat equal to A minus K e C; but as a by product of that as a good by product in fact, the objective of what we wanted, that means the error dynamics is a function of error itself is happening actually.

So, it is now decouple from from X and U, because X can be large and X tilde can be small; similarly, U can be large, X tilde can small there will be lot of numerical (( )) issues and all that (( )) actually. So, why **(( ))** that, so what you are telling, this coefficient have to be 0. So, the once we have this coefficient 0, this results is A hat is nothing but, A minus K e C that is what it is, and B hat is nothing but B. So, once you have this, then you go back to this system dynamics this observer dynamics and tell now, A hat is nothing but, A minus K e C and B hat is B. So, let me put it back and then once I put it back, I get something like this (Refer Slide Time: 45:52), right.

So, X (( )) dot is A X hat plus B U plus K e times Y minus C X hat, and this Y is your actual like sensor output, and C X hat is estimated sensor output; let me Y minus C X hat is a also called innovation residual and thing like that actually. This is Y minus C X hat is that that part of it is is called typically it is called innovation actually in the (( )) setup. That is that is the extra error information that you are giving to the filter dynamics or the observer dynamics to get it gets some better estimate of the state basically right.

So, what is what is your what is our interest here. Now, if you go back to the error dynamics part of it, our observe our objective was this should go to 0 right that is a primary objective, this go to 0 as soon as possible. That means y if I see this, this residual error dynamics then that error dynamic should be asymptotically stable actually.

(Refer Slide Time: 46:54)



So, that is what I will do here. And tell what is my residual error dynamic, this X tilde dot is now, A hat X hat that that is only this part is left out. And A hat is nothing but, A minus K e C. So, what I am left out with, A minus K e C actually. So, that is my error dynamics.

Now, let us go back and see, what is what we have done in the L Q control design that is Linear Quadratic control design using L Q theory and all that. There, if you design a controller in such a way that, the closed loop system dynamics happen to like this, this X dot equal to A minus B K times X.

And here, if the closed loop system dynamics, the error dynamics in fact, which is like X tilde dot equal to A minus K e C times X tilde. Here, the objective was to make sure that the X goes to 0, as t goes to infinity; here also the same objective X tilde goes to 0, as t go to infinity. So, almost very parallel what you have seen there actually, but only the difference is this K appears to be in the right hand side here; but, where this K e appears to be left hand side here actually, K e stands for let say estimated gain actually I mean gain of the estimator basically.

So, to avoid this problem what you what we note is that, this particular matrix all that we want is this is (( )) that means Eigen value of this matrix has to be all in the left hand side;

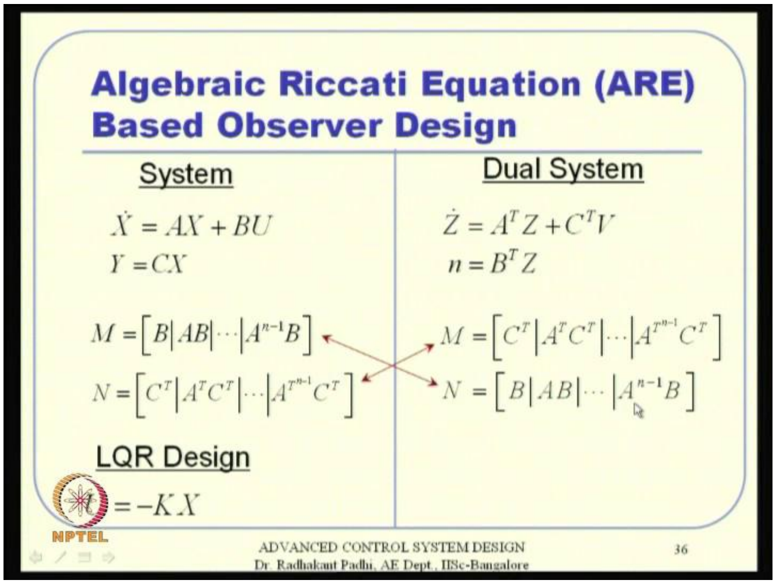
and you see, if I really I am in interested in only in Eigen values, then I can very well visualize that transpose of this matrix as well.

Now, if I visualize a transpose of this matrix this matrix is nothing but, A transpose minus this transpose will now change this order, so it will have C transpose K e transpose. So, what is what you what appears what happens I mean what appears in the left hand side now appear in the right hand side; and hence, the problem is (( )), provided we interpreted this A as A transpose here, and this B happens to be C transpose. So, A is equivalent to A transpose, B is equivalent to C transpose, then K e transpose is something that is equivalent to our kind of controller gain actually.

So, is any way that is compatible with what we call is to (( )) system and all. We have all discussed about this dual system dynamic when we are discussing about controllability observability and observer and all that actually there. Though if you are (( )) you can revise those lectures to see (( )) those actually. So, this is the observations. So, in a in control design, we have come up with something like close loop system dynamics like this, objective was like that.

Observer design that we are talking here is the close loop system dynamics is like this, objective is also like that. To make sure that, the gain appears in the right hand side, visualize the transpose problem and the transpose problem happens to be like that actually.

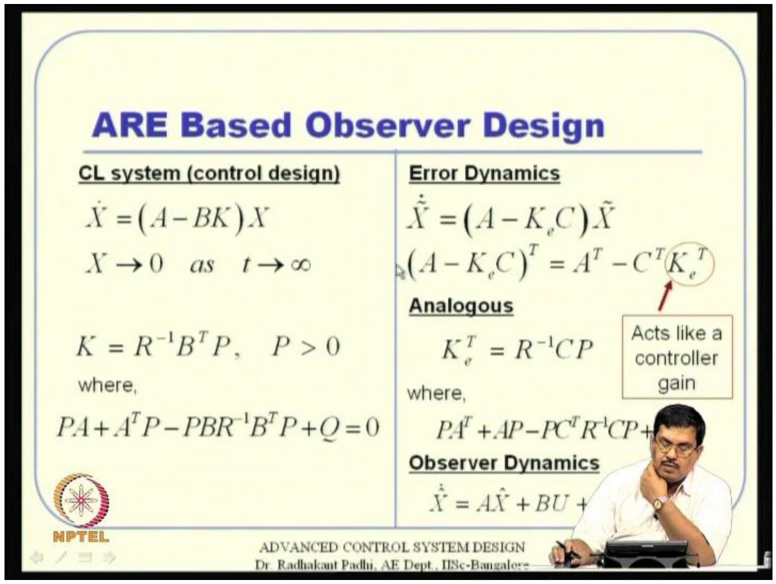
(Refer Slide Time: 49:33)



So, now we overcome see something like a like this dual system idea that is talking about. So, (( )) as a summary something like this, system is original system is like this, the dual system that we visualize is some thing like that. Now, if you see the control ability matrix and observability matrix here and similar thing here, then the control ability matrix (( )) to observe abilities here and and vice versa, observability matrix here turns out to be controllability matrix ok.

So, (( )) of designing like another observer something what it what you really do is, we design a (( )) for the dual system actually that that will be become an observer for the actual system.

(Refer Slide Time: 50:19)



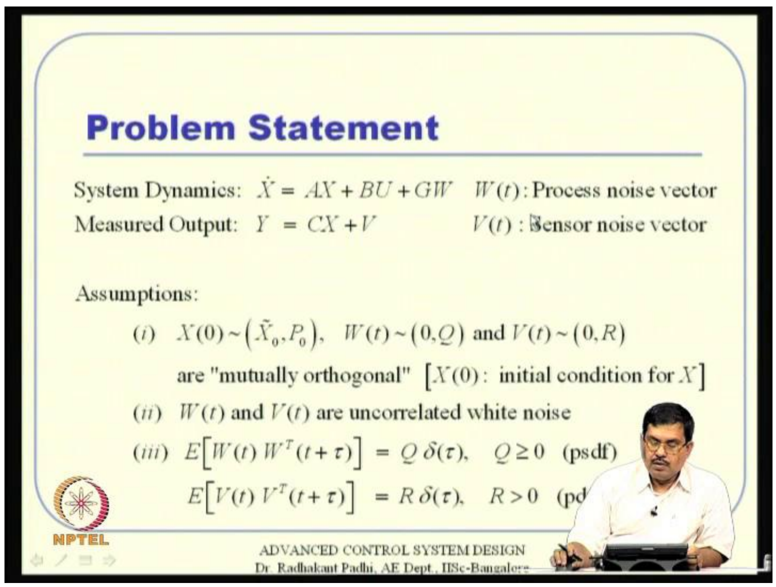
Anyway, so the L Q R designs we already know how to do it, U equal to minus K X. So, using this philosophy, you will be able to directly derive a gain for the filter I mean for the observer. So, let us go back to that, so this is closed loop system dynamics. And this gain that we are talking here was designed as something like R inverse B transpose P, where P (( )) P is positive definite matrix which comes out as a I mean which is found from the solution of the algebraic (( )) equation. So, this (( )) (( )) like that.

Similarly, the error analogues system you see here, this is like transpose and all, this acts like a controller gain as I told. And this controller gain can be directly written as something like this, remember R inverse, B transpose is equivalent to like B transpose equivalent to C, so we will put C; and this P is P, but the P is solution of observer (( )) equation not the controller (( )) equation ok.

So, you see that equivalent system whatever A transpose, C transpose, B transpose that we have, we can (( )) and then tell ok, this is what what should happen actually. So, this this start this this type of like this equation is called I mean filter regard equation or observer regard equation. So, we get a solution of this system dynamics sorry this regard equation. And then whatever solution you get you put it back that will become your K e transpose it is it is no more K e. But, if you take the transpose of that, you get K e basically, once you get K e this is your observer dynamics.

So, this observer dynamics can be propagated and as you propagate, it will it will ensure that, X tilde goes to 0; and hence, X hat will go to X actually, that is that is how it operate actually. So, this is the algebraic regard equation based observer design. So, it is nothing but, L Q observer design, a linear quadratic theory is coming into picture. And this particular idea that we just discussed can be extended to this Kalman filter design in the in the linear time invariant system case, so in continues time case actually.

(Refer Slide Time: 52:25)

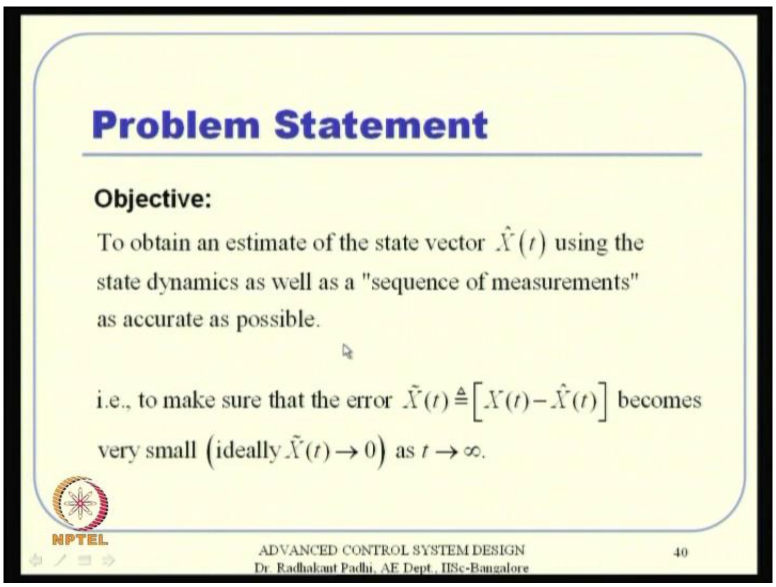


So, very quickly will see some summary sort of thing here, what it tell is system dynamics is something like this, now it is A X plus B U plus G W, and Y is nothing but, C X plus V. And this W and V are process I mean W is process noise, V is sensor noise, and they are vectors in general and thing like that.

And there are (( )) assumptions here, (( )) that this first of all, this W and V has to be like 0 mean the first is 0, that means 0 mean; and then, they have to be like mutually orthogonal, they have to white noise and thing like that. W and V are uncorrelated white noise actually they are not correlated to each other. And they are like white noise definition if you see, they are I mean this I mean there is something called this (( )) property and all that, they have to be satisfied. Lot of too much tell in this lecture, just remember that they have to be like white noise actually.

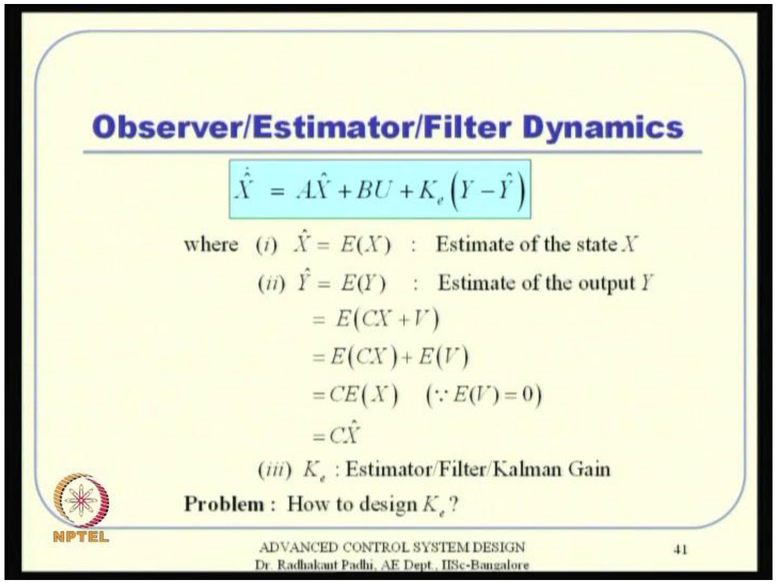
And then it also talks about like when we talk about W W transpose and then take (( )) value it will have only Q there; remember this is (( )) translated in the time actually. So, if I take only the same time, I will get some value; if I have any difference in time, it will not they will that will be 0 actually, delta delta definition is like that; so, similarly for V also there actually.

(Refer Slide Time: 53:53)



So, all this definition (( )) all this assumption where I mean with all this assumptions, we are still putting there objective that, let us estimate X hat. So, that the error goes to 0 actually as t goes to infinity. Now, can we do that, so the entire derivation I will not do it anything out of that here.

(Refer Slide Time: 54:06)

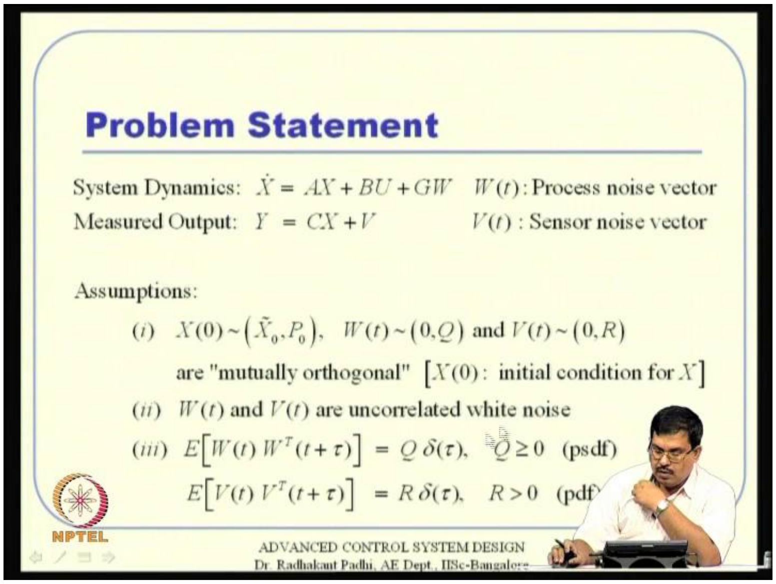


Let us, see in the summary sense this this is something very parallel to what we have done before, we will still take an estimated dynamics. And here, this K e is called Kalman gain actually or estimator gain or filter gain whatever you can talk about, exactly same as what we discussed in the in the observer actually. Whatever observer equation is there, it is same as that actually, (( )) this is X hat actually. So, that is exactly same as what you have discussed actually.

Now, you tell X hat (( )) expected (Refer Slide Time: 54:48), now X hat what we are talking actually, not the directly the same (( )) or something. Now, it will have estimate of the state itself what you called as expected value of X, expected value like like an average value for for many measurements I mean many cases and all. If you take a lot of these values and all, then take average and think average value, then that is nothing but expected value.

So all the thing that, we are discussing here is in the estimated value sense sorry expected value sense that means average value sense basically. Then similarly, the Y hat will be expected value of Y and then, when you put Y is nothing but, C X plus V and then, (( )) the expected value is linear operative.

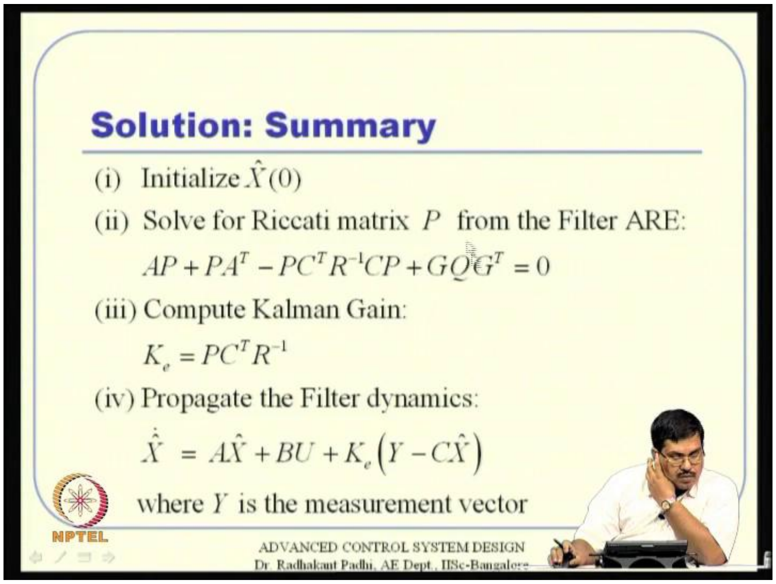
(Refer Slide Time: 55:34)



So, you can separate it out, and then C will come out, and then this is 0 by definition, the respected value this is 0 mean actually (Refer Slide Time: 55:36), both W and V are 0 mean. So, that will that will (( )) that expected value of (( )) is 0. So, we will end up with Y hat is nothing but, C X hat actually. So, whatever Y hat you are talking here nothing but, C X hat.

So, very similar to what you have done before, only difference is this X hat is expected value of straight now, is no more deterministic value.

(Refer Slide Time: 56:07)



So, the question here is we know the system I mean the filter dynamics, now how do you do design this K e and that part, I will not talk anything here, but the summary turns out to be like that. You design K e exactly like what you have done before, but this P that the Riccati matrix will be a solution of that. And if you see this, the only difference that we see here compare to this, this equation is (Refer Slide Time: 56:22), if you see this equation and this equation, the only difference is this part actually.

So, earlier there was no G, here G transpose there now it has actually. So, if you assume that, this system dynamics has G as identity, then that is nothing but, the L Q observer is nothing but, the Kalman filter with the assumption that G is identity. So, that is that is how it operates.

So, what is what is the thing here? So, we have a filter dynamics like this, which we need to propagate. So, we initialize something and start propagating, but in the but we need a (( )) value here K e, K e can be computed that way. But, in this expression we need a P, so P can be computed from this equation (Refer Slide Time: 57:05). So, you have to implemented it, then implementations (( )) has to be something like this.

You first initialize X hat 0 and and solve for this P matrix from this this (( )) equation; once you get (( )) matrix, you compute the gain filter gain. Once you get the gain, you know the filter dynamics anyway, the filter dynamics (( )) initial condition is available, so simply propagate this equation (( )) time actually. And it all make sure that, your error goes to 0 in an expected values sense actually now, (( )) mean will have to 0 and then, you will get a I mean your noise will get filter out and then (( )) that actually.

So, this is all happening if the this all happiness in continues time linear term in variant system Kalman filter theory. There (( )) much more be given that, we talk about discreet time implementation, then continuous discreet time, continues time system propagation, and discreet time measurement implementation (( )) continues discreet version; then there are extensions to make it operate for the non-linear system that is that is external Kalman filter and thing like that actually.

These are all available, but also remember that, without knowing this some of the estimation estimation theory, control design, perhaps slightly kind of incomplete actually. So, I will encourage all of you to study further in this field and then get (( )) with all that actually. So, with that I will stop this this lecture, thank you.