Optimal Control, Guidance and Estimation

Prof. Radhakant Padhi

Department of Aerospace Engineering

Indian Institute of Science, Bangalore

Module No. # 06

Lecture No. # 14

Discrete-time Optimal Control

(Refer Slide Time: 00:19)

Optimal Control, Guidance and Estimation

Lecture – 14

Discrete-time Optimal Control

Prof. Radhakant Padhi

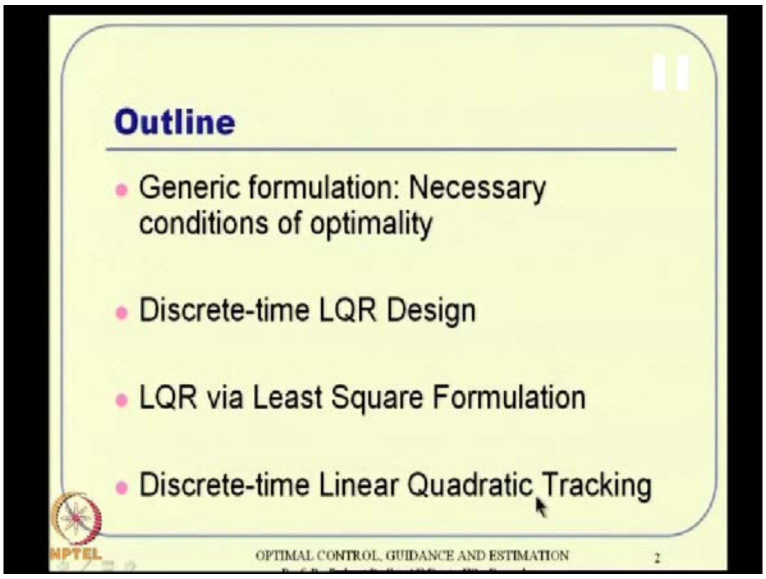
Dept. of Aerospace Engineering

Indian Institute of Science - Bangalore

Hello every one. We will continue our lectures in this optimal control guidance and estimation course. And this is a different topic, somewhat related, and we need that sometime, and hence I thought I will cover one lecture at least on this topic. It is called discrete time optimal control. So far, you have seen everything in the continuous time; I mean all the time we talk about x dot equal to f of x; U and lambda dot and thing like that. And many times discrete time formulation also has some beauty, and you can design algorithms in a better way and think like that way. In fact, some of our algorithms later that we discuss in this course, we will actually relay on this discrete time optimal control.

So, this particular lecture, I will just give you some sort of an overview of that, including generic theory, followed by L Q R in discrete time as well actually. So, that is the motivation for this particular lecture.

(Refer Slide Time: 01:11)



Outline

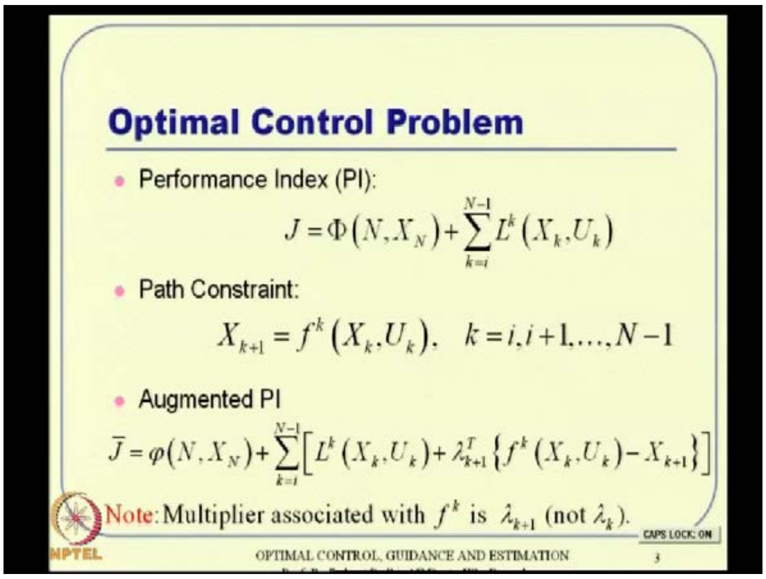
- Generic formulation: Necessary conditions of optimality
- Discrete-time LQR Design
- LQR via Least Square Formulation
- Discrete-time Linear Quadratic Tracking

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 2

The outline happens to be something like this; first is generic formulation, and essentially again it will lead us to the, this necessary conditions of optimality. Then, one example we will follow, and then will move it to some discrete time L Q R design, and for your information, this matlab also has a D L Q R function; D L Q R stands for discrete time L Q R actually. Then on the way we will also see an idea, one of the very earlier ideas; how people thought of solving an L Q R formulation, using discrete L Q R sort of ideas in least square setting basically. So, you convert the entire formulation, take the discrete grid points in time, and all the state vectors are equivalently converted to some sort of a control vector and all that actually, then an initial condition of course, where initial condition happens to be just a type. I mean just a number; it does not play a role in terms of minimization, maximization like that and think like that.

So, I will take you through that, when in the appropriate slide actually. Then we will also have some sort of a discussion on discrete time L Q tracking problem. So, far we have been talking about regulator all that L Q R, R stands for regulator, but if you really want to extend it for tracking applications, this is called something like linear quadratic tracking problems. Similar concepts are available in continuous time as well, and we have actually talked a little bit about that, I will talk more in the s d r e setting, when we talk about s d r e lecture as well. But here, we talk about discrete time L Q R, discrete time L Q tracking issues; a tracking problem, how will you formulate, and how will you get a solution like that actually. So, this is what is coming up in this lecture basically.

(Refer Slide Time: 03:01)



Optimal Control Problem

- Performance Index (PI):

$$J = \Phi(N, X_N) + \sum_{k=i}^{N-1} L^k(X_k, U_k)$$

- Path Constraint:

$$X_{k+1} = f^k(X_k, U_k), \quad k = i, i+1, \dots, N-1$$

- Augmented PI

$$\bar{J} = \varphi(N, X_N) + \sum_{k=i}^{N-1}\left[L^k(X_k, U_k) + \lambda_{k+1}^T\left\{f^k(X_k, U_k) - X_{k+1}\right\}\right]$$

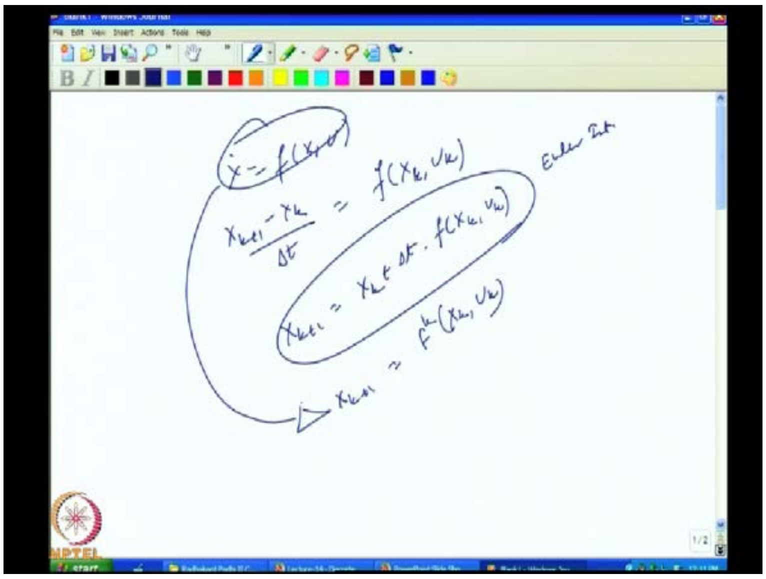
Note: Multiplier associated with f^k is λ_{k+1} (not λ_k).

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 3

So, here is equivalent representation of optimal control problem in discrete time. A performance index happens to be something like this, and this is the terminal penalty and this is the path penalty sort of thing. And when we talk about discrete time, there is no specific advantage of having a linear time invariant or time varying system. So, everywhere we will consider as much as possible, these are all time varying things actually. So, L can be a constant function, but L can be different at different points of time also, that is why this L k is represented .Here k stands for k varies from i initial condition i+1, i+2 like that up to N-1.

And actually it will lead to this X n in system dynamic equation. this is the system dynamics, so X k + 1 is a function of X k U k. most of the time we will see that L and f these are all static function, they are functions of X and U, but the function itself do not change the type, but as I told in discrete time, that has no specific advantage. So, for generality, for more generic results and all that, we will consider that, f can be a function of time as well; it means the very nature of f can differ from time to time actually. Similarly, L can be I mean L can have time dependence as well in that way. Anyway this is your performance index. This is the path constraint and all happening in terms of discrete time actually. And very easily if somebody wants to really see what is and how it can be done and all that.

(Refer Slide Time: 04:46)



$$\dot{x} = f(x, U)$$

$$\frac{x_{k+1} - x_k}{\Delta t} = f(x_k, U_k)$$

$$x_{k+1} = x_k + \Delta t \cdot f(x_k, U_k)$$

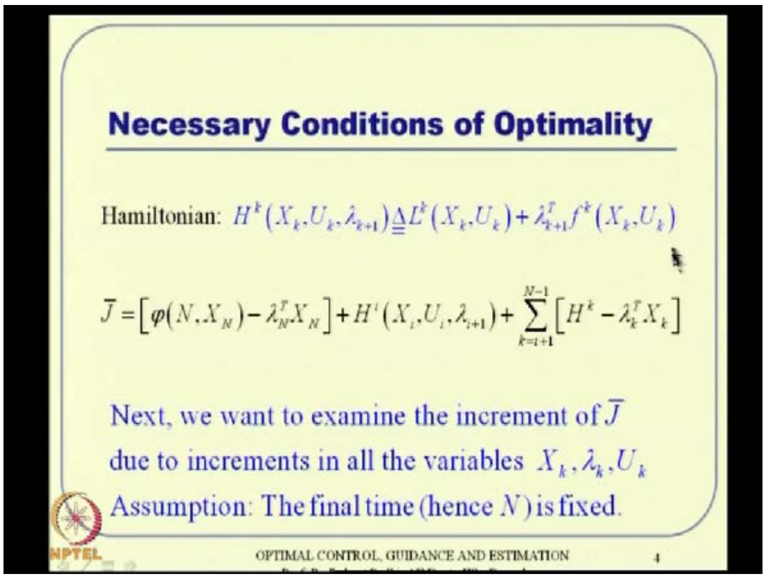
Euler Int.

$$x_{k+1} = f^k(x_k, U_k)$$

This can be done something like, if you have x dot equal to f of x U .Then you can do this X k plus 1 minus X k by delta t. This is the approximation of x dot and the right hand side is x k U k. and then you can write x k plus 1 is nothing, but x k plus delta T times f of x k U k. So, that is nothing but, this is Euler’s integration actually. So, if you use Euler integration formula, then you can see that x k plus 1 is nothing but, some function of f k of which is nothing, but x k U k assuming delta T to be constant. This is how, from a continuous time representation, you will get a discrete time representation actually. And you can use various integration formulas to get there as well this. So, this is what I was talking actually, so we have a. Similarly, you can discretize any continuous time cost function as well, using this trapezoidal rule and all that we can do that, and write it an equivalent cross function or a continuous time problem as well. So, this is our discrete cross function, and this is our discrete straight equation. So, with respect to these two, the augmented performance index can be written something like this.

J bar is nothing but, this part plus summation of all that, and this is what earlier we did here is, lambda T transpose times f minus x dot, if you remember that in continuous time. But, here in discrete time what you write here is, lambda k plus 1 transpose times; this one minus that one; that is f k minus x k plus 1 is nothing but zero, that part is we put here actually. And just to comment that multiplied associated with f k, here f k minus f k plus 1, this is what is getting multiplied, is lambda k plus 1, it is not lambda k; that is the one of the major differences really. And that is done typically to get some nice results later basically. So, that means, if you see in optimal control equivalence and all that actually that way. then x k is equivalent, I mean x of T is equivalent to x of k, U of T is equivalent to U of k, but lambda of T is equivalent to lambda of k plus 1, it is not k, that is the difference from continuous time to discrete time. Anyway, this is our augmented cross function.

(Refer Slide Time:07:25)



Necessary Conditions of Optimality

Hamiltonian: $H^k(X_k, U_k, \lambda_{k+1}) \underline{\underline{\Delta}} L^k(X_k, U_k) + \lambda_{k+1}^T f^k(X_k, U_k)$

$$\bar{J} = \left[\varphi(N, X_N) - \lambda_N^T X_N\right] + H^i(X_i, U_i, \lambda_{i+1}) + \sum_{k=i+1}^{N-1}\left[H^k - \lambda_k^T X_k\right]$$

Next, we want to examine the increment of $\bar{J}$ due to increments in all the variables X_k, λ_k, U_k

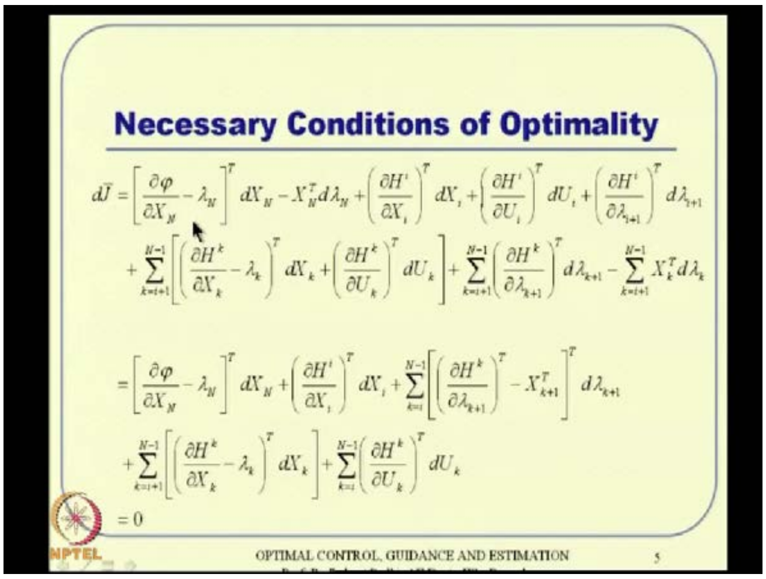
Assumption: The final time (hence N) is fixed.

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 4

And then we will follow with our necessary conditions of optimality and all that, so far. First you define a Hamiltonian H k is nothing, but L plus lambda transpose a. So, L is nothing but, L k plus lambda transpose this part, actually this part. Whatever we have it here; that is what is represented there actually; a Hamiltonian H k, even it is a function of H k U k and lambda k plus 1, and define something like this. Now with respect to this definition J bar can be written something like this. And here it is not from i ten minus 1, but we take it out from I plus 1 to n minus 1, and write the i separately basically. This is just a multiplication, I mean manipulation sort of thing. So this one, i to i plus 1 to n minus 1, whatever we have here. The very last term will also turn out to be lambda n transpose n x n; that also is represented, that also is separately written actually here. The very first one is taken out, and the very last one is also taken out.

Last one is here and first one is here, in between terms are written something like this. Next, we want to examine the increment of J bar, due to increments in all variables X k lambda k and U k. and here the assumption again is the final time, and that is in discrete notation, it is nothing but N, N is fixed actually, that is the definition, I mean that is the assumption here. So, we are talking about f x final time problems sort of thing. Anyways this is J bar, which is given this one. By the way, I have taken all these material from Frank Louis book actually. So, if somebody is interested, they can use this Frank Louis book, as a different version of the book and all that, but I have used it for some of his earlier books on optimal control actually. Anyway, this is what it is J bar is something like this actually. Now we are ready to apply our optimization conditions and all that.

(Refer Slide Time: 09:24)



Necessary Conditions of Optimality

$$d\bar{J} = \left[\frac{\partial \varphi}{\partial X_N} - \lambda_N\right]^T dX_N - X_N^T d\lambda_N + \left(\frac{\partial H^i}{\partial X_i}\right)^T dX_i + \left(\frac{\partial H^i}{\partial U_i}\right)^T dU_i + \left(\frac{\partial H^i}{\partial \lambda_{i+1}}\right)^T d\lambda_{i+1}$$
$$+ \sum_{k=i+1}^{N-1}\left[\left(\frac{\partial H^k}{\partial X_k} - \lambda_k\right)^T dX_k + \left(\frac{\partial H^k}{\partial U_k}\right)^T dU_k\right] + \sum_{k=i+1}^{N-1}\left(\frac{\partial H^k}{\partial \lambda_{k+1}}\right)^T d\lambda_{k+1} - \sum_{k=i+1}^{N-1} X_k^T d\lambda_k$$

$$= \left[\frac{\partial \varphi}{\partial X_N} - \lambda_N\right]^T dX_N + \left(\frac{\partial H^i}{\partial X_i}\right)^T dX_i + \sum_{k=i}^{N-1}\left[\left(\frac{\partial H^k}{\partial \lambda_{k+1}}\right)^T - X_{k+1}^T\right]^T d\lambda_{k+1}$$
$$+ \sum_{k=i+1}^{N-1}\left[\left(\frac{\partial H^k}{\partial X_k} - \lambda_k\right)^T dX_k\right] + \sum_{k=i}^{N-1}\left(\frac{\partial H^k}{\partial U_k}\right)^T dU_k$$
$$= 0$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 5

So, what you are interested to see, is the first deviation of J bar is equal to zero. (()) is no variations now, there are discreet variables actually. So we talk about first deviation of J bar has to be equal to zero. Hence you are interested to see what expressively takes first deviation of J bar. So, d j bar if you talk about this expression happens, because of several partial derivations actually. partial derivation of J bar d J bar happens, because there can be a partial derivation of d X n, I mean X n that is d X n, there can be something like d lambda n. there can be something like d X I, d U y, d lambda I plus 1 or sort of thing actually. So, you have to see carefully this expression, what all is a function of, I mean J bar is function of what all variables, and then keep on doing this partial derivative sort of thing and then write this expression actually. So, to begin with, let me explain one or two example one or two expressions. If you see this expression especially, the perturbation of J bar can come through the perturbation of either x n or lambda n actually.

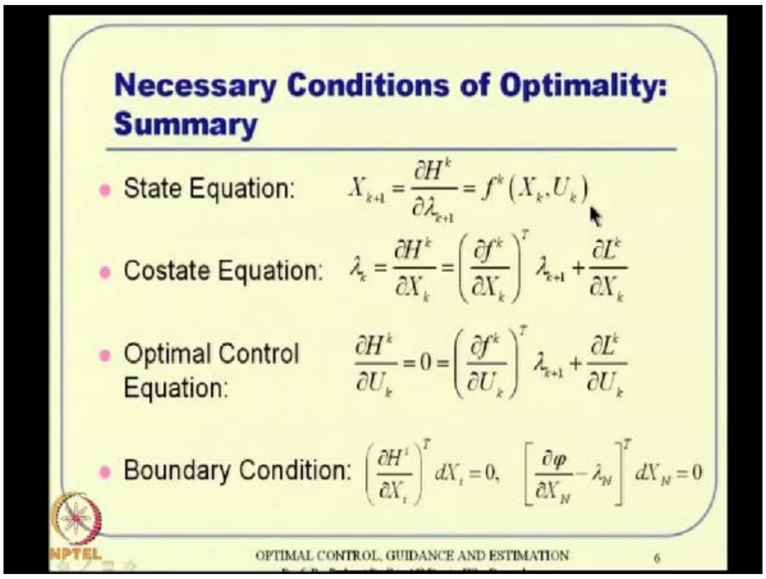
So, this particular term, we write del of this term, del y del X n into d X n, plus del y del phi, I mean del y del lambda n into d lambda n of this square bracket sort of thing. So, if you take this expression, you write it. This is what we are analyzing actually. So, first I will write this expression something like perturbation with respect to x n, then perturbation with respect to lambda n, that what we are trying to do. The first term is nothing but this, because these entire term del y del x n of entire term is nothing, but del phi by del x n minus lambda n, this one, transverse time d X n, plus this del y by lambda n which is minus identity .For this does not contain any lambda n only this one contains lambda n. So, del by n lambda n into T lambda n, that will give us this term actually X n transpose times d lambda n. then, we will come to the next expression is a function of x I U i lambda i. So, it can the perturbation can come from all, I mean terms you can write del H i divided by X i transpose time d X i plus del H i by del U y transpose time d U i, plus del H i by del lambda i plus 1 transpose time d lambda i .They are simply book keeping actually.

You observe that this is a function of x n and lambda n. Hence, we took the partial derivative that way. We observe that, this is a function, this function can be a function of X i U i and lambda i plus 1.So, you would do these three expressions, that is also that. and then coming to the last expression now remember H k can be a function of everything else X k U k lambda k plus 1 also basically. So, you account for that to we take all the partial derivatives like that, and then express in the in the form of x like this actually remember ,this one will give us perturbation with respect to lambda k also, this will give something like X k is a function of lambda k plus 1. So there part of perturbation come from lambda k plus 1, because of this term, and there is perturbation lambda k because of this term actually.

You just nothing, but a very careful book keeping, of what all things can happen from first derivations actually. Once you write it carefully, then it is time to kind of put them together, wherever its relevance things and all at there. So, then I keep the first term as it is, and then this one whatever you see here as it is like that actually. Wherever possible I can combine that now, and I can play around with this again; there is a k equal to i plus 1 to n n minus 1 here, it is i to n minus 1; that means, wherever you see i terms if possible I can include that within this term here also actually. If I separately write k equal to I outside, then whatever terms are there, I can bring it inside, by changing documentation from i plus 1 to i here. So, that careful notation you can see.

Here it is i plus 1 and here it is J basically, that will observe two terms actually. similarly, you can have a book keeping and then tell this will be like this form, and that will be that form, where I strongly suggest that all of you, actually at least do one time this algebra, it is fun to do and it is good to do and it will give you lot of confidence also, what you are doing actually. So, for the sake of time I will not explain term by term, this algebra is correct actually. So, this is to be equal to 0, with respect to all possible deviations actually; that means, as before all the coefficient have to go to zero. Now, if you notice that, then all the coefficient happens to 0.So, what about this. this will give us nothing, but the state equation, del x k plus 1 is nothing, but del x k by del lambda k plus 1, and what is del lambda del x k by dell lambda k plus 1, this is only term, so that is f k. what you are talking is x k plus 1 is equal to f k, that is nothing, but the state equation, that is how you get it there.

(Refer Slide Time: 14:33)



Necessary Conditions of Optimality: Summary

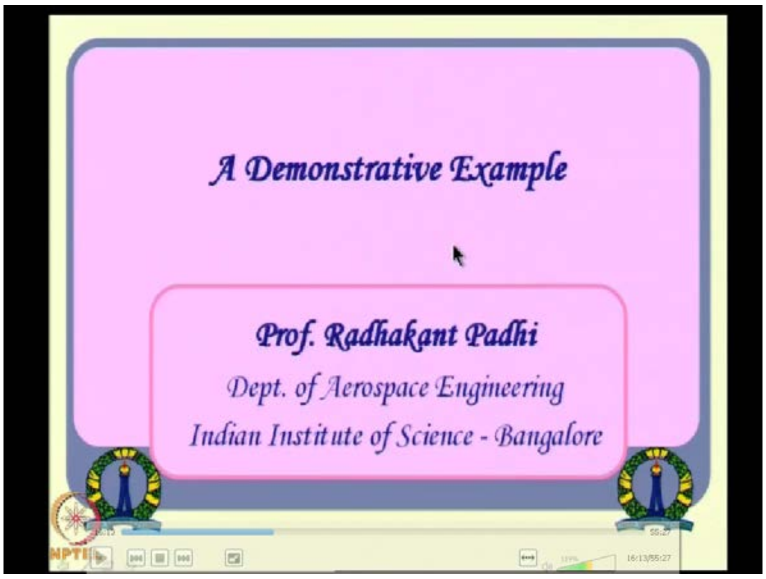
- State Equation: $X_{k+1} = \frac{\partial H^k}{\partial \lambda_{k+1}} = f^k(X_k, U_k)$
- Costate Equation: $\lambda_k = \frac{\partial H^k}{\partial X_k} = \left(\frac{\partial f^k}{\partial X_k}\right)^T \lambda_{k+1} + \frac{\partial L^k}{\partial X_k}$
- Optimal Control Equation: $\frac{\partial H^k}{\partial U_k} = 0 = \left(\frac{\partial f^k}{\partial U_k}\right)^T \lambda_{k+1} + \frac{\partial L^k}{\partial U_k}$
- Boundary Condition: $\left(\frac{\partial H^i}{\partial X_i}\right)^T dX_i = 0, \quad \left[\frac{\partial \varphi}{\partial X_N} - \lambda_N\right]^T dX_N = 0$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 6

Similarly, for the costate equation you see this expression actually, lambda k is nothing but, del x by del lambda del x k basically. So, that is what we will get costate one. Now, optimal control you will get from here, del x k by del U k equal to 0, and then boundary condition happens to be from this expression actually, here lambda n del phi by del x n this happens to be there anyway. So, there are the set of equations that we will have to note. You can carefully note that, what you what you heard earlier, is lambda dot is minus del H by del x, and here it is lambda k equal to del H by del x. So, that means, to retain this, that lambda k plus 1 taking, I mean taking lambda k plus 1 here, help such an error there actually.

If you take lambda k, then it the expression will be something different, that is the beauty that some people agree, that there is systematic representation of that actually that way; that is one thing. Second thing is lambda dot happen to be minus del H by del x, here it is plus, this is not minus actually. All other things are similar, but these observations of this costate equation should be noted very carefully. this state equation very similar to what we had; optimal control, what to very similar to what we had, and boundary condition also very similar to what we had actually .But, the costate equation happens to be slightly different, that is what it should noted careful actually. And again, I emphasize lambda of T equivalent meaning, comes through lambda of k plus 1 not lambda k, that we should never forget actually.

(Refer Slide Time: 16:13)



A Demonstrative Example

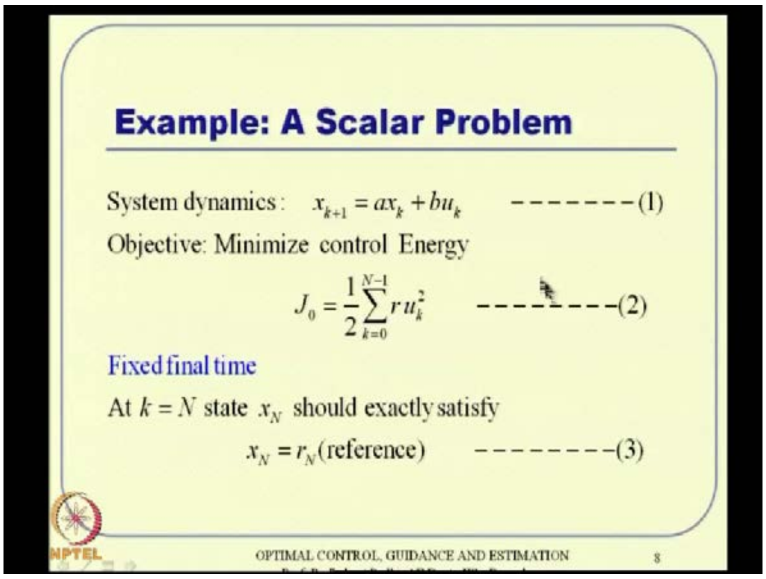
Prof. Radhakant Padhi

Dept. of Aerospace Engineering

Indian Institute of Science - Bangalore

Alright, so this is how the necessary conditions, and let us have seven examples to demonstrate, how to use in necessary condition and all that actually.

(Refer Slide Time: 16:22)



Example: A Scalar Problem

System dynamics: $x_{k+1} = a x_k + b u_k$ --------(1)

Objective: Minimize control Energy

$$J_0 = \frac{1}{2}\sum_{k=0}^{N-1} r u_k^2 \quad \text{--------(2)}$$

Fixed final time

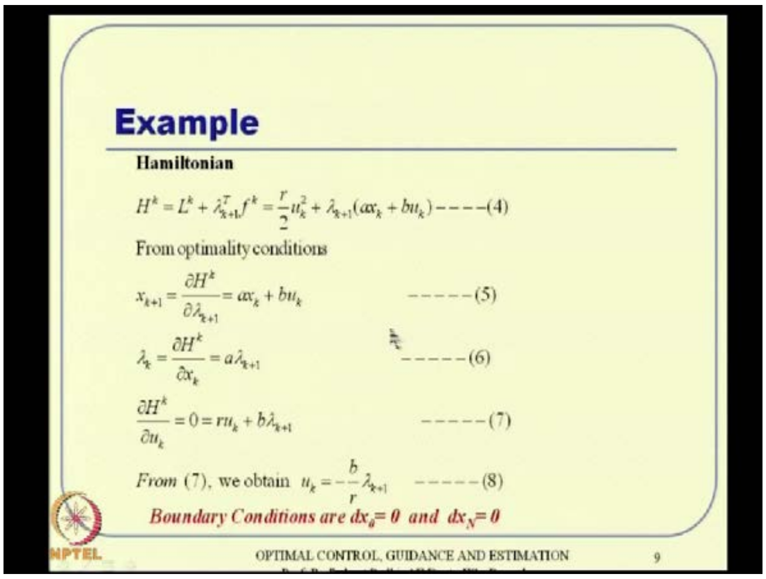
At $k = N$ state x_N should exactly satisfy

$$x_N = r_N(\text{reference}) \quad \text{--------(3)}$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 8

So we start with, we will take a linear system with a quadratic cost function, essentially; it is a L Q R problem you can think about that, but it is a, kind of a hard constraint problem. in other words even though it is a scalar problem and what you are demanding in that, that k equal to n final time, x n should be equal to r n actually. So, this is the system dynamic, I mean this is the problem. We have a linear system dynamic, and we have kind of coefficient. Objective is to minimize the control energy, subject to this final condition; x n has to go to r n. There is no choice for other things; on the way it is to minimize the control list actually.

(Refer Slide Time: 17:05)



Example

Hamiltonian

$$H^k = L^k + \lambda_{k+1}^T f^k = \frac{r}{2}u_k^2 + \lambda_{k+1}(a x_k + b u_k) \quad \text{----(4)}$$

From optimality conditions

$$x_{k+1} = \frac{\partial H^k}{\partial \lambda_{k+1}} = a x_k + b u_k \quad \text{------(5)}$$

$$\lambda_k = \frac{\partial H^k}{\partial x_k} = a \lambda_{k+1} \quad \text{------(6)}$$

$$\frac{\partial H^k}{\partial u_k} = 0 = r u_k + b \lambda_{k+1} \quad \text{------(7)}$$

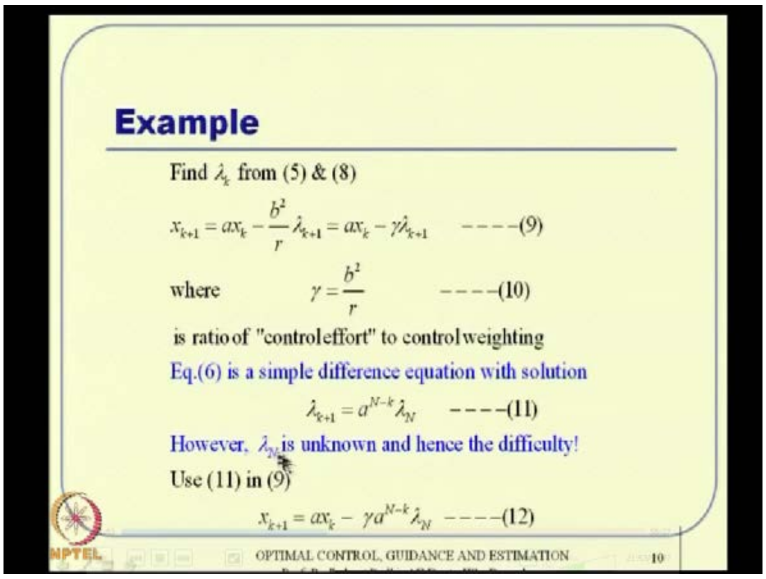
From (7), we obtain $u_k = -\frac{b}{r}\lambda_{k+1}$ ------(8)

Boundary Conditions are $dx_0 = 0$ ***and*** $dx_N = 0$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 9

So, with that formulation you will proceed to the definition of Hamiltonian, H k is nothing but, L k plus lambda k plus 1 transpose times f k, where L k you can substitute ,what is that half of r times U k square. So, half of r times U k square, plus lambda k plus 1. There is no transpose, because it is only single equation anyway here. So, lambda k plus 1 into a x k plus b U k; that is from the system dynamics, that is definition part of it. So, from optimality condition, we have this state costate in optimal control equations. So, state equation is given like x k plus 1 is a x k plus b U k, same states equation that we started with, actually same thing. And the second equation is lambda; I mean costate equation lambda k is nothing but, del H k by del x k, which is equal to a times lambda k plus 1. And let us give del U k happens to be something like this actually, del H k you know, H k right H k is like that. So, del H k by del U k we will see one term will come from here, which is r times U k plus 1 term will come from here, which is nothing, but b times lambda k plus 1. So, essentially is very close to what we know in continuous time, but the variables are slightly different and things like that actually. So, if we equate this, if we solve this equation 0, equal to r U k plus b lambda k plus 1, then U k happens to be something like minus b by r into lambda k plus 1 actually.

(Refer Slide Time: 19:14)



Example

Find λ_k from (5) & (8)

$$x_{k+1} = ax_k - \frac{b^2}{r}\lambda_{k+1} = ax_k - \gamma\lambda_{k+1} \quad ----(9)$$

where $\gamma = \frac{b^2}{r}$ ----(10)

is ratio of "control effort" to control weighting

Eq.(6) is a simple difference equation with solution

$$\lambda_{k+1} = a^{N-k}\lambda_N \quad ----(11)$$

However, λ_N is unknown and hence the difficulty!

Use (11) in (9)

$$x_{k+1} = ax_k - \gamma a^{N-k}\lambda_N \quad ----(12)$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 10

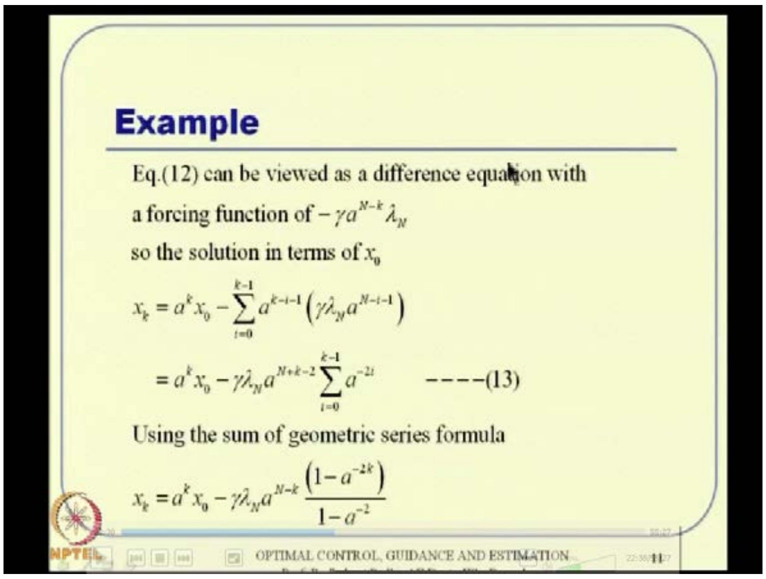
So, very close to what we know before, this kind of problems we have seen examples also, but the only difference is it is not lambda t, but its replace is lambda k plus 1 actually. So, as long as we know lambda k plus 1, we have got the controller actually U k. The question is how do you know lambda k plus 1, and also the boundary conditions, since what you are having; you have a initial boundary condition, initial condition which is fixed, fixed is a number sort of things d x not a 0. And final condition is also fixed x n as to be equal to r n. So, there is no deviation on that; that d x n is to be 0. So, these two happens to be the boundary condition, that of x on this problem actually. So, what is u, why to go ahead and solve it. So, what we do first is you find lambda k from 5 and 8 .So, I mean whatever this equation are there, now we try to solve it actually. So, what you are having is U k is there. So, this U k expression, you substitute here, so write the state equation like this basically; just for a sake of simplicity that b square by r you define gamma and then write into that way, just for the sake of simplicity sort of thing.

Now, that is fine that is the state equation, but we cannot really go ahead and solve it this equation, because the x k is left x k plus 1 is left hand side and right hand side lambda k plus 1 and thing like that. But what happens here, this particular problem, this equation happens to be independent of state actually, this itself contain sort of thing, lambda k is nothing, but a times lambda k plus 1.So, we can solve it, because it is a, I mean difference equation, involving only lambda basically. So, you can solve this difference equation; remember it is difference equation, not differential equation actually. If this difference has a solution like that, it is very easy to derive also. in other words, if you know this recursive relation sort of thing, you have lambda k is given something like this; that means, if I just start from there, which is nothing, but lambda let us say n minus 1 is nothing but, a times lambda n using this relationship what I am talking here.

So similarly, if I tell lambda n minus two is nothing, but a into lambda n minus 1, which is nothing, but a square times lambda n minus x 1. similarly, I can write lambda n minus three is nothing, but a q times lambda n minus 1 lambda, where is that, a q times lambda n basically. Sorry this is another mistake also there, minus one should not be there; alright, so this is what it is actually. We start with this expression then you can write lambda n minus 1 nothing, but lambda n and lambda n minus two is a times lambda n minus 1 but, n minus 1 is again that I can substitute, it happens to be a square lambda n. similarly, lambda n minus 3 happens to be a q lambda n things like that. So, as in general, I can write a solution lambda k plus 1 is nothing, but that kind of thing actually.

But, this solution is ok, but the problem here is lambda n is unknown, and hence the difficult actually. Lambda, remember lambda n is not known to us actually, otherwise you could have got it and solve it actually. Anyway, so now what you do is, you have this expression, now we substituted back here, this expression is available. So, lambda k plus 1 is here, we substituted back; we get it something like this actually. Now, what happens, now lambda n, we do not know correct does not a problem, but lambda n is not a dynamic variable, it is a number sort of thing actually. It is some sort of number, even though we do not know, it is not a changing number sort of thing; it is a fixed number. So, we can interpret these difference equations; state equation, as some sort of a difference equation with a forcing function sort of thing.

(Refer Slide Time: 22:38)



Example

Eq.(12) can be viewed as a difference equation with a forcing function of $-\gamma a^{N-k}\lambda_N$

so the solution in terms of x_0

$$x_k = a^k x_0 - \sum_{i=0}^{k-1} a^{k-i-1}\left(\gamma\lambda_N a^{N-i-1}\right)$$

$$= a^k x_0 - \gamma\lambda_N a^{N+k-2}\sum_{i=0}^{k-1} a^{-2i} \qquad ----(13)$$

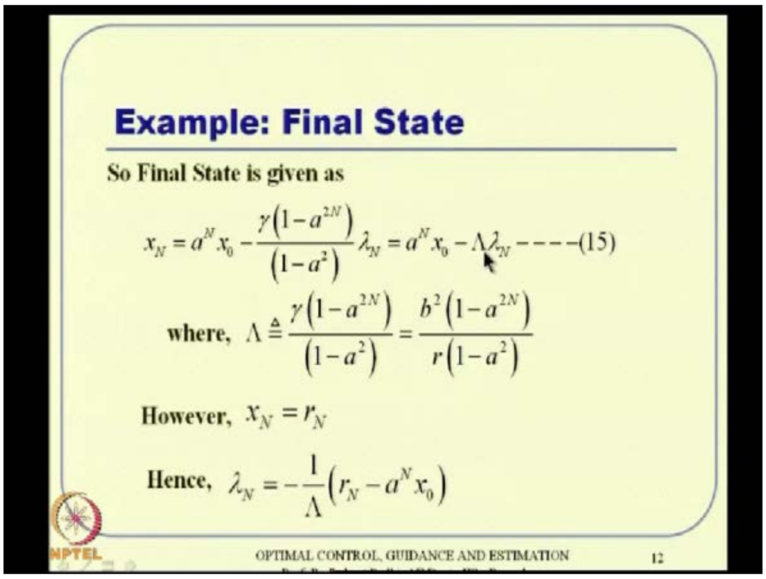
Using the sum of geometric series formula

$$x_k = a^k x_0 - \gamma\lambda_N a^{N-k}\frac{\left(1-a^{-2k}\right)}{1-a^{-2}}$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 11

So, this is right. So, with the forcing function being this one. So, it is a difference equation with a forcing function coming from this actually. So, this equation we can have a solution if you interested you can derive it yourself or you can see some sort of a difference equations math books and all that, we will tell you very clearly, that if you have a forcing function like this in a difference equation, then the solution turns out to be like that actually; remember lambda is still unknown, the solution form we know now basically. So, this turns out to be, you can write is something like this. And then you have this coefficient geometric series sort of thing. once, you have this kind of a submission term is nothing, but one plus 1 by a square plus 1 by a fourth like that actually. So, this term as a geometric formula, so using that; this series formula I can write it this way. So, I got some sort of a solution for the state, the only problem still here is lambda n is not known. So x n, if we get a x k like that, I can also write x n equal to something.

(Refer Slide Time: 23:48)



Example: Final State

So Final State is given as

$$x_N = a^N x_0 - \frac{\gamma\left(1-a^{2N}\right)}{\left(1-a^2\right)}\lambda_N = a^N x_0 - \Lambda\lambda_N \quad ----(15)$$

where, $\Lambda \triangleq \frac{\gamma\left(1-a^{2N}\right)}{\left(1-a^2\right)} = \frac{b^2\left(1-a^{2N}\right)}{r\left(1-a^2\right)}$

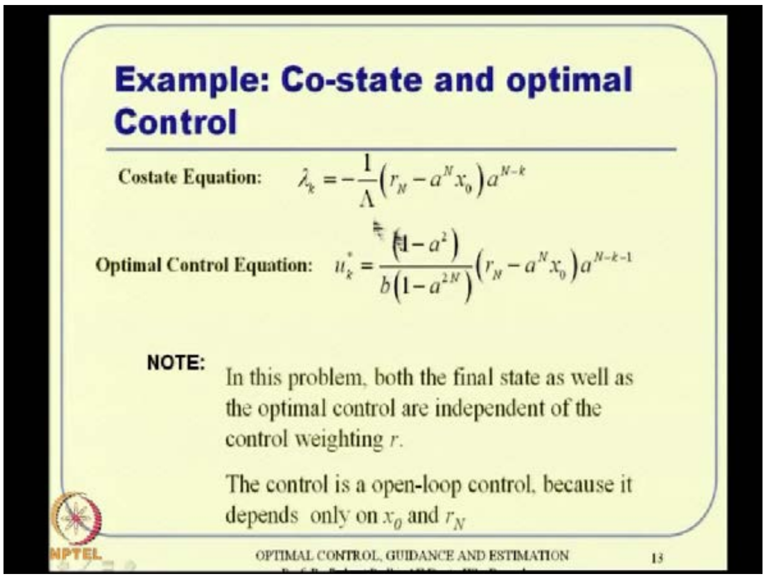
However, $x_N = r_N$

Hence, $\lambda_N = -\frac{1}{\Lambda}\left(r_N - a^N x_0\right)$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 12

And this expression I can define some terms and all that. So, that I can write in a simplified terms actually. So, x n is nothing, but some coefficient times x not minus some other coefficient time lambda n actually. And here is the case, now what we know in the problem definition; x n as to equal to r n. So now, we have a formulation, have a solution, which talks about something into x not which we know, and something into lambda n which you do not know. However, x n has to be equal to r n, that the constraint. So, if we have to put r n here, then you can actually solve for lambda n, that is what it is for here. If I substitute x n is r n here, and then solve for lambda n from this equation, then that is what I get actually; now we got a solution for lambda n. Once we get a solution your costate equation is well defined now, your costate equation happens to be this expression.

(Refer Slide Time: 24:47)



Example: Co-state and optimal Control

Costate Equation: $\lambda_k = -\frac{1}{\Lambda}\left(r_N - a^N x_0\right)a^{N-k}$

Optimal Control Equation: $u_k^* = \frac{\left(1-a^2\right)}{b\left(1-a^{2N}\right)}\left(r_N - a^N x_0\right)a^{N-k-1}$

NOTE: In this problem, both the final state as well as the optimal control are independent of the control weighting r.

The control is a open-loop control, because it depends only on x_0 and r_N

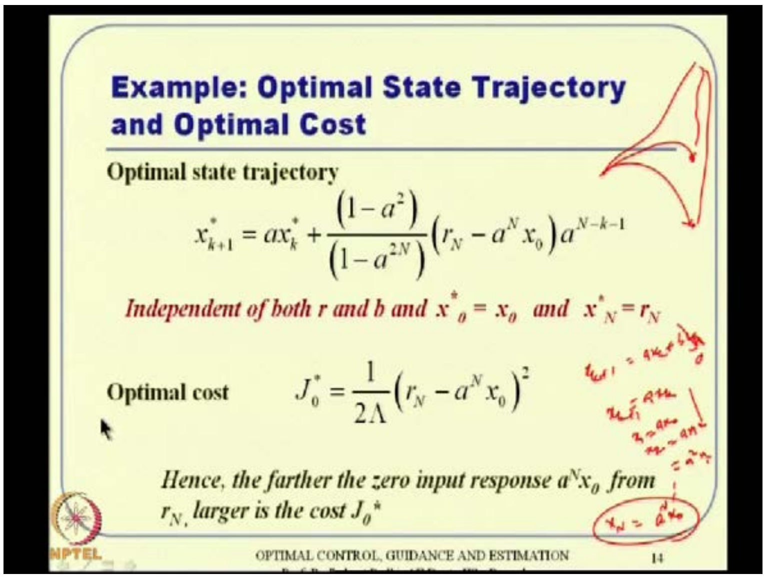
OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 13

Now lambda n is available, you can substitute that and hence you get lambda k, and finally the optimal control happens to be a function of lambda k, U k .Let me see that, this one. Once you get lambda k solution, you can lambda k plus 1 is equivalent to that, you can put it back here actually. So, all that is done and you can write it as U k is nothing, but something like that actually. Now, what can you observe here, the couple of nice observations first, I mean the first observation is in this problem both the final state as well as the optimal control; that means x n which is r n, it is a fix number sort of thing, as well as the optimal control expression, what you see are actually independent of the weighting function r, weighting matrix r.

This is r n, r n is the reference signal at then with the cost function r; we started with a cost function r actually here. So, this cost this r does not play any role actually. and this is also logical, because ultimately your drive is to take x to some final desired value, and on the way you are not kind of comparing with state minimization and control minimization, your objective is only to do control minimization. So, whether you minimize some sort of a U k square, or some factor of U k square we does not matter actually, some fraction of U k square, whatever that actually. So, essentially the control is independent of that, which is logically make sense as well.

Another observation is, the control is actually an open loop control, because what you see here, that actually depends on x knot, nothing else actually, and r n which is of course that is where you want to go, is a function of x knot and r n, anything else is a system parameter a and b, but it is a nothing to do with state some sort of solution basically. In other words U k we did not get as a something like minus k times x k some sort of thing. We did not get that actually, but essentially it can be done, we will see that in a while that discrete L Q R setting, it can be done actually. But this particular way of getting it, we need not get a state feedback solution; we got a open loop solution actually. Now, the optimal state trajectory somebody can ask like that, then you can go back to this expression, what you had for x k, x k solution is available.

(Refer Slide Time: 27:21)



Example: Optimal State Trajectory and Optimal Cost

Optimal state trajectory

$$x^*_{k+1} = ax^*_k + \frac{\left(1-a^2\right)}{\left(1-a^{2N}\right)}\left(r_N - a^N x_0\right)a^{N-k-1}$$

Independent of both r and b and $x^*_0 = x_0$ and $x^*_N = r_N$

Optimal cost $\quad J^*_0 = \frac{1}{2\Lambda}\left(r_N - a^N x_0\right)^2$

Hence, the farther the zero input response $a^N x_0$ from r_N, larger is the cost J_0^*

$x_{k+1} = ax_k + b\cancel{u_k}$ 0

$x_1 = ax_0$

$x_2 = ax_1$

$= a^2 x_0$

$\boxed{x_N = a^N x_0}$

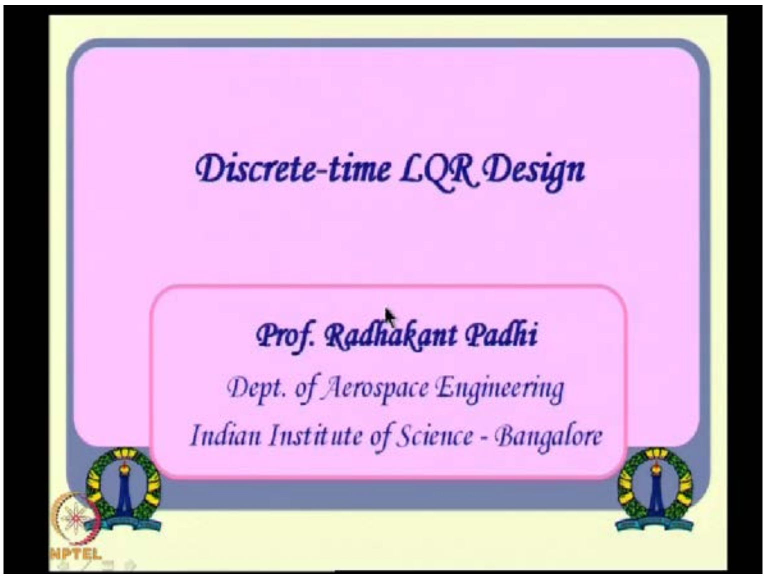
OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 14

You can substitute lambda n and you can get this x k, after optimal state equation also basically. Optimal state trajectory can be dictated by this. This is nothing, but state equation by the way. What you have is x k plus 1 is nothing, but a x k plus b U k, and b times U k happens to be something, even U k you know and b times U k we will cancel here, whatever you have is nothing, but that actually, that is all. So, this and this is I mean we want to do not need a solution for x k, because and this is a sequential equation sort of thing, if the recursive equation sort of thing. The moment you start with the x knot and this just put some formula here, where is from 0 to 1anyway. So, if you just put varying that, you keep on getting numbers here basically. If you want a solution in an x q that is that expression that you need to use directly, otherwise this is going of actually. But also remember this one; the state trajectory is again independent of both r and v and all sort of thing actually. The x knot star is nothing but, x knot, x n star is sometimes when you see books star notation, these are all optimal trajectory sort of thing.

So, whenever you get an optimal trajectory, sometimes people puts star there, just to differentiate between non optimal to optimal trajectory sort of thing. An optimal cost can also be given as something like this. Now, again it turns out, that this is nothing, but this 0 input response and a thing like that actually, what you call actually that way about. I will not talk too much on that. The whole idea here is, because the cost function is like this. The moment this is different from this is nothing, but 0 input response, this is not difficult I can also told you about that. So, you have this x k plus 1, is nothing but a x k plus b U k. So, when U k turns out to be 0, all of the U k's turns out to be 0. Then what happens a x k plus 1 is nothing, but a x k. So, what will your trajectory will be, we start with k equal to 0 then, it is x 1 equal to x knot and then x 2 equal to x 1, I mean a x knot.

And similarly, x 2 equal to a x 1 which is nothing but, a square x not like that actually. you proceed like that, what you discuss something very close to what we discuss before, and ultimately we will turn then if I land up with x n, where it should be a n minus a n times x knot, using this formula sort of thing, what you have here . So, that is your 0 input responses basically. If you do not have any control; that is what the trajectory will evolve actually. Now with control something, this is the cost function; that means, if this is one what you see here is very different from this; then obviously your cost is going to be more, which is very natural also. Suppose, I mean this picture really speaking, suppose your 0 input response takes you there, what your target is somewhere here, then you'll get some something like a deviation like that. If your target is somewhere here, then you require more control to come here, like that actually. So, whatever; this difference place a role, whatever different you see here actually, that is the message here, so it makes lot of sense also.

(Refer Slide Time: 30:44)



Discrete-time LQR Design

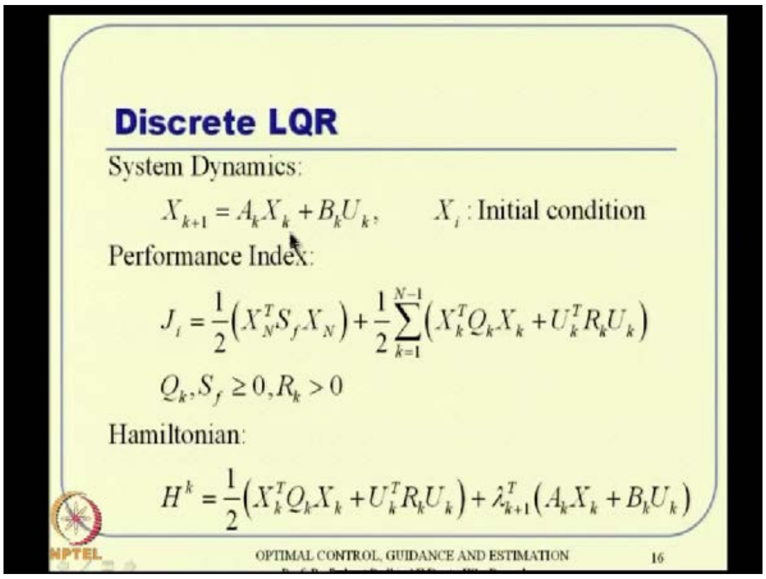
Prof. Radhakant Padhi

Dept. of Aerospace Engineering

Indian Institute of Science - Bangalore

Anyway, so now, there is a different concept, what you got is generic frame work. Now is there any equivalent of discrete time L Q R also.

(Refer Slide Time: 30:53)



Discrete LQR

System Dynamics:

$$X_{k+1} = A_k X_k + B_k U_k, \qquad X_i : \text{Initial condition}$$

Performance Index:

$$J_i = \frac{1}{2}\left(X_N^T S_f X_N\right) + \frac{1}{2}\sum_{k=1}^{N-1}\left(X_k^T Q_k X_k + U_k^T R_k U_k\right)$$

$$Q_k, S_f \geq 0, R_k > 0$$

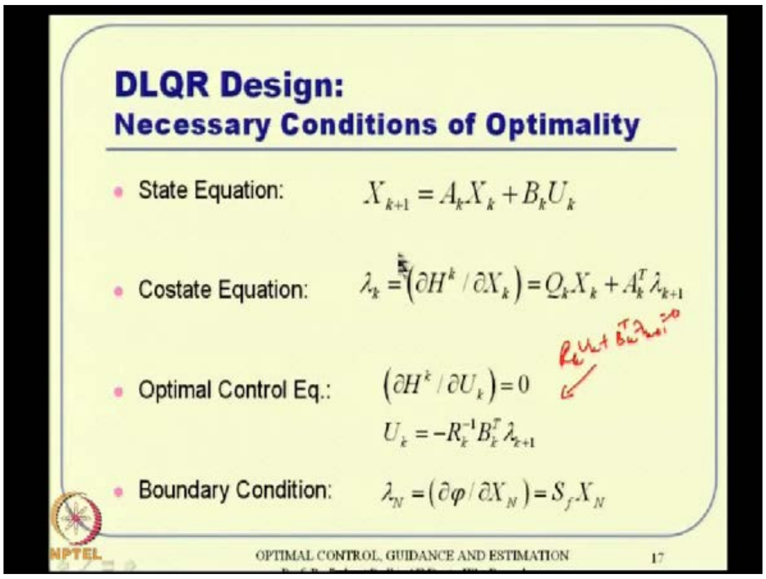
Hamiltonian:

$$H^k = \frac{1}{2}\left(X_k^T Q_k X_k + U_k^T R_k U_k\right) + \lambda_{k+1}^T\left(A_k X_k + B_k U_k\right)$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 16

The problem is something like this, we have in general; we have something like this x k plus 1 is a k x k plus b k U k, and performance index is very close to what you have seen in continuous time. Alright, so this i may not be needed actually, start with initial condition, then this i is needed, whether this is. So, J is something like that, and this is like this actually. So, your Hamiltonian definition is nothing s k is nothing, but L k plus lambda k plus 1 transpose times f k. So, L k is nothing, but this coming from here, and lambda k plus 1 times coming to, I mean f k f k is nothing, but this is what your s k Hamiltonian k.

(Refer Slide Time: 31:43)



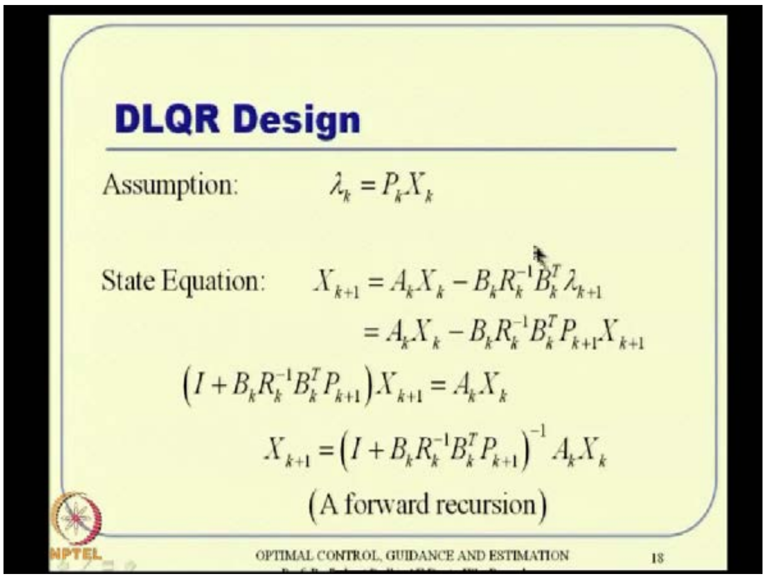
DLQR Design:
Necessary Conditions of Optimality

- State Equation: $X_{k+1} = A_k X_k + B_k U_k$
- Costate Equation: $\lambda_k = \left(\partial H^k / \partial X_k\right) = Q_k X_k + A_k^T \lambda_{k+1}$
- Optimal Control Eq.: $\left(\partial H^k / \partial U_k\right) = 0$ $R_k U_k + B_k^T \lambda_{k+1} = 0$
  $U_k = -R_k^{-1} B_k^T \lambda_{k+1}$
- Boundary Condition: $\lambda_N = \left(\partial \varphi / \partial X_N\right) = S_f X_N$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 17

Now, what is that necessary conditions, necessary conditions will tell, that state equation costate equation, optimal control and boundary conditions sort of things, we know that. So, the state equation is like this, which is already given to us. the costate equation lambda k equal to del H k by del x k. So, this expression, now H k is 0 levels from here. So, del H k by del x k is nothing but, U k x k from here, and a k transpose times lambda k plus 1 from here. So, that is what it is. There is no minus sign remember that. Lambda k is nothing but, q k x k plus a k transpose times lambda k plus 1 sort of thing. Then, optimal control equation; del H k by del U k equal to 0. If you apply that H k is again here. So, del H k by del U k is one term is r k U k; other term will be v k transpose lambda k plus 1.So, this happens to be something like this actually. So, in boundary condition happens to be lambda n equal to s f x n actually. Anyway, this expression del H k by del U k, whatever is talking is something r k U k plus b k transpose times lambda k plus 1; that is equal to 0.So obviously, let me write it otherwise; r k U k plus b k transpose times lambda k plus 1 is equal to 0. So, if you solve this for U k, this is what you get actually; very close to what we know before in continuous time, only difference is not lambda t, but lambda k plus 1, which is any way lambda T is equal into lambda k plus 1, so that is how it is. Boundary condition happens to be something like this actually.

(Refer Slide Time: 33:32)



DLQR Design

Assumption: $\lambda_k = P_k X_k$

State Equation: $X_{k+1} = A_k X_k - B_k R_k^{-1} B_k^T \lambda_{k+1}$

$$= A_k X_k - B_k R_k^{-1} B_k^T P_{k+1} X_{k+1}$$

$$\left(I + B_k R_k^{-1} B_k^T P_{k+1}\right) X_{k+1} = A_k X_k$$

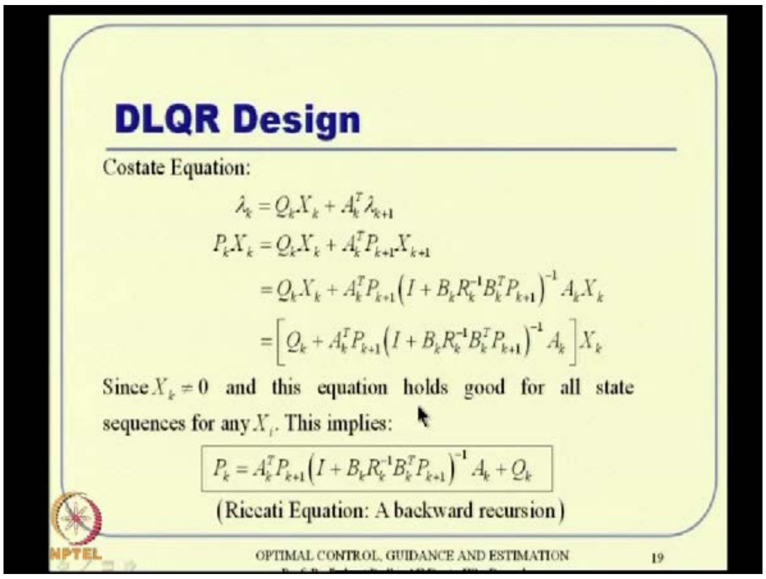
$$X_{k+1} = \left(I + B_k R_k^{-1} B_k^T P_{k+1}\right)^{-1} A_k X_k$$

(A forward recursion)

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 18

Now you have to use all these, to derive some sort of a riccati equation, equivalent in discrete time actually. So, again we assume that lambda k is to be like p k times x k. then what happens you can go ahead and tell x k plus 1, what you get state equation here nothing but, a k x k plus b k U k, but b k times U k is nothing but this, this is what it is, and lambda k plus 1, is nothing but p k plus 1 x k plus 1, so all this is substituted here. We start with a x k plus b U k, U k is like that, minus r inverse b transpose lambda k plus 1. And then lambda k plus 1 from this expression, is nothing but p k plus 1 times x k plus 1. So, that is also substituted here. Now, what you can do is, you see some x k plus 1 here, x k plus 1 here. So, you take it to one side, then write x k plus 1, is something times x k plus 1 is this equal to a k x k and hence x k plus 1, can be given some sort of, some expression like that which is nothing but, a forward recursion actually.

(Refer Slide Time: 34:39)



DLQR Design

Costate Equation:

$$\lambda_k = Q_k X_k + A_k^T \lambda_{k+1}$$

$$P_k X_k = Q_k X_k + A_k^T P_{k+1} X_{k+1}$$

$$= Q_k X_k + A_k^T P_{k+1} \left(I + B_k R_k^{-1} B_k^T P_{k+1} \right)^{-1} A_k X_k$$

$$= \left[Q_k + A_k^T P_{k+1} \left(I + B_k R_k^{-1} B_k^T P_{k+1} \right)^{-1} A_k \right] X_k$$

Since $X_k \neq 0$ and this equation holds good for all state sequences for any X_i. This implies:

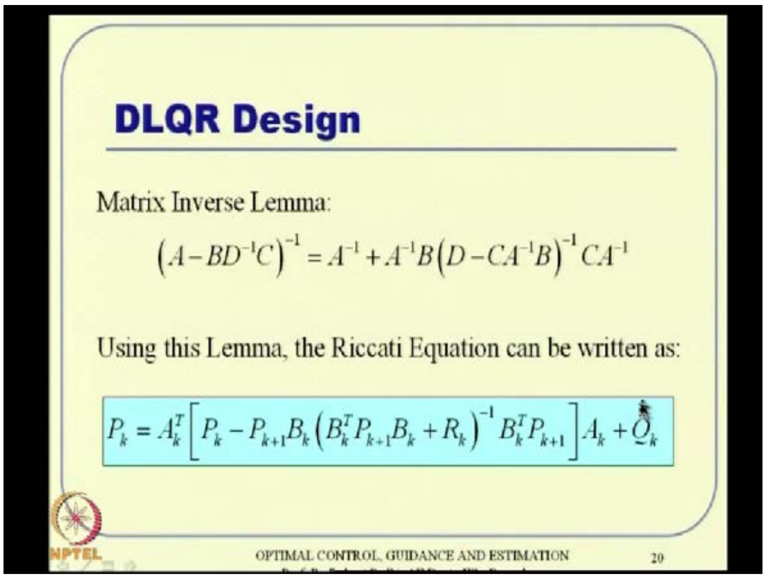
$$\boxed{P_k = A_k^T P_{k+1} \left(I + B_k R_k^{-1} B_k^T P_{k+1} \right)^{-1} A_k + Q_k}$$

(Riccati Equation: A backward recursion)

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 19

Then what about costate equation. Costate equation turns out to be like this, q k lambda k is nothing but, q k x k plus a k transpose times lambda k plus 1. Lambda k plus 1 again p k plus 1 times x k plus 1, and then this lambda k is p k x k. So, you substitute all that. Now x k plus 1 we just substituted, we just derived this expression. So, this instead of wherever k plus 1 is there I can write is that way; and then everything happens p k x k is nothing but, something big matrix actually, multiplied with x k. and since x k is not 0 in general, and this equation holds good for all state equation for any x i; that means, it is valid for x 1 x 2 x 3 everywhere actually; for all of those x will not be 0, because you are not talking about a equilibrium condition sort of thing, I mean, because this is not 0, if I take everything into one side then right it equal to 0 where x k is non 0; then obviously, using our standard philosophy that coefficient as to be 0, and that it equivalently touch starting that it is nothing, but p k, what you get here is equivalent to that actually, whatever you get here. Essentially, if you see this expression equation is nothing, but riccati equation. it is a backward recursion sort of thing; if you know p k plus 1 you can calculate p k .So, this is what it is actually. So, essentially what we got, is some sort of a discrete time riccati equation; some sort of a backward recursion, which is make lot of sense comfortable to what we know earlier, this is what it is actually.

(Refer Slide Time: 36:20)



DLQR Design

Matrix Inverse Lemma:

$$\left(A - BD^{-1}C\right)^{-1} = A^{-1} + A^{-1}B\left(D - CA^{-1}B\right)^{-1}CA^{-1}$$

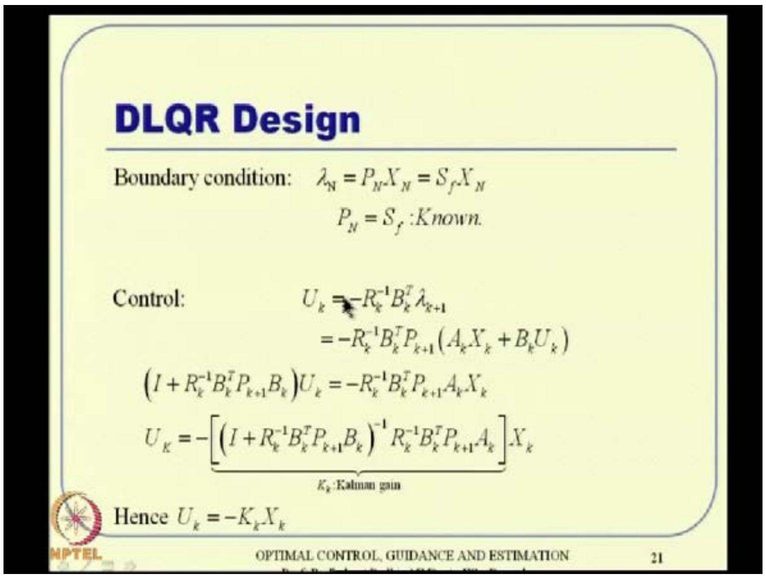
Using this Lemma, the Riccati Equation can be written as:

$$\boxed{P_k = A_k^T\left[P_k - P_{k+1}B_k\left(B_k^T P_{k+1} B_k + R_k\right)^{-1} B_k^T P_{k+1}\right]A_k + Q_k}$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 20

Now, you can stop here, but you can also use this matrix inverse lemma sort of thing, which is available, many people use it for variety of reasons. and then using; if you use this lemma then this expression what you have here, can be express it something like this; just substitute, just indentify, what is a b d c in this expression, and then a is nothing but identity, for example. So, like that actually put it there and then substitute x 1 and you will get it something like this actually. This is popularly known is this riccati equation discrete time sort of thing.

(Refer Slide Time: 36:54)



DLQR Design

Boundary condition: $\lambda_N = P_N X_N = S_f X_N$

$$P_N = S_f : Known.$$

Control: $U_k = -R_k^{-1} B_k^T \lambda_{k+1}$

$$= -R_k^{-1} B_k^T P_{k+1}\left(A_k X_k + B_k U_k\right)$$

$$\left(I + R_k^{-1} B_k^T P_{k+1} B_k\right)U_k = -R_k^{-1} B_k^T P_{k+1} A_k X_k$$

$$U_K = -\underbrace{\left[\left(I + R_k^{-1} B_k^T P_{k+1} B_k\right)^{-1} R_k^{-1} B_k^T P_{k+1} A_k\right]}_{K_k:\text{Kalman gain}} X_k$$

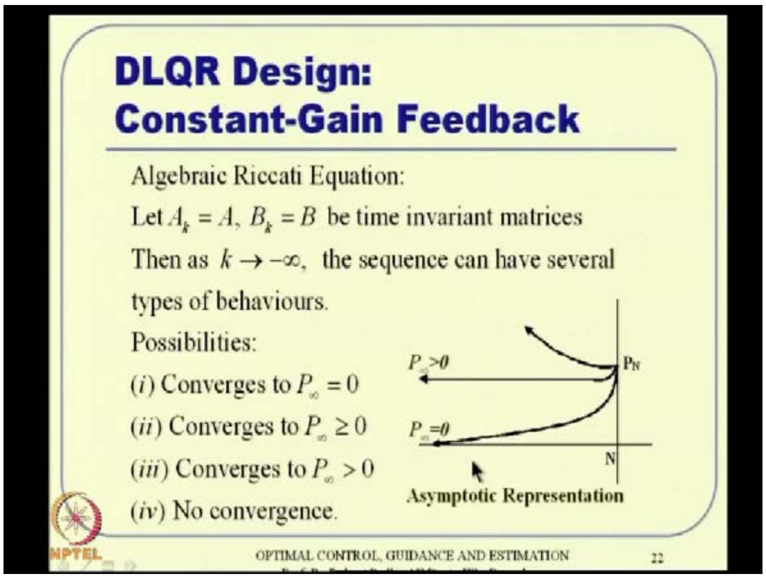
Hence $U_k = -K_k X_k$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 21

Now what about boundary condition actually. So, p n is nothing but, I mean lambda n is nothing but p n x n which is equal to s f times x n from boundary condition. So, p n is s f actually. So, what you are getting here; the same riccati equation sort of ideas with same boundary condition basically. So, we start with the boundary condition and do this backward riccati some sort of thing. Now what about, finally the control, control is nothing but, this minus r inverse b transpose lambda k plus 1 sort of thing. So, k U k is this expression and lambda k plus 1, what you got here, lambda k plus 1 is nothing, but p k plus 1 times x k x k plus 1 sort of thing, that lambda k plus 1 is nothing, but p k plus 1 times x k plus 1 actually.

Anyway, so this is what I mean here is, U k is equal to like this and then you substitute lambda k plus 1 is p k plus 1 plus into x k plus 1; x k plus 1 you can substitute using this state equation and all, and then you can see that you have this U k, appearing both sides you take everything to one side, and then keep the x k in the right hand side. Ultimately you can right U k is nothing, but this inverse whatever you see, this inverse times this matrix, times x k. So, whatever you get all the things here, if you do that,that is nothing but a gain matrix. Essentially kalman gain sort of thing, so essentially we can again write U k is nothing, but minus k k times x k, where k k comes from this expression actually. So, very close to what we know earlier, but not exactly one to one sort of thing actually.

(Refer Slide Time: 38:48)


DLQR Design: Constant-Gain Feedback

Algebraic Riccati Equation:

Let $A_k = A$, $B_k = B$ be time invariant matrices

Then as $k \to -\infty$, the sequence can have several types of behaviours.

Possibilities:

(i) Converges to $P_\infty = 0$

(ii) Converges to $P_\infty \geq 0$

(iii) Converges to $P_\infty > 0$

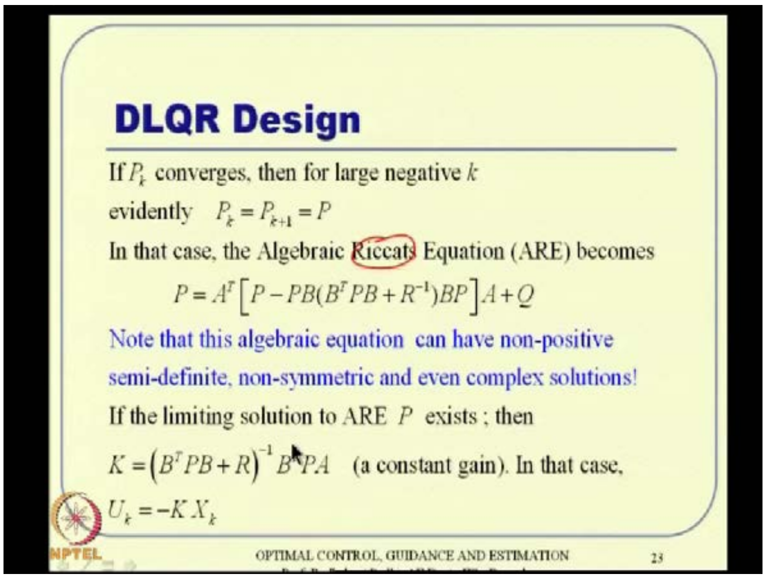
(iv) No convergence.

$P_\infty > 0$
P_N
$P_\infty = 0$
N
Asymptotic Representation

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 22

Now, it interestingly turns out that in discrete formulation, this continuous, I mean this discrete time riccati equation, need not be very well behaved as continuous time riccati equation really. In other words, the sequence can have several types of behaviors. the possibilities include something like this, we will start with p n, it can actually converts towards 0, it can go up; I mean go to infinity sort of thing, or it can I mean converse to some p, some p infinity which is actually non 0. And it can have no convergence at all also basically. There can be four possibilities; it can converge to some 0 value, it can converge to a non 0 value, it can converge to a strictly positive value, or it can, they can no convergence also basically. Anyway, that is another point; point here is, every time you apply this you may not get nice solution sort of thing basically.

(Refer Slide Time: 39:45)



DLQR Design

If P_k converges, then for large negative k

evidently $P_k = P_{k+1} = P$

In that case, the Algebraic Riccati Equation (ARE) becomes

$$P = A^T\left[P - PB(B^T PB + R^{-1})BP\right]A + Q$$

Note that this algebraic equation can have non-positive semi-definite, non-symmetric and even complex solutions!

If the limiting solution to ARE P exists ; then

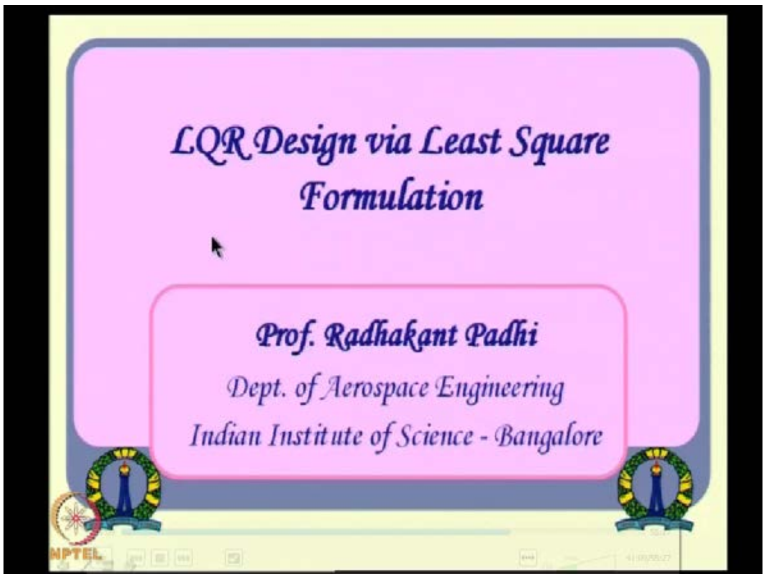
$K = \left(B^T PB + R\right)^{-1} B^T PA$ (a constant gain). In that case,

$U_k = -K X_k$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 23

If p k converge is, then for large negative k evidently, if p k converges, converges means what it converges some value basically. If it converges is then; obviously, it will get a constant value actually. So, in that case the algebraic riccati equation; is this spelling mistake actually, riccati equation. The algebraic riccati equation becomes this expression actually, and also this algebraic riccati equation can have non positive semi definite, non symmetric and even complex solutions. So, this kind of things not very good to see, but its effect actually and live with that, and in the limiting solution we have, if the limiting solution to A R E; that is this constant p matrix exists, and then ultimately we can write again something like this, which is nothing but a constant gain; in that case U k is nothing but minus k s k. So, in other words, just because your formula; it does not mean that it is a very good nice (()) formula basically. So, all these conditions are met; then you can write it actually. Anyway, so this is heart of these, this L Q R theory basically, I mean L Q R using discrete time actually.

(Refer Slide Time: 41:09)



LQR Design via Least Square Formulation

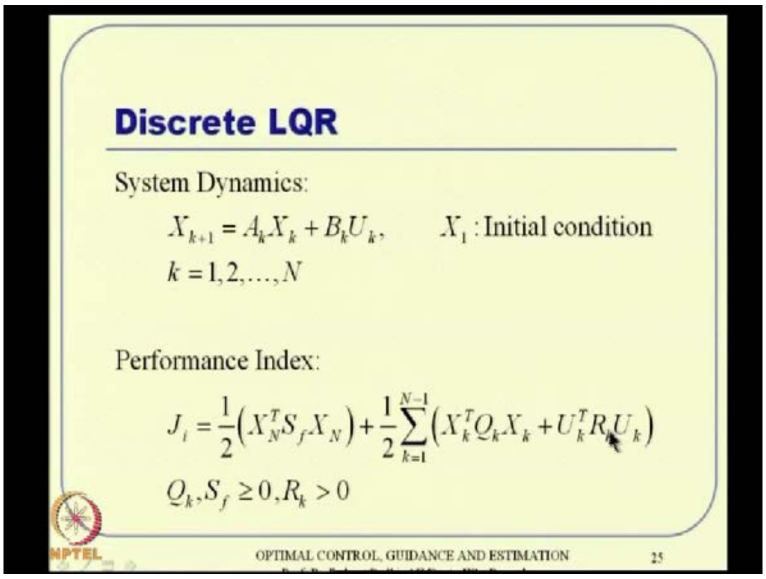
Prof. Radhakant Padhi

Dept. of Aerospace Engineering

Indian Institute of Science - Bangalore

Now, anyway as I told in the beginning, that there are some primitive ideas of L Q R design through least of square formulation sort of thing. Let us see a glimpse of that before moving further actually.

(Refer Slide Time: 41:17)



Discrete LQR

System Dynamics:

$$X_{k+1} = A_k X_k + B_k U_k, \qquad X_1 : \text{Initial condition}$$
$$k = 1, 2, \ldots, N$$

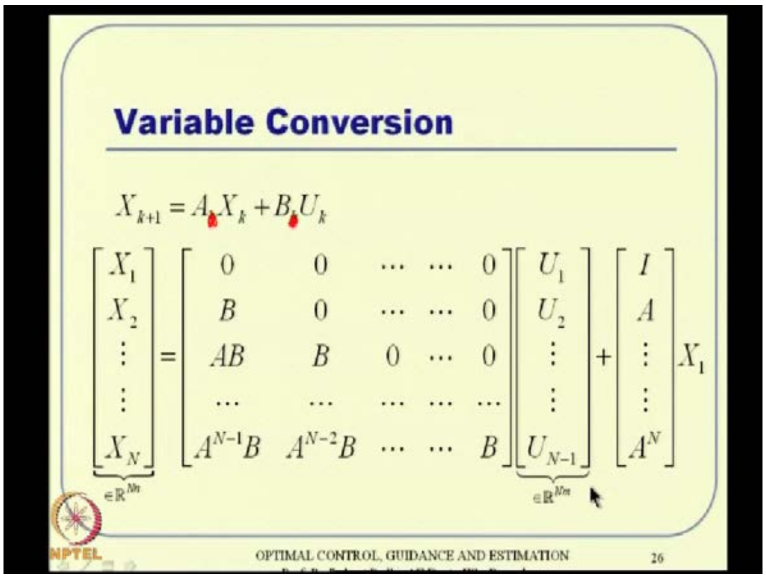
Performance Index:

$$J_i = \frac{1}{2}\left(X_N^T S_f X_N\right) + \frac{1}{2}\sum_{k=1}^{N-1}\left(X_k^T Q_k X_k + U_k^T R_k U_k\right)$$
$$Q_k, S_f \geq 0, R_k > 0$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 25

So, this is idea is very appealing and very simple actually rather. So, we have a system dynamics. the whole difficulty started, because we had to deal with state in addition to control actually, but how about representing all state vectors in terms of control vector, then everything we can substitute in the cost function also, as function of states a different grid point., and then talk about some sort of a static optimization, which will optimize which will minimize these cost function, because it will all be now function of control only.

(Refer Slide Time: 41:51)



Variable Conversion

$$X_{k+1} = A_k X_k + B_k U_k$$

$$\underbrace{\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ \vdots \\ X_N \end{bmatrix}}_{\in \mathbb{R}^{Nn}} = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A^{N-1}B & A^{N-2}B & \cdots & \cdots & B \end{bmatrix} \underbrace{\begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ \vdots \\ U_{N-1} \end{bmatrix}}_{\in \mathbb{R}^{Nm}} + \begin{bmatrix} I \\ A \\ \vdots \\ \vdots \\ A^N \end{bmatrix} X_1$$

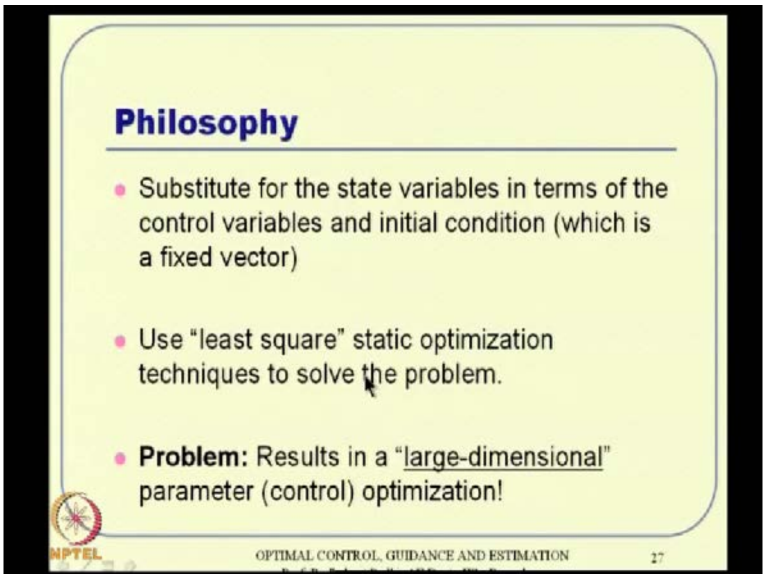
OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 26

How do you do that; this is something like this; x k plus 1, is nothing but this one we know that. So, let us say we start with x 1 which is nothing but initial condition. So, x 1 is equal to all zeros and all u's, at this are control set variation grid point actually. everything zero plus identity terms x 1, so that means first row, is nothing but x 1 equal to x 1 sort of thing. Now, x 2 when I put k equal to one, this is x 2 equal to a times x 1 plus b times; I mean a times x 1 plus b times U 1 actually. Forget this k k and all that, let us see this is the time invariant sort of thing, or either way or I consider the entries of this actually, as time is (()) thing also basically either way. So, let us be comfortable, just take some sort of time invariant and b basically. If you do that then it is x 1 is nothing but x 1, and x 2 is nothing else but, b times U 1 plus a times x 1. So, b times U 1 nothing else, plus a times x 1 here.

Similarly, if you continue, finally we get x n is nothing, but this expression actually. The start with something like a n minus 1 times b, recursively and again and again, again and again replying that, ultimately we will come up with this actually. Now, the question here is, what happens here is, all these variables can be substituted as all the variables in

U n, you want U n, I mean U n minus 1 really and x 1. So, wherever this value appears that x 1 x 2 x 3 x 4 up to x n, I can always substitute through this equivalent and expression of the right hand side; and then what we interpret it is, this optimization problem is a optimization problem of this variables only basically. So, now we can accept the idea of static optimization and talk about solution of a static optimization problem actually.

(Refer Slide Time: 43:53)



Philosophy

- Substitute for the state variables in terms of the control variables and initial condition (which is a fixed vector)
- Use "least square" static optimization techniques to solve the problem.
- **Problem:** Results in a "large-dimensional" parameter (control) optimization!

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 27

This is what is listed here, so what you do is, to substitute the state variables in terms of the control variables and initial condition, and initial condition happens to be a fixed number, fixed sort of thing. So, that does not play role and optimize, nothing that you can vary for a x 1 actually, but anyways then you can do, I can substitute this expression wake into, I mean optimization formulation, and interpret everything; this cost function and interpret that cost function now, has function of only U actually. This constraint is already we are taken care while formulating this way. But, the problem here is, actually it results to a large dimensional parameter optimization, and what is parameter is, a control vector actually. The control values at grid point one, two, three, and four. These are nothing, but the parameter. So, if you really want inaccurate good solution it requires a very huge dimensional static optimization problem, depends on the number of grids, into the number of control, into the dimension of the control vector.

Actually, if you see three controls and something like thousand grid points, and it will have a three thousand variables for optimization sort of thing actually. And typically the numbers of grid points are not thousand points; they are lot more than thousand. So, we will have that many variables, and it is not very good to see that r handle that kind of optimization problem actually. So, that is what the problem. It is also large dimensional optimization problem, which is actually not very good to see. So, that is why it is not very popular also basically. but, this is essentially the idea of this transcription method, what you have, what you can see, I mean if you can correlate very quickly and then it is nothing, but the whole idea of having the grid point and then constructing a large dimensional of optimization problem, static optimization problem, is nothing but direct transcription ideas actually. Anyway the problem is like this, and it is not that kind of that popular actually.

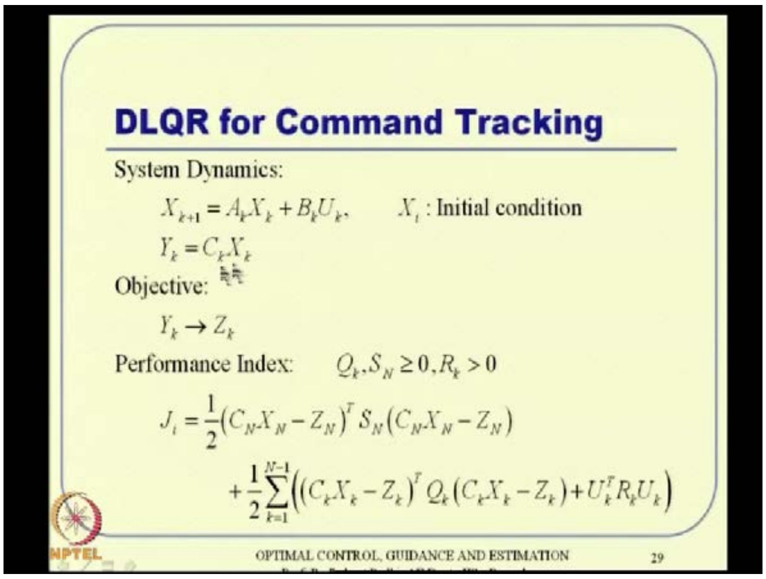
(Refer Slide Time: 45:53)

Discrete LQR for Command Tracking

Prof. Radhakant Padhi
Dept. of Aerospace Engineering
Indian Institute of Science - Bangalore

Now, the final thing before we stop this lecture is what I promised, is how do you use the discrete L Q R setting for command tracking problems as well actually.

(Refer Slide Time: 46:03)



DLQR for Command Tracking

System Dynamics:

$$X_{k+1} = A_k X_k + B_k U_k, \qquad X_i : \text{Initial condition}$$

$$Y_k = C_k X_k$$

Objective:

$$Y_k \to Z_k$$

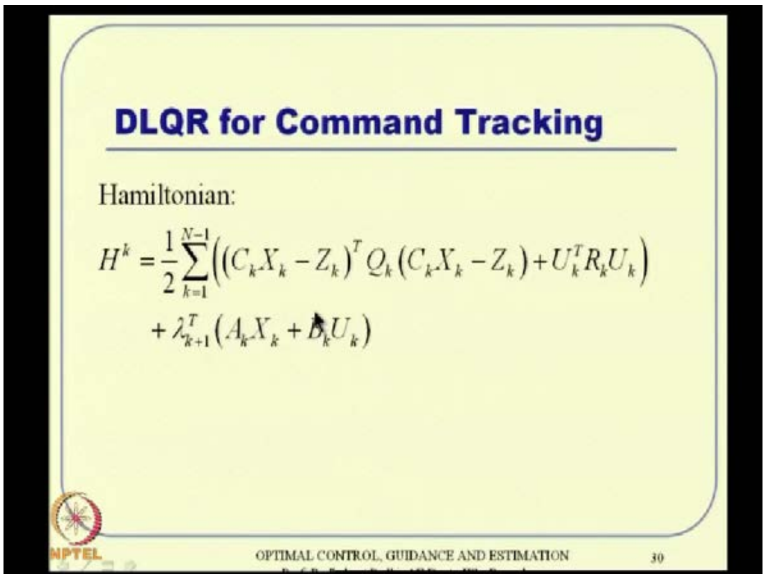
Performance Index: $Q_k, S_N \geq 0, R_k > 0$

$$J_i = \frac{1}{2}(C_N X_N - Z_N)^T S_N (C_N X_N - Z_N) + \frac{1}{2}\sum_{k=1}^{N-1}\left((C_k X_k - Z_k)^T Q_k (C_k X_k - Z_k) + U_k^T R_k U_k\right)$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 29

See; here the problem setting is like this, you have this static equation and we have a output equation also and objective is, that output your y k is it track some reference output z k; that case available and y k should be able to track z k actually. So, what is our performance index. Performance index can be, see remember what is y n, is nothing but c n x n, and that should be k is close to z n possible, that is how we formulated cos function, outside the summation sign is like this, c n x n, minus z n transpose x n times basically. On the way, this deviation has to be minimum (()) basically. So, if you take c x k, y k minus z k, that has to be minimum, and what is y k c k x k. So, y k minus z k, is nothing but c k minus c k x k minus z k. So, that has to be minimum actually, and also we want control minimum, so this is how it is possible.

(Refer Slide Time: 47:01)



DLQR for Command Tracking

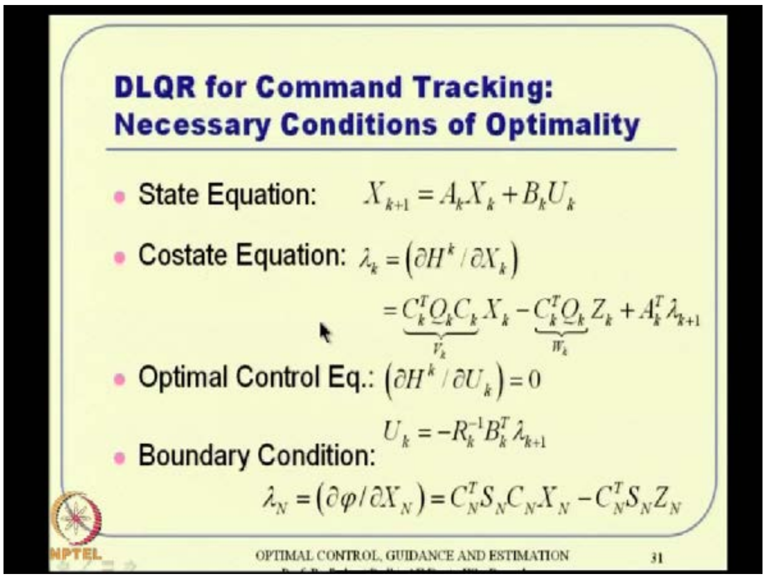
Hamiltonian:

$$H^k = \frac{1}{2}\sum_{k=1}^{N-1}\left((C_k X_k - Z_k)^T Q_k (C_k X_k - Z_k) + U_k^T R_k U_k\right) + \lambda_{k+1}^T (A_k X_k + B_k U_k)$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 30

Now, we will follow this Hamiltonian conception, and also we have this H of k, which we can write U k plus lambda k plus 1 transpose times f. So, this is nothing but U k plus lambda k plus 1 times k plus 1 transpose times k plus 1 transpose times x k plus 1; that is a nothing but a x k plus b k U k basically, now transpose f basically.

(Refer Slide Time: 47:25)



DLQR for Command Tracking: Necessary Conditions of Optimality

- State Equation: $X_{k+1} = A_k X_k + B_k U_k$
- Costate Equation: $\lambda_k = \left(\partial H^k / \partial X_k\right)$

$$= \underbrace{C_k^T Q_k C_k}_{V_k} X_k - \underbrace{C_k^T Q_k}_{W_k} Z_k + A_k^T \lambda_{k+1}$$

- Optimal Control Eq.: $\left(\partial H^k / \partial U_k\right) = 0$

$$U_k = -R_k^{-1} B_k^T \lambda_{k+1}$$

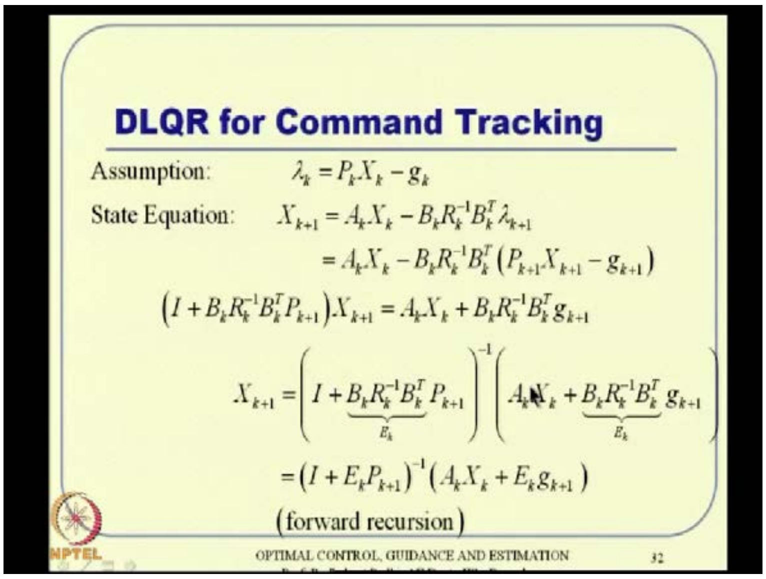
- Boundary Condition:

$$\lambda_N = \left(\partial \varphi / \partial X_N\right) = C_N^T S_N C_N X_N - C_N^T S_N Z_N$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 31

Now what you do is state equation and costate equation, optimal control, boundary condition, all that we apply, and remember the difference here, the basically fundamental difference here is the cost function actually. You have instead of a standard x transverse q x, you had something like this, and outside instead of arbitrary, I mean quadratic expression we have that actually, that is the only difference. To apply all that, now we will see lambda k, is nothing but I mean this expression basically, where it is lambda k function of x k is function of z k as well, and it is also function of lambda k plus 1 actually. This is the state forward algebra, from basic principle sort of thing. Then optimal control del x k by del U k is equal to 0. So, that explain that, you again land of with the same expression their actually. Boundary condition of course, you know del phi by del x n, is actually equal to all this expression, because phi expression is something like this actually.

(Refer Slide Time: 48:23)



DLQR for Command Tracking

Assumption: $\lambda_k = P_k X_k - g_k$

State Equation: $X_{k+1} = A_k X_k - B_k R_k^{-1} B_k^T \lambda_{k+1}$

$$= A_k X_k - B_k R_k^{-1} B_k^T \left(P_{k+1} X_{k+1} - g_{k+1} \right)$$

$$\left(I + B_k R_k^{-1} B_k^T P_{k+1} \right) X_{k+1} = A_k X_k + B_k R_k^{-1} B_k^T g_{k+1}$$

$$X_{k+1} = \left(I + \underbrace{B_k R_k^{-1} B_k^T}_{E_k} P_{k+1} \right)^{-1} \left(A_k X_k + \underbrace{B_k R_k^{-1} B_k^T}_{E_k} g_{k+1} \right)$$

$$= \left(I + E_k P_{k+1} \right)^{-1} \left(A_k X_k + E_k g_{k+1} \right)$$

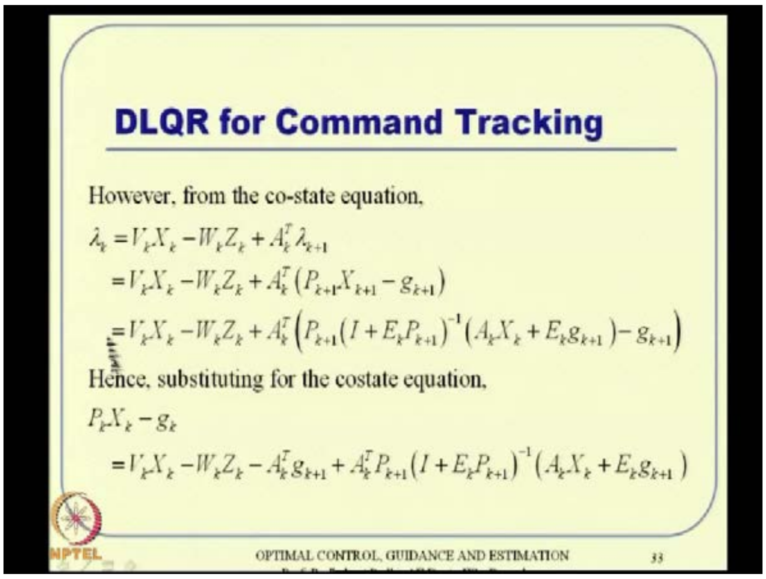
(forward recursion)

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 32

Alright, so now, what happens here; the fundamental difference is, lambda k equal to p k x k will not do the job, we will need also, me sort of time varying additional term, which will locally do the job actually; that will try to track that command, that help tracking the command and that z k, which is coming externally remember the problem is not to track the 0 signal, is to track some sort of a non 0 signal actually. Now, the state equation turns out to be something very similarly, what we have done before we will proceeds the same steps. the only were wherever lambda k or lambda k plus 1 is there you to slightly careful to use this expression, with this bias term this minus g k some sort of thing; that is what is done here, you can sort with state equation and you land up with lambda k plus 1 substitute is a lambda k plus 1 and this expression what you get here.

And then again and again k plus 1 is expression k plus 1 here and try to sort it out, take it one side and then solve it, and then tell k plus 1 has to be inverse of this matrix, times whatever is left out; that is a k x k, plus this term actually, which is coming from this term. So, essentially we land up with x k plus 1 is something like this, again forward recursion formula actually; a k x k plus e k, e k is define something like this basically. So, what you have here is i plus e k, p k plus 1 whole inverse times a k e k plus e k g k plus 1 actually, is again exactly the similar sort of ideas, similar steps and all that, but we have taken, we have to resort slightly careful to substitute lambda k s and lambda k plus 1 is properly basically.

(Refer Slide Time: 50:08)



DLQR for Command Tracking

However, from the co-state equation,

$$\begin{aligned}\lambda_k &= V_k X_k - W_k Z_k + A_k^T \lambda_{k+1} \\ &= V_k X_k - W_k Z_k + A_k^T \left(P_{k+1} X_{k+1} - g_{k+1} \right) \\ &= V_k X_k - W_k Z_k + A_k^T \left(P_{k+1} \left(I + E_k P_{k+1} \right)^{-1} \left(A_k X_k + E_k g_{k+1} \right) - g_{k+1} \right)\end{aligned}$$

Hence, substituting for the costate equation,

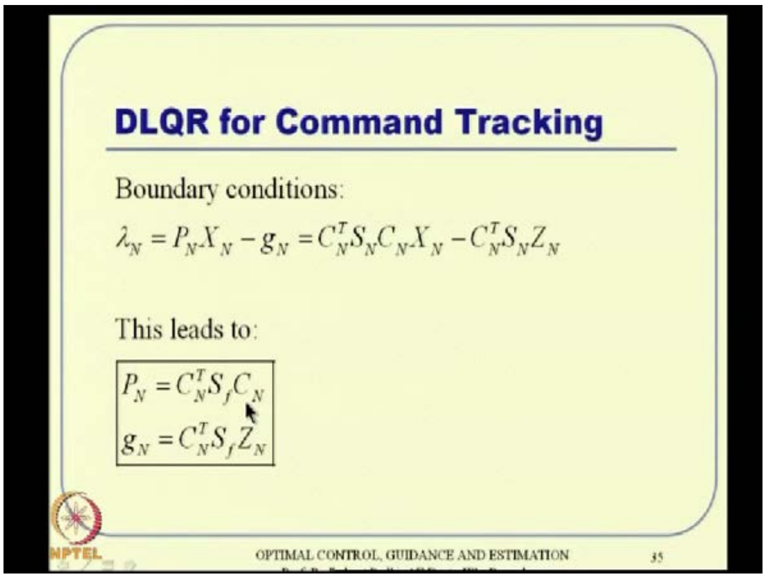
$$\begin{aligned}P_k X_k - g_k \\ &= V_k X_k - W_k Z_k - A_k^T g_{k+1} + A_k^T P_{k+1} \left(I + E_k P_{k+1} \right)^{-1} \left(A_k X_k + E_k g_{k+1} \right)\end{aligned}$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 33

So, with that, the costate equation turns out to be something like this, and we are defining some terms and all there actually. So, lambda k is a function of x k and z k and lambda k plus 1. So, that is what we see here, this is what v k x k minus w k, plus tracking transport times lambda k plus 1; that is substitute here, that is put it here, and lambda k plus 1 is now substitute, whatever that is, and then you know that actually, this x k plus 1 here actually we just solve it. So, this x k plus 1 you can substitute here. And then lambda k is nothing but p k x k minus g k; that is your costate equation; this one, x k plus 1 is something like this, lambda k is something like this, and p k and x k plus 1 is substituted, I get something like that. Now if you substitute the co state equation, then this lambda k nothing but that, because that is the definition right we started like this. So, you substitute like that, the left hand side and right hand side like that actually.

So, rearranging the terms out to be something into x k plus all this terms is should be equal to 0, and again I strongly suggest all of you to, kind of derive all this, for your clarity and think like that, you can pause this whenever you see this lecture, you can pause it here and then try to derive it this expression yourself actually probably. Anyways, so do that, and then this is what it is, get some expression like this, and since it holds good for all x, which is non 0 think like that, all sort of variation possible. Then what we can see is, the coefficient is has to be 0; that that will us the expression for p k, happens to be something like this, and g k happens to be something like this. Remember g k can we know only with the available information about g k plus 1. This is a backward riccati formula again actually.

(Refer Slide Time: 52:14)



DLQR for Command Tracking

Boundary conditions:

$$\lambda_N = P_N X_N - g_N = C_N^T S_N C_N X_N - C_N^T S_N Z_N$$

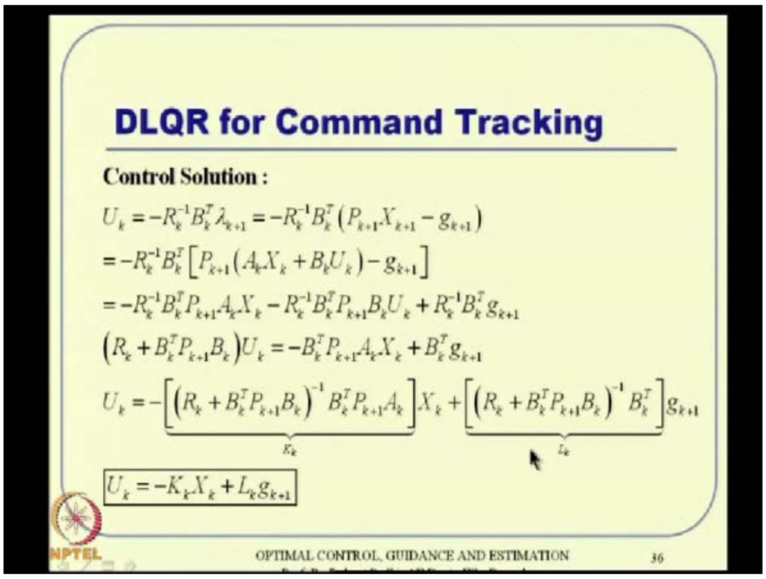
This leads to:

$$\boxed{\begin{aligned} P_N &= C_N^T S_f C_N \\ g_N &= C_N^T S_f Z_N \end{aligned}}$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 35

So, boundary condition, again go their lambda n turns out to be p n x n minus g n. So, and then lambda s n, we can substitute all the variables whatever you know there, and then this leads to this; if you consider this I mean require this coefficient something like that, and then p n happens to be like this, and g n happens to be this expression actually, that is what written here. So, what are getting, in some sort of a backward riccative relationship for p k and g k, because p k is function of p k plus 1; g k is function of g k plus 1 actually. This two backward riccati relationship also remains to boundary condition actually. So, starting from this boundary condition we can compute, keep on computing this p k's and g k's basically.

(Refer Slide Time: 53:01)



DLQR for Command Tracking

Control Solution :

$$U_k = -R_k^{-1}B_k^T\lambda_{k+1} = -R_k^{-1}B_k^T\left(P_{k+1}X_{k+1} - g_{k+1}\right)$$

$$= -R_k^{-1}B_k^T\left[P_{k+1}\left(A_kX_k + B_kU_k\right) - g_{k+1}\right]$$

$$= -R_k^{-1}B_k^TP_{k+1}A_kX_k - R_k^{-1}B_k^TP_{k+1}B_kU_k + R_k^{-1}B_k^Tg_{k+1}$$

$$\left(R_k + B_k^TP_{k+1}B_k\right)U_k = -B_k^TP_{k+1}A_kX_k + B_k^Tg_{k+1}$$

$$U_k = -\underbrace{\left[\left(R_k + B_k^TP_{k+1}B_k\right)^{-1}B_k^TP_{k+1}A_k\right]}_{K_k}X_k + \underbrace{\left[\left(R_k + B_k^TP_{k+1}B_k\right)^{-1}B_k^T\right]}_{L_k}g_{k+1}$$

$$\boxed{U_k = -K_kX_k + L_kg_{k+1}}$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 36

Now, the control solution finally, that is what our main aim; U k equal to minus r inverse r k inverse b k transverse lambda k plus 1, and this actually can, that lambda k plus 1. You can substitute it like this, and then x k plus 1, we can substitute like this, x k plus 1 is nothing but the state equation. So, that is a x k plus b k sort of thing, we can substitute here and then we expand the terms x k term and U k term and g k plus 1 terms like that. Now you have a U k in left hand side and U k in the right hand side. So, you combined together and then solve for U k actually, and U k will turn out be something like this, which is. and this part of the solution can be define some sort of gain matrix, and this gain matrix some sort to be, all this expression and then there is a bias gain matrix sort of thing basically, this a reference signal remember g. this bias, g starts from something like this actually the g k somewhere here, so this bias term coefficient.

So, essentially we will line up with formula which will tell you that, U k is not only minus k into x k, minus k into x k, but we have a additional term which tells U L k time g k plus 1 actually, which make sense, which you want to track some sort of signal reference, signal actually. That is it is a, I mean in this particular lecture, I wanted to give some sort of overview of discrete optimal control followed by discrete L Q R and there are many things around that, but now with the advancement of this digital computational all that, you can actually implement any continuous time control formulas with a very small grid size also, and then the discrete of optimal control will a very close meaning, is what is continuous optimal control as well actually. So, with that knowledge will stop this lecture and also remember that this discrete L Q R formulation, and discrete of optimal control formulation all that. We will use it some part of time later, in a slightly different starting actually, to come up with numerical algorithms for non-linear optimal control also basically. So, with that motivation I will stop here; thank you for your attention.