

Optimal Control, Guidance and Estimation

Prof. Radhakant Padhi

Department of Aerospace Engineering

Indian Institute of Science, Bangalore

Lecture No. #21

**Approximate Dynamic Programming (ADP), Adaptive Critic (AC) and.
Single Network Adaptive Critic (SNAC) Design**


Hello everyone, we will continue with our lecture series. Last class, we saw this dynamic programming and we realized, that actually it, it is a very good approach in general in the, in the sense, that it, it gets into this close form solution or control. However, we at, towards the end of the lecture, we also saw, that it actually results in this huge computational difficulty and all, which we have now termed as curse of dimensionality. So, there are ideas there, which to, to kind of avoid that and then get some meaningful solutions, at least, for limited class of problems actually. So, we will see that in this particular lecture.

So, one approach, which is called as approximate dynamic programming and here, you will see some sort of a fusion between these two ideas of calculus of variations and dynamic programming actually. And followed by that we will, we will use this, those results in this method, called adaptive critic and a variation of that in a little bit computationally more efficient sort of ideas there, which is called single network adaptive critic, actually. So, let us get started there.

(Refer Slide Time: 01:22)

Outline

- Approximate Dynamic Programming
- Adaptive Critic (AC) Design
- Single Network Adaptive Critic (SNAC) Design
- AC vs. SNAC: A Comparison Study

 OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 2

So, this is outline of this particular lecture. First, we will see, study what is approximate dynamic programming, and then we will try to see, what is adaptive critic design followed by the single network adaptive critic. And we will see examples where we will give some comparison between, I mean, performance comparison between AC and SNAC. That, we will see in two, three example problems actually.

(Refer Slide Time: 01:51)

**Approximate Dynamic Programming:
Discrete-time Framework**


Problem:
Find an admissible control U_k (at time t_k)
which minimizes a “meaningful” *cost function*:

$$J = \sum_{k=1}^{N-1} \Psi_k(X_k, U_k)$$

subjected to the constraint of *system dynamics*:

$$X_{k+1} = f(X_k, U_k)$$

and appropriate *boundary conditions*
(Note: $N \rightarrow \infty$ leads to infinite horizon problem)

 OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 4

Let us see the first thing first, which is approximate dynamic program actually. If everything will happen in discrete time domain here and so, the integral will substitute

like a summation and things like that actually. So, the objective is still very, very similar to what you have seen before and that means, to, to optimize this J is a cross function, so find a U_k , so that you can optimize this actually, remember. In other words, formally speaking, to find an admissible control U_k , which minimizes this meaningful cost function given this way or it is also subject to the constraint of system dynamics, which is of the form this way actually.

(()) with appropriate boundary conditions in this entire class, we will assume, that N tends to infinity, that you were talking about infinite time problems actually. And typically, these are, when you talk about infinite times, these are typically regulated problems. So, even though generally cost function will be used for, for desiring result and all, most of the examples that we will follow will have quadratic cost function actually, getting there. So, we write this first thing as this J, we start from something like J_k . Remember, k starts from 1, then it is J. How you can define J_k ? Starting from k itself actually, wherever that is, then n minus 1, actually.

(Refer Slide Time: 03:07)

Approximate Dynamic Programming: Necessary Conditions of Optimality

- Write: $J_k = \sum_{\tilde{k}=k}^{N-1} \Psi_{\tilde{k}}(X_{\tilde{k}}, U_{\tilde{k}}) = \Psi_k + J_{k+1}$
 - Cost-to-go from time t_k (points to J_k)
 - Utility Function at time t_k (points to Ψ_k)
 - Cost-to-go from time t_{k+1} (points to J_{k+1})
- Define: $\lambda_k \triangleq \frac{\partial J_k}{\partial X_k}$
- Optimal Control Equation:** $\frac{\partial J_k}{\partial U_k} = 0$

NPTEL OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 5

So, you define this cost-to-go from time t_k is something like J_k , so then there will be some other variable here. We substitute this k tilde sort of thing, which is the dynamic variable sought of thing. But k tilde varies from k to N minus 1, so this cost function, cost-to-go from time t_k , that is, J_k is written something like that.

And obviously, if you just take out the first term, that is, k tilde equal to $k + 1$ for one term, and then from k plus to N minus 1. So, this if you just take out it with k , then that is nothing, but Ψ_k and then k plus 1 to N minus 1 is nothing, but k minus 1, sorry, J_{k+1} plus 1.

So, as I told you in the previous lectures, this is something called utility function at time $t = k$ and this is something called cost-to-go from time $t = k + 1$, actually. So, like the dynamic programming we will again define this λ_k is something like, $\frac{\partial J_k}{\partial X_k}$. And then, the optimal control equation will be in this form actually, that we know, that $\frac{\partial J_k}{\partial U_k}$ ultimately has to be 0, basically. Because J_k , I mean, one of the conditions, that needs to be satisfied is, like necessary conditions, that needs to be satisfied is $\frac{\partial J_k}{\partial U_k}$ has to be 0, actually.

(Refer Slide Time: 04:33)

**Approximate Dynamic Programming:
Necessary Conditions of Optimality**

$$\begin{aligned} \frac{\partial J_k}{\partial U_k} &= \left(\frac{\partial \Psi_k}{\partial U_k} \right) + \left(\frac{\partial J_{k+1}}{\partial U_k} \right) \\ &= \left(\frac{\partial \Psi_k}{\partial U_k} \right) + \left(\frac{\partial X_{k+1}}{\partial U_k} \right)^T \left(\frac{\partial J_{k+1}}{\partial X_{k+1}} \right) \\ &= \left(\frac{\partial \Psi_k}{\partial U_k} \right) + \left(\frac{\partial X_{k+1}}{\partial U_k} \right)^T \lambda_{k+1} \end{aligned}$$

Optimal Control Equation: $\left(\frac{\partial \Psi_k}{\partial U_k} \right) + \left(\frac{\partial X_{k+1}}{\partial U_k} \right)^T \lambda_{k+1} = 0$

NPTEL OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 6

So, what is $\frac{\partial J_k}{\partial U_k}$? If you go back, if you go back to this J_k definition, J_k is nothing, but this Ψ_k plus J_{k+1} . So, I substitute that and then, I mean, here I will substitute that, so J_k is nothing, but Ψ_k plus J_{k+1} . So, this partial derivative $\frac{\partial J_k}{\partial U_k}$ is nothing, but this term plus this term, actually. This term I will keep it as it is because Ψ_k is something that I know from this, this cost function definition and all that. So, that will be a direct derivative sort of thing, so I will keep it that way.

What is this J_{k+1} ? $\frac{\partial J_{k+1}}{\partial U_k}$, the effect will be felt, that means, J_{k+1} will be plotted by U_k through $k + 1$, that is a system dynamics actually. In other

words, if I, if I (λ_k) control U_k , that will result in some part of (λ_{k+1}) in $k+1$ because our system dynamics is like this. If I apply a U_k , it will result in some difference in X_{k+1} , but, but J_{k+1} is a function of X_{k+1} . So, because of that there will be some change in J_{k+1} actually, that I have to keep it actually.

So, how do I keep it? I will keep it through this chain rule of derivative actually. Remember, these are all vector matrix algebra, so it was slightly more careful the writing of this chain rules and all that actually. We have seen some of these partial derivative results chain rules and all in very beginning lectures where we gave you this matrix algebra and things like that actually; you can see some of those if you have forgotten that. So, this $\frac{\partial J_{k+1}}{\partial U_k}$, I am writing it something like partial derivative chain rule sort of thing, write in this way.

Now, what is this? This is nothing, but by definition λ_{k+1} . Remember, λ_k is defined as something like this, $\frac{\partial J_k}{\partial X_k}$, so λ_{k+1} is nothing, but $\frac{\partial J_{k+1}}{\partial X_{k+1}}$, so that is what is λ_{k+1} . So, ultimately, what my equation tells me, that this has to be equal to 0; that means, this expression has to be equal to 0. Finally, this is my optimal control equation. However, how do I get this λ_{k+1} , that is the, that is the bottom line, actually. So, for doing that we need to go back to the other side of the story, that is, the costate equation sort of thing and as I told, this part can be evaluated from, from system dynamics actually. This, this particular (λ_k) and this will be directly evaluated.

(Refer Slide Time: 06:59)

Approximate Dynamic Programming: Necessary Conditions of Optimality


- **Costate Equation:**

$$\lambda_k = \left(\frac{\partial J_k}{\partial X_k} \right) = \left(\frac{\partial \Psi_k}{\partial X_k} \right) + \left(\frac{\partial J_{k+1}}{\partial X_k} \right)$$

$$= \left[\left(\frac{\partial \Psi_k}{\partial X_k} \right) + \left(\frac{\partial U_k}{\partial X_k} \right)^T \left(\frac{\partial \Psi_k}{\partial U_k} \right) \right] + \left[\left(\frac{\partial X_{k+1}}{\partial X_k} \right) + \left(\frac{\partial X_{k+1}}{\partial U_k} \right) \left(\frac{\partial U_k}{\partial X_k} \right) \right]^T \left(\frac{\partial J_{k+1}}{\partial X_{k+1}} \right)$$

$$= \left[\left(\frac{\partial \Psi_k}{\partial X_k} \right) + \left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \lambda_{k+1} \right] + \left(\frac{\partial U_k}{\partial X_k} \right)^T \left[\left(\frac{\partial \Psi_k}{\partial U_k} \right) + \left(\frac{\partial X_{k+1}}{\partial U_k} \right)^T \lambda_{k+1} \right]$$
- **Costate Equation on Optimal Path:**

$$\lambda_k = \left(\frac{\partial \Psi_k}{\partial X_k} \right) + \left(\frac{\partial X_{k+1}}{\partial X_k} \right)^T \lambda_{k+1}$$



OPTIMAL CONTROL, GUIDANCE AND ESTIMATION

7

So, coming back to this, this costate equation, this, remember this is the definition of lambda k. Lambda k is nothing, but del J k by del X k, now that is equal to this del psi k by del X k plus this term del J k plus 1 by del X k sort of thing. Now, remember this, this particular thing. Now, psi k is a function of X k and U k both. So, this particular thing, first, first of all is a direct function of X k. So, in a, in a given example, I mean, typically it is a quadratic cost function, so let me give something like psi k. psi k is nothing, but X k transpose Q k R k plus U k transpose R k U k, that kind of thing and some people put half actually. So, (()) however, remember that our control U k is also a function of X k, that is what we ultimately want actually, some function phi of k, actually.

So, what is happening here? del psi k by del X k plus this term, but this particular term del as del psi k by del X k. What you are interpreting is psi k is a function of state and control, both. So, one thing is direct function, direct partial derivative I can take in. The second one is because control is going to be a function of state. I will also account for that and write this chain rule actually. So, the first term is direct, the second term is this one. Similarly, the term, when I talk actually, this term will be first actually, J k plus 1 is a function of X k plus 1. So, this entire thing, what you see here, what, whatever you see here is nothing, but del X k plus 1 divided by del X k. This del x, this particular term, this, this term what I see here, this entire term is nothing, but this del X k plus 1 by del X k transpose, actually. So, this, this is again the chain rule. This, this term, this del X k plus 1 by del X k. This term and this term taken together is nothing, but that actually, but

this particular term again is, one thing is, remember this x^{k+1} is just saw, that X^{k+1} plus 1, one second, X^{k+1} is nothing, but f of X^k , U^k , alright. So, that is again a function of X^k and U^k , both actually.

So, we keep the, keep all those things in mind and write this particular term what you see here. One term is direct because it is a directly a function of X^k and the other term is through U^k , but U^k is also a function of X^k . So, this is what the script kept here actually. So, just this is little bit of derivation, I understand that, but just I follow little bit book keeping here actually. First thing is this is just one definition and this term we clearly explain, that this one is direct, the other one is through u basically.

Here also, similarly this J^{k+1} by X^k , remember that X^{k+1} will also be perturbed the moment you perturb X^k . So, because of that this there is non-zero, that means, this one comes directly from X^{k+1} perturbation, actually. So, I, first I write it something like this term, $\frac{\partial X^{k+1}}{\partial X^k}$ transpose times this one. But this particular one you can realize, that this is a function of this X^{k+1} is a function of both X^k and U^k and U^k is a function of X^k again actually.

So, this particular term is because of that first is directly and the second one is through U^k basically and I understand this is small little bit use of notation here, but I would think with all partial derivative I am writing it, the similar notations sort of thing, but let us ignore that for a second actually, that as long as you understand what you are doing is fine, actually. So, all these partial derivatives are, are expanded this way, actually. Now, once I do that I can, I can think about reorganizing the terms and all that here. Remember, this is nothing, but λ^{k+1} actually. So, what I do? I will take the first term here and first term here and remember the transpose will be nothing, but this transpose plus this transpose, which is something like this transpose into this transpose actually; $a^T b$ whole transpose is nothing, but $b^T a$, you can do that actually.

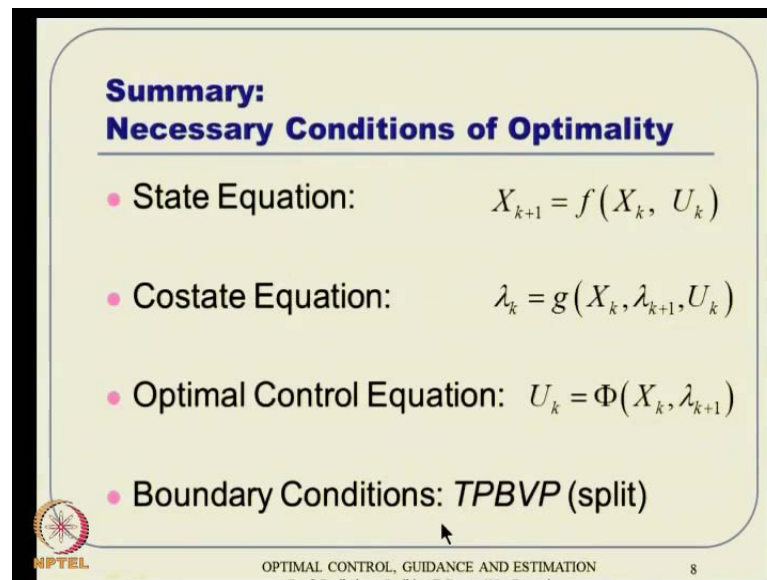
Anyway, so the first term here and a first term here and λ^{k+1} from here, so I will keep it this way. Then, the second term we can see, that $\frac{\partial U^{k+1}}{\partial X^k}$ plus by $\frac{\partial X^{k+1}}{\partial X^k}$ transpose is common here and this will be, this transpose is also in the left hand side actually, that will come after expansion. So, both of the terms will be in the left hand side. I will take out this, this is common and I write it that way. So, my λ^k is actually, theoretically speaking is, actually, this big expression here actually. There is a

there is a transpose here let, me probably risk out again, so let me write, this is the transpose actually, alright. So, this is what it is.

However, if we can, we can see, that, that this, this equation on the optimal path, this equation is also valid, this is nothing, but 0 actually. So, what you do here is if I put it there, this entire expression becomes 0 actually. So, that means, if I am worried about only optimal path, then my lambda k is nothing, but only the first term actually, this is what it is. So, this is the recursive equation sort of thing. That means, if I know lambda k plus 1, I can know lambda k.

And typically, from boundary position we know this lambda N actually and even if it is finite time problem and all that, we know lambda n is del phi by del x and all that actually, that way. So, you start from there and this actually tells us there is a backward recursive equation sort of thing and also, remember in, in 2.1 (()) problem calculus variation approaches, we actually have lambda dot equation, but lambda dot is typically propagated by codes, actually. So, all these are comparable actually that way.

(Refer Slide Time: 13:22)



Summary:
Necessary Conditions of Optimality

- State Equation: $X_{k+1} = f(X_k, U_k)$
- Costate Equation: $\lambda_k = g(X_k, \lambda_{k+1}, U_k)$
- Optimal Control Equation: $U_k = \Phi(X_k, \lambda_{k+1})$
- Boundary Conditions: *TPBVP* (split)

NPTEL OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 8

So, what is the summary? Summary of necessary conditions is something like that, state equation is, is available to us and the costate equation is something like this. Remember, these are all functions of state and control X k and U k and obviously, lambda k plus 1 also here. So, I can write it as lambda k is some function of X k, U k and lambda k plus 1 as well and optimal control is ultimately a function of this one, X k, lambda k plus 1.

Remember, this is the equation, this is the equation if you solve for U^k from here. Ultimately this is going to be a function of X^k and λ^{k+1} actually. Remember, X^{k+1} will be substituted by that, f of X^k, U^k . So, that means, what you get here is just a function of X^k, U^k only, basically. So, ultimately, this optimal control equation can be written something like that.

So, essentially, (λ) conditions are also split. So, what you see observe here is a, we actually landed out with something similar to what we know, what you knew earlier in calculus variation approach, actually. So, your state equation, costate equation, optimal control and boundary condition, that is what was the calculus variation approach actually. However, we started with something like this utility function and all that actually in cost-to-go and things like that. These are the concerns coming from dynamic programming actually and these are all happening in discrete domain, so obviously, these are some approximations involved and all that, actually. So, that is why it is approximate dynamic program actually.

(Refer Slide Time: 14:54)

References

- **Werbos P. J.**, Approximate Dynamic Programming for Real-time Control and Neural Modeling. In D. A. White, & D. A. Sofge (Eds.), *Handbook of Intelligent Control*, Multiscience Press, 1992.
- **R. Padhi, N. Unnikrishnan and S. N. Balakrishnan**, A Single Network Adaptive Critic (SNAC) Architecture for Optimal Control Synthesis of a Class of Nonlinear Systems, *Neural Networks*, Vol. 19, No. 10, Dec. 2006, pp.1648-1660.

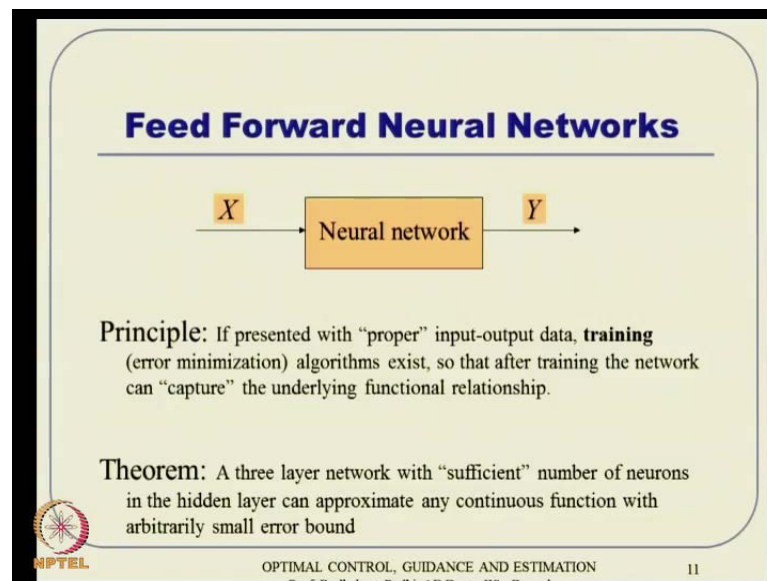
NPTEL OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 9

Some references, if you want to see some of these derivations little more carefully and all you can see this reference, which is a very standard nowadays. This is actually a chapter written in, in a handbook. The handbook is written for intelligent control in 92, about twenty years back actually. But there is nice chapter here, which talks about approximate dynamic programming actually. And we can see our paper also where these

derivations are summarized again actually, but this paper will talk about more on what we are going to do next in the adaptive critic and, and single network adaptive critic and all that actually.

Alright, so this, this adaptive critic design for infinite time regulatory problems, let us see that one first, then we will talk about SNAC as in some sort of an extension in that actually.

(Refer Slide Time: 15:49)



So, now, here I assume, that you have some little bit idea of neural network and if you do not have much of ideas, then nothing to worry so much. There are tool boxes available in MATLAB and other softwares and all that where neural network training algorithms are available to experiment actually. And again, what is neural network training? You can think about neural network is something like a function approximation block work sort of thing where you, if you given some sort of an input you get some sort of output, but if your weights are selected properly, then you will get proper output given a set of input actually. And how do you get this weights adjusted properly? That require some sort of a training and then, that is, that is a different ball game altogether. There are various kinds of training available and all that, what you are interested here is very standard weight propagation supervised learning basically, supervised training.

It essentially tells us, that it may, why, I mean the whole idea here is I can explain little bit here, let me ask, let me talk about something, like a some function of x and a bunch

of weights, actually. Let us say, then x is known to me because that is input, but w is something not known to me, actually. Now, what I, what I think here, what the problem, problem is, this is the actual output, that I am getting from here. After selecting a set of ϕ and all that, little more, little more explanation I can think about w , I mean, y equal to something like w^3 , let us say ϕ reason for that, w^3 transpose, something like ϕ^3 of x if I write it, where ϕ^3 is a set of basic functions, which, which I select. So, I know about that actually. Then, I will do something like, I will take ϕ^2 , then I will multiply with w^2 transpose and then I will take entire thing, something like ϕ^1 and then, then what I do is I write something like w^1 transpose and this it can continue actually.

Let me stop here because it is sufficient to do actually this much only, so this is my y actually. So, what I am getting here is all this ϕ^3 , ϕ^2 and ϕ^1 . These are defined functions actually, predefined functions, typically taken as sigmoid function, tangent sigmoid things like that, whereas w^1 , w^2 , w^3 are unknown parameters to be fixed and it needs to be fixed in such a way, that given my set of x , whatever my input variable, my output variable should match that actually. So, for that we, we, let us say we required some sort of a training data, that means, sorry, that means, what I am telling here is let us say this y , what I am telling y is should be y^* like that, actually. That means, if it is not, y is not y^* to begin with, but if my weights are selected properly, then y will be a y^* actually. So obviously, there is an error δy , which is $y - y^*$ and what I am doing here is I want to fill it, I mean, kind of minimize a cost function, error cost function, let us say something like $\frac{1}{2} \delta y^T \delta y$ actually. So, this is actually a function of all this weights, the weights w^1 , w^2 , w^3 .

So, what is done here typically is $\frac{\partial e}{\partial w^1} = 0$, $\frac{\partial e}{\partial w^2} = 0$. This is, this is also $\frac{\partial e}{\partial w^3} = 0$, like that. So, if you put it in and then go through this chain rule of derivatives and all that you will be able to come up in some sort of necessary conditions and typically, this optimization is carried out in some sort of a numerical optimization sense actually. That means, this close form solutions are typically difficult to get. So, this, this equations, even though we know, we do not put it, we try to directly start with some sort of initial guess values and then try to update the sequential update sort of thing, which will lead to this, this error minimization actually. That means your y , actual y will, will converge to y^* . It will ϕ in the like a desired output in the trained domain wherever in the domain, that is selected for training within

that domain it is. So, going to (()) like that actually, that is the whole idea of neural network synthesis training and all that, actually. If interested, you can see many, many books available for, for neural networks, actually.

Now, coming back what it tells us, that if presented with proper input-output data when that proper is again, you have to really do a careful job to provide a proper training that are basically, then what it tells is, that there are training algorithms essentially, error minimization algorithms that exist, so that after training the network and capture the underlining functional relationship. These are not number fitting; these are function fitting, actually. That means, once the function is fit actually, then given a different attacks, remember, x of bunch of numbers.

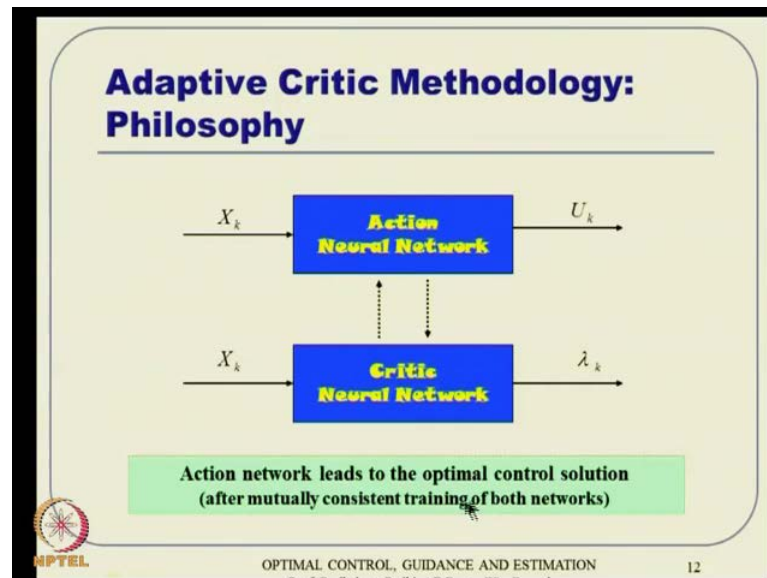
Now, actually, if you have a bunch of numbers and you got some y star and you have to train the network in such a way, that Δy is minimum actually, that means, this y star selected in such a way, that Δy is minimum, so that means, the functional relationship is captured. So, in other words if I present a different set of input state x actually, whatever that is, then I will again get some sort of a similar output, that I would have otherwise got, actually, using all this optimality, I mean, all these ideal conditions, what I mean actually. So, this is the idea of a neural network.

So, and then there is a theorem, which also tells us, that three layer network with sufficient number of hidden layer can approximate any continuous function within arbitrarily small error bound in compact set, obviously. So, set has to be kind of closed and bounded and all that actually, that is how let me quiet, let me complete that in compact set. Alright, so this is, this is what the basic knowledge, that we will want, that is all, actually. There is a neural network, which can approximate some underlining function. There are training algorithms, that exists, which, which can do the job for us and it. There are also powerful theorems, which tell us, that only three layer network, something like this can approximate any continuous function that is more important, actually. So, that means, there is some degree of theoretical guarantee thing, that, that comes along, actually.

But still, there is also, remember the way of selecting this w , the weight, weighting matrix dimensions and the function selections and all that are still art, actually. That means, there is no clear cut theory, which will tell you what kind of functions, that you

need to select for ϕ_1 , ϕ_2 and ϕ_3 and it will not also tell you what is the dimension of that. That means, what is the, what is the number of functions, basis functions from $\{ \phi_i \}$, that you want to select in each layer. They are still subject to, kind of, open question basically, but normally it turns out, that actually not a very large network, it is about four, five the hidden layer neural that have, actually.

(Refer Slide Time: 23:31)



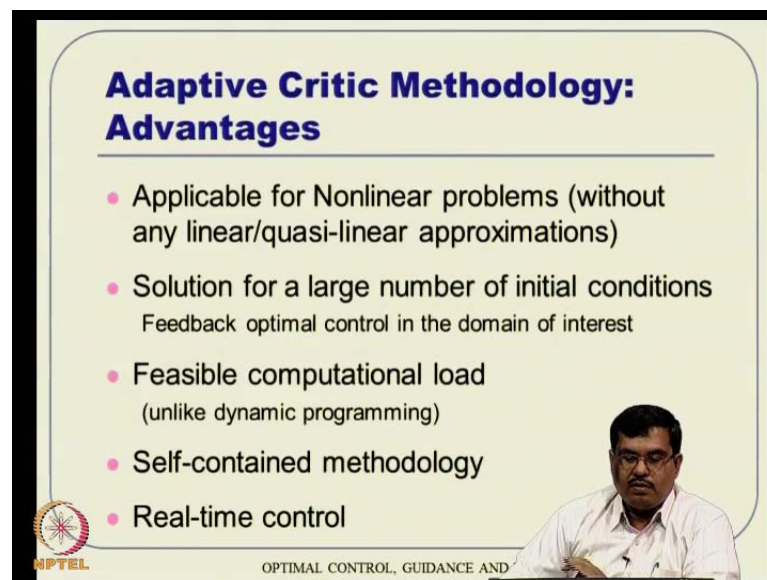
Now, with this background, **what is**, what is adaptive critic methodology? Let us say this is the methodology, which tells, that let me now assume a neural network, which I call it is as X_n network, for the all the detailed is given set of state of k , it will predict my U_k , that is my ultimate objective, remember. Now, y is my U_k , here y is U , then I have got a close form solution for control basically. Just imagine, that is, this y is essentially, this y is essentially my U function, this is my U . So, what I got? What I, what I got is actually a control, which is a state feedback from actually, this is what, what I can do actually.

How about to get that? There is a necessity of knowing λ s and all that, the information, that we have is only X_k , so there is a necessity of another network, which we call as critic network, which will nothing, which will do nothing, but predicting λ_k actually. That means, this networks job has to predict an optimal λ_k knowing k and this network's job is to predict an optimal control U_k knowing k . So, ultimately, this is the network, that you are interested in, but on the way we need this also and there is an interaction that will go on actually. That means, while training, X_n

network will assume, that critic network is optimal and while training critic network we will assume, that action network is optimal and that has to be some sort of cycle training and all that.

And ultimately if everything converges, then, then we will tell, that this network, that has converged here is nothing, but my optimal control in the solution actually, U k, this is what it tells you. Finally, this X n network leads to this optimal control solution, but that will happen only after mutually consistent training of both the networks.

(Refer Slide Time: 25: 18)



**Adaptive Critic Methodology:
Advantages**

- Applicable for Nonlinear problems (without any linear/quasi-linear approximations)
- Solution for a large number of initial conditions
Feedback optimal control in the domain of interest
- Feasible computational load
(unlike dynamic programming)
- Self-contained methodology
- Real-time control

NPTTEL OPTIMAL CONTROL, GUIDANCE AND

What are the advantages of this method? Before you see the little steps and all that is applicable to, for non-linear problems in general without any linear or quasi-linear approximations of the system dynamics, you can directly take the system dynamics as it is, actually. It is also solutions for a, for a large number of initial conditions (()), what you are looking for is some sort of a semi-global solution and all that. Wherever the network is trained in that domain, it will, it will operate something like an optimal control actually.

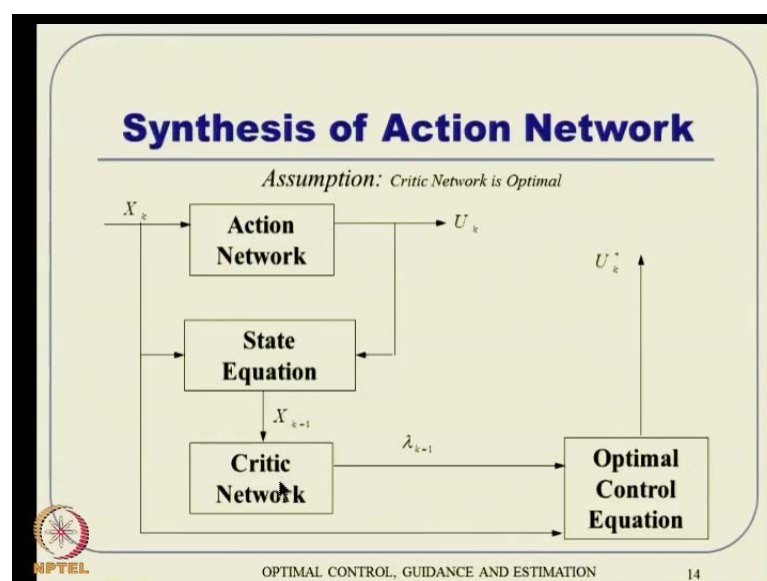
We are getting a close form solution, remember that this expression what you saw here actually close form solution in a way, but this, this will happen to be a close form solution only within the domain of training. Wherever you have trained the network, within that domain is, this relationship is valid actually. So, because of that it is a, we can interpret, that the some sort of a semi-global solution and all that because the domain can

be expanded as we like actually, at least theoretically. So, we are getting something like feedback optimal control in the domain of interest, that is, that is the advantage actually. There is something like a feasible computational load, even the problem dimension increases and your control application duration increases and all that actually, still it is not like infeasible computational load, like dynamic programming.

It still need sometime couple of hours probably to train the network depending on the computer first and then software, that you are using and all that still needs computational thing of the order of minutes and hours probably, but not like infinite infeasible computational load. For example, if you really use this classical technique for dynamic programming for a particular response guidance problem, it takes about some twenty, twenty-two years altogether, actually. So, that is not we are not talking like that, still minutes and hours, but still it is feasible actually.

Then, it is a self-contained methodology; it does not talk about some other method, solves it and then just ends the network. So, it does not talk about, that it is a methodology by itself actually, ultimately leads to real time control because what you are getting is controlled as a close form solution, finally. So, it is given as a, after the training process is over you have got the optimize guides already. So, what you are doing here is just simply evaluating this, this function U is a function of state actually.

(Refer Slide Time: 27:42)



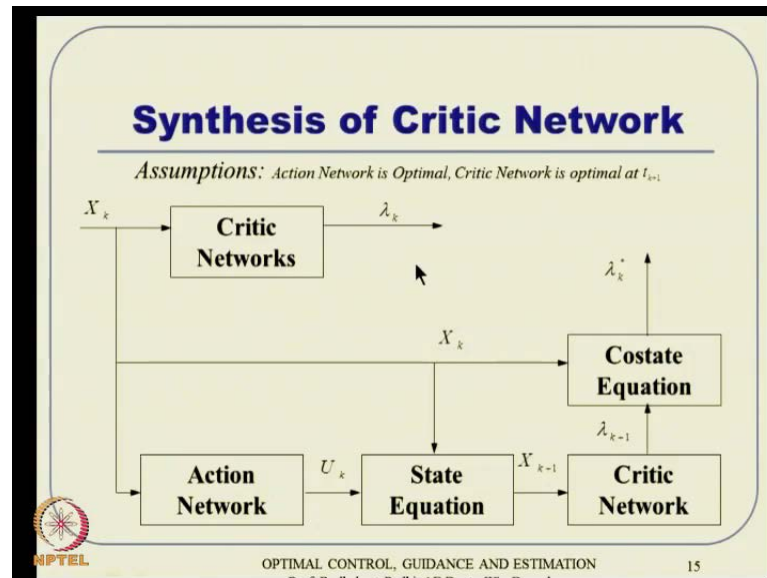
Now, the question is how do you do that? So, first thing is let us see how to synthesize the X_n network by a synthesis. What I mean is, how do you come up with the, the desired values of the output actually. Remember, X_n network takes X_k as an input and gives U_k as output, but this training process demands, that you have a corresponding U_k^* , which is the desired value of U_k actually.

So, how do you get that there? So, let us take it through, and the results that we have is state equation, optimal control, and, and costate equation, you know that actually. So, we give some, assume certain network, which $(())$ and all that, actually. So, we get some sort of a U_k and by the way, this initialization does matter. And I will talk about a process of good installation also actually, but anyways, there is, after initialization is X_k , if I feed it to the network, I will get U_k and now, that I have got X_k and U_k I have got, I can feed that into state equation and hence, I will get X_{k+1} and if I get X_{k+1} , remember I am assuming, that critic network is optimal here, so this, using this optimized critic network I will get λ_{k+1} . Now, that I have got λ_{k+1} and X_k also available with me, I can feed it both, $X_{\lambda_{k+1}}$ and X_k and optimal control equation and I will get U_k^* , actually.

Now, given this input this should be my output, this is my actual output. So, there is an error between the two. So, using this error minimization algorithm and all I can, I can adjust the weights, that is, that is my training actually. So, this training process continues like this and remember, this is, even though it is can be a single point, but ideally speaking we have to, we typically do this, I mean this batch processing sort of thing. That means, you take a, a set of, I mean, state vectors, that means, let us say hundred states together, hundred states means is, let us say the system has X_1, X_2, X_3 , then the value for X_1, X_2, X_3 selected hundred times actually from the domain. We have hundred things like that and hundred outputs like that. So, we have this $(())$ algorithm, but the cost function accounts for error in the each components of actually. So, the idea here is if you do that batch processing, the, the local minimum trapping possibility is minimum actually. So, that is why you do that.

But having come here for each other state, that you take you will get corresponding U_k^* actually that way. So, we have this error and then try to optimize this weights based on that information, actually. So, that is X_n network training.

(Refer Slide Time: 30:15)



What about critic network training? This critic network training will talk about something like this. We have this X_k , I mean, gives you λ_k , but you need to get a λ_k^* actually, these are desired λ_k . So, how to do that? This is like, this we got X_k , we can also feed that in the X_n network, which we assume is already optimized. Then, you get some sort of optimal control U_k . Now, X_k and U_k is available, so you got X_{k+1} . And remember, this, there are two assumptions here, one is, X_n network is optimal and critic network, that you are going to train for t_k is also assumed to be optimally t_{k+1} . The reason being, finally, my boundary condition is known for λ_n is actually a function of X_n . So, that means, I know a final condition actually from the boundary conditions, actually.

So, I can, I can assume, that my future time, my, my critic network is already optimal actually and in, in fact, it, that entire observation gives you some sort of a training approach for critic network, actually. In other words, for finite time problems we will like to train the network first by using this, this boundary condition and then proceed backwards actually for infinite time (()), I mean, final value of λ happens to be 0. So, we train it in the, in the domain of very close to 0 and all that actually and then, try to enlarge the boundary all that way.

But anyway, coming back, what it means? The critic network is a, is already optimal t_{k+1} . So, that means, if I put it will get a λ_{k+1} also. Now, I got X_k and

lambda k plus 1, so using costate equation I will get it a desired lambda and again, I can train this based on this error value between that way, actually. Alright, so this is the procedure. If I train action network, receiving critic network is optimal and the way and the vice versa, but also remember these are on, on the way, we are using state equation, optimal control equation, as well as, costate equations. So, all the three equations are getting used and hence, after mutually consistent training we will get this optimal control basically.

(Refer Slide Time: 32:20)

Initialization of Networks: Pre-training


- Linearize the Problem
- Formulate a Lumped Problem and Obtain LQR Solution

$$X_{k+1} = A_d X_k + B_d U_k$$

$$J = \sum_{k=1}^{\infty} (X_k^T Q_d X_k + U_k^T R_d U_k)$$

$$\lambda_k = S_d X_k \quad U_k = -K_d X_k$$

- Train the networks



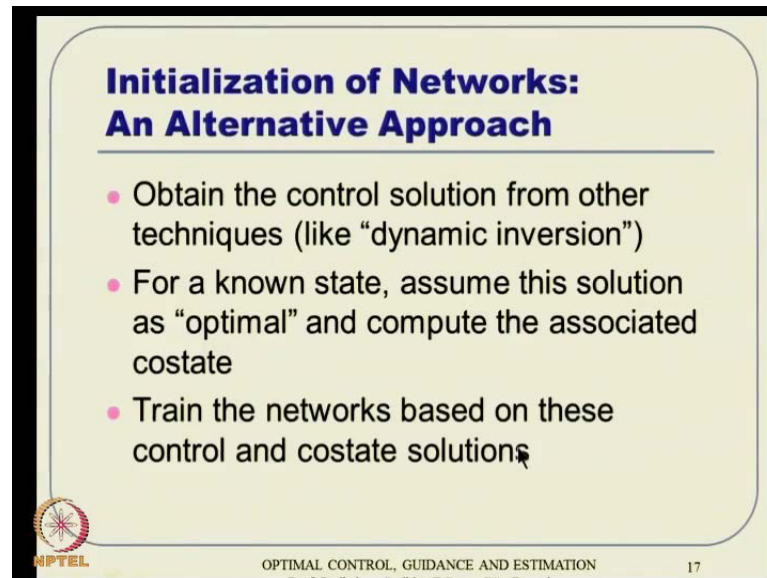
OPTIMAL CONTROL, GUIDANCE AND ESTIMATION

16

How do you initialize the network? As I told that is also important and for initialization this is what standard method that you propose, that you keep linearism the problem first and formulate something like a problem in a linear domain actually. That means, you have this, formulate, what this is a small error here, formulate rather, not lumped per se. It is LQR problem, linear quadratic regulator problem, because the cost for the state equation becomes linear and cost function is quadratic is LQR problem and obtain the LQR solution for that. This solution will tell us, this is, this is our standard result coming from (()) equation and things like that as the discrete optimal solution and in that, that is also one lecture I have already covered actually, can see that if want. So, lambda k is S d into X k and U k is minus K d into X k where X T and K t are available from this LQR solution. So, I solve in the Ricard equation and all that actually. So, now, because these relationships are available, you can actually train the network using these relationships that is called, pre training actually.

So, we start with very arbitrary set of ways, but then first you train the network, so that the networks can mimic this relationship at least. That means, they are actually mimicking this, they are actually predicting this optimal control relationships for linearized version of the problem actually. So, that actually gives some sort of (()) initialization, so that the non-linear problem training can happen quickly, actually.

(Refer Slide Time: 33:59)



**Initialization of Networks:
An Alternative Approach**

- Obtain the control solution from other techniques (like “dynamic inversion”)
- For a known state, assume this solution as “optimal” and compute the associated costate
- Train the networks based on these control and costate solutions

RPTEL OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 17

One of the alternative idea also exists, that you can also obtain some sort of a control solution from other techniques, let us say, of using dynamic conversion. Then, for a non state assume, that the solution is already optimal even though it is really not optimal and there you can complete the associated costate. But this is the trick, trick, you think here because you know, the control is actually a lesser dimension, but the costate is actually a dimension, actually. So, you will know some sort of a studio inverse and all that where I personally sometimes do not like actually.

If the dimensional discrepancy is high you have one control and let us say, in eight, nine states, then obviously, this is not the good approach, actually. But if it is something, you, compatible dimension, it is are three, three stage, three control or probably something like three, state two control like that, then you can use this actually. The advantage here is we are not using this linearization technique, we are not linearizing the problem actually. (()), so if you do that, then you can train the (()), this control and costate solutions, which you can use it actually for (()).

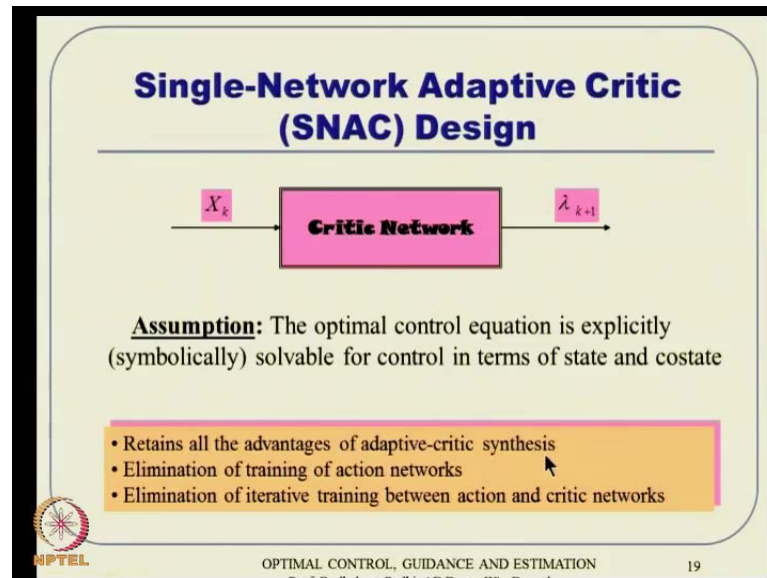
So, this was the idea of adaptive critic, but on the way, as I told, we also have this idea of single network adaptive critic. So, what happens here is if you go back to this approach, this idea here, there are actually two networks. So, essentially, it is something like, I forgot to tell you, that this, this training procedure, that you are talking action critic, we first train the action assuming critic network is optimal, then you go to train the critic network assuming the action is optimal.

Now, what you are doing here? You are assuming critic is optimal and you are, you are saving this actually. So, what you are telling here is this action network is not yet fully trained, we partially trained it. But using this I am actually training the costate equation, but once I change the critic network I have actually changed this network already here. So, there is necessity of retraining the action network again and once I do that whatever action network I assumed in the critic network training, here is already trained. So, I again have to come back to critic network and train actually.

So, this process what you are seeing here, action network training and critic network training, this needs to be mutually consistent. That means, you come, once you train critic network go back to action network, train that again. Then, once you are done that come back to the critic network and train it again. Once you have done go back to that and then train it again taking like that actually and after a while you will find, that no further training is necessary in either of the cycle and that is where you call, that the mutually consistent training has happened actually. So, that is called cycle training sort of thing.

So, the observation here is the cycle training is necessary because this two network structures, basically can you get rid of that because cycle net to cycle training is typically, first of all, it is computational intensive. Second thing, there is a possibility of misguiding actually. In other words, if one network goes to the other one, (()), actually.

(Refer Slide Time: 36: 44)

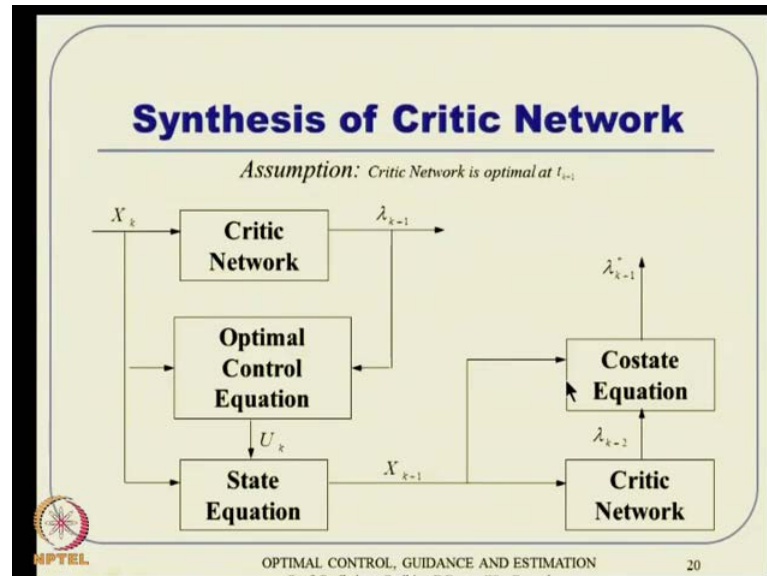


So, here the idea comes something like this. It can be done provided the critic network is actually remodified, I mean, a kind of, instead of assuming λ_k is output what about assuming λ_{k+1} as output rather than let us interpret what is going on and see what is going on, actually. And the assumption here is, most of the time as we have seen in, like derivations and then other approximate dynamic programming and all the U_k is going to be a function of X_k and λ_{k+1} . So, all that I need is U_k is some, something like an explicit close form solution in the form of X_k and λ_{k+1} , which is typically true from most of cases, actually. Now, once that is there, then what it tells us is if I, if I just use a network, one single network, which will give me X_k as a function, I mean, λ_{k+1} is a function of X_k , then if I know X_k , I know λ_{k+1} and I know the optimal control solution as some sort of, like an explicit solution symbolically available as the function of these two variable and $(())$ actually. So, this is what it is.

Once as I, the advantages are like this, it retains all the advantage of integrity, that eliminates the action network and hence, it also eliminates the $(())$ training between action and critic networks, actually. These are the advantages, which, which sets us lot of training time actually in a way, also eliminate some of these approximations in all from critic network anytime. You have a network as the function approximation, that approximation already has also gone by eliminating that actually and it also leads us

these, this consistency the quick training sort of ideas and all that way. So, that is a kind network adaptive critic.

(Refer Slide Time: 38:20)



So, how do, how do (()), let us go back to the structure again. So, we have X_k , we have a critic network to train. So, we have got λ_{k+1} , all that you are assuming here is critic network is optimal at t_{k+1} while we are training for t_k . So, and x_k , I will get λ_{k+1} . Once I use this λ_{k+1} in optimal control will get U_k now and that k and U_k is available, will use it in state equation and get X_{k+1} . And because I am, my X_{k+1} is available I can use it in critic network to get λ_{k+2} rather and because X_{k+1} and λ_{k+2} is available, I can put it in a costate equation and get λ_{k+1} actually.


So, what I am getting here is, this is the desired λ_{k+1} and this may actual λ_{k+1} . So, the difference between that is error based from, that I will minimize, that to train this network actually. So, remember this optimal control state equation and costate equation, all the tree equations are used in the single structure basically here, that results in simplified algorithm programming becomes simple, the training time sets and also things actually.

(Refer Slide Time: 39:28)

Initialization of Networks: Pre-training

- Linearize the Problem and Obtain LQR Solution $\lambda_k = S_d X_k$ $U_k = -K_d X_k$
- Obtain the relationship between X_k and λ_{k+1}

$$\begin{aligned} \lambda_{k+1} &= S_d X_{k+1} = S_d (A_d X_k + B_d U_k) \\ &= S_d (A_d - B_d K_d) X_k \\ &= \tilde{S}_d X_k \end{aligned}$$
- Train the networks


OPTIMAL CONTROL, GUIDANCE AND ESTIMATION
21

How do you realize? Remember, the LQR solution can give us only this, lambda k is some k, it is not lambda k plus 1 is not available. So, how do you do that? But you know, that this is also available, some hands you can construct it, that means, lambda k plus 1 is nothing, but this is a constant matrix, just being to the k plus 1, but X k plus 1. We can substitute the state equation, linearised state equation, U k can substitute by function of X k and hence, we will get something like lambda k plus 1 as a function of X k. So, using this relationship you can pertain the networks and then go for the training, actually.

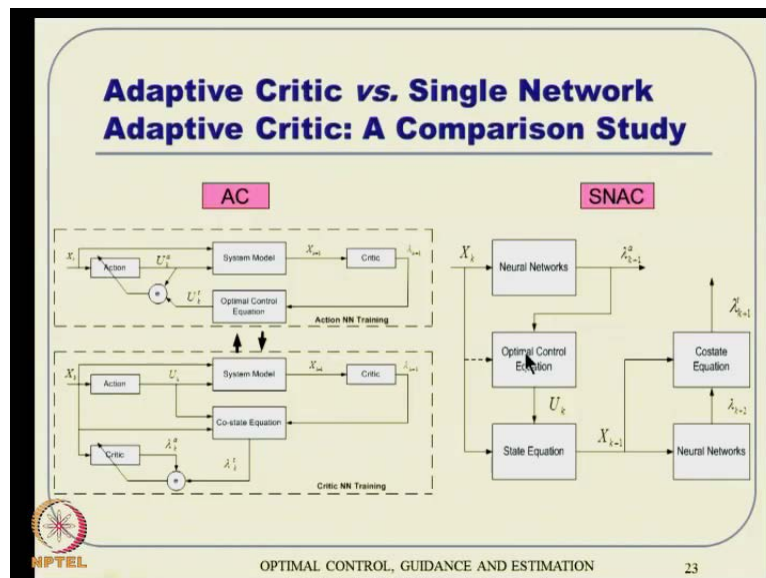
So, now for the rest of the time we will talk about some two, three example problems and give you some sort of a comparison study to see where performance improvement happens. Remember, the result-wise it will not vary much if you do consistent training actually, but, but other things will matter also. Remember, this, this because of this assumption, that are this lambda k plus 1 critic network is already optimally t k plus 1, that you are repeatedly using and for that there is a necessity of the cycle training, I mean, this telescopic training and all that actually.

So, what it means is, let me, let me (()) that suppose you talk about a domain of training, that means, we know that our state, let us say, some sort of a scalar example. This is my time and this is my X actually. My X is going to vary, but it is going to vary within that actually, no matter what. Wherever the initial condition it will vary, but it will vary within that actually. So, this, this is the assumption actually, whatever that. So, then I do

not have to worry about outside this bound and all that. However, what I need to do is, I mean, this is my total training (()). What I need to do here is something like this, even though I know, that this is my X max and this is my X min, then what I do is I just slice this thing with the small domain first actually.

We will train this network in that domain because close to the 0 line my (()) thing, which is a linearised expression and all are valid actually. So, we will train this. Once it is done I will expand this little bit more, once it is done I will expand this little more and once it is done will expand this little more actually. So, this is initially starting with this domain, then starting with that, then going to that and that and things like that. This is called telescopic training actually and while you enlarge the domain, that this in for not forgetting also, that means, even if you talk about some sort of a full domain, let us say, this you are generating something like thousand data points from there, then about ten percent I will still recommend, that you keep on generating from here explicitly basically. That means, this larger bound should be nine hundred, then this smaller bound should be hundred actually. The reason for that is the error of larger values should not put away the error between the smaller values actually. So, it means, as you should not forget the relationship close to 0 basically, that is the whole idea about this, this (()) called telescopic training actually, which is invariably helpful. So, all that is this.

(Refer Slide Time: 42:42)



So, there is a comparison (()) set of thing. You know, the adaptive critic operates in this kind of idea, you have action training, critic training, there is cycle training involved and things like that, whereas single network adaptive critic does not talk about, that is a strains in one single activation sort of thing, actually There is no cycle training involved, there is no action network per se actually, there is only one set of (()) there.

(Refer Slide Time: 43:02)


**Comparison Studies:
Three Motivating Examples**

Scalar Problem: $\dot{x} = x - x^3 + u$

Van-der Pol's Oscillator : $\ddot{x} + \alpha(x^2 - 1)\dot{x} + x - (1 + x^2 + \dot{x}^2)u = 0$

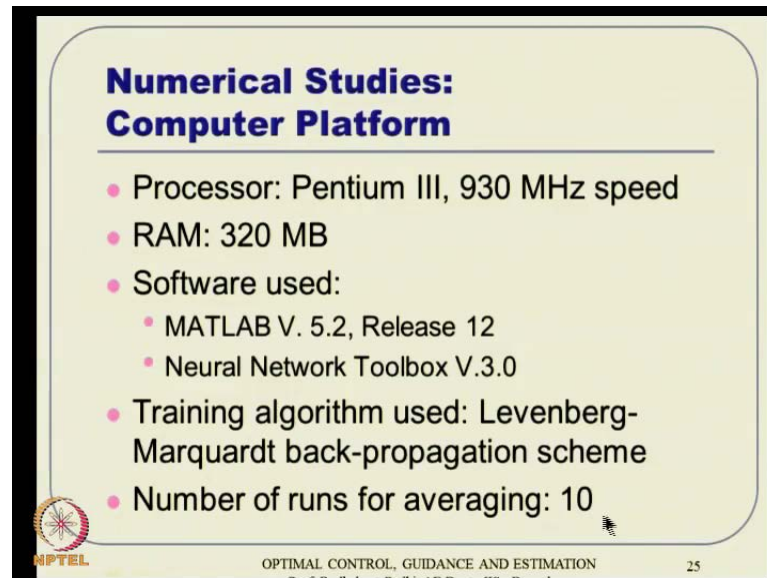
Electrostatic Actuator : $\dot{Q} - \frac{1}{R} \left(V_{in} - \frac{Qg}{\epsilon A} \right) = 0$

$$m\ddot{g} + b\dot{g} + k(g - g_0) + \frac{Q^2}{2\epsilon A} = 0$$


OPTIMAL CONTROL, GUIDANCE AND ESTIMATION
24

So, we will talk about three problems here. One is a standard scalar, sub-scalar problem that we discussed in the previous lecture. The other one is the two-dimensional thing, in case two state, actually, this (()) equation. This is (()) equation, again a very standard benchmark problem, Van-der Pol's oscillator. There are reasons for that, this actually goes for limit cycle oscillations and hence, the control. If the control is not good you will not be able to regulate the system, it will not be able to drive the stage towards (()), basically. Then, I will extend that little more practical problem, which talks about (()) system, (()) system, which is the (()) problem at micro electro-mechanical system, is electrostatic sort of thing, alright.

(Refer Slide Time: 43:52)



**Numerical Studies:
Computer Platform**

- Processor: Pentium III, 930 MHz speed
- RAM: 320 MB
- Software used:
 - MATLAB V. 5.2, Release 12
 - Neural Network Toolbox V.3.0
- Training algorithm used: Levenberg-Marquardt back-propagation scheme
- Number of runs for averaging: 10

NPTEL OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 25

So, first problem first. This is a tie for all these things, what you have done is a little bit older computer and all, that used to be same for all exercise for fair comparison. We used something like a Pentium-three processor with a little more, little less I mean, 930 megahertz speed sort of thing, RAM was less, 320 only, software used was MATLAB version 5.2. Now, versions are quite advanced, it is almost like MATLAB 2012 is available actually, but anyway, that is, that is not the point. Neural network toolbox is also improving, but the, the exercise is carried out with the toolbox version 3.0 basically and training algorithm uses something called Levenberg-Marquardt algorithm, which is very standard in neural network techniques. So, the kind of the most powerful algorithm, that exists in the network training domain actually and the number of runs for averaging is 10. So, also remember, that when you run this algorithm there are other processing things also goes on there. So, what you are doing here is kind of taking around ten times and then trying, kind of averaging all that, actually. So, we will smooth out some other relationship here actually. It will smooth out some of the noisiest relationship very less fluctuation, but it will actually smooth out that part also, basically.

So, this is the scalar problem, we have the standard equation, we have a standard cost function also, q and r equal to 1, 1. Then, this is the, cost, this is the solution that we have derived in the previous lecture also. This is the $(())$ because of the availability of the closed form solution and hence, you can compare your results with that actually. So, we have, first thing is discretize the state equation using all the integration formula,

discretize the cost equation as well, discrete the cost function as well and the costate equation can be derived using this approximate dynamic programming results like this and optimal control equation turns out to be like that. Remember, q and r are both 1 here, actually.

(Refer Slide Time: 45:37)

Implementation

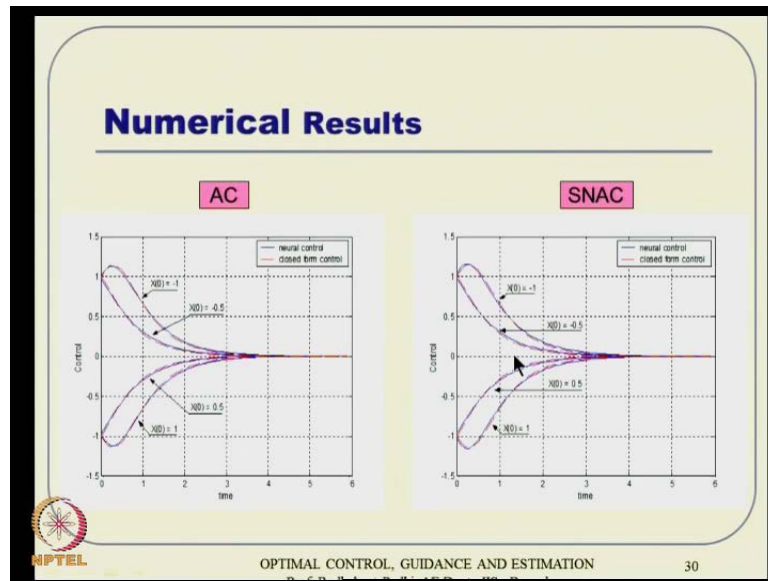
AC	SNAC
<ul style="list-style-type: none"> • Critic Network : 1-4-1 • Action Network:1-4-1 • Convergence Criteria: <p><i>Critic</i>: $\ \lambda_p^i - \lambda_p^a\ _2 / \ \lambda_p^i\ _2 < 0.01$</p> <p><i>Action</i>: $\ u_p^i - u_p^a\ _2 / \ u_p^i\ _2 < 0.01$</p> <p><i>Cycle</i>: $err_{c_n} - err_{c_{n-1}} < 0.0002$ $err_{a_n} - err_{a_{n-1}} < 0.0002$</p>	<ul style="list-style-type: none"> • Critic Network : 1-4-1 • Convergence Criteria: <p>$\ \lambda_p^i - \lambda_p^a\ _2 / \ \lambda_p^i\ _2 < 0.01$ $p = 1, 2, \dots, n$</p>
<p>Weights: $q = r = 1$</p>	

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION

28

So, the network structure is 1, 4, 1, 1, 4, 1 for here adaptive critic, here it is 1, 4, 1. The convergence criteria selected is something like 1 percent, but here I have to select for two networks and the cycle convergence also we need to select, that was selected very small actually. Remember, this is 0.02 percent and things like that actually. It is not percentage per se, it is an absolute value here. So, these are percentage errors, but this is absolute values actually. So, select something like this, but this is actually not needed here because I know, cycle training here wherever things are needed, these are comparative, actually.

(Refer Slide Time: 46:12)



So, result-wise you see, both the results are, this is adaptive critic results compared with close form solution, which is very close to each other. So, the adaptive critic is working. SNAC, which is also working, this is also very close to the close form solution. So, the point here is both are working, it is not the, I mean, but the control solution is also, you can see both are happening to be fairly close to the close form solution actually.

So, both are happening, that is not a problem, but what, what is the advantage? The advantage is in SNAC style actually.

(Refer Slide Time: 46:45)

Time Comparison:

AC				
# OF ITERATIONS	CRITIC TRAINING	ACTION TRAINING	CONVERGENCE CHECK	TOTAL
	TIME (sec)	TIME (sec)	TIME (sec)	TIME (sec)
1	5.713	4.48	3.747	67.994
	5.104			
	5.058			
	5.026			
	5.026			
	5.104			
	5.073			
	5.026			
	4.917			
	5.088			
2	4.995	4.137	3.699	42.088
	4.995			
	5.057			
	4.854			
	4.745			
	4.792			
	4.808			
	4.776			
	4.183			
	3.668			
3	4.776	4.183	3.668	12.627

TOTAL TIME (sec) = 122.709

SNAC			
# OF ITERATIONS	CRITIC TRAINING TIME (sec)	CONVERGENCE CHECK TIME (sec)	TOTAL TIME (sec)
1	5.625	1.266	6.891
2	5.406	1.25	6.656
3	5.516	1.25	6.766
4	5.5	1.234	6.734
5	5.675	1.25	6.928
6	5.375	1.25	6.625
7	5.532	1.25	6.782
8	5.375	1.25	6.625

TOTAL TIME (sec) = 53.907

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 31

So, let us see what happens here. See, the computational time sense, you can see, that number of iteration; that means, the cycle training sort of things and all. It talks about some time of computation and all that, wherever SNAC talks about only iterations like that. What you see here? One, two, three, four, five, six, seven, eight, nine, ten, ten iterations, this algorithm is supposed to converge actually. Then, you go to the 2nd iteration, 3rd iteration like that actually, here is not needed. So, what is ten here is simply one, eight iterations here we have done, actually.

So, essentially, if you see all this, ultimately the time to train is what matters actually. And finally, if you see the total time taken for all these, you add up all that, you will get it something, the total time here and even the total time is something like this. So, essentially, it turns out, that time of training is essentially 43 percent of adaptive critic in SNAC. So, that is roughly the kind of 40 percent we can say, which is the lot of advantage for problems, actually.

(Refer Slide Time: 47:44)

**Cost Function Comparison ($t_f = 6$)
For different initial conditions**

INITIAL STATE X(0)	COST (SNAC)	COST (AC)
-1	1.4499	1.4591
-0.8	1.0521	1.0586
-0.6	0.6461	0.6476
-0.4	0.3032	0.302
-0.2	0.0779	0.0773
0.2	0.0779	0.0772
0.4	0.3031	0.3021
0.6	0.6461	0.6478
0.8	1.052	1.0588
1	1.4497	1.4592

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 32


Now, cost functions for each of the initial condition, we can actually evaluate the cost function and plot it in the, the claim here is again cost functions are fairly similar to each other, actually. So, what is your claim here? Both will, both will work, but SNAC will work slightly better in the sense of computational advantage actually, primarily and it will also eliminate your other programming difficulties and small approach (()) and things like that, actually.

(Refer Slide Time: 48:14)

Van-der Pol's Oscillator

- System Model:

$$\ddot{x} + \alpha(x^2 - 1)\dot{x} + x - (1 + x^2 + \dot{x}^2)u = 0$$
- Cost Function: $J = \frac{1}{2} \int_0^{\infty} (X^T Q X + R u^2) dt$
- Cost Function: $J = \sum_{k=1}^{N \rightarrow \infty} \frac{1}{2} (X_k^T Q X_k + R u_k^2) \Delta t$
(discrete)



OPTIMAL CONTROL, GUIDANCE AND ESTIMATION

33

(Refer Slide Time: 48:23)


Necessary Conditions of Optimality

- State Equation:

$$\begin{bmatrix} x_{1,i+1} \\ x_{2,i+1} \end{bmatrix} = \begin{bmatrix} x_{1,i} \\ x_{2,i} \end{bmatrix} + \Delta t \begin{bmatrix} x_{2,i} \\ \alpha(1 - x_{1,i}^2)x_{2,i} - x_{1,i} + (1 + x_{1,i}^2 + x_{2,i}^2)u_k \end{bmatrix}$$
- Costate Equation:

$$\begin{bmatrix} \lambda_{1,i} \\ \lambda_{2,i} \end{bmatrix} = \Delta t \begin{bmatrix} q_1 x_{1,i} \\ q_2 x_{2,i} \end{bmatrix} + \begin{bmatrix} 1 & \Delta t[(2x_{1,i}u_k) - 1 - (2\alpha x_{1,i}x_{2,i})] \\ \Delta t & (1 + \Delta t)[(2x_{2,i}u_k) + (\alpha(1 - x_{1,i}^2))] \end{bmatrix} \begin{bmatrix} \lambda_{1,i+1} \\ \lambda_{2,i+1} \end{bmatrix}$$
- Optimal Control Equation:

$$u_k = -r^{-1} (1 + x_{1,i}^2 + x_{2,i}^2) \lambda_{2,i+1}$$



OPTIMAL CONTROL, GUIDANCE AND ESTIMATION

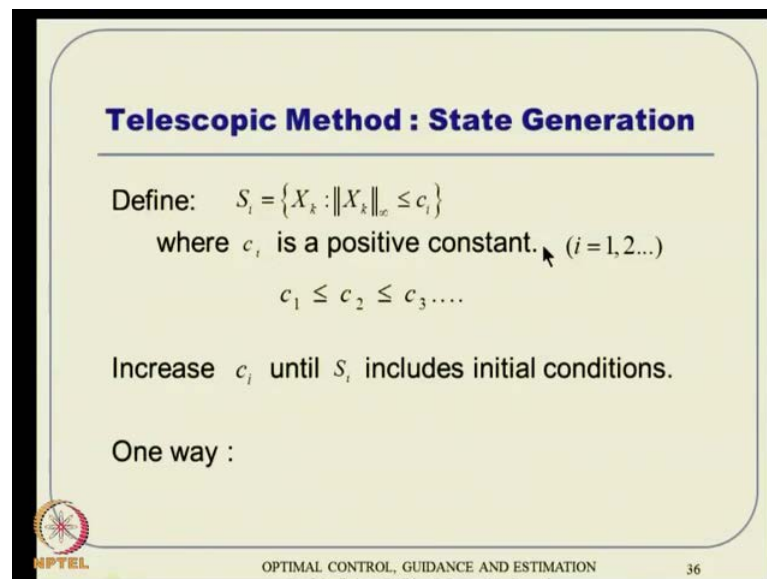
34

Coming to the second equation, second problem, this is a wonderful oscillator, again we will talk about a quadratic cost function and things like that. Cost function is discretized; state equation, remember, is the two state, two state problem now. So, this is also discretized and written in the form of Euler integration. Costate equation can be derived from the standard costate equation that you discussed about; optimal control is like this. So, using this thing we will we will formulate this problem and then selection criterion has to be selected properly. So, critic network, action network and then there is 5 percent error here. And then, absolute values, error in the cycle training is something like this

and wherever these things are comparative, I mean, wherever things possible we need to be comparative.

So, the critic network here is 2, 6, 2 structure, is also 2, 6, 2 structure and convergence criteria here is 5 percent, here is also five percent actually. And also, there is one point to note here, that nice own experience, I will recommend something like a breakup network actually. In another word, even though the critic network can be actually, theoretically solved in, in terms of single network really, but because there are two outputs involved I will rather prefer, I have two sub networks, which is one network is 2, 6, 1 and the 2, 6, 1 also. And then, I will consider, that the entire structure is something like a single network actually, but that is my own preference, need not be like that always, actually. So, this is like this and then it, once you do the comparison study, q and r are selected that way.

(Refer Slide Time: 49:46)




Telescopic Method : State Generation

Define: $S_i = \{X_k : \|X_k\|_\infty \leq c_i\}$
where c_i is a positive constant. $(i = 1, 2, \dots)$
 $c_1 \leq c_2 \leq c_3, \dots$

Increase c_i until S_i includes initial conditions.

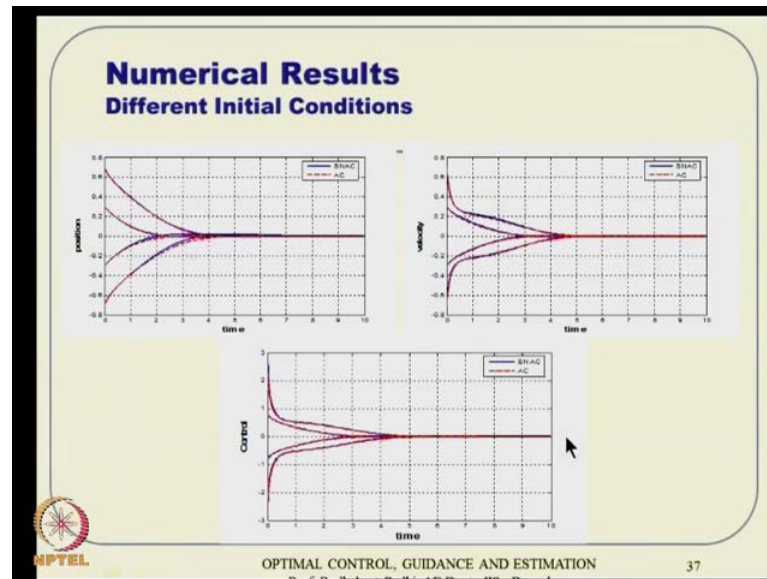
One way :

 NPTEL

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 36

And here, there is the cycle training idea there, sorry, telescopic training ideas. First, you train with the smaller domain, then enlarge area and things like that. So, this first is 5 percent of the total domain and then, subsequently it is increased by 5 percent is a normal, actually.

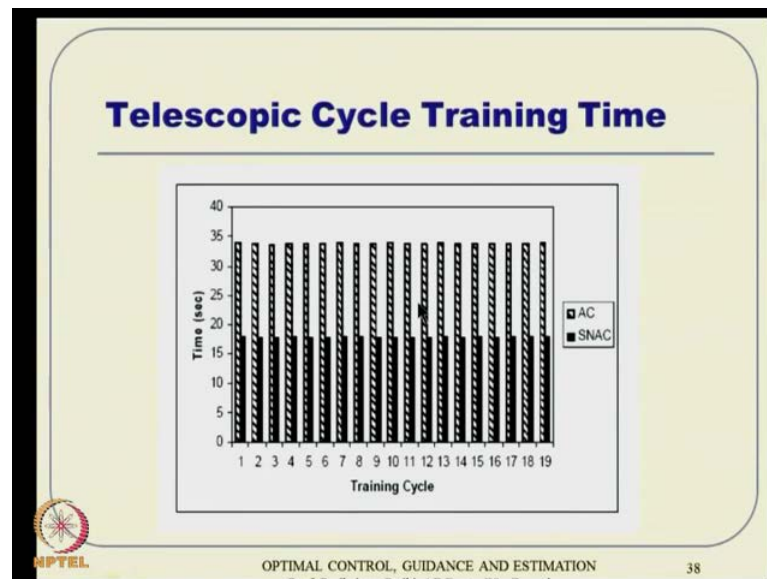
(Refer Slide Time: 50:01)



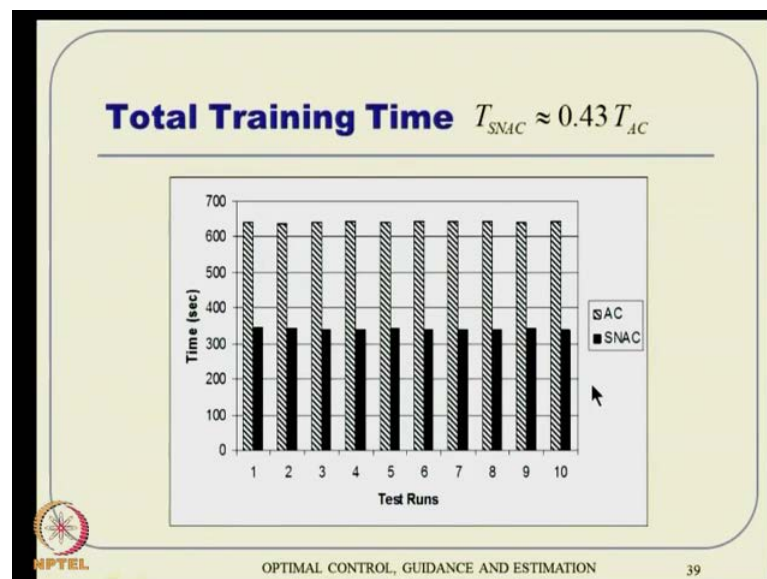
So, again you can see if there is no $(())$ solution we cannot, we cannot compare individually, but comparison with respect to $(())$, I mean, adaptive critic and SNAC is always possible. So, you train both those things, both, which we solve and then plotted together for the different initial conditions and all, actually. So, you can see, that the position and velocity both are given to 0 and remember, this is a limit cycle problem and otherwise if control is not zero for any non-zero initial condition, you will never be able to go towards zero conditions. You will not, you will never be able to drive this state towards 0, it will go towards the limit cycle.

And what is the limit cycle $(())$, that if you do not know you can see this, if I plot the face plot, that means, x versus \dot{x} , what I am looking for is starting from something I will go to 0 actually. That is not happening, this, this is something like a close form, close look function sort of thing, even this is supposed to be stable limit cycle. That means, if I start with anything, anything, I will merge into the limit cycle and then continue oscillating in the limit cycle actually. But ideally speaking, if I start somewhere, I will, I want to go to zero actually. That is not going to happen unless my control is good actually. So, that is the problem here, alright. So, this is what it is, they both are working.

(Refer Slide Time: 51:19)

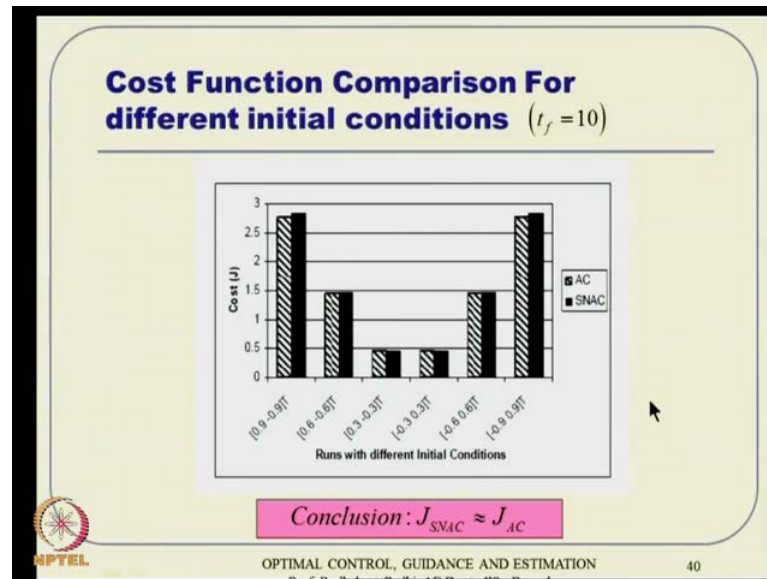


(Refer Slide Time: 51:35)



However, comparison, since performance comparison, since you can see, that the time taken is invariably half actually and picture really it is, is put in a bar (()) sort of thing, so telescopic training time, if you see that, then training time turns out to be half actually and total training time if you see, it is believe it or not, it is exactly 43 percent it turns out to be. So, the claim here, which is going to be slightly lesser than above 50 percent, are roughly about 50 percent of the time we will be done, actually.

(Refer Slide Time: 51:50)



Cost function comparison, again, both has to be similar, so that is not much of a concern, but to make adaptive critic work you really need to be lot more careful in your programming and all that compared to SNAC. So, that is another big advantage looking from simplicity of the structure point of view, actually.

(Refer Slide Time: 52:09)

Electrostatic Actuator

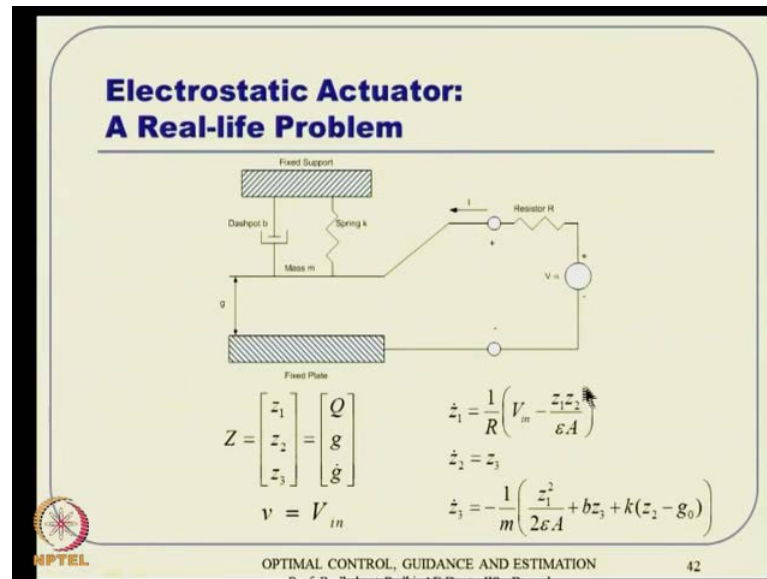
- System Model:
$$\dot{Q} - \frac{1}{R} \left(V_{in} - \frac{Qg}{\epsilon A} \right) = 0$$

$$m\ddot{g} + b\dot{g} + k(g - g_0) + \frac{Q^2}{2\epsilon A} = 0$$
- Cost Function:
$$J = \frac{1}{2} \int_0^{\infty} (X^T Q_w X + R_w u^2) dt$$
- Cost Function:
$$J = \sum_{k=1}^{N \rightarrow \infty} \frac{1}{2} (X_k^T Q_w X_k + R_w u_k^2) \Delta t$$

(Discrete)

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 41

(Refer Slide Time: 52:28)



So, finally, this is the electrostatic problem. This is the system dynamics is something like this, there is a mechanical part, which oscillates and there, there is a charged part, which is electrical, actually. So, what is the objective here is actually to maintain some sort of gap, g stands for gap actually, there are two as in, I think I have a picture somewhere, and g stands for this. So, this gate has to be maintained at a particular value of g naught actually. So, this is in control, input is control voltage actually, v in is your control input. So, this v input k has to manipulated in such a way, that g has to be maintained at a particular separation, g naught actually, the system and this is your spring mass dumper. This is a spring and this is your dumper, this is the mass, which oscillates actually. There is a fixed plate, and this, this gate is controlled by, by driving this, this voltage input actually and there is a resistor in here, which (()) current flow and things like that actually, alright. So, this is what it is, is the system dynamics.

And then, the cost function is something like this, it needs to be minimized. But what is x that used to be thought about? Actually, x is your state, but that state I will derive it, I mean, I will tell you now what this state actually. State has to be deviation state actually. So, that means, g , g is not the state, even though, even though the dynamics is given in terms of g , but what you are considering is the different g minus g naught happens to be the state, one, one state actually, that we will see that (()). Cost function is also in the discretized form, something like this. So, ultimately what you need is your state equation, we write it in a, in a state phase form.

So, we define this z_1, z_2, z_3 sort of thing. These are the states Q, g and \dot{g} and then we direct the control v as something like v input and write this equation. This state equation should be given in the form of this $\dot{z}_1, \dot{z}_2,$ and \dot{z}_3 function of $z_1, z_2, z_3,$ as well as, control input, actually here. So, this is what we want to see in any given control, control synthesis problem. We want to see it in the form of state phase equation, actually. So, we wrote that in a state phase equation.

(Refer Slide Time: 54:29)

Plant Parameters

PARAMETER	SYMBOL	VALUE	UNITS
Area	A	100	μm^2
Permittivity	ϵ	1	$\text{C}^2 / \text{N}\mu\text{m}^2$
Initial gap	g_0	1	μm
Mass	m	1	mg
Damping constant	b	0.5	mg / s
Spring constant	k	1	mg / s^2
Resistance	R	0.001	Ω

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 43

(Refer Slide Time: 54:51)

Electrostatic Actuator Dynamics

- Equilibrium point $Z_0 : z_2 = 0.5, \dot{Z} = 0$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{v}{R} - \frac{z_1}{2R\epsilon A} \\ z_3 \\ -\frac{z_1^2}{2m\epsilon A} - \frac{k}{m}(0.5 - g_0) - \frac{b}{m}z_3 \end{bmatrix}$$

- At equilibrium (operating) point

$$Z_0 = \begin{bmatrix} \sqrt{\epsilon A} \\ 0.5 \\ 0 \end{bmatrix}$$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 44

But now, remember there will be a necessity of finding out the equilibrium state actually. So, for that you need this problem has all these parameter values, given something like

this. And then, at equilibrium point what will happen? You consider this, remember z_2 is the g and g is the desired value given to us actually. So, that is what you put it as and then, make sure, that the equilibrium conditions \dot{z} is 0. That means, all the derivatives should be 0 actually. And v_2 is, I mean, this where z_2 is 0.5, that is, that is a desired value, that, that we have. So, we put 0.5 here, whatever is active, the three equations.

Now, in terms of three unknown variables and what are the unknown variables? z_1 , z_3 and v . So, $z_3 = 0$, so that is 0, z_2 is already there and z_1 can be solved from this equations, so that is already available actually. And once you have z_1 , then you put it back here and, and try to solve, no, no, sorry, the z_1 has to be solved from these equation and this turns out to be something like that, and once you have z_1 , you put that equation in the first equation and solve for v , actually. So, that will give you corresponding v_{naught} . So, this is the desired value of state at the operating point or equilibrium point and this is the associated control value at that particular point, actually.

(Refer Slide Time: 54:54)

**Electrostatic Actuator:
Error Dynamics**

Error Dynamics : $X \triangleq Z - Z_0$
 $u \triangleq v - v_0$

$$\dot{x}_1 = \frac{1}{R} \left(u - \frac{x_1}{2\epsilon A} - \frac{x_2}{\sqrt{\epsilon A}} - \frac{x_1 x_2}{\epsilon A} \right)$$

$$\dot{x}_2 = x_3$$

$$\dot{x}_3 = -\frac{1}{m} \left(\frac{x_1^2}{2\epsilon A} + \frac{x_1}{\sqrt{\epsilon A}} + kx_2 + bx_3 + \frac{1}{2} + \frac{k}{2} - \frac{g_0}{k} \right)$$

NPTEL OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 45

So, now, that the desired states and control, both are available. We can formulate the derivation dynamics and for that we defined x is the deviation state is z minus z_{naught} and u is the deviation control, which is v minus v_{naught} , actually. So, with the definition we can derive the system dynamics is something like this, actually. These are all happening in the form of deviation state and deviation control now, basically.

(Refer Slide Time: 56:14)

Necessary Conditions of Optimality

- State Equation:

$$\begin{bmatrix} x_{1,i+1} \\ x_{2,i+1} \\ x_{3,i+1} \end{bmatrix} = \begin{bmatrix} x_{1,i} \\ x_{2,i} \\ x_{3,i} \end{bmatrix} + \Delta t \begin{bmatrix} \frac{u_k}{R} - \frac{x_{1,i}}{2\epsilon A R} - \frac{x_{2,i}}{R\sqrt{\epsilon A}} - \frac{x_{1,i}x_{2,i}}{R\epsilon A} \\ \frac{x_{1,i}^2}{2\epsilon A m} - \frac{x_{1,i}}{m\sqrt{\epsilon A}} - \frac{kx_{2,i}}{m} - \frac{bx_{3,i}}{m} - \frac{1}{2m} - \frac{k}{2m} + \frac{g_0}{km} \end{bmatrix}$$
- Costate Equation:

$$\begin{bmatrix} \lambda_{1,i} \\ \lambda_{2,i} \\ \lambda_{3,i} \end{bmatrix} = \Delta t \begin{bmatrix} Q_{v11} x_{1,i} \\ Q_{v22} x_{2,i} \\ Q_{v33} x_{3,i} \end{bmatrix} + \begin{bmatrix} \left(1 - \frac{\Delta t}{2R\epsilon A} - \frac{\Delta t x_{2,i}}{R\epsilon A}\right) & 0 \\ \left(-\frac{\Delta t}{R\sqrt{\epsilon A}} - \frac{\Delta t x_{1,i}}{R\epsilon A}\right) & 1 \\ 0 & (\Delta t) \end{bmatrix} \begin{bmatrix} \left(\frac{-\Delta t x_{1,i}}{m\epsilon A} - \frac{\Delta t}{m\sqrt{\epsilon A}}\right) \\ \left(\frac{-\Delta t k}{m}\right) \\ \left(1 - \frac{\Delta t b}{m}\right) \end{bmatrix} \begin{bmatrix} \lambda_{1,i+1} \\ \lambda_{2,i+1} \\ \lambda_{3,i+1} \end{bmatrix}$$
- Optimal Control Equation: $u_k = -R_w^{-1} \frac{\lambda_{1,i+1}}{R}$

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 46

So, it is a standard regulatory problem now. So, and hence, you can select this cost function, quadratic and things like that actually. That way, actually, you know the cost function was given in the form of, I mean x actually, not in the form of z and this x is deviation state and this u is deviation control itself, alright. So, this is what it is and this is deviation state.

So, now, you are ready. So, the cost function is a state equation discretized and corresponding costate equation is derived, optimal control is also derived.

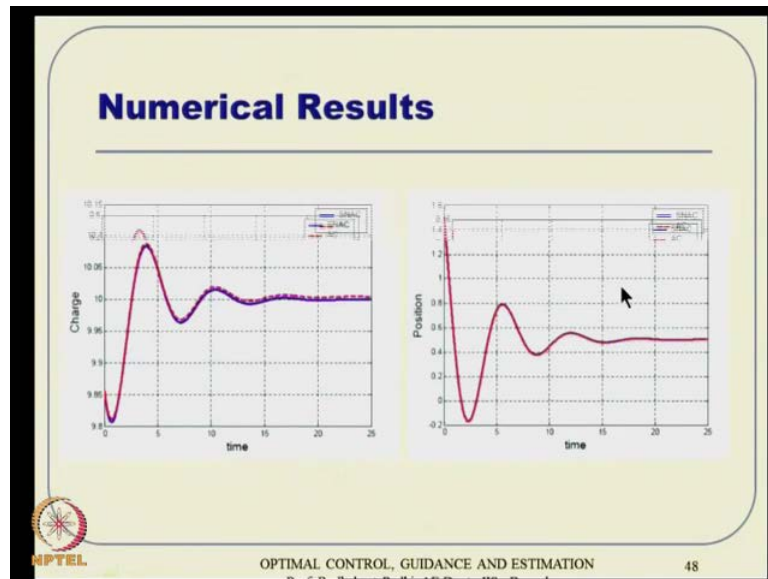
(Refer Slide Time: 56:44)

Implementation

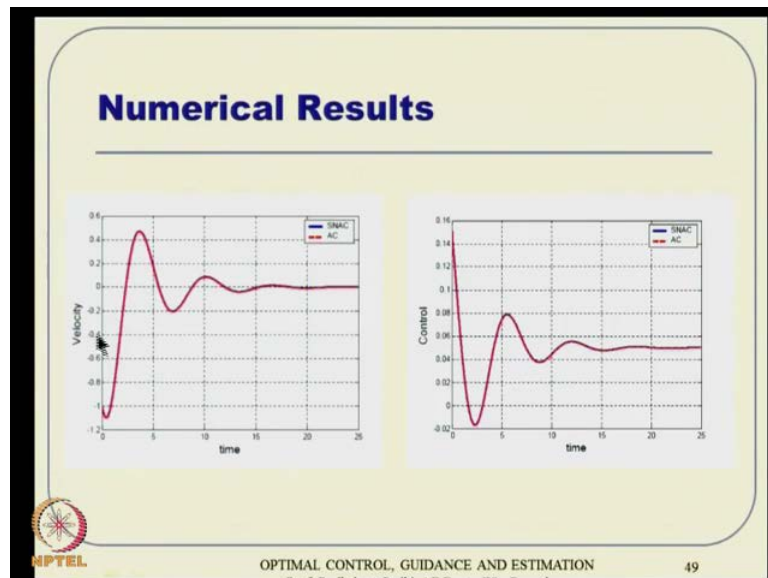
AC	SNAC
<ul style="list-style-type: none"> • Critic Network : 3-6-3 • Action Network:3-6-1 • Convergence Criteria: <p style="margin-left: 20px;">Critic: $\ \lambda_p^k - \lambda_p^a\ _2 / \ \lambda_p^k\ _2 < 0.05$</p> <p style="margin-left: 20px;">Action: $\ u_p^k - u_p^a\ _2 / \ u_p^k\ _2 < 0.05$</p> <p style="margin-left: 20px;">Cycle: $err_{e_n} - err_{e_{n-1}} < .01$</p> <p style="margin-left: 40px;">$err_{a_n} - err_{a_{n-1}} < .01$</p>	<ul style="list-style-type: none"> • Critic Network : 3-6-3 • Convergence Criteria: <p style="margin-left: 20px;">$\ \lambda_p^k - \lambda_p^a\ _2 / \ \lambda_p^k\ _2 < 0.05$</p> <p style="margin-left: 40px;">$p = 1, 2, \dots, n$</p>
<p>Weighing matrices: $Q_W = I, R_W = 1$</p>	

OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 47

(Refer Slide Time: 56:51)



(Refer Slide Time: 57:29)

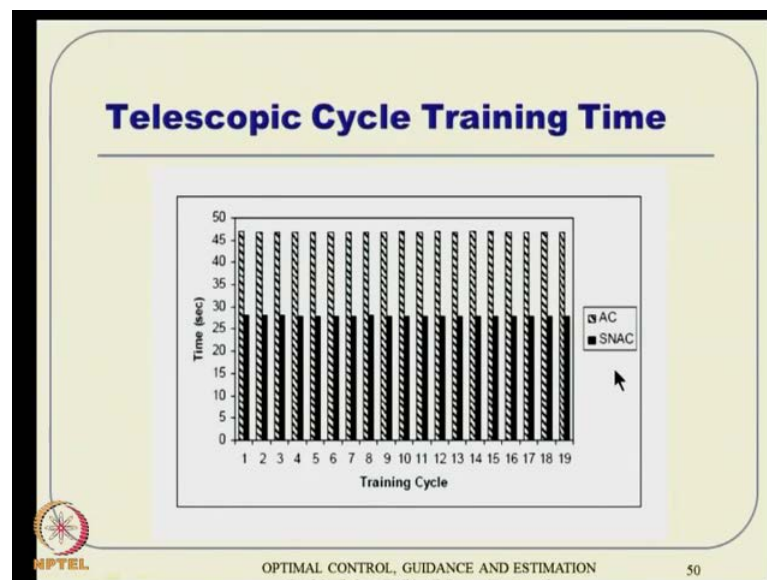


Then, again the two structures need to be compatible. So, the compatibility is maintained, actually. Your (()) are taken, 1, 1, and again the comparison results in (()) stimulation. Both happen to be similar and hence, I mean, both are kind of, both work actually. So, I do not claim, that one works better than the other. In fact, it, theoretically speaking if training is consistent AC is nothing, but SNAC and (()) actually, so both (())). Similarly, if you see these are the two states, I mean, this is the charge and Q and then the position it is, I mean v_2 , this position is Z_2 basically. So, what you are getting is R, Q and position z_2 and then z_3 and then control, actually. Remember this after you solve this you can, you can actually recompute your original state because the definitions are

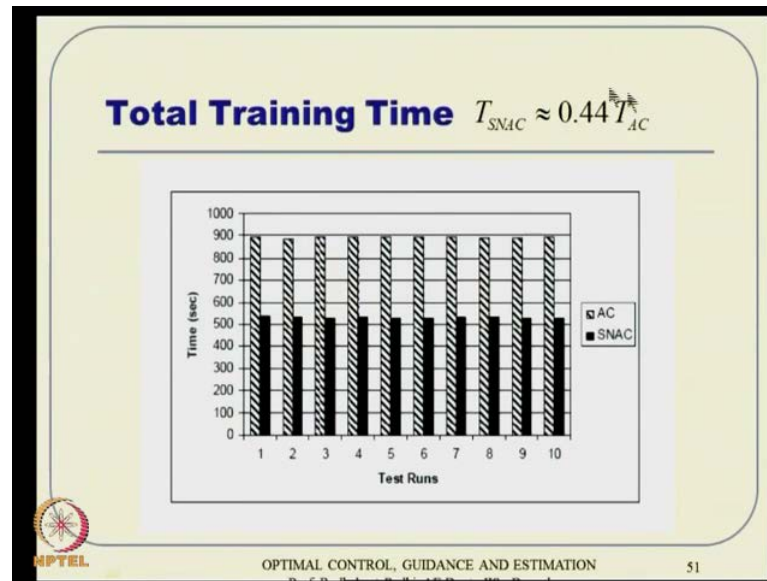
already here. So, once I get x and u , I can recompute my z and v . Actually, z is nothing, but z naught plus x and v is nothing, but u plus v , so u plus v now. So, I compute this v rather and the corresponding z will be available actually from there, but you can also think about doing other way round, just simply compute v , u equal to, I mean, this v equal to u plus v naught, then you have z naught initial condition.

So, you have original system dynamics. These original system dynamics can be simulated directly, actually. The system dynamics, once you have control and initial condition you can simulate the system dynamics and validate equation from the non-linear system point of view, actually. So, that is what is done here. So, this is what is done and it is all happening, it is working and it is working together.

(Refer Slide Time: 58:30)

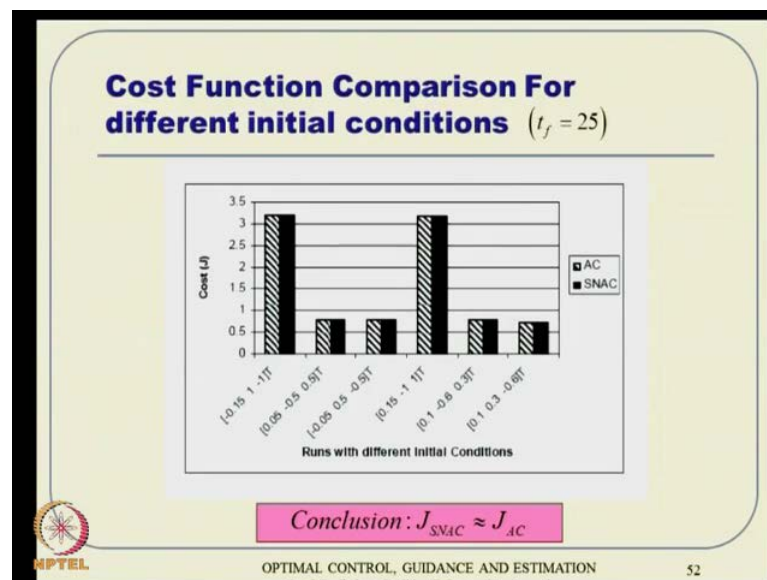


(Refer Slide Time: 58:33)



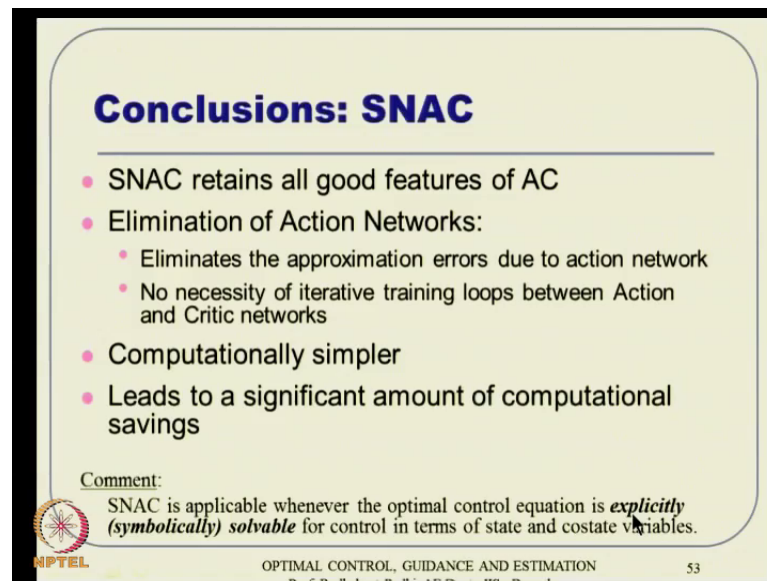
And computational thing point of view, it is happening in a, it is happening in a, it is very close to same 43 percent, now it is 44 percent, actually. So, little less than half of the time will be done and simplicity will be maintained, that is more important actually

(Refer Slide Time: 58:43)



A cost function evaluation, since both are again equivalent actually, (C) actually.

(Refer Slide Time: 58:48)



Conclusions: SNAC

- SNAC retains all good features of AC
- Elimination of Action Networks:
 - Eliminates the approximation errors due to action network
 - No necessity of iterative training loops between Action and Critic networks
- Computationally simpler
- Leads to a significant amount of computational savings

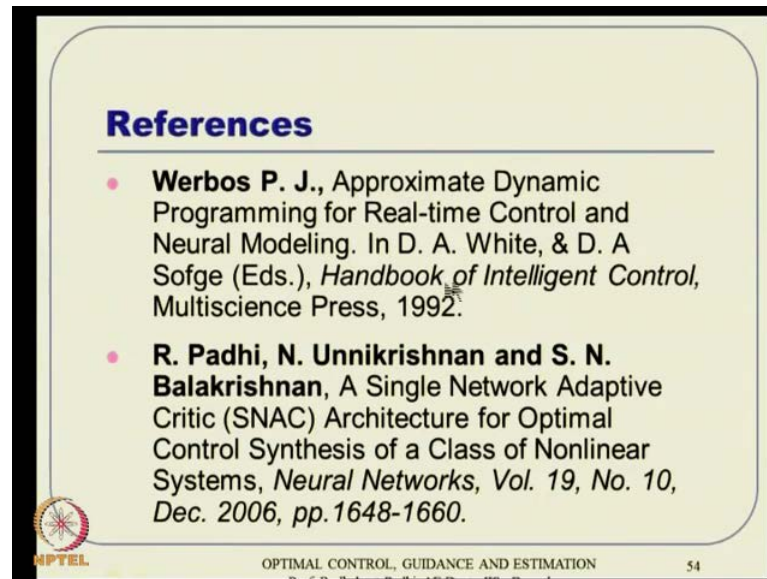
Comment:
SNAC is applicable whenever the optimal control equation is *explicitly (symbolically) solvable* for control in terms of state and costate variables.

NPTEL OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 53

So, conclusions about SNAC, SNAC retains all good features of adaptive critic, eliminates the action network and hence, it eliminates the approximation error due to action network and also there are no necessary iterative training loops between action and critic networks. Computationally, it is simpler and hence program, writing the program and all that also becomes simpler and it, it leads to a significant amount of computational savings actually, little more than half actually.


But all that happening with the assumption of the, that SNAC is applicable whenever the optimal control equation is explicitly or rather, symbolically solvable for control in terms of state and costate, that you should not forget actually. And if you have quadratic cost, cost function and control defined system dynamics, then this condition is always valid actually, you just remember that.

(Refer Slide Time: 59:38)



References

- **Werbos P. J.**, Approximate Dynamic Programming for Real-time Control and Neural Modeling. In D. A. White, & D. A. Sofge (Eds.), *Handbook of Intelligent Control*, Multiscience Press, 1992.
- **R. Padhi, N. Unnikrishnan and S. N. Balakrishnan**, A Single Network Adaptive Critic (SNAC) Architecture for Optimal Control Synthesis of a Class of Nonlinear Systems, *Neural Networks*, Vol. 19, No. 10, Dec. 2006, pp.1648-1660.

 OPTIMAL CONTROL, GUIDANCE AND ESTIMATION 54

The references, again, that these two references are the backbone of this particular lecture. Actually, one is chapter from that particular handbook, the other one is our own paper, actually. We see more on that, alright.

Thanks for attention, actually.