Optimal Control Guidance and Estimation

Prof. Radhakant Padhi

Department of Aerospace Engineering

Indian Institute of Science, Bangalore

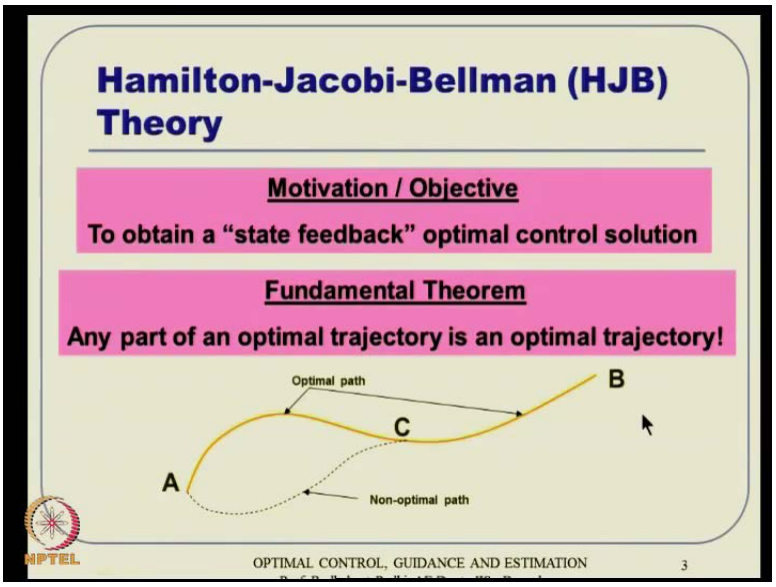
Module No. # 17

Lecture No. # 40

Take Home Material_ Summary – II

Hello everybody. Here, we are at the end of the lecture series and this is the last lecture of this course on, "Optimal Control Guidance and Estimation". Last two lectures, as I told in the previous lecture, we are reviewing this material what we have a covered in rest of the 38 lectures. Then the lecture 39, just previous lecture, we summarize many things like up to LQR theory and SDRE also. So, let us continue that series and then kind of quickly review the rest of material as well.

(Refer Slide Time: 00:51)

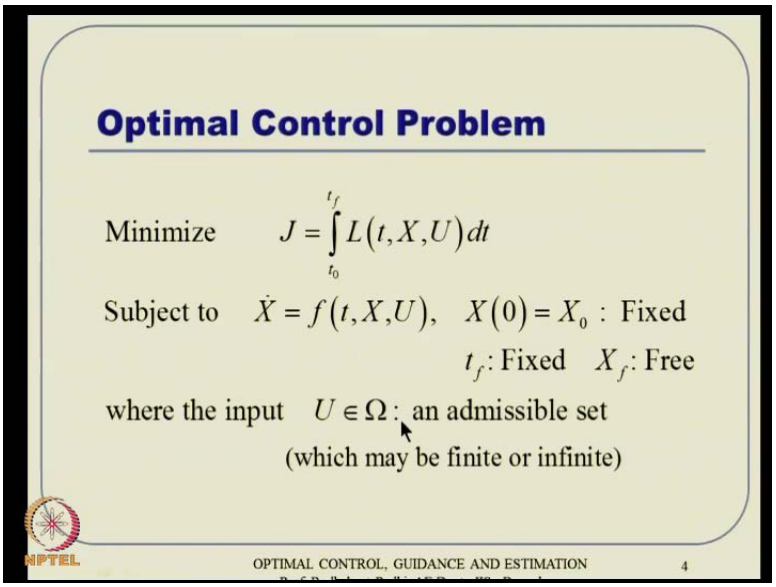


So, next topic that we discussed is something called dynamic programming and this was kind of motivated with the objective that we want to have state feedback optimal control solution really; open loop sort of all things, we are not happy with that any more. Then there is a fundamental theorem of optimal control, which tells us that any part of the optimal trajectory is also an optimal trajectory, and that was slightly explained in this

way. That if it happen that A to B and via C; if you claimed that this path is optimal, the colored one, colored solid one and then if is there is something like a non-optimal path from A to C; it is not going to happen, because in that case then this path A to C via dotted line and then C to B, would have been the optimal path.

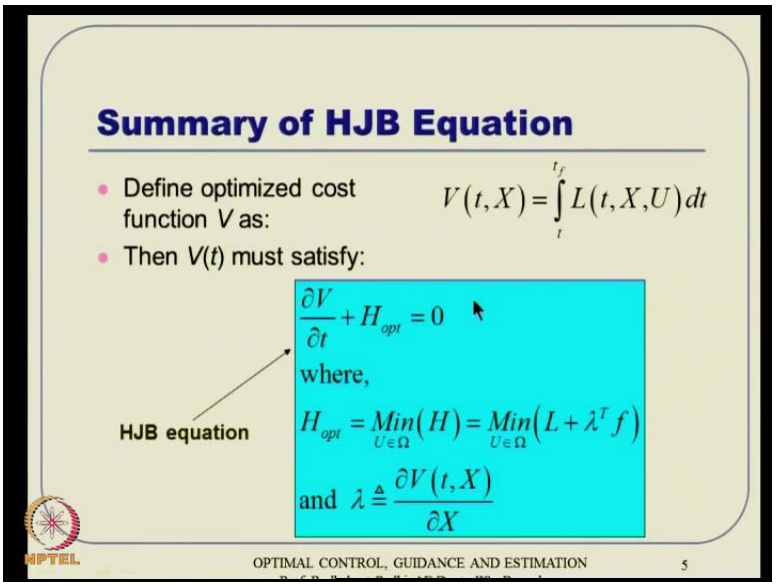
But, by definition, we are telling that to begin with, this path is optimal and hence this is not allowed. That means if you confine our problem from A to C, then this is also going to be an optimal by its own way. So, that means, any part of the optimal trajectory is an optimal trajectory and using that and expanding the cost function something like cost to go and thing like that.

(Refer Slide Time: 01:54)



We come up with this optimal control problem formulation and the associated solution like this. We started with minimizing this cost function, there is no terminal penalty remember that. Subject to this non-linear state equation with initial condition and final conditions like this, where U now happens to be kind of an admissible set, where it cannot be arbitrary, but it must belong to something like an admissible set. It may be finite or can be infinite, which can depend on the situation.

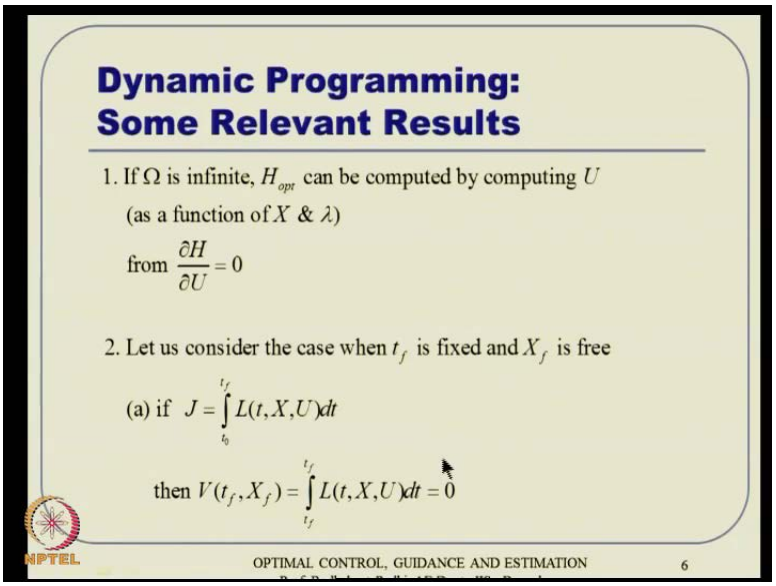
(Refer Slide Time: 02:27)



Then after the analysis we end up with these formulation that if you define something called optimal cost, means that these cost function is evaluated on using optimal control trajectory as well as optimal state trajectory, then that we define as optimal cost. This optimal cost must satisfy these equations, which is famously called as HJB equation that means Hamilton-Jacobi-Bellman equation.

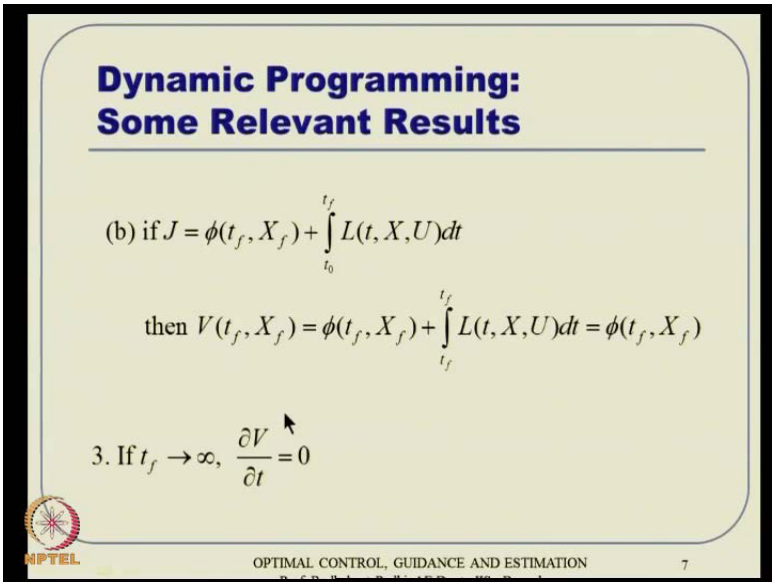
So, this Hamilton-Jacobi-Bellman equation, you can see that these Hamiltonian is not any Hamiltonian, but these Hamiltonian is Optimal Hamiltonian, which this Hamiltonian evaluated on the optimal path. And on the way the Hamiltonian takes lambda also; lambda happens to be Del V by Del X here and then you can see that H of t is function of lambda and lambda is a Del V by Del X; that means, this equation is essentially a non-linear partial differential equation.

(Refer Slide Time: 03:24)



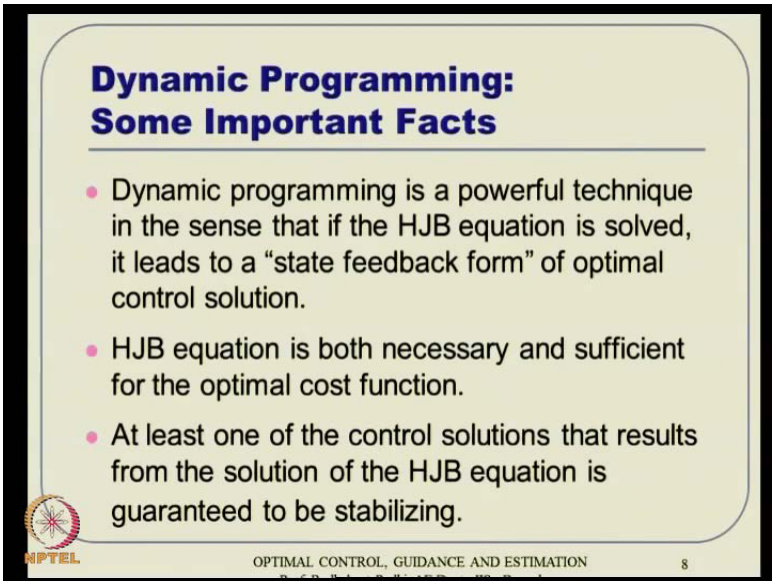
So, that is the observation and to get into these further observation is if omega is infinite; that means, there is no control bound per se, then H opt can be computed by computing this for standard things like del H by del U equal to 0. Otherwise, you have to account for these control bound and then find the solution. Now, you can tell if you consider that t f is fixed, then V of t f, X f, as the integral happens to be from t f to t f; that means, integrally is 0. So, that will give us the boundary condition.

(Refer Slide Time: 03:57)



Now, if there is a terminal penalty also, the integral part will become 0, but V of t f, X f will be phi of t f, X f and that is only the change; the HJB equations remain the same. The boundary condition happens to be instead of 0, it will be happen to be the same function evaluated at X of f. And then another observation that if t f goes to infinity then Del V by Del t is 0. So, we have only that part of equations, which is coming from H opt (Refer Slide Time: 04:22) equal to 0. But, still remember it is still a partial differential equations and lambda is a vector valued function, Del V by Del X. If it is a scalar problem then only it becomes something like ODE problem after that.

(Refer Slide Time: 04:40)

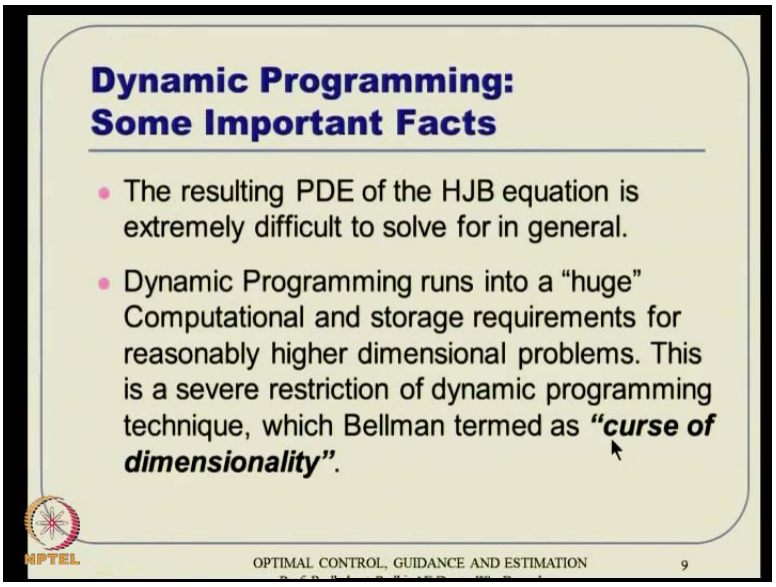


So, this is how it is. Then some fact also to be studied on the way that a dynamic programming is an essentially a powerful technique in the sense that if the HJB equation is solved, then it is essentially leads to "state feedback form" of optimal control solution. The second is HJB equation is both necessary and sufficiency; necessary and sufficient for the optimal control function. For optimal control problem, we do not have to realize. Once you solved for HJB or we demonstrate that some solution satisfies the HJB equation. Then it is both necessary and sufficient condition.

Then also that it even though HJB equation is non-linear PDE, essentially it means it can have multiple solutions. Then this result tells us that it is also at least one of the control solutions that results from the solution of the HJB equation is guaranteed to be stabilizing. So, we have a final set of solution out of which will be able to will be able to

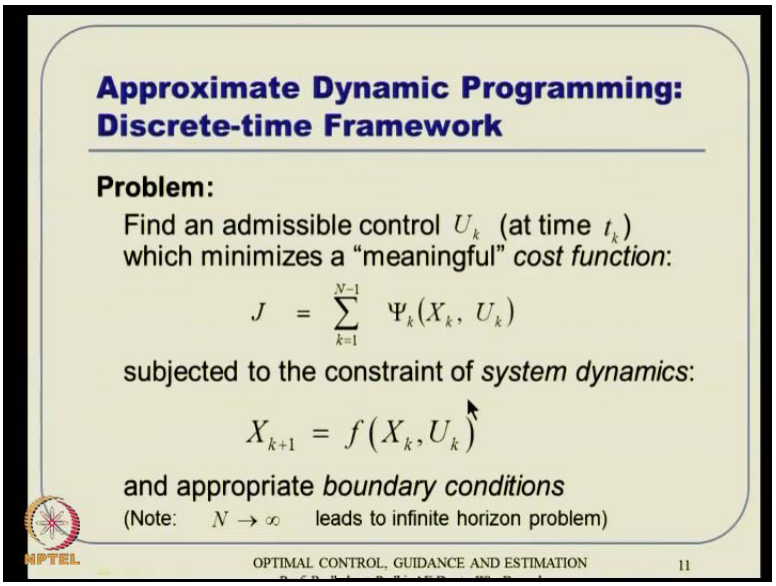
select one stabilizing solution at least and that will be our optimal control solution also basically.

(Refer Slide Time: 05:40)



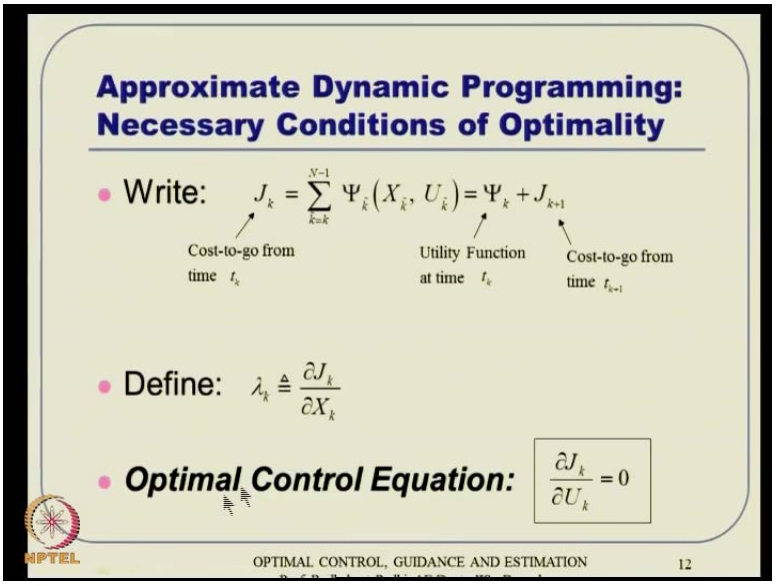
But they are difficulties also that the resulting PDE of the HJB equation is extremely difficult to solve for in general. If you really run into these numerical ways of solving these then it runs into these huge computational storage requirement. And to the extent that is the modern computers are also not useful, it runs into the computational complexity problems and all that. So, this is essentially called as "curse of dimensionality", which is the well known terminology term and defined that if control duration of control application duration is longer or here the number of states are higher; the dimensional of state vector is higher, then with these problems I mean the computational requirement grows exponentially.

(Refer Slide Time: 06:31)



So, that is not possible in general and to avoid that they are recent ideas called Approximate Dynamic Programming and then this formulation in discrete time frame work. The cost function happens to be like these and the state equation is in discrete domain where X k plus 1 happens to be like these.

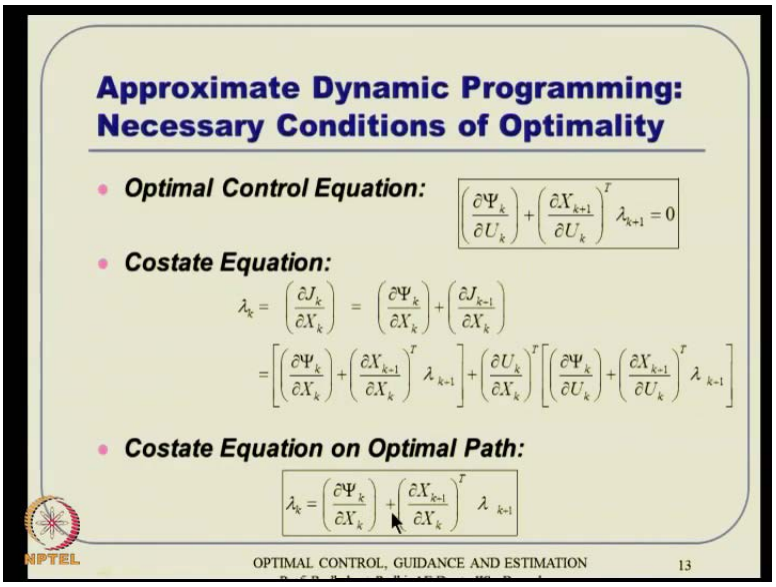
(Refer Slide Time: 06:43)



Then we write in a very similar way that we wrote for dynamic programming equations. The cost to go from time t k we can be written something like utility function at time t k plus cost to go from t k plus 1. Then defining lambda k is something like del J by del X k

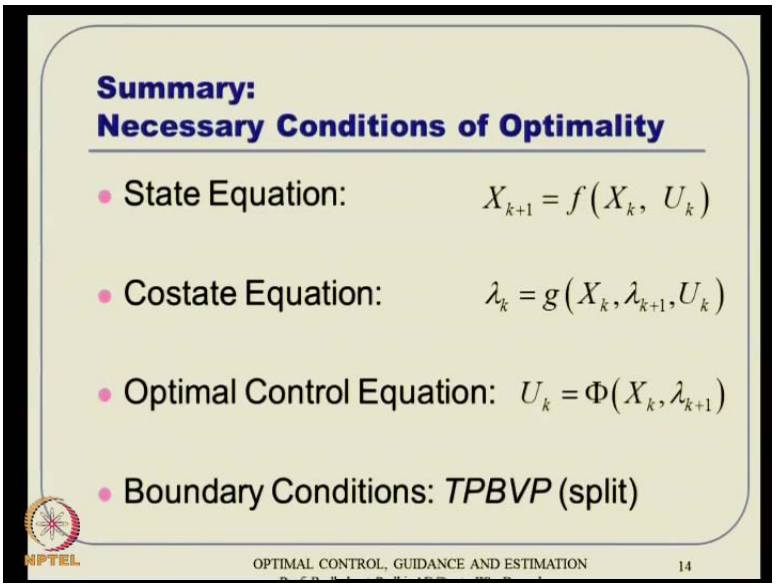
and optimal control equation that means to be satisfied; del J by del U k equal to 0 and this leads to these condition that this expression has to be 0.

(Refer Slide Time: 07:04)



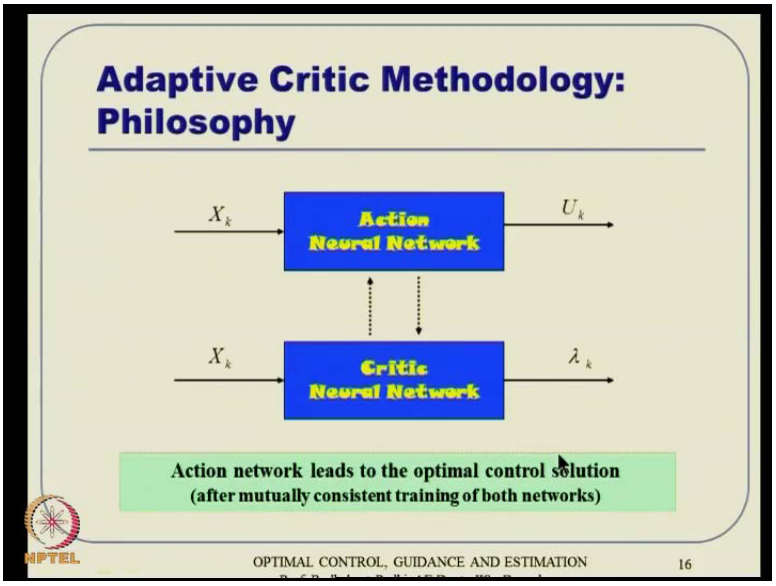
Now, lambda k plus 1 is coming now here. Similarly, the Costate Equation, if you analyze that turns out to be like long expression, but on the optimal path this is 0 and hence these expression is 0. So, we will end up with only that. So, this is the costate equation on optimal path.

(Refer Slide Time: 07:28).



Essentially, we will end up again with these state equations, costate equation, optimal control equation, boundary conditions thing like that, even though, if we start with these dynamic programming ideas. We will end up with simultaneous sort of formulation of necessary conditions, where we had the state costate optimal control equations essentially. Using these we also talk something like Adaptive Critic and single network Adaptive Critic Design on how do you make use of these approximate dynamic programming designing .

(Refer Slide Time: 08:02)



The philosophy was like this. We have two neural networks; one takes the input X k and gives the output U k, the other takes the input X k and gives the output lambda k . Now, after mutually consistent training, eventually this action network will capture function, the optimal control function that lies between X k and U k. Hence, after successful training, we keep using this, which will produce optimal control relationship as a function of X k. So, that was the idea there. So, action network leads to the optimal control solution or of course after, mutually consistent training of both the networks.

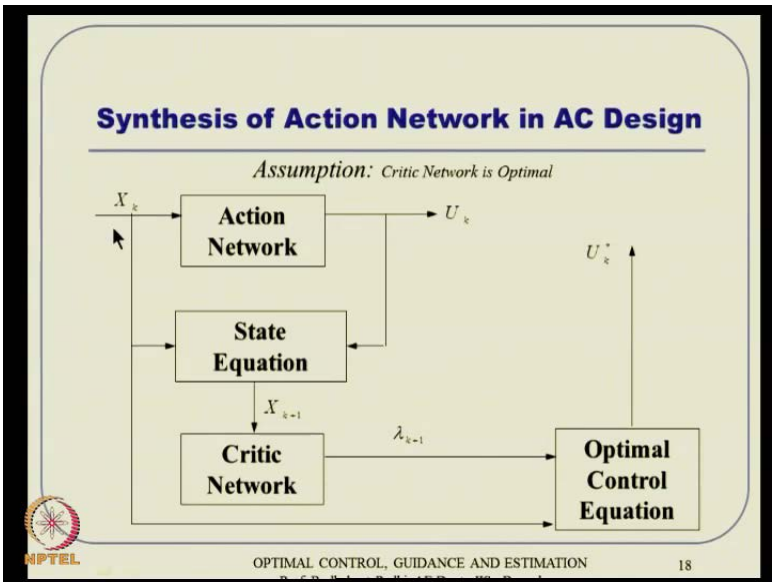
(Refer Slide Time: 08:43)

So, there are advantages and the first advantage was it is applicable to non-linear problems in general and that does not require any approximation like linear or quasi linear like that. We do not need any of that. The second is that it is possible for the solution is valid directly for a large number of initial conditions. Essentially, it leads to these feedback optimal controls in the domain of interest. It gives some sort of computational load sense and it is feasible because it do not really run into these huge computational requirements of running into months and years.

Here, it is possibly to do that in a few minutes or maximum like tenth or twentieth of half an hour. So, it is feasible computational load even for neural network training. Remember, after training it is just be evaluation of a close form of expression. I mean that is certainly possible to use in a real time , but even during the training it does not take too much of time, it takes maximum about, I mean, I will put something like 20 to 30 minute to train as network.

Then it is self contained methodology; it does not talk about relying on others. Essentially, it is possible to use it for real time control applications, ultimately after training, all that is evaluation of something like a close form expression from this sector, using and utilizing this train in a neural network is a close from function of X k, so that is possible to do online computation basically.

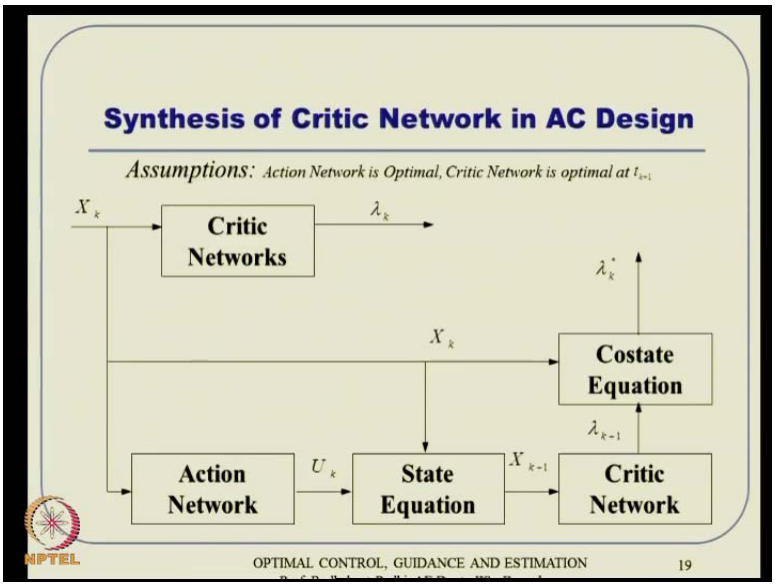
(Refer Slide Time: 10:14)



How you do this synthesis? We have come up with this structure where we assume that while action network training critic network is a optimal and vice versa. So, while action network training, we randomly select X k and pass it through action network and that way you get pass U k, utilizing X k and U k in the state equation, we get X k plus 1. Utilizing that in the critic network, which is assume to be optimal you get lambda k plus 1 and again utilizing this X k and lambda k plus 1 you get the desired U k star.

Now, you remember that out these three conditions of state, costate and optimal control, you have use state equation and an optimal control equation while training the action network. Now, this is the U k where as this U star k is the desired U k. Now, if these two are very closed to each other and essentially the difference if you do take the norm is to be small number; if there close to each other then you stop; that means, of the action network is trained. If not, if you keep training, I mean that in other words, take this as input and take that is output, the desired output and then rejects the weight and then repeat the cycle with respect to different set of X k. this trainings are typically down in batch mode training where you do not use only one particular X k, but generate bunch of X k, let us say 100 X k together and things like that.

(Refer Slide Time: 11:32)

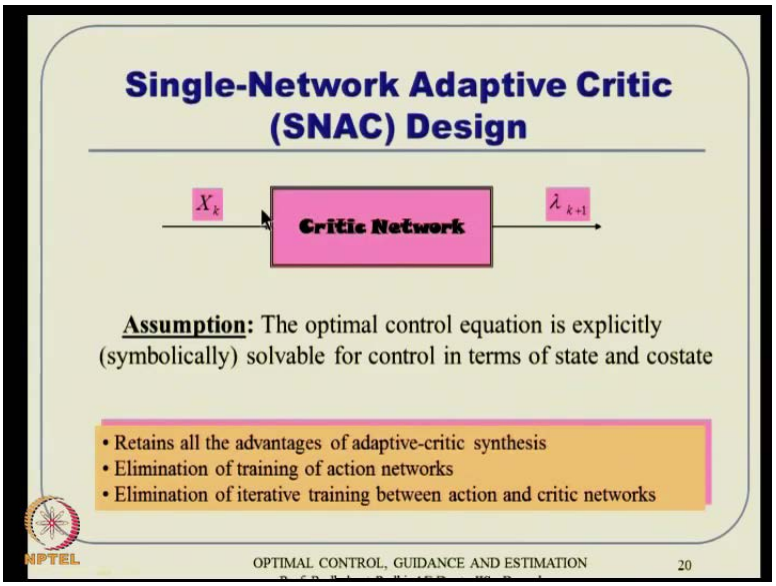


Now, going back to the critic training and that structure is something like this. Here, we are assuming that action network is optimum and then the critic network at the next time step ahead is also the optimum. That is justified because the boundary condition if you see in costate that is given at the final time. So, if you come from that side of story, it tells that I will start from equal to t f, the close approximate it to zero basically in the regulator problem and then the utilizing this feat in two philosophies, that action network is optimal and critic network is also optimal for future time.

Then I can come up with this algorithm, I mean this idea of training this. Start with a set of X k, randomly selected again and get my actual lambda's, but the same X k can be used in action network and then get U k and once X k is also there, U k is also there. I can use that state equation an get X k plus 1, then use that in critic equation, critic network, you get lambda k plus 1 and utilizing X k and lambda k plus one, I will get the desired lambda k . So, these two are close, I will not train, the network is already optimal and if not I will try this as input and that as output and train it.

So, I will keep on doing that and then go back to action network training, come to critic training and things like that, ultimately when there is no cycle training improvement, we assumed that is the training is consistent and then we tell whatever converge network from there that gives us the optimal control for that particular plan.

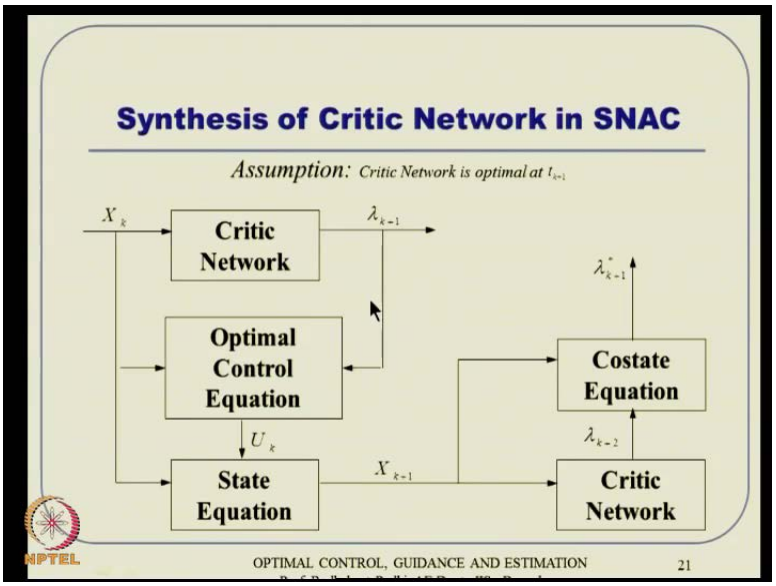
(Refer Slide Time: 13:02)



But moving on we also had this Single-Network Adopted Critic Design where we do not really have to do two networks for all type of problems. For a class of problem, where the optimal control equation is explicitly solvable or symbolically solvable for control in terms of state and costate, all that I need to know is to get my lambda k plus 1. X k is known and some of I get lambda k plus 1 then I can utilize this closed form of solution that is available with me to evaluate my U k.

So, I really do not need to have an action network training as a part of a training procedure. The advantage is it returns all the advantages of adaptive critic synthesis. Essentially, eliminate the action network and also eliminate the iterative training between action and critic. So, that is also gone there. So, this leads to several advantages or improvement, especially it save a lot of computational time even for off line training. It also eliminates the approximation involved in the action network that is no more required and because action network is also not required, the cycle training is also not required. So, all that you do is one set of training for the critic network and you are done.

(Refer Slide Time: 14:17).



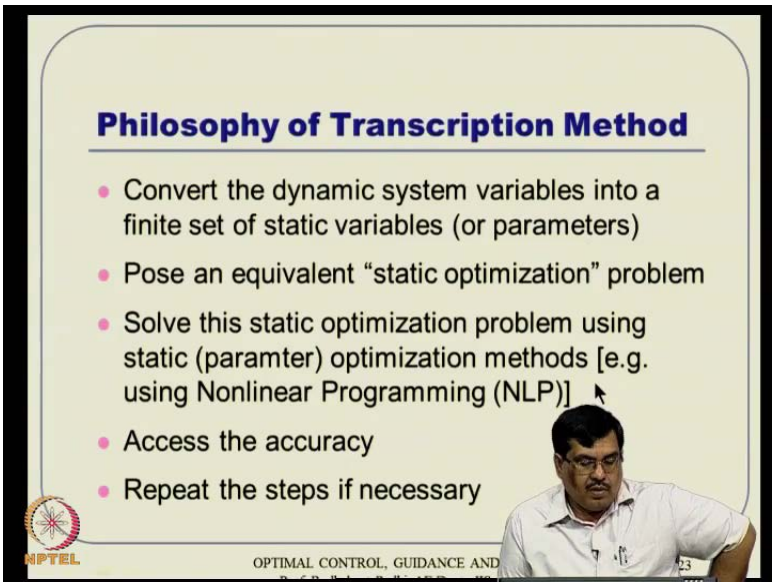
So, that is an idea of single network adaptive critic. How do you do it? All the three necessary conditions are use in these settings. you take X k and get lambda k plus 1 and use both X k and lambda k plus 1 get U k. Now, you can get U k I mean use X k and U k to get X k plus 1. Once, you have a X k plus 1 you can lambda k plus 2 assuming the critic network is optimal for the future time. Then X k plus 1 and lambda k plus 2, if you used in costate equation, you get decide lambda k plus 1. Again, if these two are close, do not train and training is already done. if not then you train this network again in a batch training sense .

Here, you can see if optimal control, state equation and costate; all three are getting used and by using this structure you are also using this structure, your also using this boundary condition, because in the future time we are assume the critic network is optimal. That is guaranteed by fast trainings of the network using the boundary condition. So, it start with the boundary condition that lambda t f whatever of that del phi by del X n for finite time or infinite time problems you that lambda t f goes to zero, but the state also close to 0. That means if you start with very small numbers around 0 and then if the corresponding lambdas should also be close to 0; that means, LQR solutions is valid in that sense.

So, utilizing that LQR solution, if you to pertain the network properly then the boundary conditions enclosed in this network and that validates that justification counts form that

sides. This is how it is done and if it is not close to each other, again keep training and keep validating. So, that is how the network training proceeds.

(Refer Slide Time: 16:05)



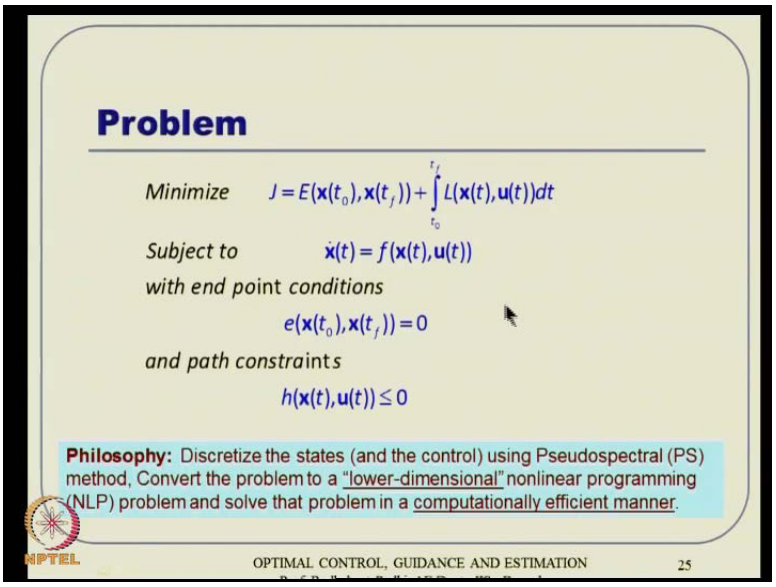
Then we moved on and we also had a glance of that is called transcription method that people have solved it. The philosophy turns out to be something very close to static optimization really. So, what peop\ what what one idea I mean what is this idea tell as is something like this. We convert the dynamic systems variable into a finite set of static variables or what you call as parameters now and pose an equivalent static optimization problem. Solve this static optimization problem using this various a static or parameter optimization methods using non-linear programming technique.

Variety of methods are available in literature, which you can recite now and then solve this and then go head getting that. Then access the accuracy and the repeat the steps if necessary. The question is how do you pose the static optimization problem depends on a variety of ways. Then we started with idea of finite difference and then you can use this Euler integration method formula and then set of grid point solution and all that we came up with for the state equation. And for the cost function we thought we will take trapezoidal rule something like that where we can discretize the cost function. Now, it will, but at the every grid point, every node point or every grid point the state equation is satisfied as a kind of algebraic constraint of variables.

So, if you see that for better accuracy we need lot of grid point and then if you have lot of grid points, then essentially it will sub the side dimension optimization problem, which has its own difficulties. So, this kind of issues, which comes to focus then people have use something like sparse of algebra or then start with something like grid refinement and then variety of ideas where you start with course grid and then once you get converge solution for that, refine the grids and again start solving it and things like that. That is grid refinement and then we have this sparse matrix algebra where you can eliminate this algebra around lot of zeros.

So, using that people have come up with somewhat efficient algorithms, but still they are I think they are not suitable for online sort of application where you really need very fast computation. But, then we had this glance of these evolving methods called pseudo spectral transcription which hold lot of promise in my view.

(Refer Slide Time: 18:28)

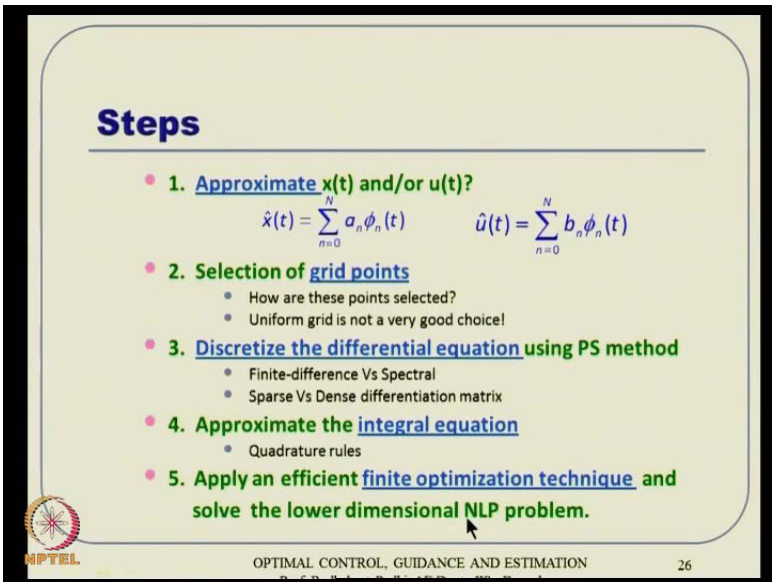


And the idea was something like this, we wanted to have a very generic formulation while want to minimize this generic cost function with this terminal penalty and path penalty, subjected to the cost function and with n point conditions, both in equality sense as well as path constraints and equality sets. The whole idea is to have an equivalent discretized formulation, by using this is so called pseudo spectral descritization. But, essentially the idea here is to convert the problem into a lower dimensional non-linear

program problem. The dimensionality of the problems should not grow actually. We should able to do the job with very less number of grid points.

And then we are not interested in sparse matrix formulation sort of things. Essentially, it can lead to dense matrix algebra basically. So, it is lower dimensional, but we do not waste our computation by having a sparse matrix formulation sort of things. That is the whole idea of how do we go further and people have come up with various things for last 10 years, from about 2000 onwards this gained quite big momentum.

(Refer Slide Time: 19:32)

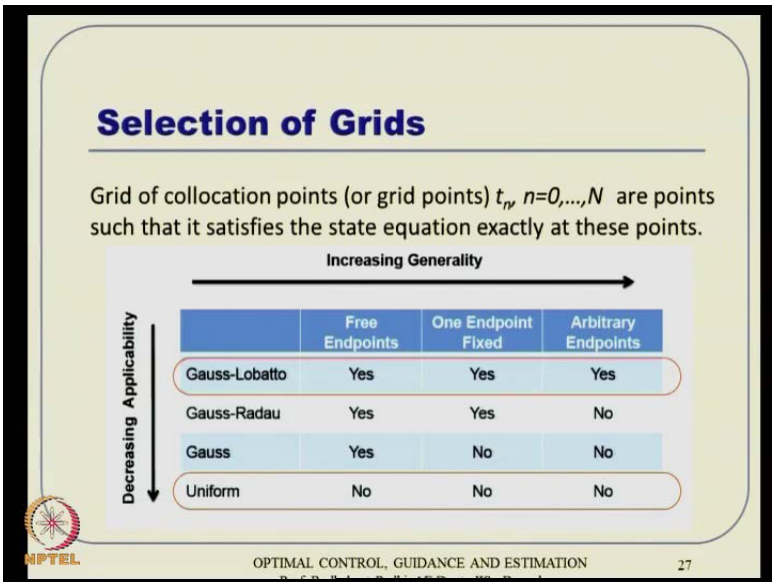


So, one idea is like this. We can start something like this. We can have a control parameterization in the form of basis function that typically (( )) polynomial, legendary polynomials like that and control also can be parameterized. Sometimes, people tell control need not be parameterized because if you parameterize the part of this continuous function sort of things then essentially this discrete controls cannot be approximate it..

So, if you want to do want that kind of generalization with control bounds and all that then probably that not a very good idea. That is not the point I have, the point is to see the philosophy. So, we have this start. We can start with control I mean state using approximated that way and control using approximated that way with a set of basic functions. Now, we have to select a set of grid points and how is this grid points selected and all sort of things are sort of questions. But the whole idea here is we have discretize the state equation and costate I mean cost function by utilizing this thing as well as the

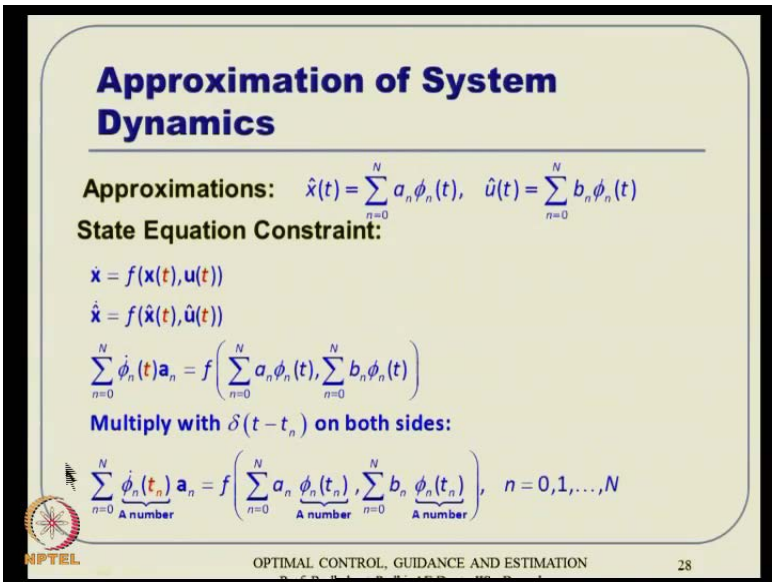
system dynamic and then we formulate this lower dimensional non-linear programming problem.

(Refer Slide Time: 20:42)



So, selection of grid points has this something called collocation coincide. So, they are points such that it satisfies the state equation exactly at these points. So, they are called collocation points or simply called grid points. And there are ideas like what kind of grid points you want to select because uniform grid points are no more good for this kind of things. As one idea is to have if you select uniform things then if you see free end points one fixed end points, arbitrary end points, then none of the boundary conditions are satisfied there. But, slowly we can see Gauss flowing I mean Gauss collocation points or Gauss-Lobatto, Gauss-Radau, like those are available in literature. Now, an increasing generality, I mean satisfying all the boundary condition and variety of things, now the universal recommendation is something like Gauss-Lobatto points.

(Refer Slide Time: 21:49)

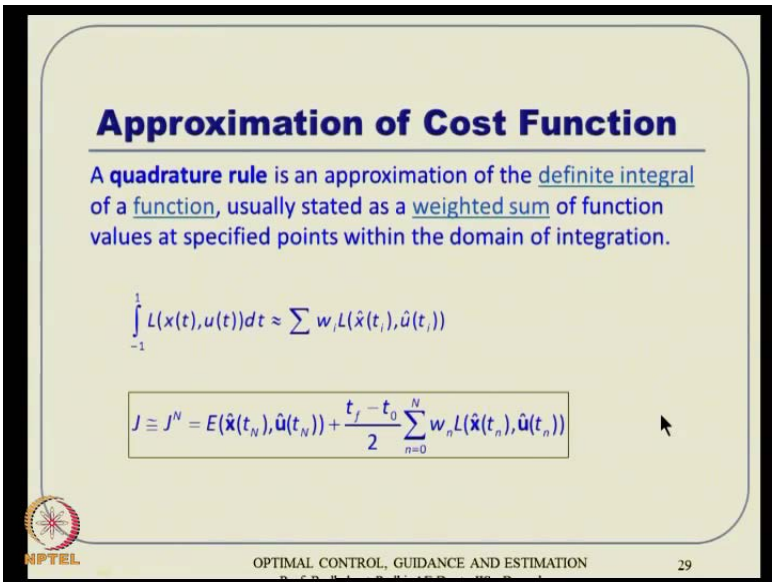


So, we can select something like this. Now, proceeding further with the algebra like that. It essentially is a non uniform grid point side where the problem dimensionality gets reduced quite a lot and satisfies the boundary conditions well. Now, coming back, you have these approximations, so the state equations constraints, once you start something like this and then we are interested in that, I mean we can substitute X dot as X hat dot. then see if this X hat dot coming for this expression takes the form of something like this where this phi n of t is essentially function of time, but it is an explicit function of time. So, phi dot of t is essentially a number once evaluated at any grid point.

That is phi n dot of t is an expression of time, explicit expression of the time, and hence I can evaluate that any point of time to get a number for phi n dot. In the right hand side, where ever X is there, I will replace that with X hat. So, this expression comes here where ever U is there I will replace that with this U hat I mean this expression comes.

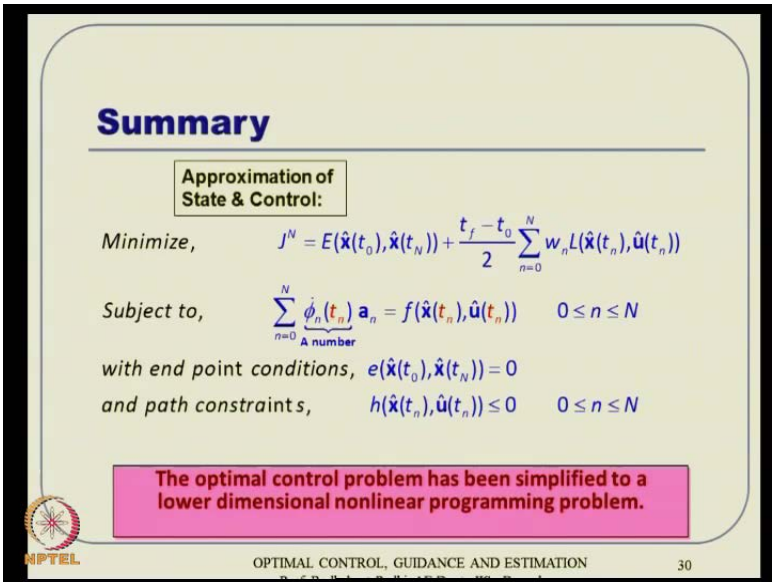
Now, multiply both sides with this delta function I mean situated at t n. Then you can essential telling that evaluate this now phi n t at that particular gird point. So, that becomes a number now and all these things become a number in the right side also. So, essentially this tells us now algebraic constraint because this has become a number; no more differential equation form and all that. Once, it becomes a number, this is a constraint equation in the form coefficients a n and b n and that is how it results in algebraic constraints starting from the state equation.

(Refer Slide Time: 23:14)



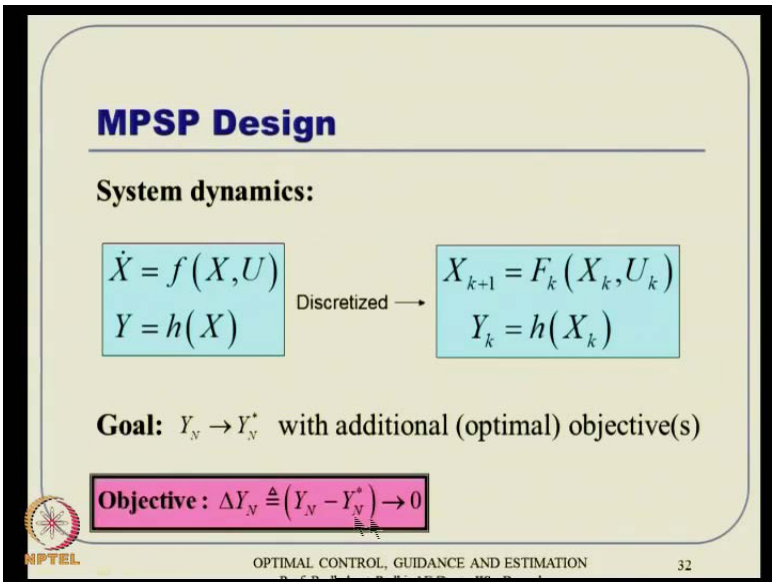
Now, coming to the cost function again similar things can be done, but there are ideas like something like quadrature rule and all that, where this cost functions can be evaluated in the much better accuracy sense basically. So, that can be also discretized in that from. Now, this goes to, I mean utilizing this constraint, algebraic constraint and this discretized cost function we have parameterized cost, I mean formulation other words static optimization problem .

(Refer Slide Time: 23:42)



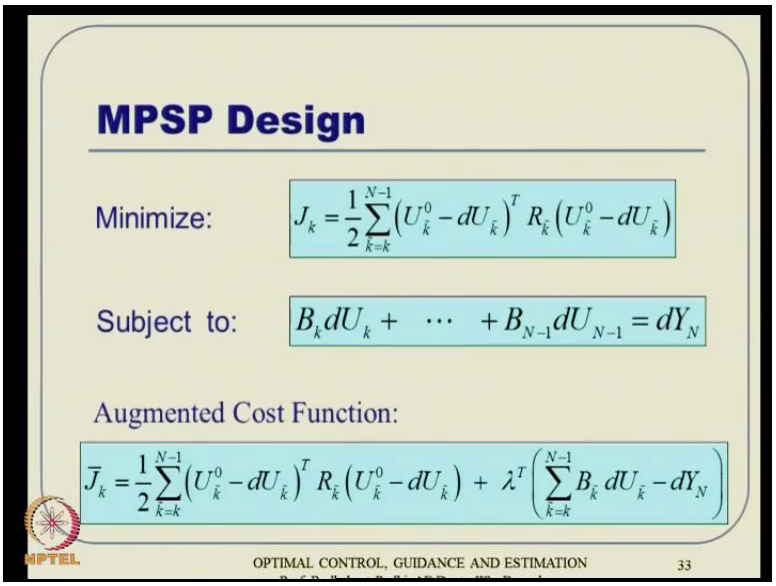
The way to solve it this static optimization problem and essentially it leads to lower dimensional non-linear programming problem and hence it is computationally efficient. There are varieties of application problems, variety of convergence guarantees, and variety of analysis tools, generalization, and particular problem, everything available in our last ten year in variety of literature.

(Refer Slide Time: 24:13)



Then we moved on this idea of MPSP, model predictive steady programming design where we started with some problem formulation like this. We had this system dynamics and this output equation. So, if you discretize that it takes this form and the idea was that t equal t f; that means k equal to n, our objective was this output Y n should go to a particular set Y n star the particular values. That means if I take this around delta Y n that should go to 0 basically.

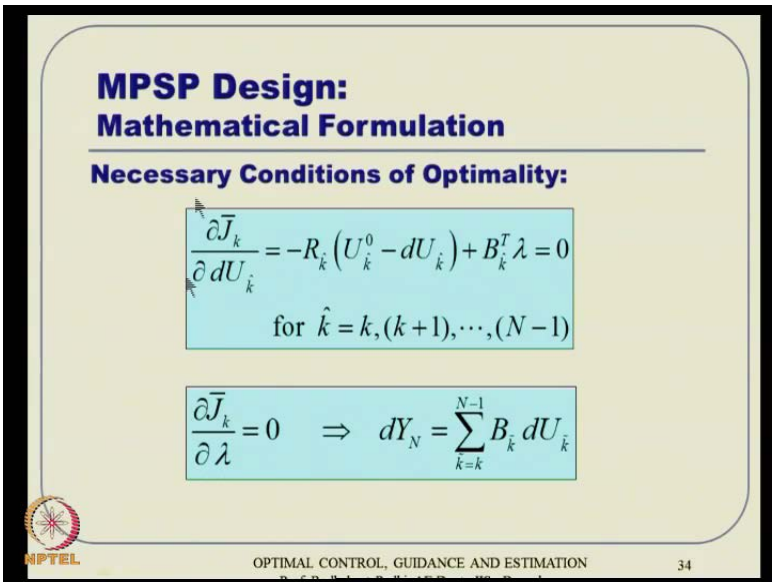
(Refer Slide Time: 24:36)



Now, this essentially leads us to this constraint formulation, if you start with a guess history and try to evaluate this and then finally, you have to see why this comes because at every grid point I met some error in the control and then I have tell that as dU1, dU2 and all those are error over about my previous guess history. Then essentially it leads to this kind of a formulation where this linear sort of constraint comes in to picture. If I want to minimize the quadratic cost function, I can do it that way. It essentially starts from t k; t k is my current time in that sense.

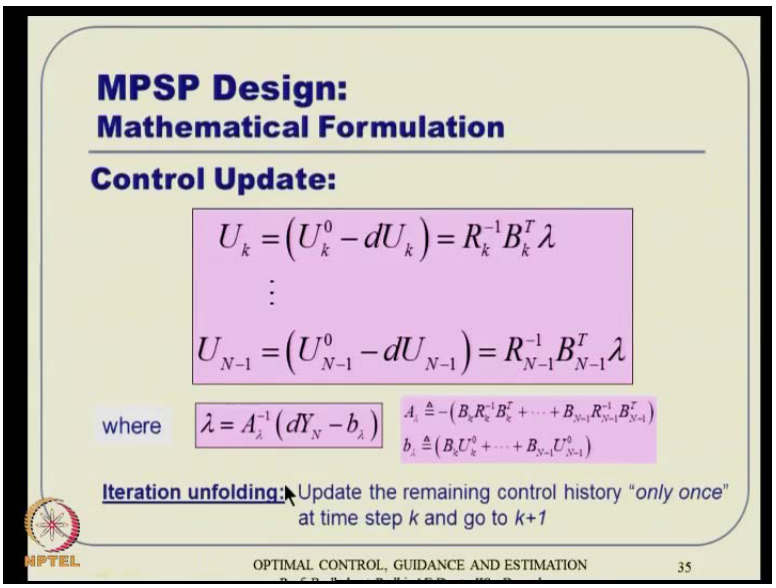
And then we have this standard quadratic cost function with linear constraints. So, it is possible to solve this in close form. Now, the point to observer here is this is a static equation. This is not a dynamic equation and the dimensionality of this equation is also small. It is a state dimension happens to much more than the output dynamics. So, that essentially tells us that this is nothing but a parameter optimization problem where the constant alpha I mean constant lambda will do the job. We do not really need to have a dynamically varying lambda basically. So, that essentially leads us and not only utilizing that we can also get close form solution for these errors in the control, so that the iteration can proceed faster. On the way we have compute this sensitivity matrices also, but, that can be computed recurs severalty.

(Refer Slide Time: 25:59)



So, because of all this it leads to a very fast computation essentially and hence we think that it is possible to kind of utilize it for online applications, essentially. The necessary conditions have to be is this a static optimization problem, so the very standard way of approaching that is to formulate an augmented cost function and then utilize this, I mean use these necessary conditions of optimality. These partial derivatives have to be 0.

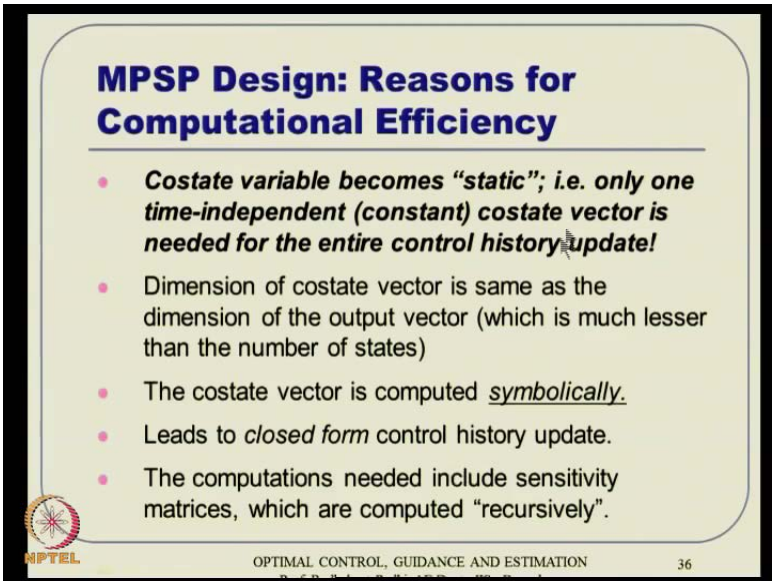
(Refer Slide Time: 26:26)



So, solve these two equations and then that can be this. Then arrive at this kind of control update I mean formula utilizing these lambda and lambda can be given something like

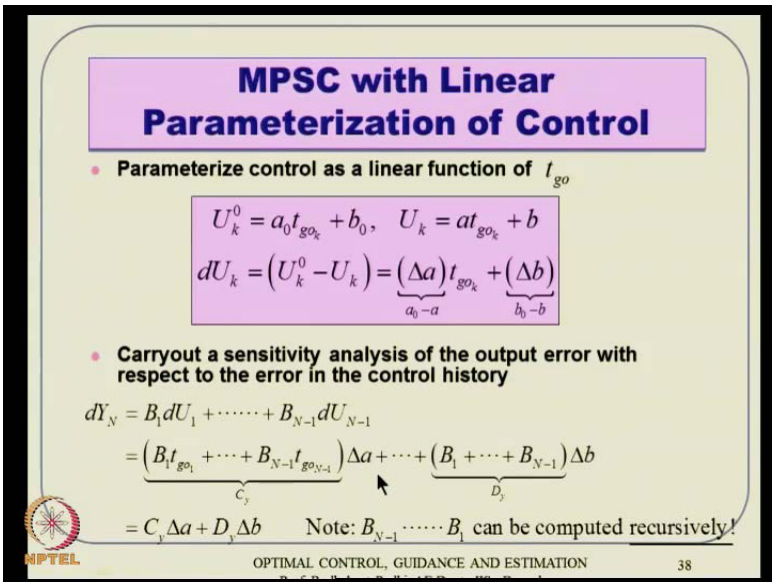
this where a lambda and b lambda can be computed that way. Also if somebody wants a finite number of iterations at any grid point then this idea of iteration unfolding can be exited there.

(Refer Slide Time: 26:45)



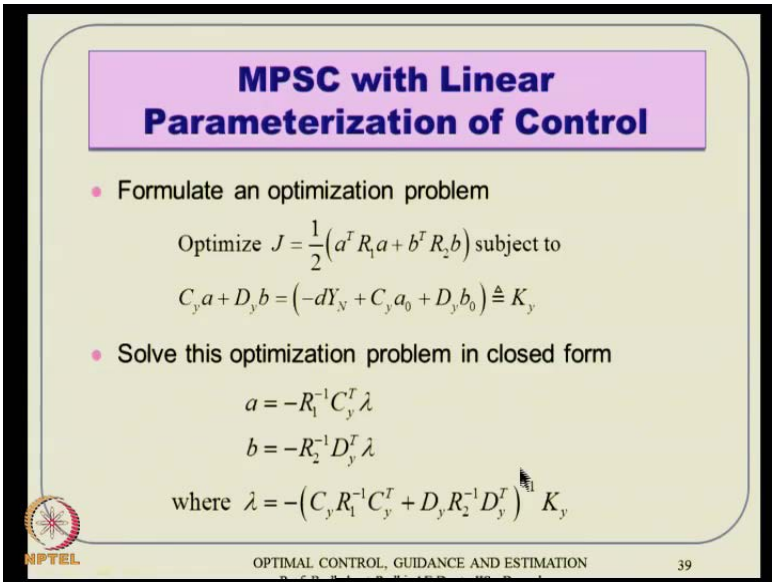
So, there are reasons for this computational efficiency and I have told many of that. So, this costate variable becomes static and that is the major point I mean the dimension of the costate vector is also small. The costate vector can be computed symbolically in closed form solution. Then this computation of these recursive matrices, I mean computation of sensitivity matrices can be done recursively. Because of these, this method happens to be computationally quite efficient. Then we thought of extending this idea to something called model predictive spread control approach also,

(Refer Slide Time: 27:19)



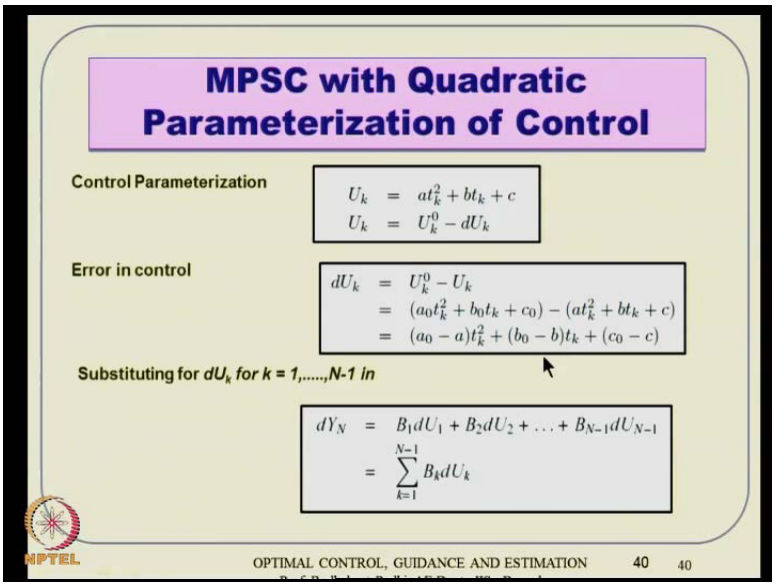
Where the control is parameterized in the form of a kind of a polynomial expression, let us say. So, if you parameterize that in something like a linear or quadratic thing like that, essentially it further reduces the dimensionality of the variable, pre-variables and also guarantees control smoothness by enforcement. So, utilizing this idea, we come up with I mean we put this expression back into this error equation and essentially we will kind of constraint the freedom to the parameter freedom only basically.

(Refer Slide Time: 27:48)



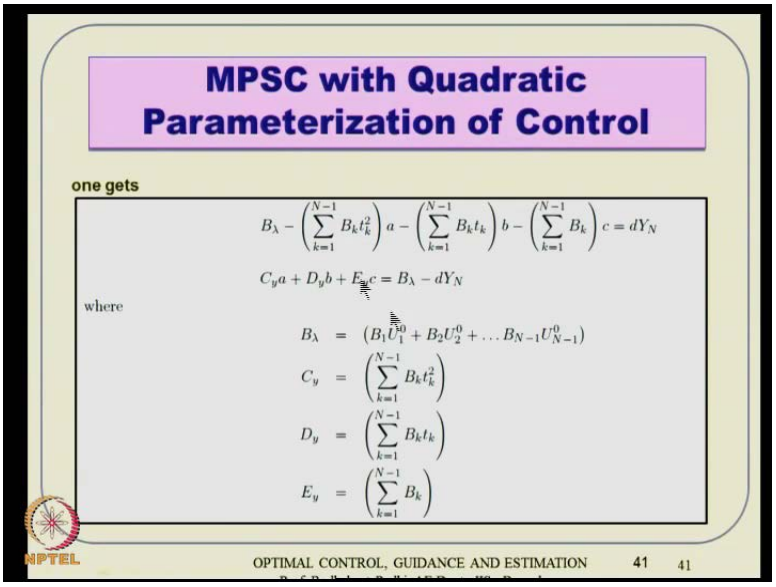
Then it all is ok, we can formulate a parameter optimization problem in the framework of a's and b's and then we get a quick solution from that side of the story.

(Refer Slide Time: 27:57)

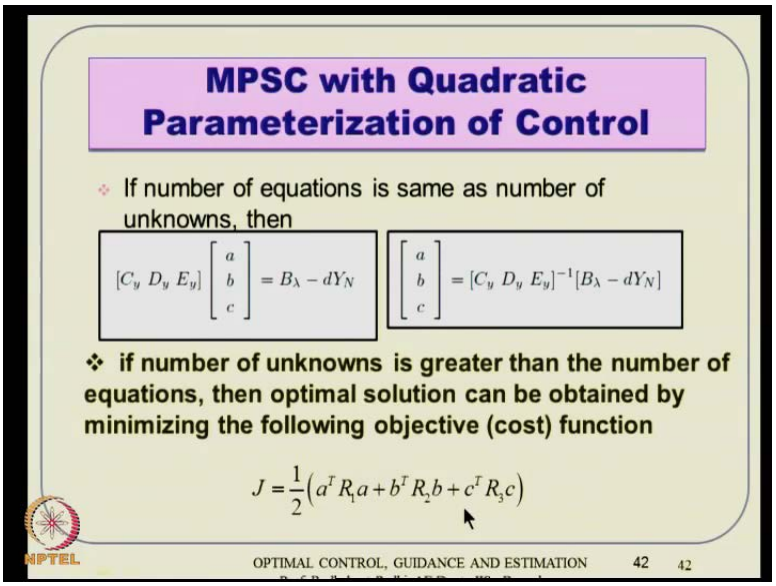


And then the same idea can be done for quadratic problems also and this quadratic formulation d U k cast in that way. There errors in a b c and then we can come up with this constraint equation.

(Refer Slide Time: 28:08)



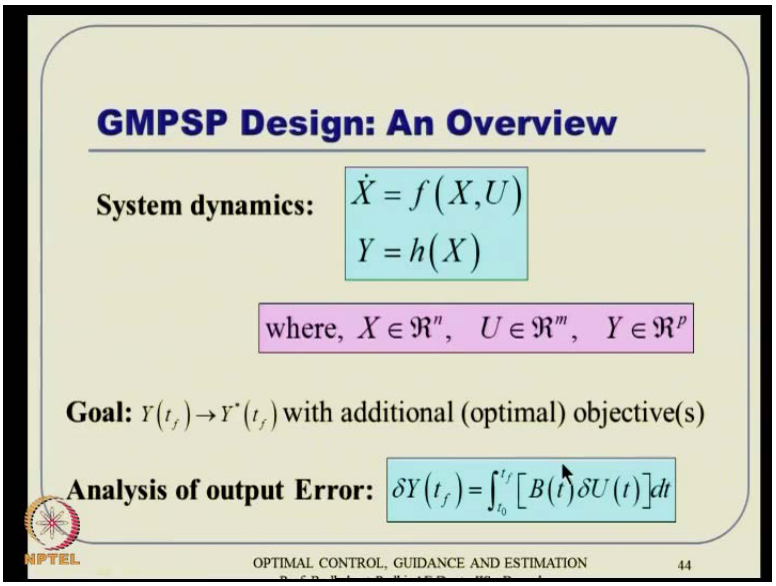
(Refer Slide Time: 28:11)



Now, the question is if it is a square equation; that means, we have something like three variables and three constraints then you can directly solve it like this, but if it is of only two constraints and you have three freedoms like that, you can talk about some optimization also. This optimization if you select a quadratic function like that in the set of parameters. And then I mean it leads to this fast computation of this enclosed form also.

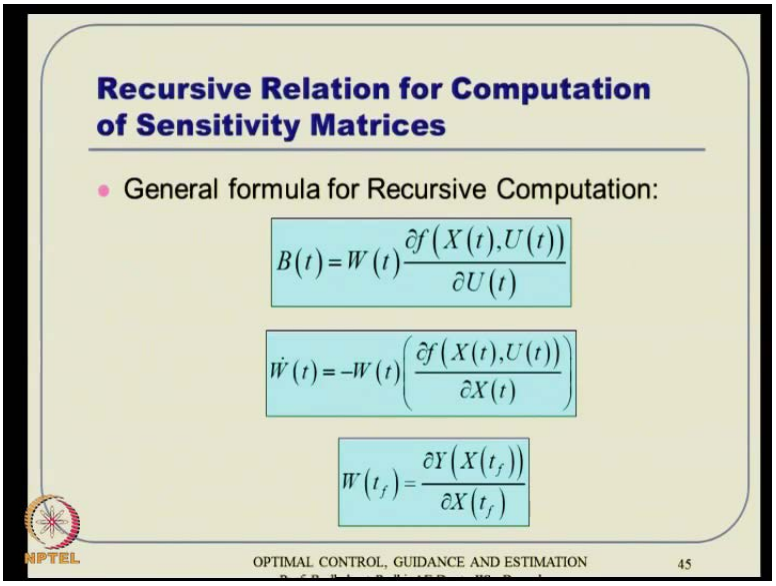
So, very quickly you can update these parameters and proceed further. So, that is the idea of spread control and you have given good real life example problems of missile guidance and all that as part of the lectures in this course. Then you entered and thought about putting this very recent idea that one of my students have proposed in his Phd is something called generalized model predictive static programming; generalized MPSP. That is the whole idea that somehow we do not want to start with discretization basically. We want to do all the things all the algebra in the continuous time framework itself.

(Refer Slide Time: 29:12)



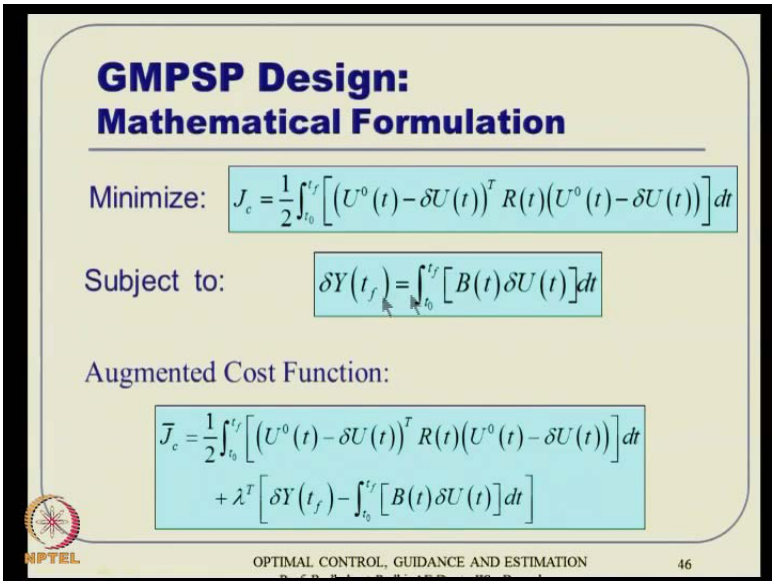
Then the formulation does not demand that you start with a discretization. So, start with the original system dynamics in continuous time and we have this output also in continuous time. The whole idea is at t goes to t f, Y of t f should go to Y star of t f. Then analysis of this output error essentially landed up with some expression like this. The details are there in the lectures and you can see the corresponding lecture you can find that. Then if you have this then, this B of t has to be computed. This is sensitivity, a kind of a time variant sensitivity matrix.

(Refer Slide Time: 29:51)



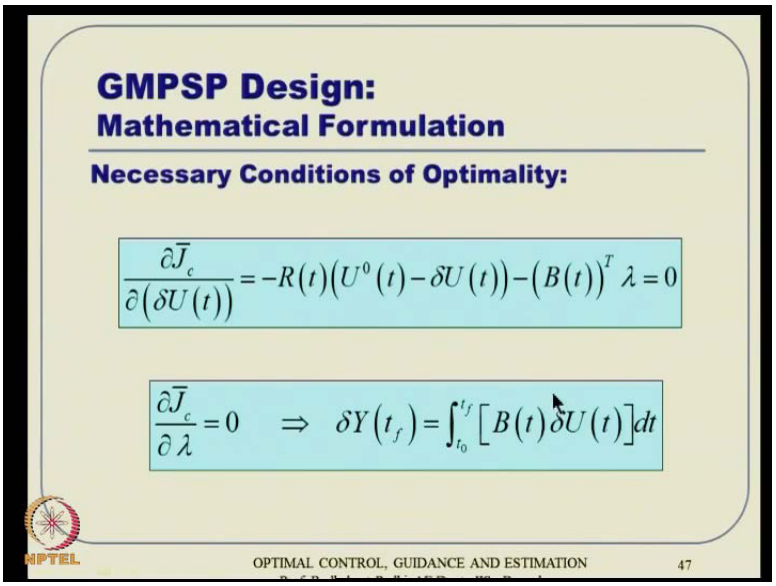
And that can be computed like this; B of t is nothing, but W times this one where W whereas, W of t can be computed backwards. So, this final boundary condition turns out to be like this and W dot turns out to be like this. You start with this final boundary condition and backward integrate it from t f to t not, then evaluate this B of t all over. So if you do that essentially it is called as very closed MPSP, but it generalizes that, so that you do not get confined to various kind of discrete formulation and all that way.

(Refer Slide Time: 30:24)



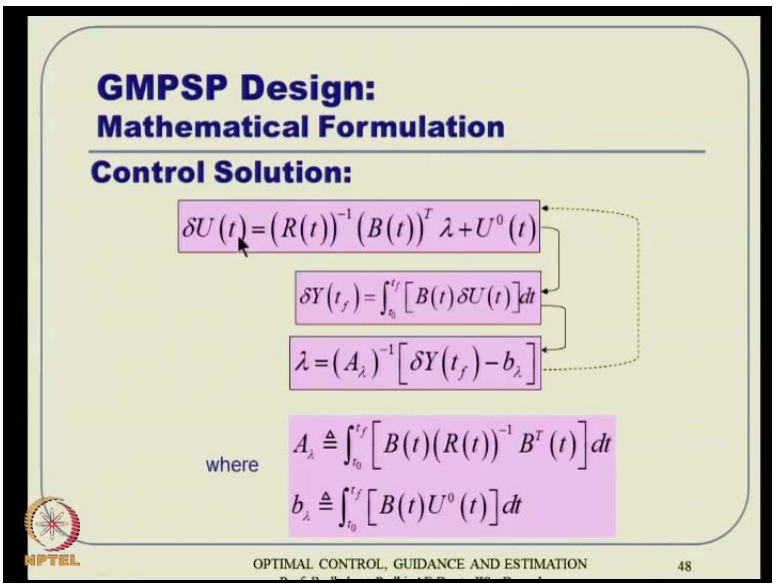
So, the idea here is again you can have or you can formulate a quadratic cost function control variable for minimization, subject to this constraint equation now. How this equation came and all, I encourage you to see my other lectures for details. Then if you solve this parameter I mean this optimization problem what you see here, then essentially you have to have this. Remember, this is not necessarily we are talking about like parameter optimization problem here. We are still talking about something like integral function minimization and all that way. So, we have this augmented cost function, which is like this. But still this is algebraic constraint, it is not a differential constraint remember that. So, this constraint comes outside the integral.

(Refer Slide Time: 31:11)



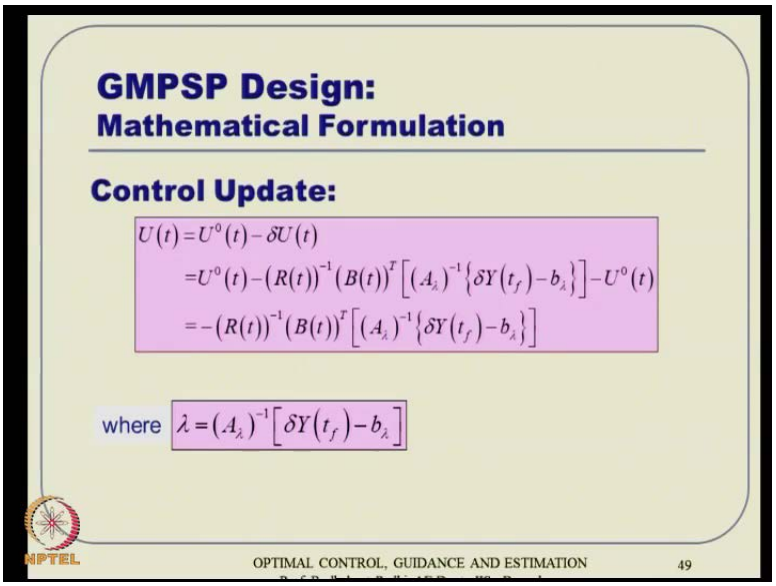
And then as utilizing the necessary conditions of optimality and in that way we land up with some expressions this way. These two expressions for which we need solve for this control and all that.

(Refer Slide Time: 31:20)



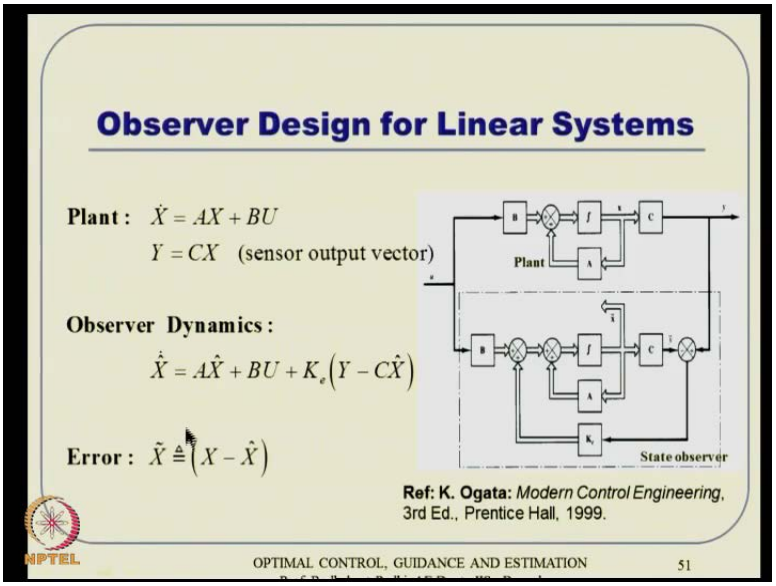
So, that can be solved. So, we start with these and that. So, you can substitute this expression and solve for lambda, then back substitute here and get this delta U.

(Refer Slide Time: 31:31)



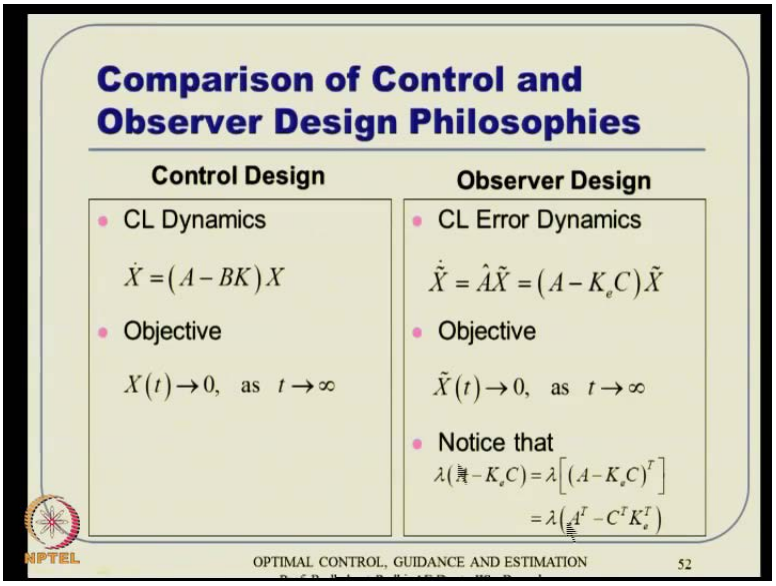
So, the control update happens to be in this form where lambda can be evaluated that way. That was generic MPSP ideas and all that. These are all a various control ideas that we talked about and little more further towards end also we talked control constraint problems and all that again back in the control design. But that point of time I mean after talking, after discussing these concepts, we thought we will go back to the estimation side of the story and have these ideas on estimation as well.

(Refer Slide Time: 32:04)



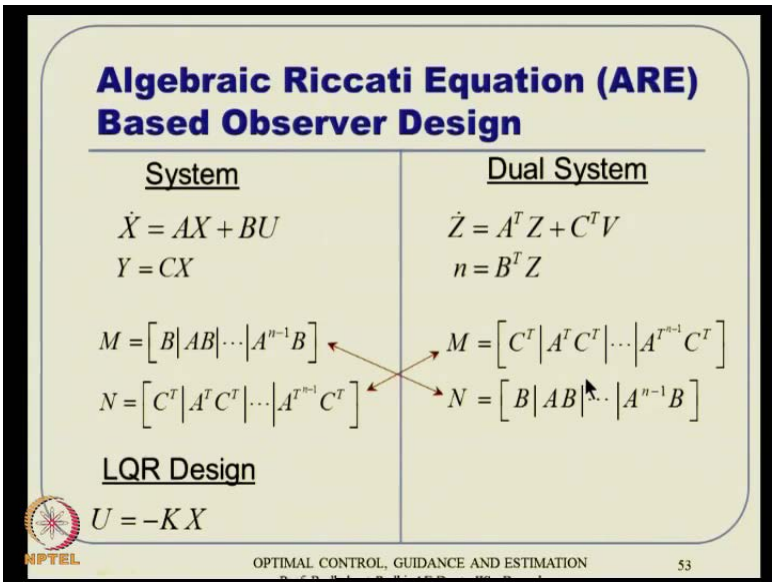
So, we started with LQ observer design and we thought if the system is linear and then we have this sensor output vector something like this, then we fit an error observer dynamics like that. The difference between this and that is this term where this is an actual output of from the sensor and this is a predicted output sort of thing. So, if you take the difference and augment these in the observed dynamics and especially if you design K in smart way, in a good way, then the error that is X tilde is X minus X dot; this error will go to 0, I mean asymptotically. That is the whole idea of observer design.

(Refer Slide Time: 32:38)



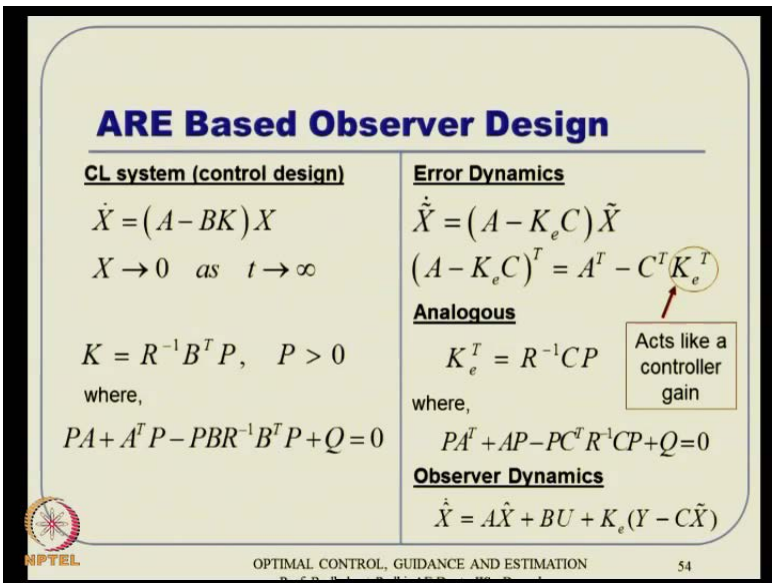
Then we analyze these two ideas; one is control design, one is observer design. This happens to be very close to each other. So, the only problem is here the gain matrix appears right whereas, this gain matrix appears left here. So, about this, we observe that we can accept this eigen value concept. So, eigen value of any matrix is same as the eigen values of its transpose. So, utilizing that fact, we can write it something like this and then we can now correspond, I mean we can now correspond to this observer to control formulas and all that.

(Refer Slide Time: 33:12)



Finally, you can have this concepts of system versus dual system; one is controllable and the other is observable, the other is one is observable the other is controllable and that kind of ideas you can excite.

(Refer Slide Time: 33:25)

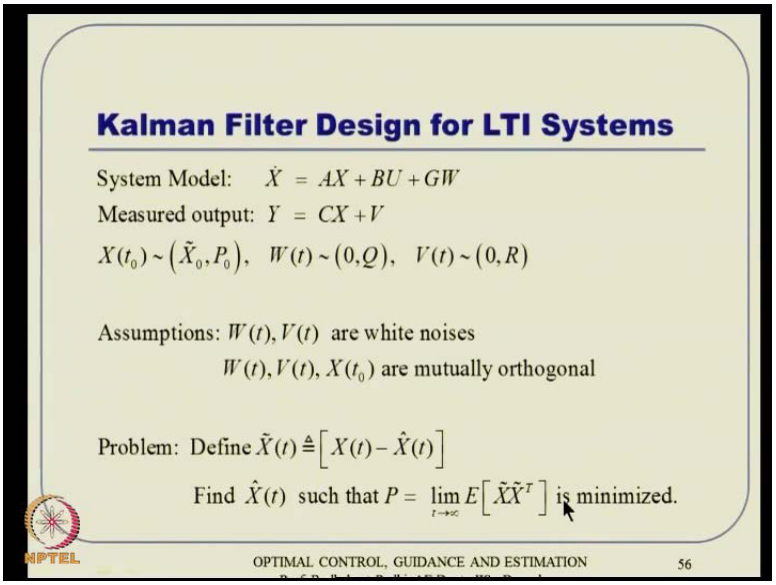


Because you know the LQR design, the final results you can actually come up with observer designs also, which talks about K e transpose and K e transpose is given something like that and hence K e will be given as something like this transpose of this entire thing. Then these are the designees and where this is the Riccati equation for

control design whereas, this becomes the Riccati equation for the observer design. So, this is called the observer Riccati equation and all that.

And also remember that this happens to be a kind of continuous Kalman filter where Kalman filter gives us much more tuning capability. Whereas, this observer technique is a filter, but it does not give that much flexibility for tuning and that is the only difference there. Then we entered and after discuss some probability theory random variables and all that to get ready ourselves for Kalman filter and then finally, switch over to, I mean going towards Kalman filter design and solve the derivations in detail.

(Refer Slide Time: 34:18)

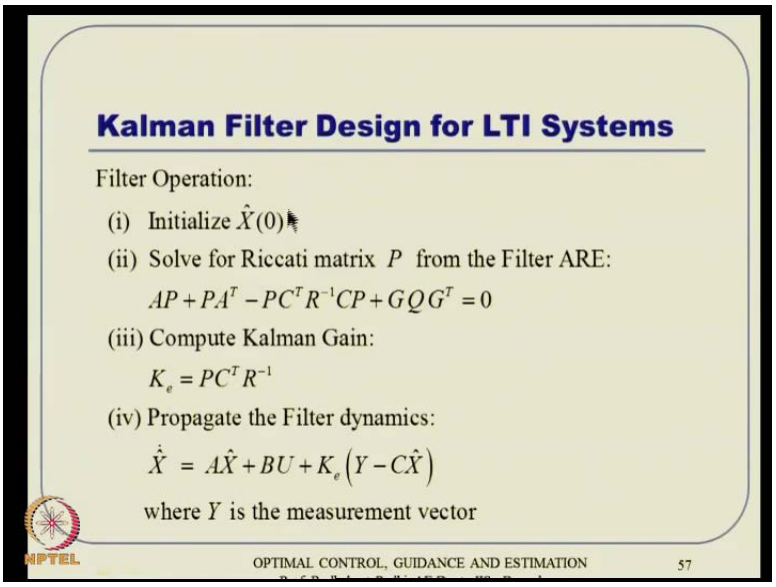


So, that is the problem here. We have this system dynamics. X dot is A X plus B X plus G W and the measured output is something like C X plus V. Here, now G W; that means, W is a process noise and V is a sensor noise, they are now start appearing in the system dynamics and we are assuming the variety of things that initial condition and the process noise and sensor noise are given by this characteristics. Especially, the process noise and sensor noise are 0 mean; white noise essentially.

We also tell that they are mutually orthogonal; that means, if you take cross covariance matrix and all that they will turn out to be 0. Then the whole idea is to define this error between the actual state and an estimated state and then we define the covariance matrix something like X tilde times X tilde transpose, expected value of that as limit tends to

infinity. The whole idea is somehow we have to find this X dot such that this error, this covariance matrix has to be minimized and we had a lot of algebra analysis and all that.

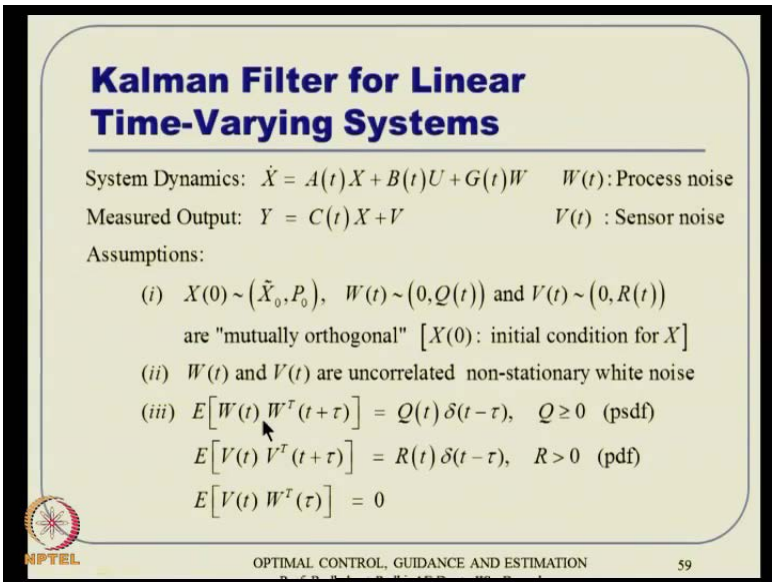
(Refer Slide Time: 35:20)



So, finally, it gives us something like this. We have to initialize with some guess value and then it lands up with this filter Riccati equation in this form very close to observer equation, remember that. If you see this observer equation here, and then if you see this filter equation here is very close, only this G times G transpose is not there before. Now, this process noise influence matrix whatever this comes here it will start appearing explicitly.

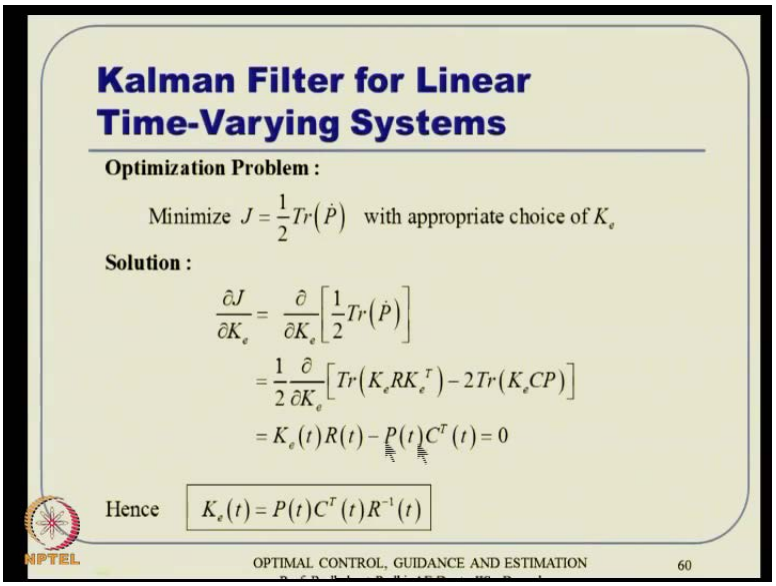
So, that is our continuous time kalman filter. Once you solve this Riccati equation, filter Riccati equation we have this gain computation that way and you can propagate the filter dynamics like this and Y happens to be the measurement vector and CX hat is nothing, but the predicated output sort of thing. So, this error is also called innovation and all like that way. So, I mean if you continue to operate it from an initialized state and for linear system it is guaranteed to converge by the way. Ultimately this error will go to 0 and hence X hat of t will be a good representation of X of t. That is the whole idea of kalman filter basically.

(Refer Slide Time: 36:31)



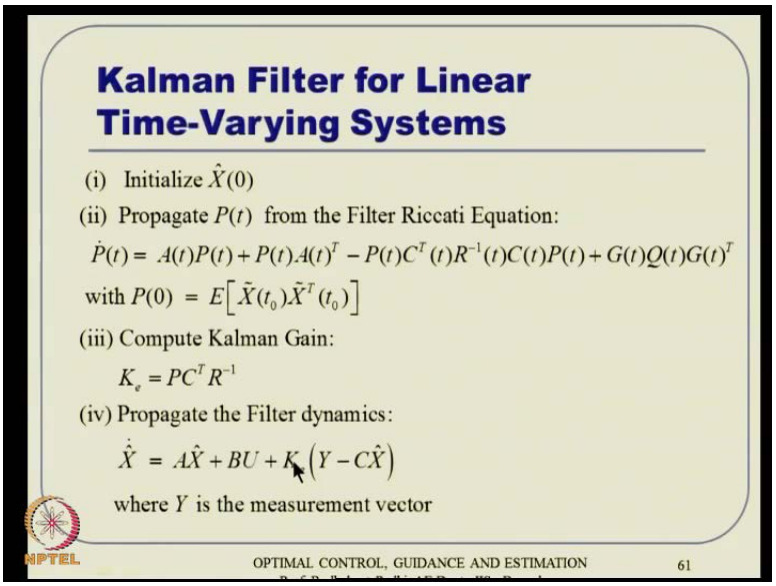
Now, extended that for time varying systems and how do you that and all that, so we are now instead of our constant matrix, this A B C, all these are time varying matrices now. So, how do you do about that? The assumptions were fairly similar, we had this process noise and sensor noise here and both of them are assumed to be 0; mean white noise all the time. Whereas, then at the initial condition, process noise and sensor noise they are all assumed to be mutually orthogonal and then we have this W and W t and V t are uncorrelated, non stationary white noise. We assume that kind of a thing. And this expected value happens to be, I mean white because of these conditions. The expected value of W, W transpose, they are at the Q only when t is equal to tau sort of thing; tau not equal to t, and then they are 0.

(Refer Slide Time: 37:21)



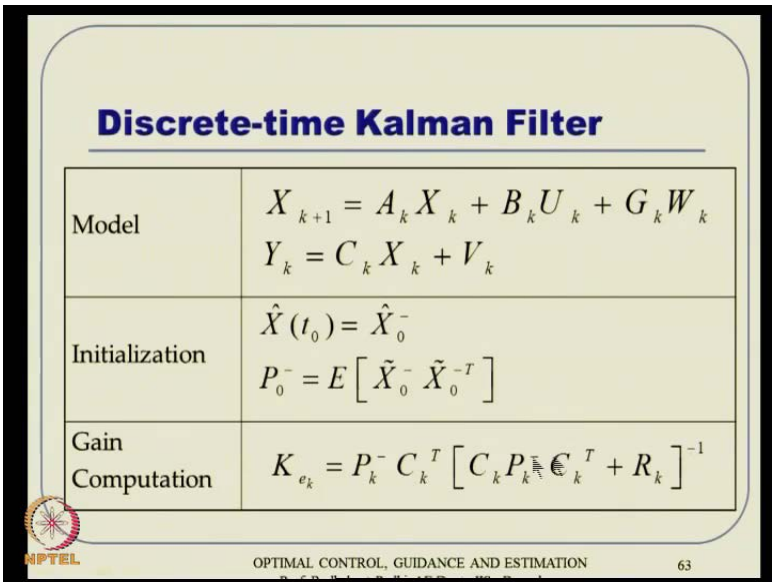
Utilizing that now there is no guarantee that this matrix Riccati equation will, I mean there is nothing called matrix Riccati equation now. So, we have this P dot equal to the differential Riccati matrix what I mean. Now, the idea is if P cannot be minimize then can we at least minimize P dot, so that leads to the idea of selecting this cost function to minimize trace of P dot essentially a norm of P dot. The solutions represent to be like this. If I evaluate this partial derivative Del J by Del K e that has to be equal to 0 and then this expression gives us that K e is nothing but that. Very close to what you have before, but the Riccati equations is no more stationary. This is not equal to 0, but it is equal to P dot.

(Refer Slide Time: 37:59)



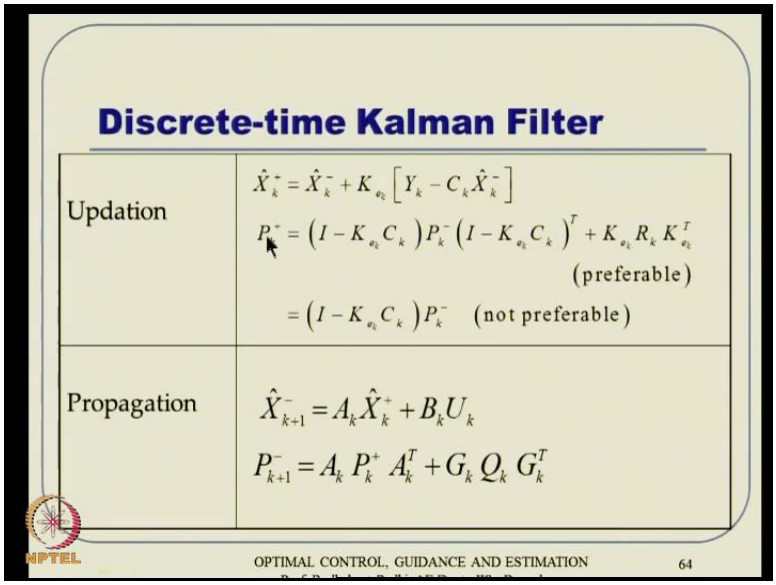
But we start with some P dot I mean P naught, some guess value for P, and then propagate this P dot equation. Remember, the nice thing about filtering theory is everything is propagated forward. Nothing like control theory, something like the Riccati equations is solved backwards and things like that. But here because of dual system properties and all that it turns out that even Riccati equation have to propagate forward and basically. So, we have to propagate P of t starting with this initial condition and using this equation and then every point out time you can compute a kalman gain is time varying and then use that in the filter dynamics or observer dynamics.

(Refer Slide Time: 38:50).



That we have this discrete time kalman filter, DKF sort of thing, essentially the summary is like this. We have a discrete time complete discrete time formulation. It initializes the state and initializes the covariance matrix also then you compute the gain this way.

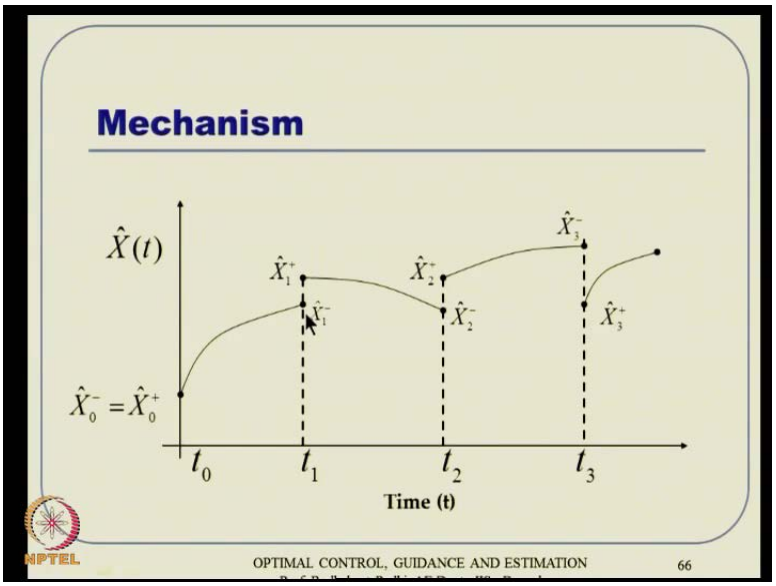
(Refer Slide Time: 39:03)



And then we have this update, once you compute the gain you can update the state to the next time state, sorry at this same time step. Then you can update the P k also at the same time step. So, this minus plus notations are necessary because minus is something that either you start from guess value, or you kind of propagate there from the previous value. But, once the measurement value keeps coming at that point of time where gain is ready. So, we can update the values depending on the measurement error here.

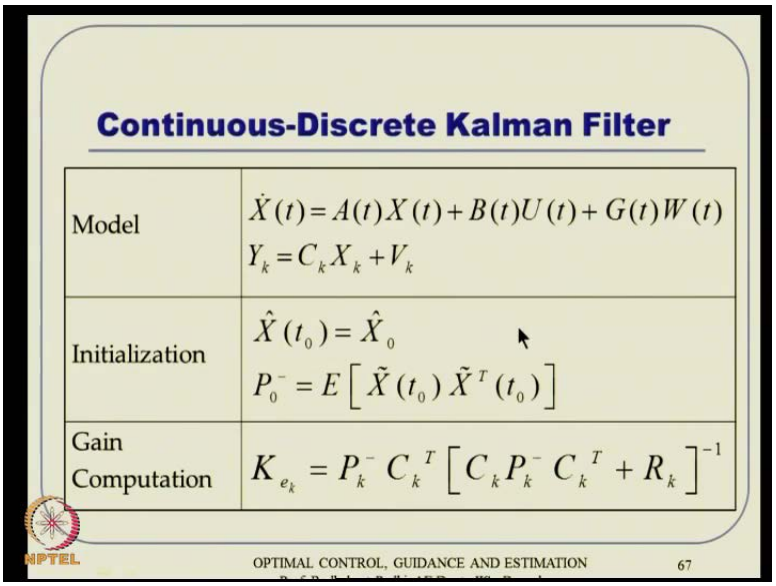
Error between the actual measurement and predicted measurement whatever is the error that will help us in giving a different value compared to this minus. Similarly, the covariance matrix needs to be updated also and then once these values are updated then you again propagate it using this noise free system dynamics like this. Then we have the whole motivation of continuous time discrete time and everything is to be is the fact that we are interested in continuous discrete klaman filter; that means, the system dynamics is continuous, but the process that is the sensor output I is strict.

(Refer Slide Time: 40:11)



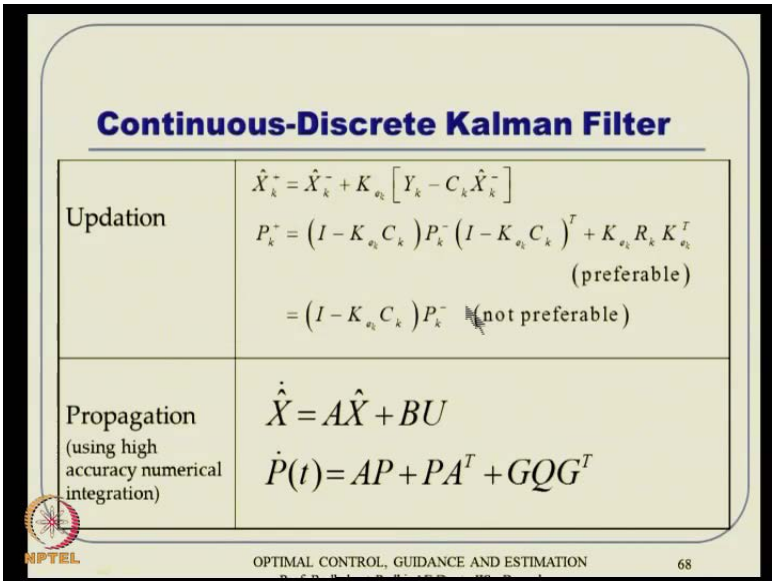
So, in that setting what happens? So, that is where the idea comes that I start with some guess value t naught, but I can propagate in a much better way like utilizing higher order numerical methods for integrating continuous time system. then I land up with some predicted value and there when the output comes, the sensor output comes, I can update that and then I can keep on predicting until the next centre output comes at t 2 and then update then continue like that. So, for the prediction part, we want continuous system dynamics in a continuous time of frame work, but for updating you know the sensor cannot be continuous and output cannot be continuous. It can from only a discrete time discrete points of time. So, as the prediction becomes continuous where as update becomes discrete.

(Refer Slide Time: 40:56)



So, in that setting the result turns out to be like this. We have this model, which is a continuous time state equation, linear still and then we have this discrete time output equation. We initialize this again with some values of X hat of 0 and P 0 minus of I mean these two values have to initialize and then you compute the gain.

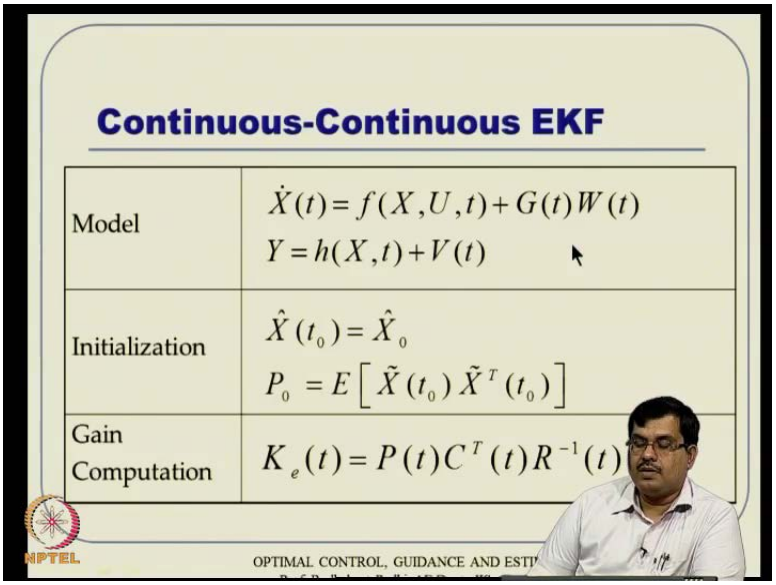
(Refer Slide Time: 41:20)



When the measurement comes we will update it in the discrete time setting, the discrete time results sort of thing. Now, you can update the covariance matrix in both way so, but, again it is always preferable to use this to avoid the state difficulties. Even though

this expression turns out to be quite kind of simplified compare to that, but this retains the symmetric properties and all that. So, it is just still advisable to use this expression instead of this expression. Anyway, the propagation again, after update, you have to propagate and propagation can be done this way. The only difference is this additional non-linear term of P is not here. That happens, because the output is not continuous and hence it does not get correlated in each other sense. Anyway finally, there was an idea that all this we studied because of understanding this so called EKF or extended kalman filter. That is what it used in practice.

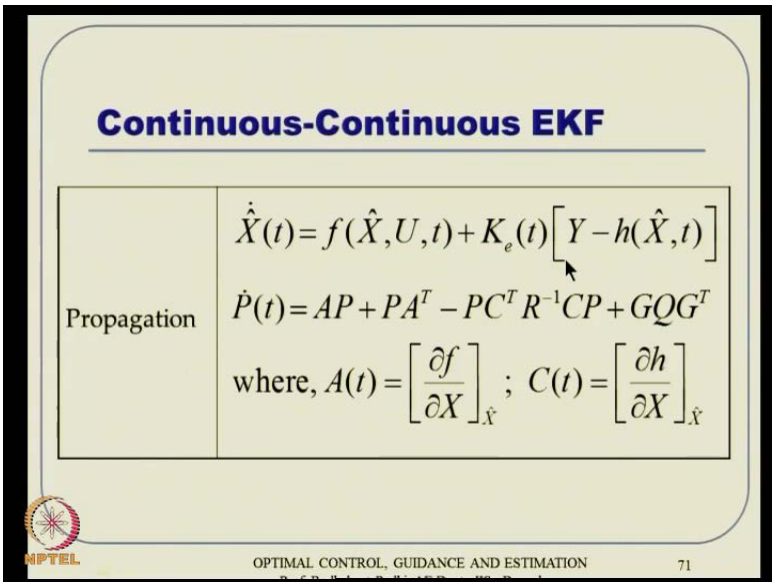
(Refer Slide Time: 42:18)



So, here the system dynamics in non-linear and the output equation is also non-linear. So, then we have this idea of continuous EKF or in the sense that system dynamics is continuous and output is also continuous, but if you can generalize that and tell that you can also have continuous discrete formulation, where the system dynamics is continuous where as the output is discrete.

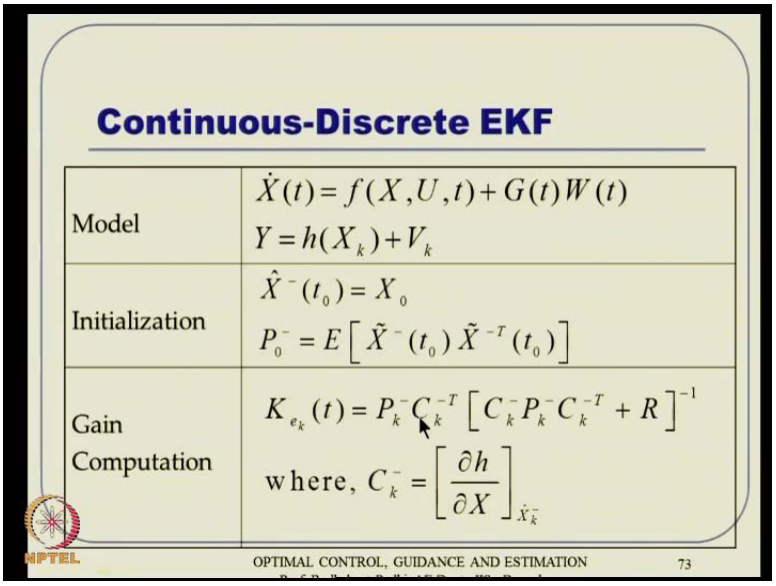
The initialization, again the procedure is very similar. Again, I do not want to derive and explain and all that here in this last lecture, but I hope all of that is done. So, essentially the idea is we can linearize these two non-linear equations about the current value of the state and then you can utilize the kalman filter theory for the corresponding linearize problem.

(Refer Slide Time: 43:20)



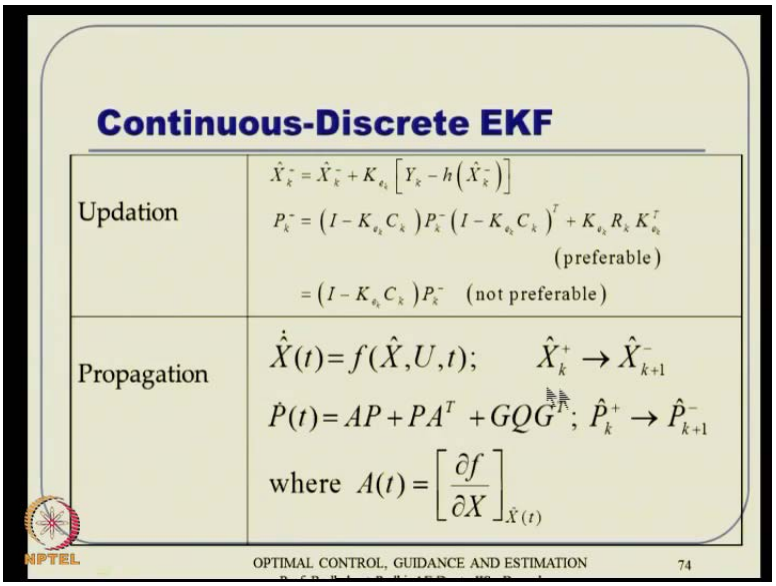
That is the whole philosophy there. A operation happens to be like this, you start with initial value and initial guess value for the state and initial guess value for covariance also, covariance matrix, then you compute the gain and then you can propagate the system dynamics as well as the Riccati matrix in these two equations actually.

(Refer Slide Time: 43:34)



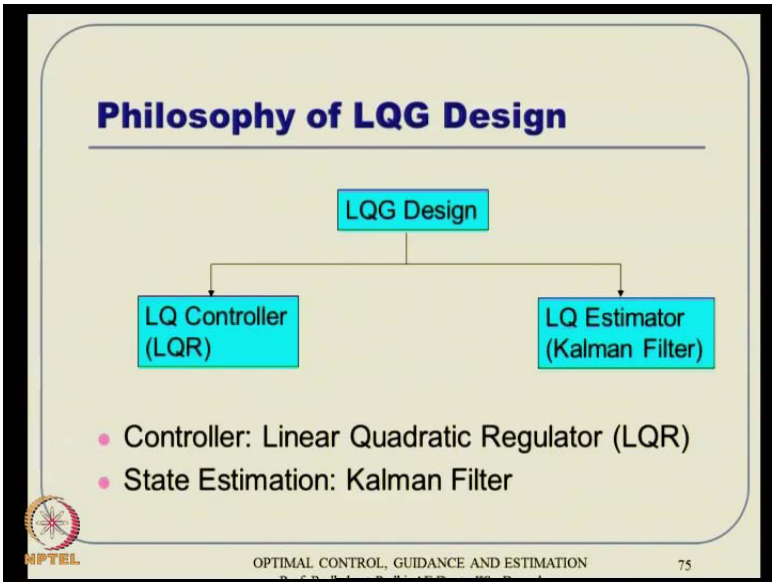
Eventually, this continuous discrete EKF and that is what we need, the system dynamics happen to be continuous whereas the output happens to be discrete and that sorting was a mixture of ideas.

(Refer Slide Time: 43:49)



Again we have to initialize this and then compute the gain at every point of time, in that way and then update as soon as a measurement comes, we involve the discrete relationship and update that and then after we are done with the update, we can predict it or go for propagation node using the continuous time theory. Once again the non-linear term is not there because that the measurement happens to be discrete that is the reason.

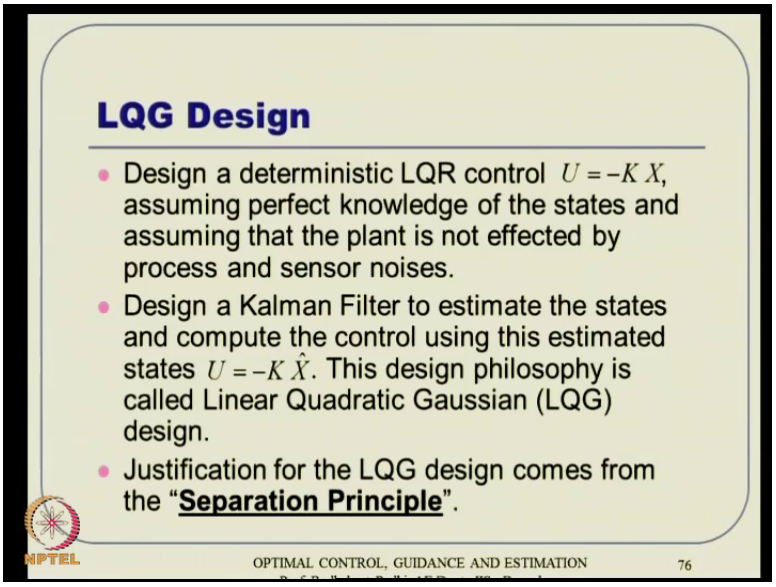
(Refer Slide Time: 44:11)



So, after sorting kalman filter, we also had a glance of what is called LQG design in the certain of non-linear control theory again, so essentially the philosophy was we have this

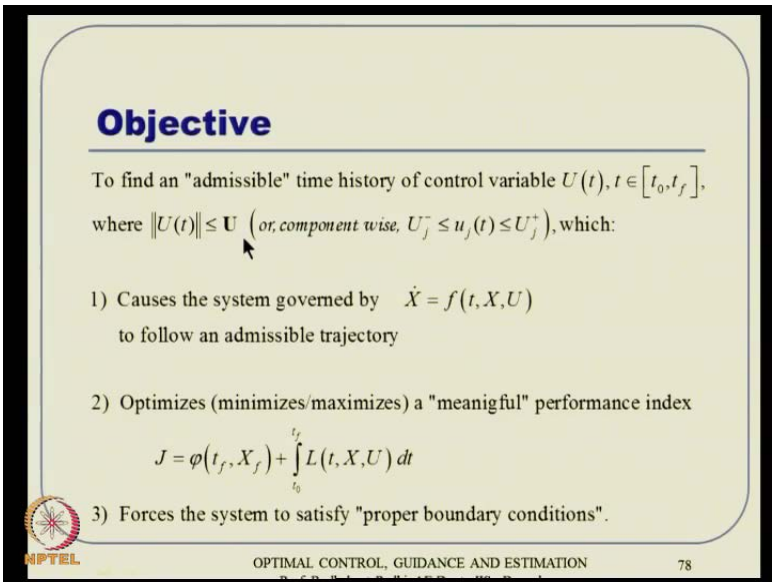
LQ controller or a rather LQR controller and LQ estimator or kalman filter in that loop. So, if we put both the things together then the thing called LQG design. Essentially, we are using the control formula for LQR, but the state U equal to minus K X, instead of that you are saying U equal to minus K X hat. So, X hat is nothing, but the estimated state in that sense basically.

(Refer Slide Time: 44:47)



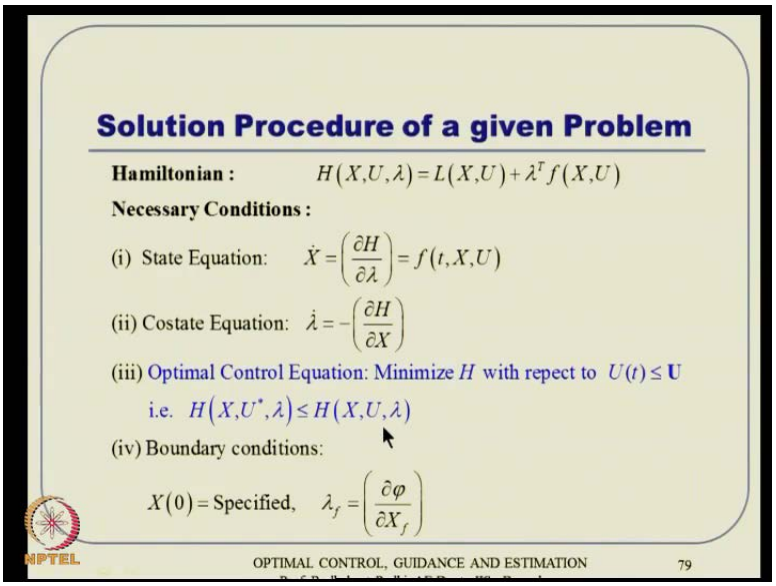
So, this is what it is written, instead of U equal to minus K X, we use U is minus K X hat. And justification comes from the fact called separation principle, which tells us that these two thing designs can be done independently without harming the close loop behavior. Essentially, if you see that analysis tell you that the closed loop, I mean the closed loop system dynamics and the error dynamics if you send eigen values of both the things and then put them together and analyze the eigen values; they do not get altered actually. So, in other words they happen to be kind of a separate. So, that is the basis of these LQG design and remember that is valid only for linear system. Even the people do that for non linear system, there in a no proof for sake for non-linear system and all that.

(Refer Slide Time: 45:43)



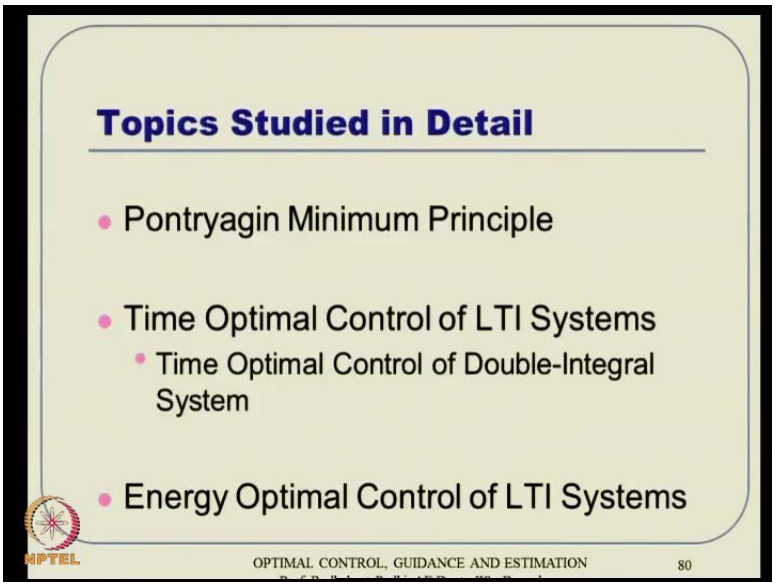
So, towards the end we also discuss something called control constrained optimal control entity. So, we went back to the objective and tell ok we start with the same objective whatever the regular control, optimal control is suppose to do that should find an admissible time history of control and this region, so that this cause the system governed by the system dynamics to follow an admissible trajectory, and on the way it should optimize the cost function, it should satisfies the boundary condition as well. But, at the same time it should be confined to a set; that means control is not really infinity and it confined to allowable limits or component wise they can be restricted between some minimum value and maximum value.

(Refer Slide Time: 46:20)



So, if you do that, the essentially this is a great contribution from Pontryagin and it essentially tells us that the regular theory can be used; you can still define Hamiltonian in that way and carry out the necessary conditions optimally. The only difference is the optimal control equation. So, instead of Del H by Del X equal to 0 and try to solve the control out of that, you carefully analyze this relationship; that means, Hamiltonian expression evaluated for optimal control has to be less than equal to that same expression evaluated for the non optimal control. And utilizing this relationship carefully you should be able to kind of come up with some sort of a control function. This equation has to be honored also by the way. I mean this state, costate and boundary condition has to be honored also on the way.

(Refer Slide Time: 47:08)

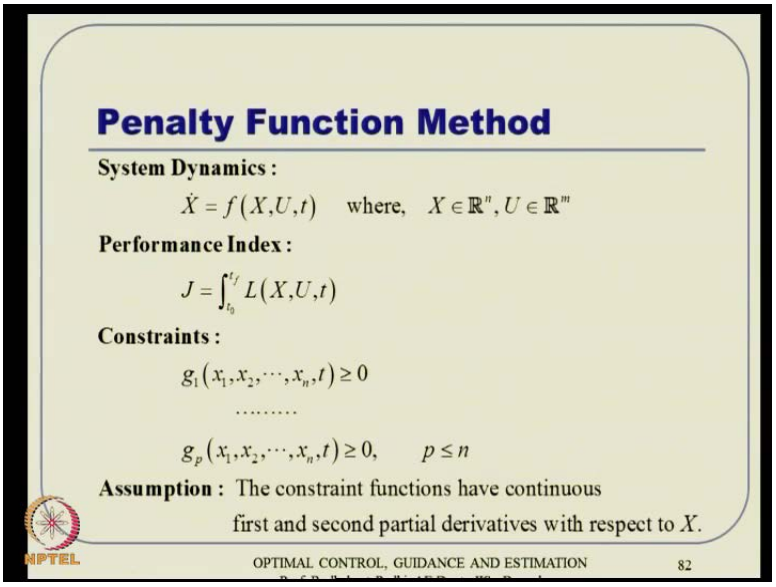


So, topics that we studied in detail talks about Pontryagin minimum principle; some sort of a cursory proof is also there, the idea behind how it comes and how we will end up with that. And then we talk about a great deal of analysis for time optimal control system, the minimum time control problem. We have been taken a double integral system to demonstrate the idea in fact in detail actually. We essentially come up with switching control sort of ideas; we define some sort of a switching function and then switching regions and thing like that. We told that it can be; control can take a discontinuous sort of thing, one time it can be discontinuous, but with some sort of appropriate switching at appropriate time, we should be able to do this job and try the system towards origin even if the control is constrained.

We also studied energy optimal control for LTI system, in general, where things were much more benign compare to this time optimal control. On the way also I told that there is another concept called fuel optimization problems and which was left out for self study. So, both this time optimal control and fuel optimal control leads to this singular arc, I mean this discontinuous control or singular arc problems like that. Whereas, the energy optimal control where you want to minimize something like a quadratic cost function of control variable is much more benign and it naturally leads to something that we feel intuitive. In other words, as long as the control is away from the bound then you apply the regular value and the moment it goes away from the bound you apply the bound value.

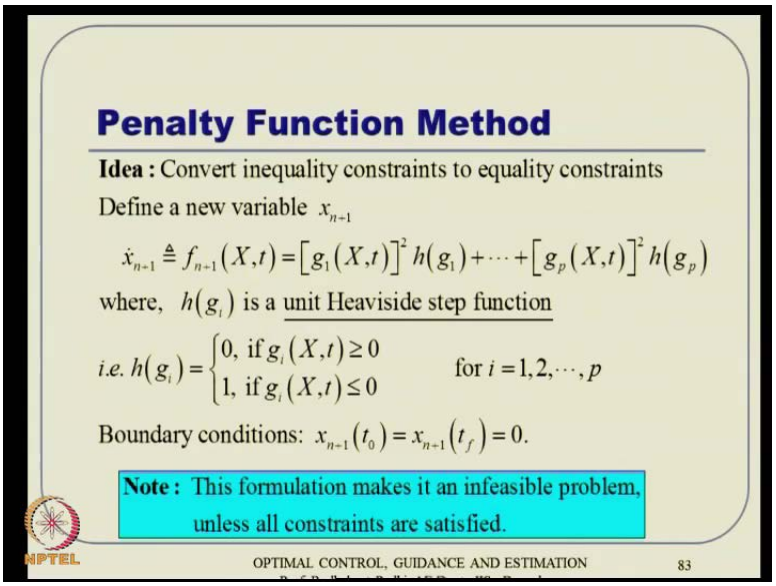
And that is possible only when we have this nice formulation of quadratic cost function and control variable and all that is called energy optimal control. The details you can see that in the lectures and you can study the corresponding books also basically. Then towards the end we also talked about state constrained optimal control; that means, it need not be only control constrained, inequality can come instead variable also.

(Refer Slide Time: 49:12)



There we studied two methods; one was a penalty function approach, where you had these problems like this. We have the system dynamics like this and performance index was given like that, but also had a bunch of inequality constraints as functions like g 1 to g p in that way. So, we assumed that the constants are smooth. In other words they have this continuous first and second partial derivatives and things like that.

(Refer Slide Time: 49:36)

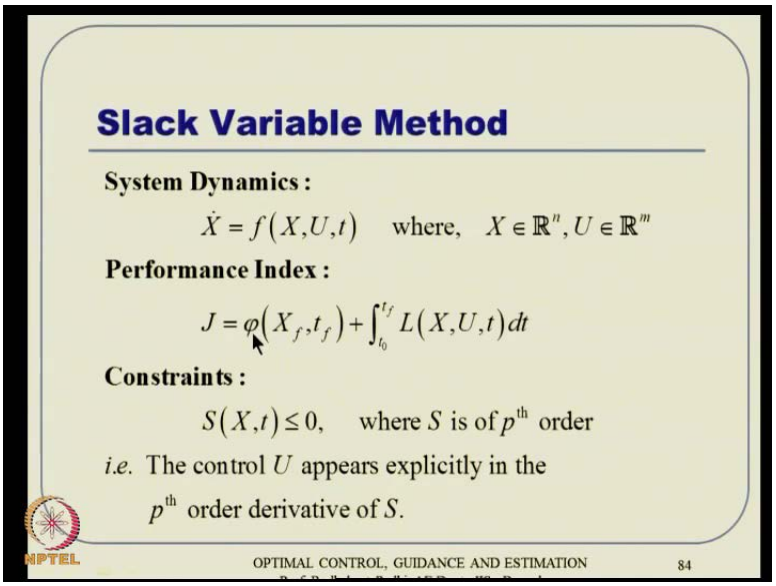


Then the whole idea here is to formulate an auxiliary state equation. So, X n plus 1 dot is something like this. It is something like a quadratic function times Heaviside function like h of g 1 and up to g p is exactly similar way where this Heaviside step function is defined something like this. If it satisfies the constraint, it is 0 as that is not active, but if it does not satisfies, this become one; that means, this function is suddenly active .

But the critical observation was like we are not only this state equation, along with the state equation we want to put this boundary conditions also; that means, this equation which is guaranteed to be positive the moment any of the constraint is not satisfied the X dot is guaranteed to be positive. Hence, these boundary conditions will never be met together basically. So, essentially it tells you that this formulation makes it some sort of an infeasible problem unless all constraints are satisfied.

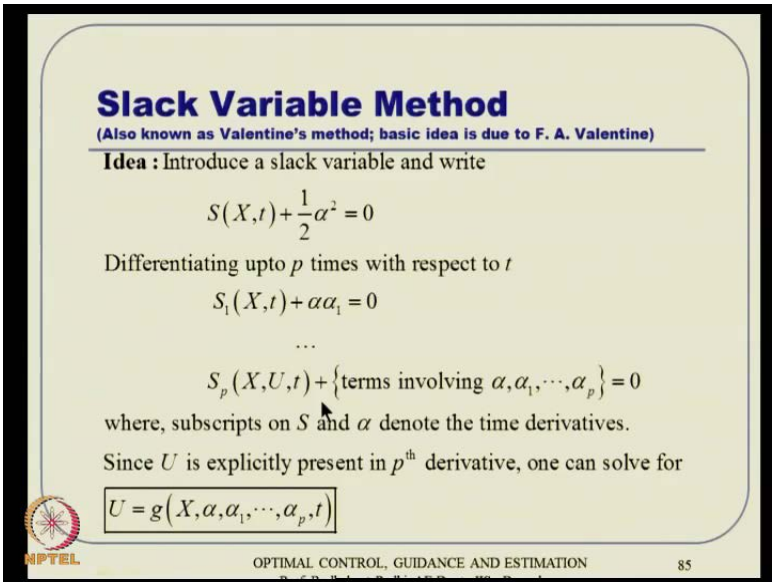
So, now the idea is I will take the original problem as it is, but I will augment that with this state equation and this boundary condition as well. So, essentially I will solve some sort of an artificial problem, which talks about problem in n plus 2 dimensional spaces.

(Refer Slide Time: 50:48)



So, that was done and I mean that is the one idea basically. The other we talked about is also slack variable method and in my personal view this is a kind of better than the other one. Here, the problem was again revisited in a little more generic sense, this terminal penalty is also there here, but that can also be there in the other formulation. I mean that that is not a restriction pursue. The idea is here is we have these constraints even if we have something like this.

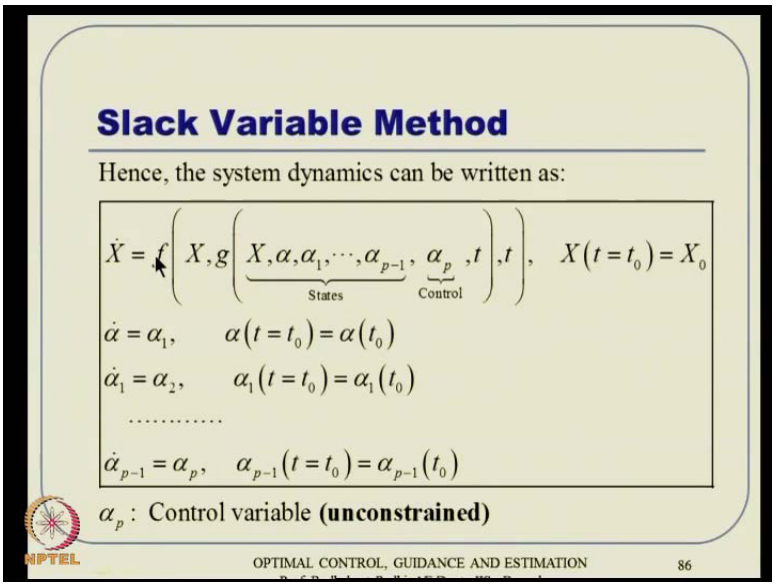
(Refer Slide Time: 51:13)



Then we wanted to formulate this equality constraint out of this inequality by introducing this half alpha square. Remember, this is a scalar constant. So, if you have multiple constraints, you have to do multiple times all this steps. So, if a scalar constant like this we can have this equality constraint now, but remember going back to the static optimization problem where we kind of a did the similar thing for well deriving KKD conditions, but here the idea is not to neglect the value of this slack variable, but to compute the slack variable and its derivatives together.

So, to do that there was an observation that this equality constraint, if I take a derivative this leads to that. If I take subsequent derivative of the same expression, ultimately I will end up with some equations like that. But, also remember that this assumes that this polynomial is of p th order; that means, if you take p th order derivatives of this polynomial then control U appears explicitly basically. By utilizing that fact, so from this expression, I can solve for the control in terms of other variables. So, the last derivative that p th order derivative equal to 0. If I utilize that thing then I should be able to solve control in terms of other variables.

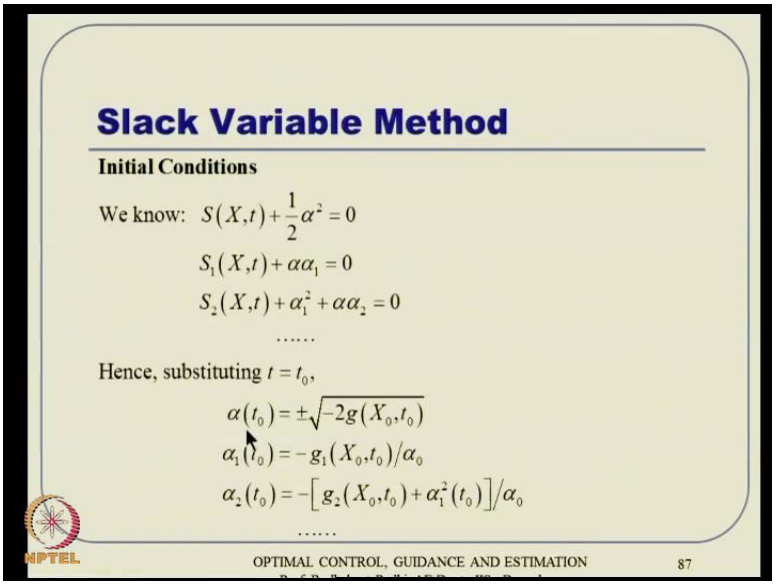
(Refer Slide Time: 52:33).



Now, once I do that I can revisit the state equation and I put the state equation is nothing but f of X u, but u is now something like this. If I put it there here and in this I will consider alpha p is control, but all other things are states. So, I should be able to augment these state equations also which is very much straight forward.

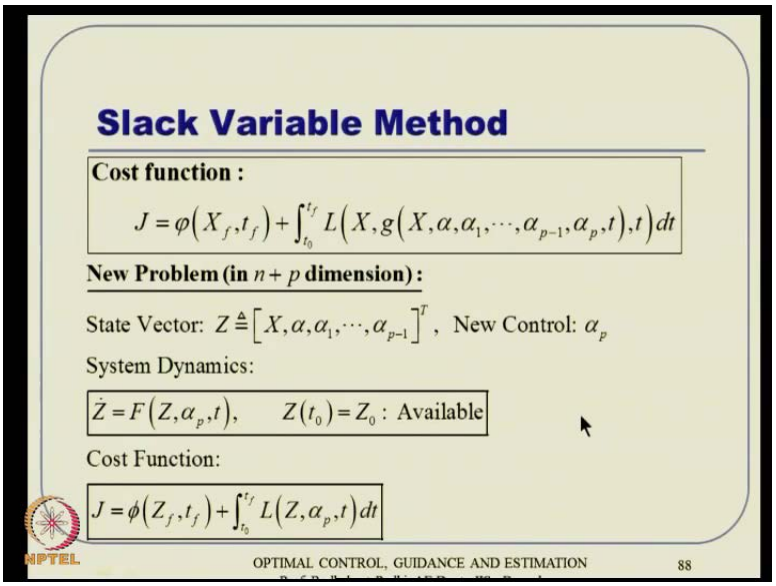
Alpha dot is alpha 1; alpha 1 dot is alpha 2 like that I will be able to do that and alpha p happens to be a control variable. Say, under the fact is alpha p is unconstrained also. So, the same constrained constant has been taken care on this equation itself. So, alpha p it turns out to be unconstrained actually. So, again we had a kind of p states to the n states. In other words, we increase the state dimensionality, but ultimately we are able to kind of convert a constraint optimization to a free optimization problem.

(Refer Slide Time: 53:25)



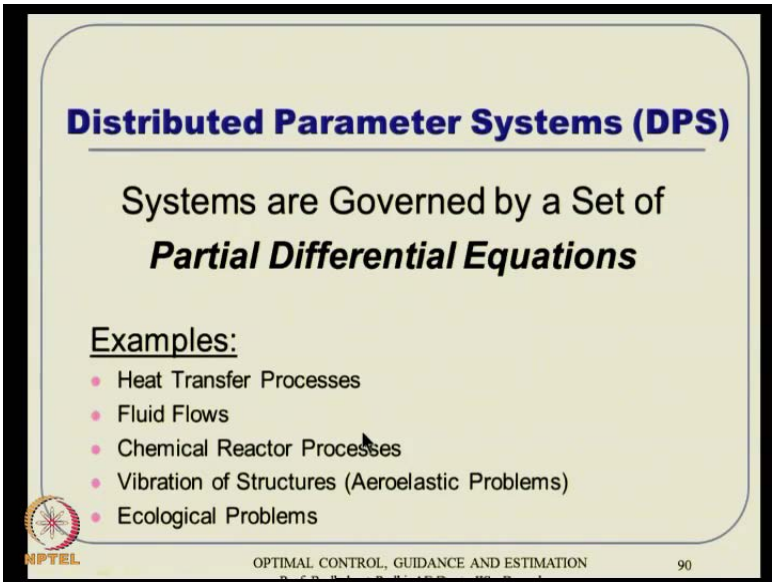
Now, if you introduce this state equation, we also need one step initial conditions for that and that was unfortunately, it is possible to solve or the initial condition as a function of the initial condition for the state and time. So, utilizing this equation, you can actually solve this alpha t 0, alpha 1 of t 0, and things like that. So, this formulation is kind of very compatible in that sense.

(Refer Slide Time: 53:47)



So, again the formulation takes a little bit a higher dimensional, but essentially ends up with a free optimization problem, when alpha p is considered as a control variable. That was given there and you can go ahead and solve this problem in that way. So, these are the two methods that we kind of discussed as part of the state control, state constraint optimization problem.

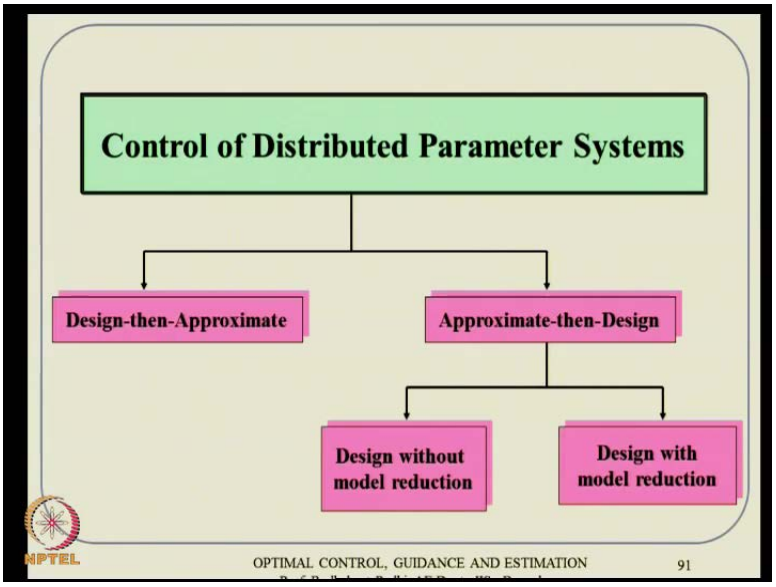
(Refer Slide Time: 54:19)



Now, towards the end of this course, we also thought we can extend this scope of this optimal control to the distributed parameters systems, as well. Distributed parameter

systems, by nature are governed by partial differential equation, which is very different from whatever you seen so far. But there are many examples where we have to talk about partial differential equations. For example, heat transfer process fluid flows, I mean chemical reactor process and vibration of structures and then ecological problems many things are there where ODES are not sufficient.

(Refer Slide Time: 54:43)

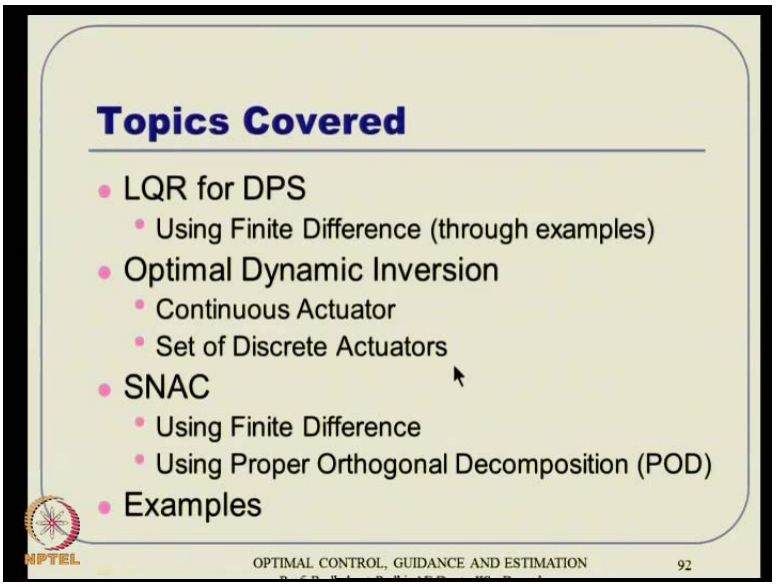


So, how do you handle with those kinds of problems? Now, going back and seeing what can we done for this distributed parameter systems. There are two approaches; one thing is a kind of design and then approximate that is what mathematician will love to do that. They invoke this infinite dimensional operator theory and does all sort of nice algebra there and then finally, come up with the control expression, then they discretize it ultimately and then try to implement it. That is kind of design and then approximate. But, the engineering approach follows their little bit reverse way. First of all, it does approximation and then does the design.

For that approximate problem and that way it can be done in two ways; one is design without model reduction, which is a not very much advisable, because you really end up with a large number of grid points again. But, then we can also utilize the concept of design with model reduction; that means, not only we approximate the system, but you introduce the system, a kind of model order reduction sort of ideas there, and they come up with a lower dimensional representation of the same thing and then utilizing that

lower dimensional system you can design like a control system So, that is the idea of design with model reduction.

(Refer Slide Time: 55:48)



Then the various topics that we covered under that in relatively fair detail are something like this. First we started with the idea of putting finite difference and then utilize the same idea of LQR for linear distributed parameter systems. And we gave some examples also to demonstrate the ideas there. Then we took a little digration after that where you talked about optimal dynamic inversion and then that was done by utilizing this continues actuator and the set of discrete actuators subsequently. As not very much in the frame work variation calculus and all that, but I still thought it is a nice idea to discuss about. Variation calculus is used not in the frame work of what optimal control frame work would like to do in away.

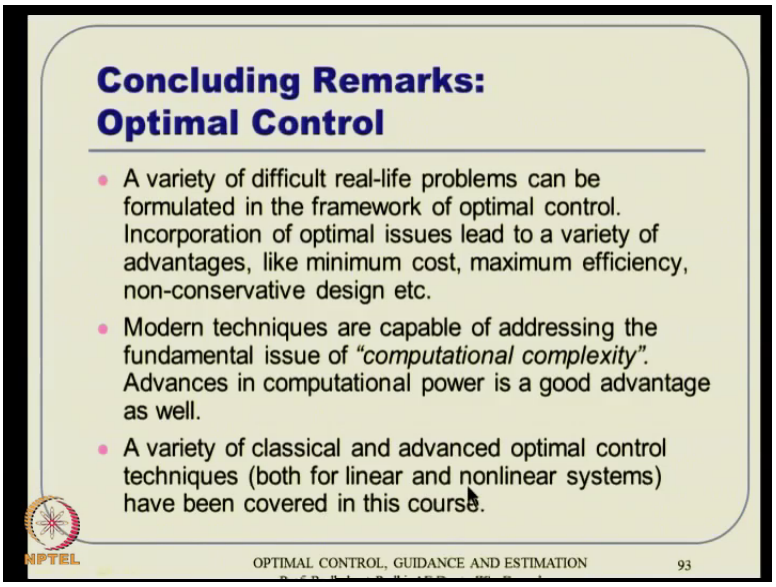
In other words, we have this quadratic error quantity and then quadratic error was enforced to go to 0, utilizing the dynamic inversion ideas there. Why do you enforce this stable linear error dynamics and all that, but that will give you only a constraint with infinite freedom. So, there is a scope for getting this calculus of variation little bit and then we solve for controls assuming both continues actuator and also subsequently we had this discrete actuator formulation and all that, where this control singularity starts coming as the objective starts getting met. So, there is a necessity of switching power

and then we having some two goals; one in the beginning and one towards the end like that. We demonstrated that utilizing examples also basically.

Also we have this single network adaptic critic based optimal control formulation and that we can directly do that utilizing only directly finite difference and then putting the grid points and then solving the problem on the grid points only. But, we can also have this idea of invoking this model reduction through this basic function design using this proper orthogonal decomposition sort of ideas, where you can starts something like set of snapshot solution. Utilizing the snapshot solutions, you can design the basis functions which are tailor made to that particular problem.

And utilizing those basis functions, you can actually come up with those basis functions to be utilized in Galerkin approximation, to come up with something like a low dimensional ODE representation of the system dynamics. Then utilizing that ODE in the model form you can come up with some set like a state feedback control design. The details of that can be seen in the particular lecture basically also, given examples on the way basically.

(Refer Slide Time: 58:21)



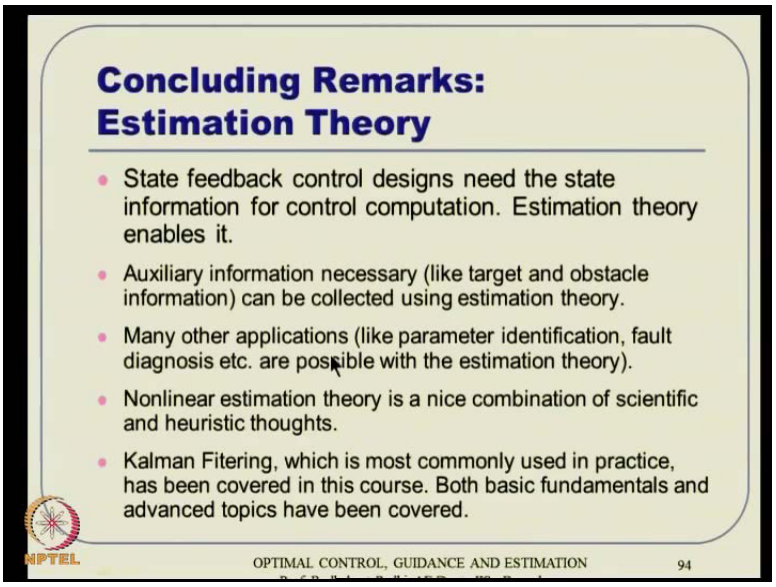
So, finally, at the end of the course there are some concluding remarks. First remark is variety of difficult real life problems can be formulated in the frame work of optimal control. Do not forget that. An incorporation of optimal control issues leads to a variety of advantages like you can talk about minimum cost, I mean minimum time or whatever

maximum efficiency or non conservative design things like that. So, optimal control frame work gives us a very good platform, very good tool for doing that.

And then modern techniques are capable of addressing the fundamental issue of what optimal control theory was kind of suffering from for long time and that is nothing but computational complexity. So, in computational complexity, various good algorithms are come up and then still developing actually and that is a resource for doing a good research also. But, I have said that many techniques available like pseudo spectral method, MPSP, SDR techniques like that which kind of try to address computation complexity issue in some sense basically.

Then this computational advantage coming from the computer side; that means, the computational power is increasing day by day that also helps. Now, it does not make sense to kind of get confine to the old idea that optimal control cannot be used in real time. It can be and in this particular course we discussed variety of classical and advanced optimal control techniques both in the setting of linear and non-linear framework. So, both have been covered in this course also basically.

(Refer Slide Time: 59:53)



Then coming to the estimation theory part of it, we also remember that it is a very good compliment for the control design part of it and state feedback control design needs the state information for control computation. That is the first thing why we want to study estimation theory. An estimation theory, once we know it enables the optimal control or

any of the state feedback control design in a very good way. So, do not forget that part of that is a good motivation for us to study estimation theory. Then sometimes we need auxiliary information like the target information, obstacle information, like that for coming up with strategies; that means, guidance strategies and things like that.

For that we need to have target estimation, obstacle estimation or things like that, which is not the states of the own vehicle or own system, but states of the other system, which is necessary for doing some computational of another system. For that also we need estimation theory. So, many other applications are also there like something like parameter identification, fault diagnosis and many things like that, where if we know estimation theory in a good way, then it enables you in a very good way. Non-linear estimation theory is a kind of a combination of scientific as well as heuristic thoughts.

So, for example, extended Kalman filter, even though there is no theoretical guarantee, but we know that it works very well, but it has difficulties in tuning also. So, tuning part you can put it into heuristics or experience sort of thing, but once it starts working it works in a very good way. Finally, Kalman filtering, which is kind of most commonly used in practice has been covered in this course in detail and both basic fundamental as well as advanced topics something like EKF, even some little bit kind of (( )) ideas also we have discussed.

So, you can go back and study this course in a good way. I mean if you do not understand take your time and thus there will be practice problems on the way. Try to find and solve those problems, get a better understanding. Now, that after going through the course I am confident that you can also be able to read many books and then enrich knowledge further. So, with that let me sign off from this course thank you and good bye.