Optimal Control Guidance and Estimation

Prof. Radhakant Padhi

Department of Aerospace Engineering

Indian Institute of Science, Bangalore

Module No. # 11

Lecture No. # 23

Model Predictive Static Programming (MPSP)

Optimal Guidance of Aerospace Vehicles

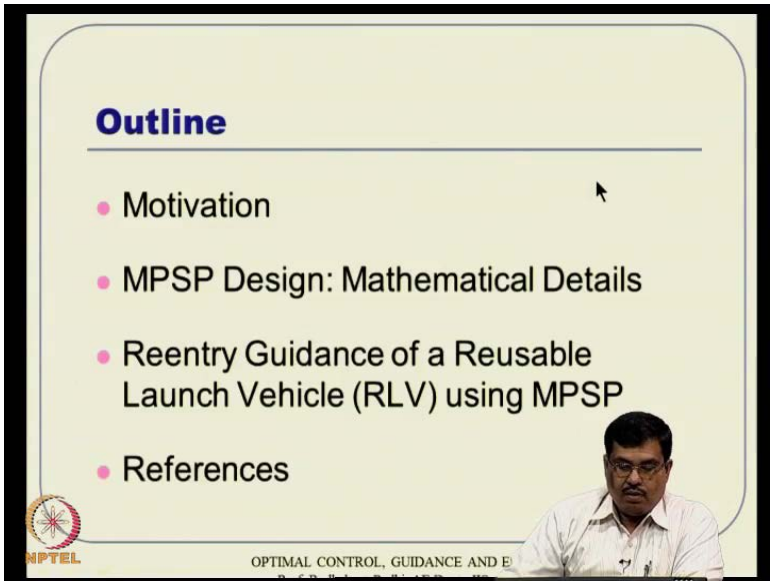
Hello everyone. We will continue with our lecture series on this optimal Control Guidance and estimation course. And so, far we have seen calculations and various approach which leads to this two point boundary problem followed by this L Q R which is a subset of that and then the little bit extension of that in the in the prime work and theta d. And then last couple of lectures we have also seen what is that this Dynamic Programming approach? Where it leads these HJB equations? However, the HJB equation is non-linear partial differential equation on thus. So, that lands up with this curse of dimensionality. The first approach leads up curse of complexity. This is a kind of approach leads to curse of dimensionality.

But anyway so, there are there are techniques, evolving techniques and all that which strives to avoid those problems in a limited sense at least. And one of that happens to be approximate Dynamic Programming which we have already seen there. And similarly we will, I mean slowly what your, what I am telling is, we are going towards these. These are called Real Time optimal Control. That means, the the solution techniques are so powerful that the solution is available very quickly. Actually, that means, you can think of using these algorithms really online.

So, this one of these techniques happens to be this Model Predictive static Programming which I am going to talk in this lecture as well as next lecture. And we will see some application of this particular technique in the frame work of optimal Guidance of aerospace vehicles. So, essentially if you know, if any Guidance problem is a is a truss optimization problem. That means, it can be I mean ideally it should be formulated as a as an optimal Control problem, but because of this curse of dimensionality or curse of

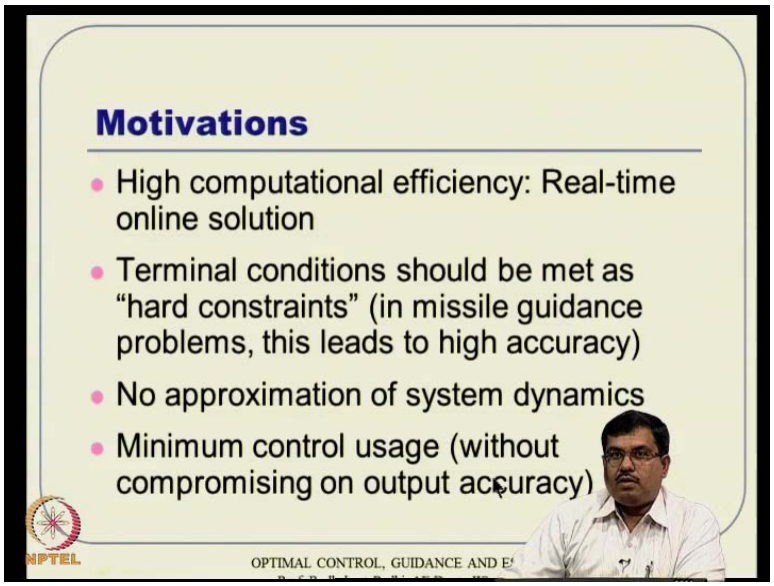
complexity and these are these are typically never used online. However, as I told there are several methods revolving which strives to kind of overcome that and one of that happens to be this Model Predictive static program.

(Refer Slide Time: 02:21)



Now, let us let us see what it is. So, outline of this lecture will be something like this. First is Motivation, why we do that and then some Mathematical details of this this particular Design and then I will take you through a little bit intensive way. This Reentry Guidance of a reusable launch vehicle using the MPSP technique and ultimately I will give you some references to follow it off as well. Alright. So, let us get going. The first is Motivation.

(Refer Slide Time: 02:49)



And let us see what motivations are there for this particular technique. The very first thing that comes to mind is high computational efficiency. That means, what you are looking for is some sought of real time online solution. Actually the second thing what you are looking here is, Terminal conditions should be met as hard Constraints and especially in missile guidance problems. This essentially leads to High Terminal accuracy. That means, if you are if you are working on a missile guidance problem. The very first thing that you want is, to go towards the target as close as possible and in that sense hard constraints are typically much better.

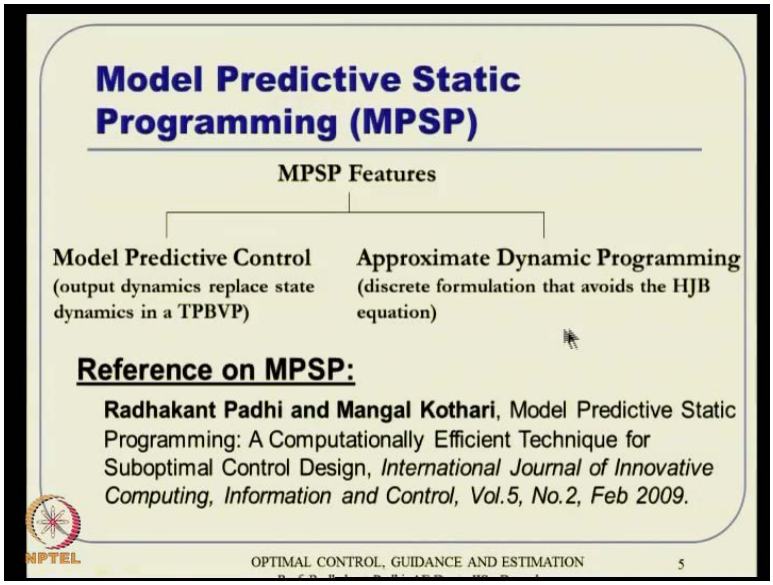
I will take you through some examples in in next class and all that actually. So, this is the next thing and especially it is not necessarily only missile Guidance. I mean in any aerospace Guidance or ground vehicle Guidance, underwater Guidance, anything that you think about. The ultimate aim is when t goes to t F or final time goes to current Time goes to final time you want certain objectives to be met actually. In other words your position vectors should be very close to the target position Vector and your velocity angles especially should be in some particular desirable values and all actually those thing can also we met actually ok.

So, it can be essentially use for any aerospace vehicle and especially very particular it can be for missile guidance as well. So, the next one that you are looking for is no approximation of system dynamics. That means, as we saw that in the adopt equity

techniques and all, we do not want to kind of linearise the system dynamics or and think like that and then talk about lot of analysis which essentially leads to some sought of closed form or semi closed form solution. However, the very very the fact is, we have actually started with linear the system dynamics which you do not want to do ok.

The next one is on the way, minimum control usage is also our our objective. Of course, that should not be compromising the output accuracy at the at the final time. So, without compromising on the output accuracy at the final time what you really want is, a is a path from a given missile condition which will take actually minimum control and lead to extremely high accuracy. And we really want a very highly computational efficient method which will essentially lead to real time online solution actually. So, that is the type of Motivation that we are looking for here.

(Refer Slide Time: 05:13)

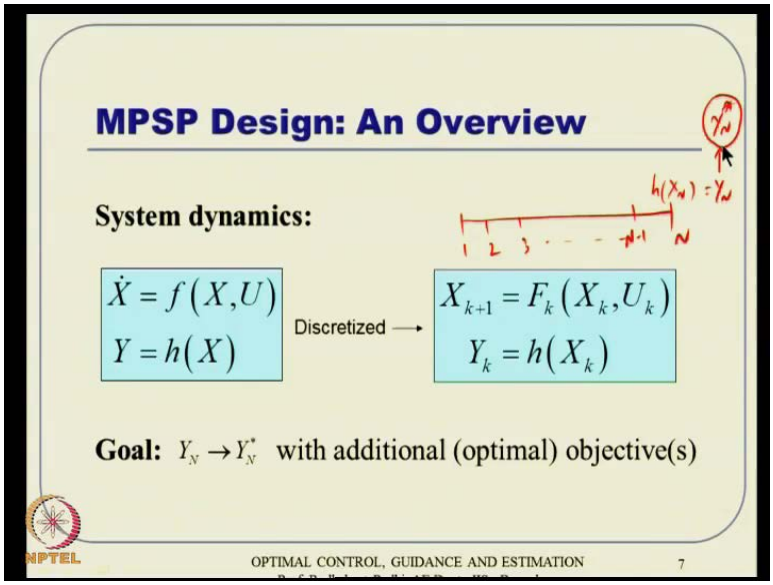


Alright. So, this this Model Predictive static Programming; obviously, you can see that a partly they very name suggests model predictive it the philosophy is partly from Model Predictive Control. So, this Model Predictive Control is something like output dynamics replaces the state dynamics in a in a 2 point boundary problem. And that is what the features some of the features of this MPSP technique. And the other side is actually realized on that part. What about the other side of the study? It is also inspired from the philosophy of approximate Dynamic Programming and both of the techniques we have seen so, far. So, I hope you are able to connect what I am talking here actually.

So, sincerely what it what is the approximate Dynamic Programming? Essentially, a discrete formulation that avoids the HJB equation in a way. So, here we also talk about a discrete time frame work in this in this particular design where the the particular steps numbers and think that are are inspired by Model Predictive Control slide actually. And if you want to refer, I mean the details part of it and all you can actually see this reference is the first publication that we published in about in 2009, in a journal. actually This we can see that in I will also list out few more journal papers and conference papers and think like that in the end of this lecture as well as the next lecture actually.

Alright. So, let us get down with the mathematical details part of it now. Then we will slowly go through the steps and try to understand what is what is really happening here. actually

(Refer Slide Time: 06:51)



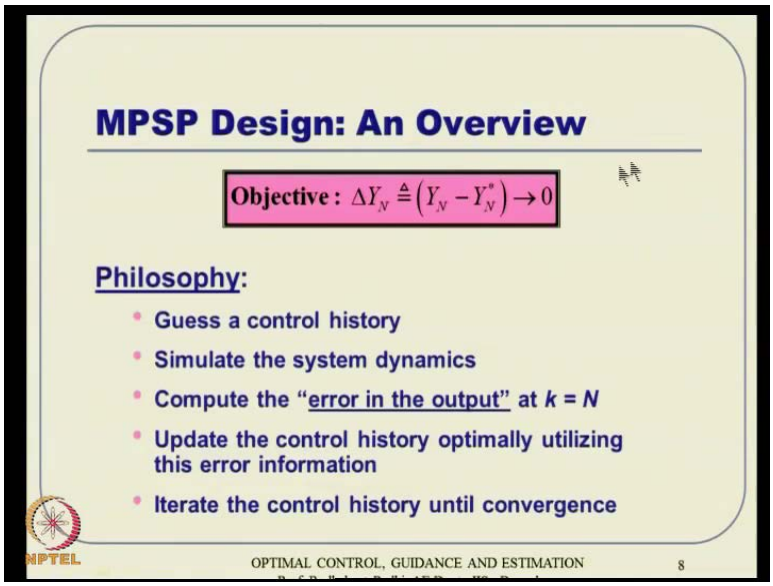
Now, the very first thing is you as I told it is also inspired from discrete sonic dynamics and all that. So, for you first thing to do here is we have a system dynamics in this form. Remember this complete non-linear form and then there is a output which is also a non-linear Function of X. actually sorry ok.

So, this is this is actually a non-linear system dynamics and also kind of a non-linear output and also remember that this this output that I am talking here is actually the desirable output it is an it is not really the sensor output. actually ok.

So, both the things are discretized. you Typically it is done through this the system dynamics. Discretized is typically done through this Euler integration actually. So, what is the objective? Objective is as we go along; that means, k varies from actually this if you interpret that in the form of time domain. This is one. k k varies from 1 2 3 4 like that and ultimately we have this this N. And just before that is N minus 1 and remember Control typically ends up to N minus 1 where state evolves of two time step actually. So, what you are looking here is the at time step N. We have this. I mean the subjective to be met is this Y N which is nothing, but h of X N should go towards a particular desirable value Y N star. And of course, as it as I mentioned before with additional some optimal objectives and all way will assume on the way actually. So, this is the particular problem when we you I mean when the state starts evolving at the final point N. We have a X N alright the final point N. We have this X N and this X N, if we evaluate h of X N then, you will get Y N and this Y N should actually go to certain Y L star actually. That is that is our objective ok.

alright . So, what is the philosophy here? Obviously we start with some sought of a guess history. I mean that is typically with respect to I mean, this typically were typical way of solving optimal control problems. We start with a Control guess history and we have initial condition. So, we start propagating the system dynamics and ultimately when we reach there, we we arrive at some sought of Y N value. But obviously, that is not close to Y N star because because the control that we have applied is actually a guess history Control. So, there is an error there and. So, the the objective here is how do we adjust these guess value values guess value of the control history? So, that after the update the which actually lead to that particular X N which is if of, I am using, which if you evaluate Y N that will be closer to Y N star actually.

(Refer Slide Time: 10:08)



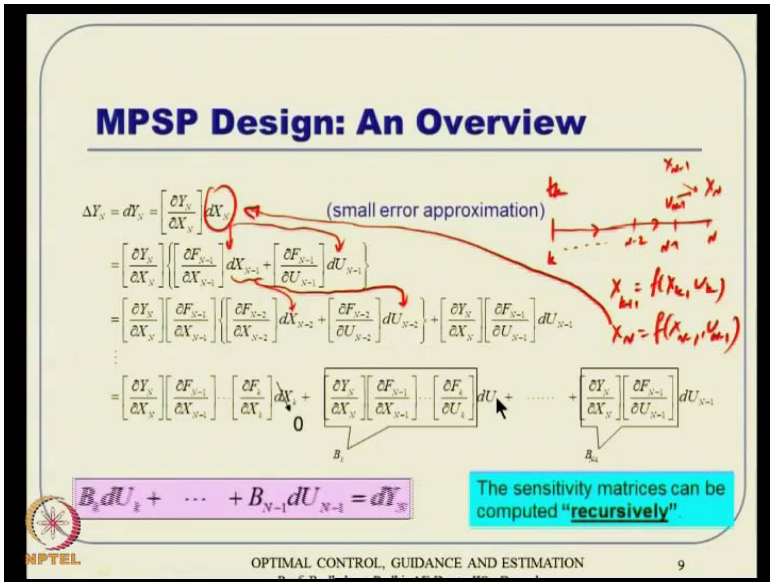
So, that is the objective here. So, so Mathematically speaking we can define some sought of error delta Y N. This delta Y N is is nothing, but Y N minus Y N star and that is should go to zero. Remember Y N should go to Y N star so, obviously, the difference would go to zero ok.

So, what is the philosophy? First you have to guess a control history and then simulate the system dynamics from the initial condition that you are interested in. Then we to compute the error in the output at the final time and you have to utilize this error information to update the control history optimally. Then obviously, we have to repeat this literalism until convergence and it have to equate it somebody can see that the this steps if you see it little carefully, this is very close to what we have seen in suiting method actually in a way. Then obviously, were somebody can ask what is the beauty here? The whole beauty here is the update happens iterate the control history until convergence. What you what you see in the last step? This iteration happens very very fast actually. So, why it happens? And all that. I will I will tell you as we go along actually.

Alright. So, let us start analyzing this is see as I told we we started with with a guess history of the control. But that leads to this this error in error in output of the, I mean the output at final time. So, now, we want to analyze this error and tell how much the error we have done in in the control everywhere actually. We more we have a we have a

control here in this time history. If you are aware, you have a control U 1 here, you have U2 here, you have U3 here and and think like that. actually All these have to UN minus 1. You the think this this values are Arrhenius. So, you want to update it so; obviously, you want to update it you want you to everything. So, that the updated control history here will take us to the desirable value ok.

(Refer Slide Time: 12:17)



Alright. So, you want to analyze first how much error we made and what is the effect of that error in this delta Y N. actually How much error we made in the control everywhere all the time stage and how much effect it had in this particular delta Y N actually. So, here we introduce this this small error approximation. I mean, obviously, the demands that you start with a reasonably good way basically it may not be very good, but ultimately it turns out that it is not that sensitive actually. We can start with larger result, but theoretically speaking it it demands that the delta Y N is approximated like a d Y N actually. So, that is a small Y small error approximation and using this small error approximation what you do here is the delta Y N by delta X N into d X N actually. That means, this d Y N what you see we analyze that where does it come from actually. Because ultimately remember Y is a Function of x. So, if there is an error in Y it has to come through the error in X. It cannot escape from that actually. So, we assume that that where it is and also assume that we are not starting from really, I mean this this is actually can be applied from any time.

So, what we are to telling here is, well let me, we will start with not really one every time. Suppose we are we are operating a time point I mean, time step three, it then becomes our initial time. That means, we start operating from any any k actually really. So, that what we are doing here? So, we want to analyze backwards up to time step k basically. But anyway. So, coming back this this was an error delta Y n. So, this is this smaller approximation d Y N and this calculus simply tells us that this d Y N has to happen because of d X N and this is the relationship actually. Now the question here is what is this d X N? Here this this time step N and then this is N minus 1 actually. There was a problem in X n; however, this X N if you if you see the system dynamics. System dynamics tells us that X k plus 1 is a function of F k X k U k basically ok.

So; obviously, the if there is a there is an error in X N then then there is some error in X N minus 1 and U N minus 1 actually. So that means, X N minus 1 and N minus 1 X N minus 1 and U N minus 1 both have an effect towards X N actually. So, that is why any amount of error that I make in X N, there is certainly some error in an in X N minus 1 and U N minus 1 both actually. So, that is why I want to expand this d X N in the in the, I mean in the variables d X N minus 1 and d 1 minus 1 actually. And remember this is the will instead of going back behind what we have here is X at k plus 1 is actually F of k X U k. So, obviously, X N X N happens to be a Function of X N minus 1 and U N minus 1 actually. So, if you see these then, obviously, d X N if you if you see that from this explanation you can you can relate to this this d X N. This particular thing you can find out and then you can tell this this should be related to something like d X N and then there is something like d U N actually. So, this is what what you want to do. So, you we do not want to touch that that we keep as it is, but d X N, I want to expand it backwards and then I write want to write it is something like this d F N minus 1 by d sorry del F N minus 1 Y del X N minus 1 into d X N minus 1 plus del F N minus 1 Y del U N minus 1 into d U N minus 1 ok.

Now, here what you see here is the this control variables are all decision variables. That means, if you made some error there then we there is a direct chance of correcting that actually. So, it does not come from anywhere because because we took care on decision that is why this d U N minus 1 is coming actually. However, what about d X N minus 1 again going back to that any time the state evolves because of the previous state and the previous control actually. So, again you see that d X N minus 1, I can expand it in the

form of d X N minus 2 and d U N minus 2 actually. This particular thing what you see, d X minus 1 is expanded N the form of d I X N minus 2 and d UN minus 2 actually.

So, d d U N minus 1 you keep as it is. That we do not want to change actually. This this 1 is kept as it is actually. Now, what happens is you continue this exercise until this until the final, I mean until the initial time and as I told here in this problem, we we interpret the initial time as up to k actually. So, I start with N then N minus 1 then N minus 2 and I keep on continuing this way and here I will end of with this time step t k. This represents t k. That is why my problem starts actually ok.

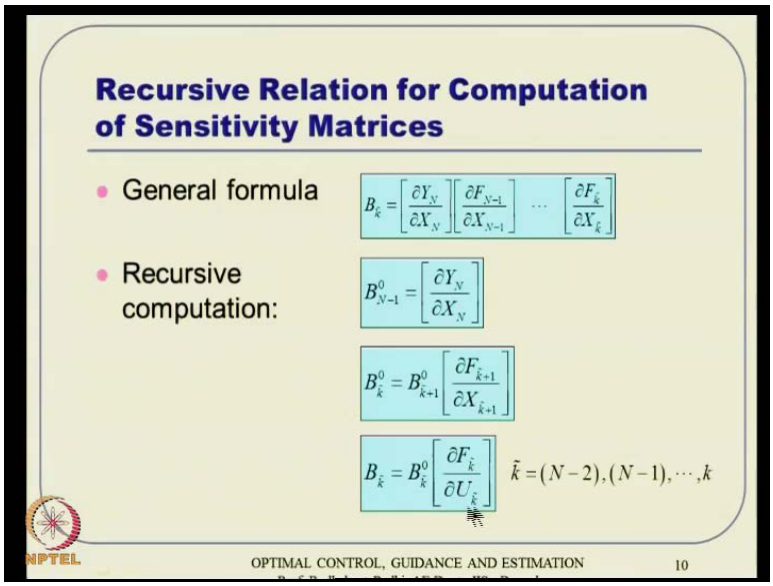
So, I will continue up to this time step k basically. So, I will do that and then I I can see that that this the entire expression will lead to something like this. actually These first there will be some sought of a error in the initial condition d X k, that is my current tie in condition of the state. So, that is my error in initial Condition and then there is a bunch of expressions which will tell us that there is a error in d U k, the next term will tell us there is an error in d U k plus 1 and things like that. actually This will continue up to this d U N minus 1 actually. And some of you can can now see that there is some league of kind of inspiration from approximate Dynamic Programming. That is what I mean this sensitivity analysis part what we are talking here is this kind of a root from that side actually.

Alright what next actually? We have done that, but here we observe that the there cannot be any initial condition error because we we assume that the initial condition is known to us actually. That is from, that is where we want the solution anyway. So, this goes to zero. So, ultimately what it leads to is something like this actually. So, this d Y N what you is what is evaluated delta Y N is can be can expressed, I mean can be expressed as d k times d U k plus B k plus 1 times d U k plus 1, all the we have up to d N minus 1 times d U N minus 1. Now these matrices what you see here and refined as V k V k plus 1 and all that here. the Somebody can argue that what is happening here it is actually we if you see the first thing then it is actually multiplication of two matrices. The very previous one will be a multiplication of three matrices and it keeps filing on actually. That means your number of matrices computation of for computing the sensitivity matrices is essentially large actually in a way. So, how can that computation and efficiency be retained? Now if you if you look at this lightly closely it it turns out that this matrices actually can be computed recursively.

So, this Recursive Computation actually is one of the key features. Why it is computational efficiency actually? I will tell you there are other reasons as well actually. And also notice that the the dimension of this equation here what you see here is dimension of Y N. It is it is nothing to do with dimension of state actually really and typically this dimension of Y happens to be much lesser than the dimension of the state. That means, the ultimately the the number of equations that we are talking here is dictated by how many outputs you want to control at the final time really. So, that actually helps and that gives us another reason why it is computational efficient actually. And also also notice a big thing here that, this particular relationship what you are getting here is actually an algebraic constraint. There is no differential equation involved and all that here. That is probably the biggest reason why this is computational you know efficient actually. That means, somehow there is a hope that we can actually formulate a static optimization problem actually. That is why this name Model Predictive static Programming is given. By the way if we as I told before programming stands for optimization actually. So, Model Predictive static Optimization. Think about that ok.

Any way, coming back this is the constraint equation that we got. What it tells is we made these much errors this d U k d U k plus 1 and all that. So, this much errors are we made and hence we will ended off with this d Y N because these are the sensitivity matrices associated with them. d k is a Sensitivity of d U k with respect to d Y N like that actually alright. So, this is what it is and we will see how these matrices can be computed recursively. This is the relationship that you can see above if you talk about any B k (( )) this (( )) is actually dynamic variable like this. Then any V k (( )) can be expressed like this. So, if you observe little closely first first we can probably define this B N minus 1 0 something like the coefficient that multiplies everywhere and then we we operate on this kind of hydration actually B k o is B k I mean B k tilde is B tilde plus 1. That it essentially is backward recursive actually. So, this one multiply by that actually.

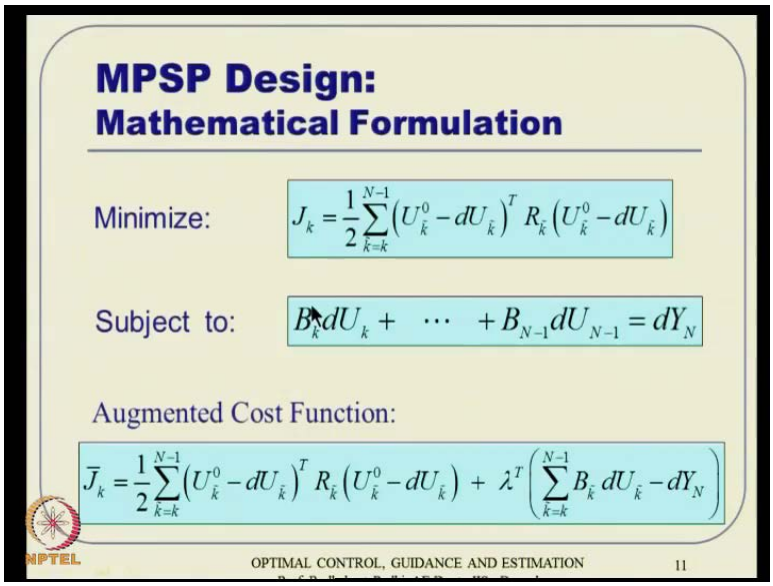
(Refer Slide Time: 22:46)



Whatever addition term you are getting here will multiply with that. Ultimately after computing all these B k 0 all that we do is, we multiply all this B k tilde B k tilde 0 into del F k by del U k. That means, what you see del F k by del U k you take it, out separate it out and the in between terms that remains we just start kind of getting recursively, ultimately we just multiply one shot actually. So, that is the way of recursive computation of the sensitivity matrices actually. Now, what about this now? What you what you see here this constraint equation is a small dimensional algebraic constraint equation that is clear here, but also remember how many free variables were have. This number of free variables depends on the dimension of the control variable multiplied by the number of time steps available ok.

So; that means, there are so many freedoms out here unless you assume that you are at the very final time and things like that. Forget that. We can assume that there are enough time steps here available so, that means, we have lot of freedom and we what you want is actually a less number of constraints. That means, it is severely under constraint equation actually. When you have something like that an under constraint equation there is a scope of optimizing actually that optimize that essentially lead us to this the static optimization ideas actually.

(Refer Slide Time: 24:11)



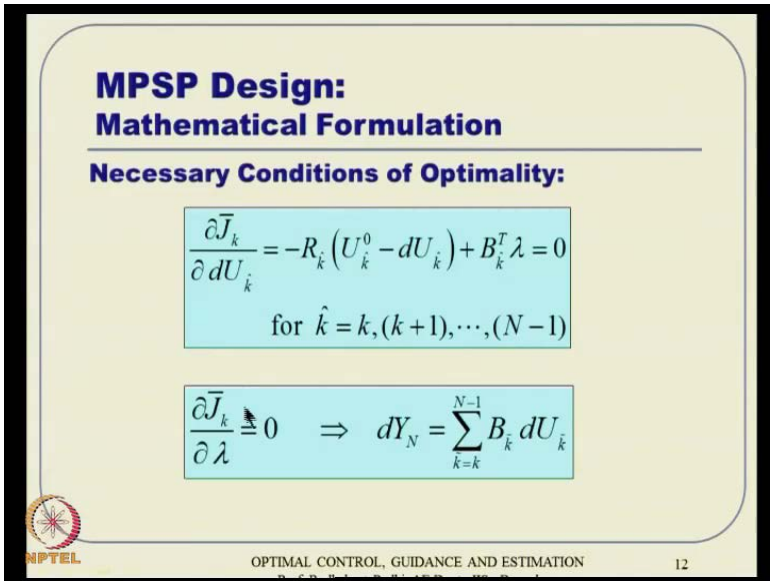
So, what you do here is not only you want to satisfy this constraint, but we also want to I mean think about minimizing the control history this way. So, why this this term here? This is actually the the updated control. So, the control history will be updated that way here. We have this previous value or the guess value of the control to begin with or the previous value and this is what the correction you want. So, this is what the updated control value will be and this one transpose time are into d U k and are into U. This essentially what you are telling is U transposed are U sort of thing ok.

So, essentially you are telling that I i want to have some sought of a control minimizing solution actually and also remember there are enough freedoms out here. I mean little bit more freedom out here, but in some particular some particular problem it may so happen that you will want a solution which will actually Minimize d U k not U not U k in general actually. Why? Because if you really, I will see that next example and all if you if you are control history that you want has to be bounded between some maximum, minimum value you will typically like to operate in the middle segment actually. That is severely non I mean typically non zero. Basically in that those situations you would like to have this d U k transpose or the U k instead of U u k transpose are U k sort of thing.

But any way just to maintain generality, we will assume that it is a is a control minimizing solution. So, we will take this cost function subject to this constraint whatever constraints we have taken have derived already basically. Now we can clearly

see that is actually nothing, but a static optimization problem actually. So, we can follow the various standard formulas I mean, standard rules logics whatever you have studied before for static optimization and then the procedure tells that we have to we have to first construct some sought of a Augmented cost Function and this Augmented cost Function is something like this we have the original cost function plus lambda transpose this entire equation minus d Y N sort of thing. Once you have that this is actually a problem for I mean were J k bar the 3 variables here is d U d U and lambda as well actually. So, we have to take derivatives of, with respect to both actually.

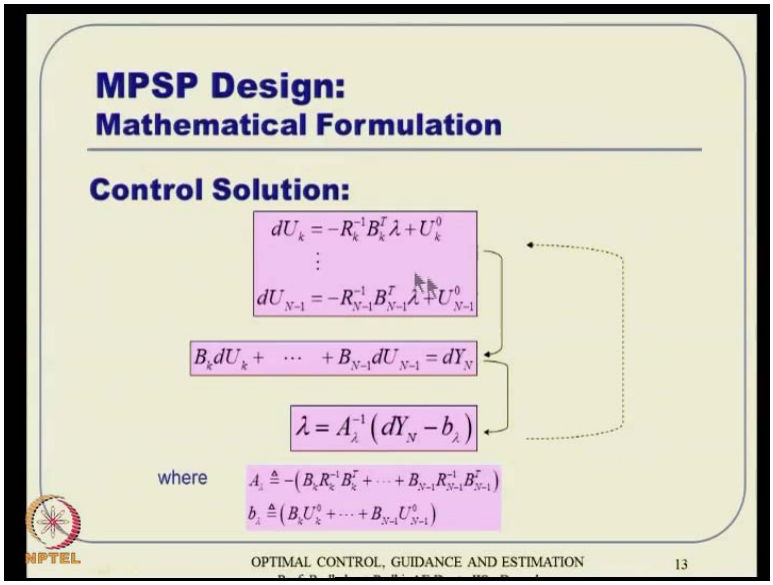
(Refer Slide Time: 26:29)



So, we take derivative with respect to d U k till the first. So, that essentially leads to something like this and then we have this second equation del J k bar by del lambda 0 which talks about something like this. Now these two equations needs to be solved together to get your control history get our Control history update actually. Now, what you observe here again, this because this is static equation that we we started with the cost rate variable that you require here is also static. That means, just a constant Vector that we need actually and that is probably the biggest reason why it is computationally efficient. In other words utilizing a single lambda single lam single Vector lambda will be able to update the entire control history of which is a major departure from this 2.1 boundary value problem solutions actually.

Any way coming back these are the two equations that you want to solve and really essentially eliminate lambda. So, how do you do that? First you solve this in terms of lambda then, substitute solve for lambda and go back actually.

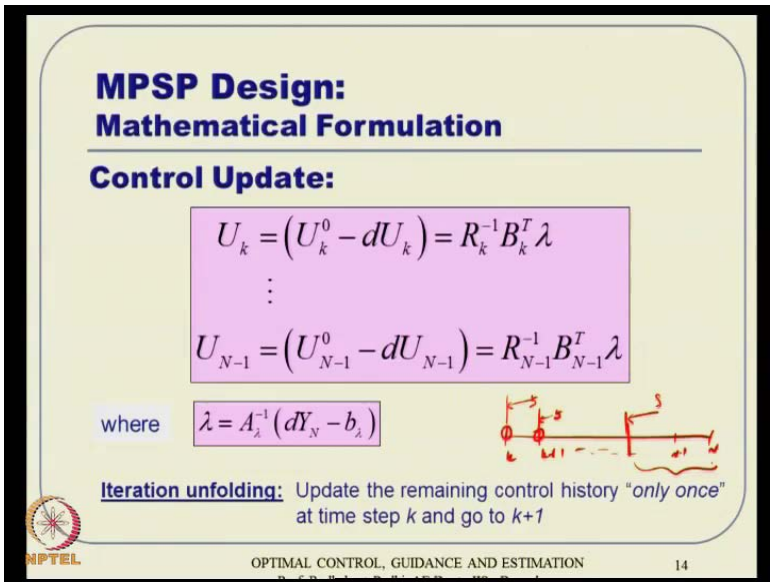
(Refer Slide Time: 27:31)



That is what it is done here. You first you solve this various in terms of lambda, then you go back to the constraint equation, put the value expressions there and solve for lambda. Once you solve for lambda you can again go back and put the lambda values here and you can get get the solution that you require actually. So, essentially it will tell us that lambda can be solved something like this where A lambda is defined like this and V lambda is defined like that. Once lambda is known now d U k is also known actually ok.

Again I emphasize or I repeat that with with a single coasted variable, you are able to compute all the necessary corrections for for control variables at all the grid points actually aright. So, this our also closely you can, if you notice this if you if you see the this A lambda and b lambda are typically functions of only this I mean this sensitivity matrices largely and R k inverse and all R k critically selected as part of the cost function and this U k 0 is is your previous solution. That means, previous either guess history or or the previous hydration value actually. So, knowing the previous values and knowing the sensitivity matrices, we can very quickly compute lambda. Remember sensitivity matrices can be computed recursively. So, using that you can compute lambda and once you compute lambda, your control history updates are all known actually.

(Refer Slide Time: 29:01)

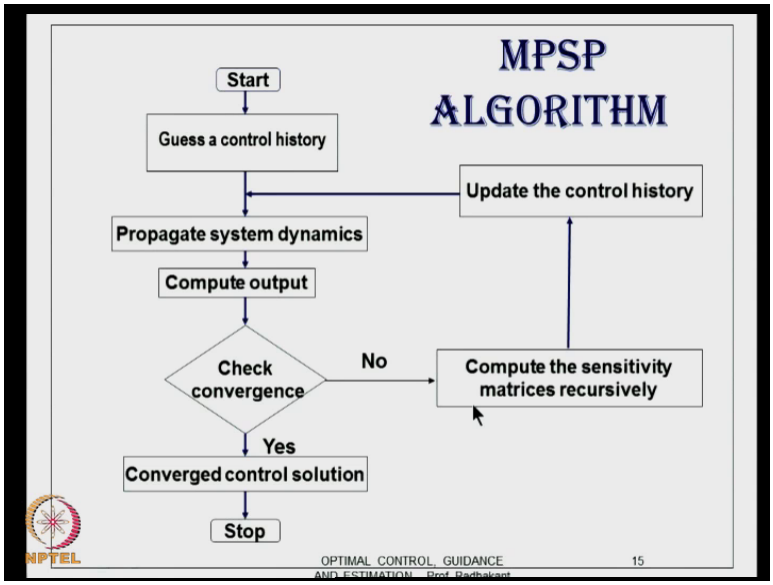


So, this is why it is why is computational efficient actually. So, finally, the control update can be written something like this and I mean A lambda is given like that actually. Also I said, this point of time somebody can argue that I mean in our own observation most of the times this this entire method converse is very repeatedly also. We just need about 3 I mean about 4 5 iteration rate at the maximum for getting some good solutions actually. However, you can also think that even if that is not allowable, I mean if suppose my time sequence is very large and then even one at takes very long time and think like I am see or how many iteration it will take and think like that. Then it will, I mean you can think about doing I mean incorporating this. So, this idea of hydration of folding, that means, it is simple. it All that it talks is, tells is, I start with k I have k plus 1 and I i will have to go up to N minus 1 any way.

So, what I will do is I will operate a based on the fixed number of iterations. So, I will suppose I operate based on something like I will not wait on until it converse, but I will operate a based on the five iterations only and and my computation will show that I can actually do the five iteration in the available time actually. So, every time I will I will either operate on the fixed number of iteration or I can send that this way I can operate five iteration here, five iteration there and think like that or as you remember, as you note that the when you go closer and closer your time time available become shorter and shorter. That means, number of times times greed's actually. So, number of time greed's becomes smaller and smaller here. So, then you can think, wait I can actually do more

number of iteration necessary I can probably do eight iteration, but these number five, eight and all that will be priory fixed depending on a problem actually and depending on the computational power and and real time algorithm that we use and think like it actually. So, nobody forgets from doing that, but how about my observation my comment here is, in in many of the problem that we have solved it it actually takes about four five iterations only to converge any way.

(Refer Slide Time: 31:12)

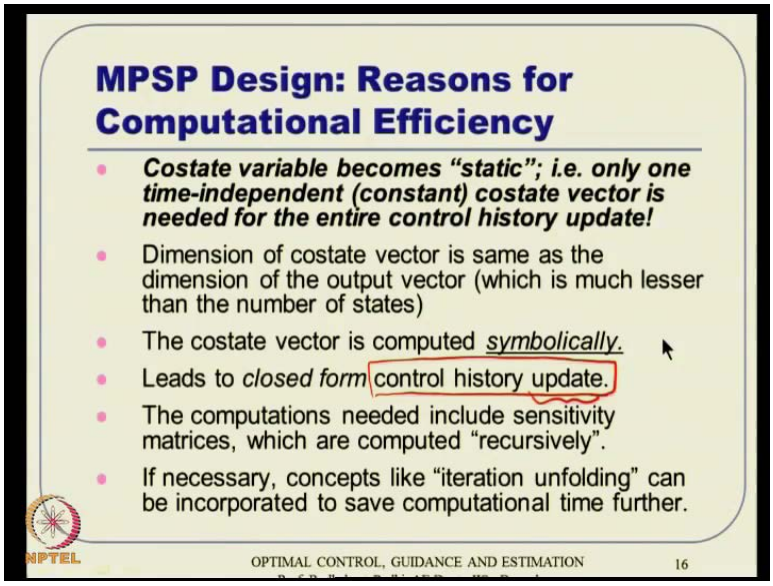


So, the entire algorithm can be can be represented like this. This is very close to a kind of soothing method if you if you think about. We start with some sought of a guess control history, propagate the System dynamics, compute the output check for convergence and lastly when I talk about convergence, I talk about output convergence whether Y N is gone to Y N star actually and also remember when Y N goes to Y N star or at the same time it turns out that they also goes to some sort of I mean the cost function also tries to kind of goes towards convergence actually anyway. So, this check for convergence if it is it is converse the take the converged solution and stop actually and if it does not then, you just go there and then and then update the, compute the sensitivity matrices recursively and then update the control history and go. Now, also remember a small comment out here that is for control to adjust it all depends on this d Y N actually ok.

Because these algorithm if you see, the convergence check that you are doing here is d Y N whether it has gone to zero or not. Now now remember these are non-linear problems

that you may have multiple solutions and think like that. That means, if your guess happens to be good even though the need not be optimal from cost function point of view you know actually get eluded it that Y N has gone to Y N star hence my solution is optimal actually. So, my suggestion for that is do not get eluded that way. if you a you allow for at least one iteration forcefully. That means, do not bother about checking convergence from the very beginning. Just allow one iteration from the second iteration onwards you can probably check actually. So, that is why it is actually alright.

(Refer Slide Time: 32:50)



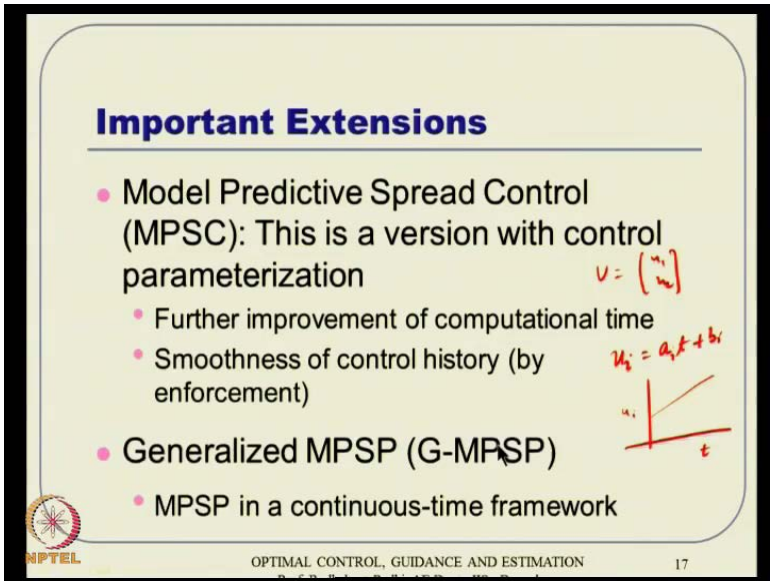
So, I have all told everything on the way, but let me summarize reasons for computational efficiency. Why it is computational efficient? The very first thing is as I told the Costate variable becomes static. That means, only one time independent. That means, constant Costate Vector is needed for the entire control history update. Basically that is the primary the the major reason why it is computational quite efficient and then dimension of the Costate Vector is same as the dimension of the output vector which is typically much lesser than the number of states actually ok.

Essentially, this is lesser then the dimension of this the lambda we also becomes lesser actually. It is directly I mean it is same as the output vector dimension actually and the Costate Vector is is computed symbolically. Costate vector well, it is theoretically speaking it is Costate. Costate vector with reiteration and is computed symbolically and then it leads to some sought of a close from control history update actually it it is does

not lead to closed from control per say closed from control history. But it leads to this closed from control history update were able to kind of compute very fast actually.

Alright, the computation involves largely the sensitivity matrices which are also computed recursively. That is another reason why it is computationally efficient and if necessary as I told concepts like iteration unfolding can can be incorporated to serve computational time even further actually. So, these are the even last issues is a primarily and mechanization issue, but all other things the first five issues, five points are actually the algorithmic centric actually. This happens because of the algorithm actually.

(Refer Slide Time: 34:43)



And there are important extensions also we have proposed and I will probably take you through some of these and then the extensions is first thing is is the variation of this which talks about something called Model Predictive spread Control. This is a kind of a version with control parameterization. That means, you you parameterize the Control variable ok.

Let me talk about that little bit philosophically. Suppose U contains something like U 1 and U two. So, what you do is, U Y U let us say you parameterize like a constant vector or like a a i t i or t goes sought of think initial guidance will do that. So, a i t plus something like Bi; that means, with respective to time what I am looking for is something like a sorry for that what what I am looking for is with respect to time. I am looking for something like a straight line equation actually or you can think of d I 0 then,

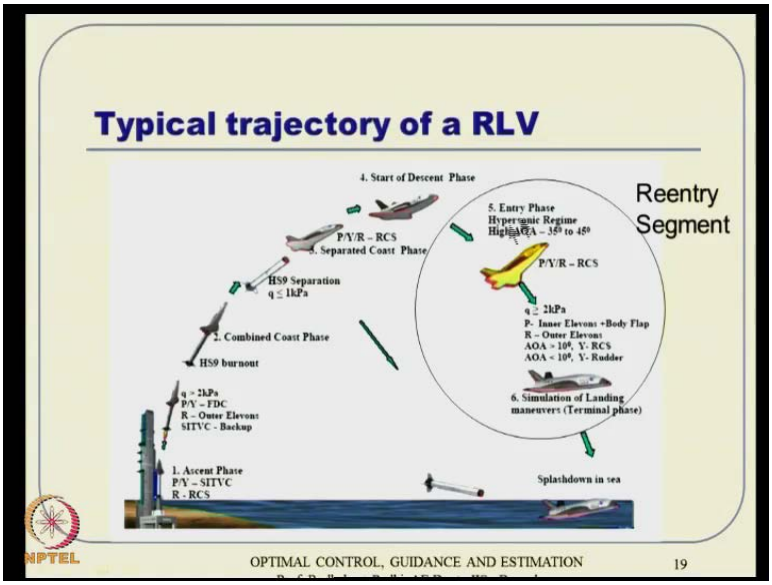
this this straight line passes through there the region or probably you can think of a i 0 then it is just a constraint thing and all that. So, but the constraint itself is updated in a in a systematic manner using this philosophy and all that. So, that is Model Predictive spread Control.

What it does it further improves the computational time because instead of entire control history grid point by grid point what you are doing here is, updating the coefficients only grid points are automatically taken care because of the polynomial fatigue that we have. So, essentially it further improves the computational time and also it smoothness the control history by enforcement. So, remember the other one may or may not happen because every grid point it is treated as some sought of independent control value basically. So, smoothness may happen and it does happen in most of the cases, but theoretical guaranty comes when you have something like polynomial fatigue. So, no matter whatever a I and B I you take this is a sincerely some sort of a what you what you see here is this is U I and this is t ok.

So, this is guaranteed to be smooth actually. So, that is this part of it. We will we will see that I mean probably one or two classes down the line actually. And then there is another extension which is called generalized MPSP very recent development ah around 2011 2012 like that. So, what happens here is we started with something like a discrete time frame framework. So, essentially we have revisited the entire problem formulation and we are propose some sought of equivalent development in continuous time framework also. That means, somebody I mean it not necessarily that you have to start with the another formulation and all that actually then carry on with the continuous time frame work from beginning to end and of course, for implementation sense you have to always do some Discretization at the end actually. ill ill I will talk about these two methods in after one more class probably. We will see some of this generalized and think like that alright. So, this is, these are extension part of it. Now coming back to some some example problems and all, so I thought here I thought instead of going through this typical text book sought of small small examples. So, we will rather go to a very practical example and see what the implications there are actually. So, this essentially talks about reentry guidance of a reusable launch vehicle and you know this reusable launch vehicle is also at in the kind of draws lot of attention allover because essentially by repeated launch of the vehicle ok.

If you are essentially aiming to bring down the cost of the launch of the satellite actually so, that is what it happens, but it its throws its own challenge and all that. So, how do you overcome and all we will see actually.

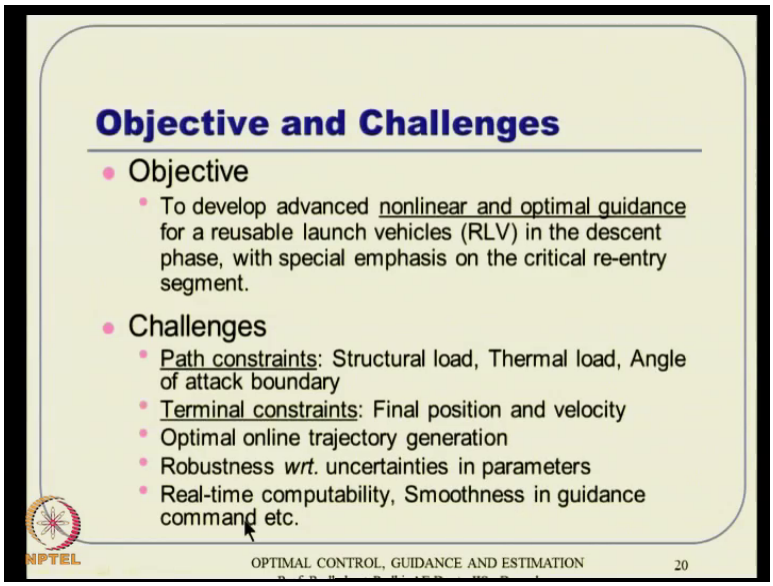
(Refer Slide Time: 38:33)



So, this is a typical kind of trajectory and all that from beginning to end. The launch vehicle takes some sought of a vehicle here to stop. On the way the launch vehicle itself drops out and after that this vehicle has to be recovered and it has to essentially go to a run way what also you can think about something like this splashdown in see and think like that, but we can essentially run to some sort of run way actually. So, this is the reentry segment that you are interested in and then we are talking about these numbers and all you can forget these are the, I mean typical values and all it does not matter. It depends from vehicles to vehicle varies from vehicle to vehicle like that actually.

Essentially the the approach is somewhat similar. It it goes to somewhere and then after sometime or it takes actually launches a satellite there and then after launching the satellite it has to come down and and recover itself. So, what you are interested in here in this particular problem is the reentry segment. This this Reentry segment is a very critical segment because its several constraint sections of vehicle actually.

(Refer Slide Time: 39:36)



So, how do you do that? And even these MPSP taking that you are talking here, there are certain kind of limitations as well which we did not talk probably. The very first limitation is let me kind of go back to that and probably show some of that basically. So, the cost function that you have assumed conveniently rather happens to be a function of control variable only ok.

So, the boundary conditions are there. That means, at equality that there are certain objectives for the output and all that is that is there what on the on the on the path actually where the cost function is not a function of the state it is still a function of a control only basically. The second thing is when the sincerely what you are talking here is well the constraint, control constraints and all that. So, those things are still not accounted for actually ok.

So, sincerely by by using this necessary conditions here. What you are assuming is the control is actually unconstraint. So, these are the primarily two kind of limitations that that adds on this algorithm as of now. Of course, you can think of doing further research to kind of relate some of these conditions actually anyway. So, this is what it is. This is reentry segment and where interested in something like the idea here is to develop advance non-linear optimal, non-linear and optimal guidance for an RLV in the descent phase with special emphasis on critical way reentry segment actually.

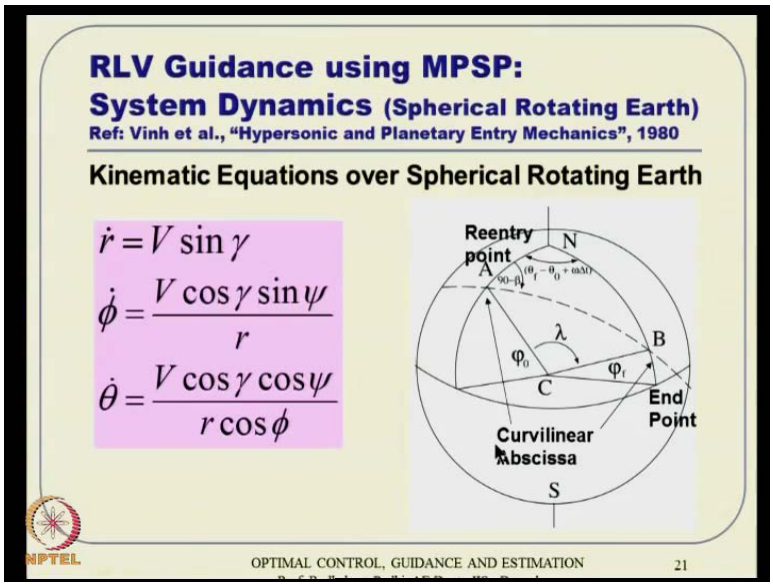
So, what are the challenges here? First challenge is Path constraints and Path constraints, there are several constraints, the structural load, thermal load, angle of attack boundary the primarily three things together. Then there is this and remember this angle of attack plays a very critical role because it essentially gives two things; one is controllability of the vehicle and also very strongly couple to the structural load actually. Structural role is a direct function of angle of attack really. actually ok

So, this has to be kind of handled very carefully actually. So, there is a Path constraint which essentially talks about structural load, thermal load, angle of attack boundary like that. And there is also a terminal constrains that ultimately at the end of the reentry we want to go to some position and then a specific velocity vector sense also basically. So, that means, the velocity magnitude as well as its direction has to be in specific direction. So, that ultimately we can recover the vehicle and then do vehicle actually.

Remember this RLV does not have any thrust. That means, you get only one chance to land actually and you cannot keep on moving around an airport and try to find out and think like that. It just gets one chance and then with that one chance we have to get it back actually and then we also aim to generate some sort of optimal trajectory and then, obviously, it has to have robustness with respect to uncertainties parameters and what you mean parameters? Essentially in inner and aerodynamics parameter and typically they are not very accurate. They will have some some twenty thirty percent inaccuracy aerodynamic parameter and some about five ten percent inaccuracy in parameters like going to finish a mass and all that actually.

So, the the algorithm has to be reverse to that actually when there is real time computability issue. And obviously, there is another issue of smoothness in guidance command generation. Also why do you need smoothness? Because ultimately what you what you interpret is control variable in guidance loop happens to be nothing, but the output I mean, reference command input to the inner loop where it needs to be tracked perfectly. Actually at track lay as close as possible. So, that is why we are having a smoothness, smooth history in the guidance command actually helps the, so, these are the things that we are looking for here.

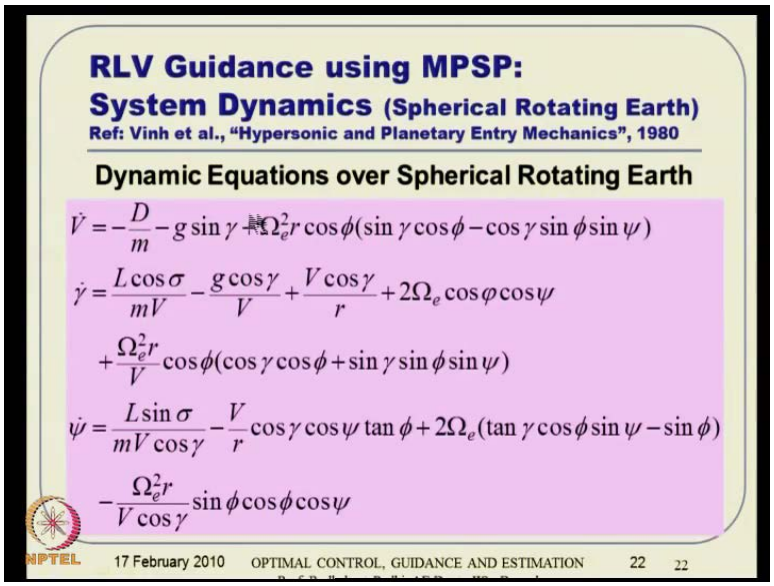
(Refer Slide Time: 43:31)



Now, when you go back to this dynamics it, I have taken this dynamics from this from this reference. Essentially I think which is this book is available probably free to free download or something these days I have seen soft copies actually. But anyway so, otherwise you can think of locating in some library and all that actually ok.

So, anyway this is obvious these three equations are essentially they reduce from center of earth and then, this and this latitude and longitude actually. So, essentially talks about spherical earth and rotating earth as well actually. I talk about everything spherical rotating earth. If you assume that you talk about spherical trigonometry and all that that way in the coordinates position, coordinates that is defined is in terms of r phi and theta taught you essentially and this I mean this latitude and longitude actually.

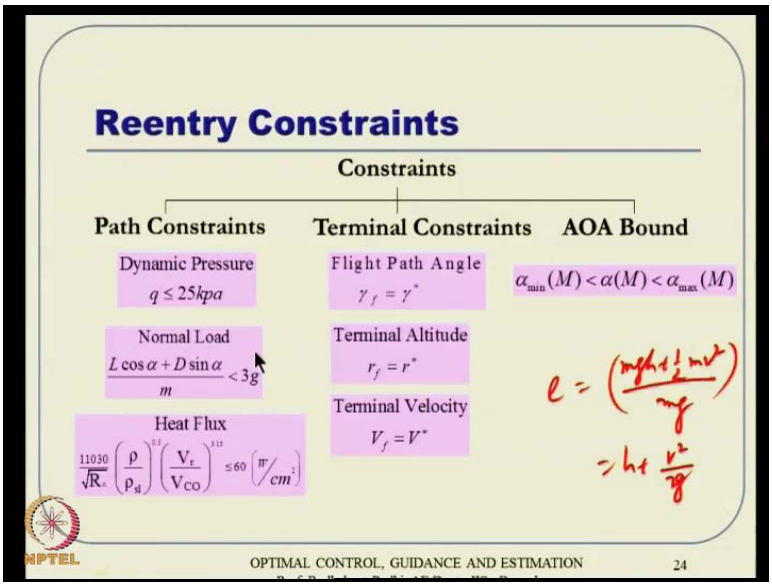
(Refer Slide Time: 44:27)



But this is Kinematic component what about Dynamic component? Dynamic equations will contain V gamma and psi which V is the velocity velocity magnitude and gamma and psi are essentially the Flight Path Angle and heading angle actually. So, these are if you see that this equation is already become quite kind of a mess actually ok.

So, first first thing what it is formulated only with pitch plane maneuvers; that means, we thought back angle can be suppressed and wards back angle all that it assures is just moving the vehicle in the one single direction without changing the direction of the velocity vector sideways actually. It just goes up and down vertical sought of thing and also we neglected this earth rotation form part of it actually for location short duration mess if it happens or earth rotation does not matter that much. So, what happens there actually.

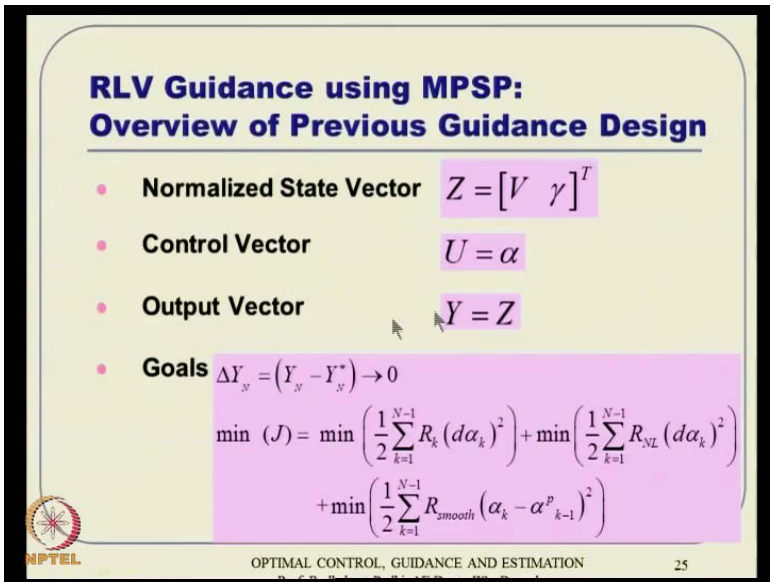
(Refer Slide Time: 45:19)



So, in with that results these are these are the something some reentry conditions that you constrains that you think about, they some of the values that we we applied to the kind of problem and then we thought to keep what about getting the solution out of predictor. So, then they told this this formulation in terms of really what is called as specific energy, a specific energy is nothing, but this kinetic energy e. e is something like kinetic energy and V h plus half m V square divided by m g for unit weight actually.

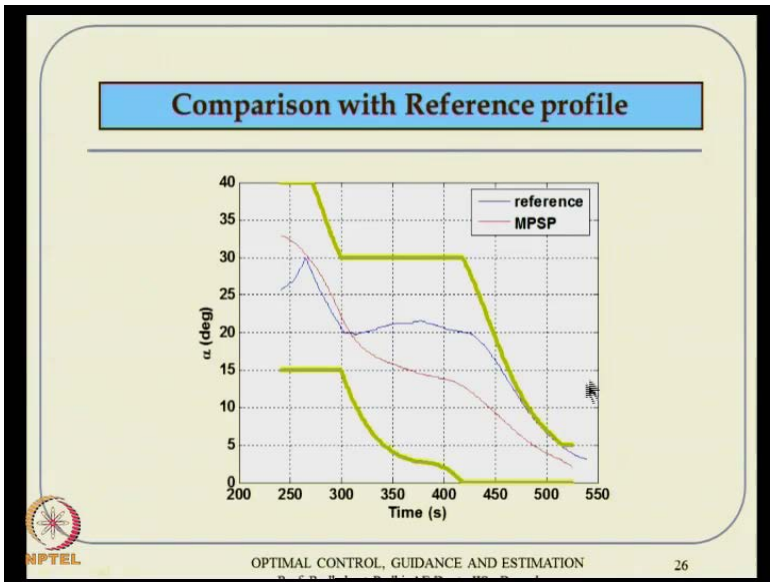
Essentially h plus V square by 2 g sought of thing actually. So, this is, if you if you know e and then if you know V the h is given actually that way. So, this three are linearly independent now basically.

(Refer Slide Time: 46:05)



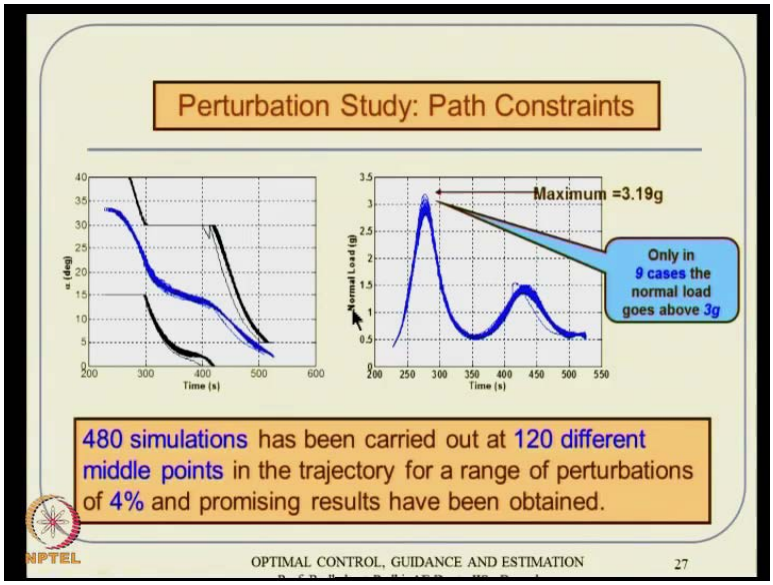
So, we talk about V and gamma essentially and then control vector is alpha and the output vector happens to be sort of thing whatever is the anyway. So, this is the formulation that we that we followed and also remember this is angle of attack is rounded between these two. So, we wanted a middle value as close to that is possible. So, the cost function assumed was something like deviation minimization actually. So, part of that this this part does deviation minimization, this part kind of assures normal load factors would be minimum and these essentially kind of guaranty is smoothness actually any way ok.

(Refer Slide Time: 46:44)



So, what you do here is the, if you see this result this is some sought of another method, solution actually what you what you have in blue line here. Take that as initial guess actually and then rotate through this algorithm and then found out that the, this this problem can equivalent be solved using this trajectory rather what you see here in this lines sought of things. What is beauty here? This this this line compared to the blue one this one is not only smoother, which is much more smoother actually and it also happens to be lying somewhere in the middle part. Whereas, this blue line is actually touching to the upper side right here which is not allowable actually ok?

(Refer Slide Time: 47:24)

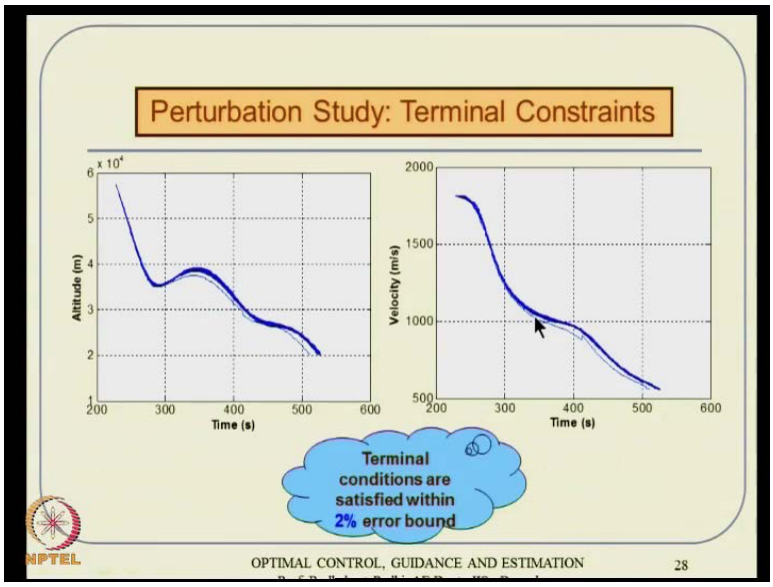


Any way so, coming to the the Perturbation study and things like that we have done some some some sought of Perturbation study is using this randomly wearing coefficients and all that and essentially them some 480 simulations and all that and some different different middle point, semi cell points and things like that. Essentially you can see that the the Normal load sense. It is slightly getting valuated here. It is value is 3g, but is getting around 3.2 g sort of thing. And but the number of cases that got violated is also small actually ok.

So, if you in other words if the structural limit becomes 3.2 g instead of 3 3 g when it is within that, but even otherwise the number of are very small actually and every time it happen to be in the middle of the boundaries basically for the alpha. They also remember the bound values themselves are function of might nerves. So, they are not fixed

numbers actually with respective to time. The max number is you know is velocity vehicle velocity on velocity the dynamic variable. So, the bound itself ah bound values itself change depending on the situation actually. So, that is why the, you can see once a bound c values is here actually.

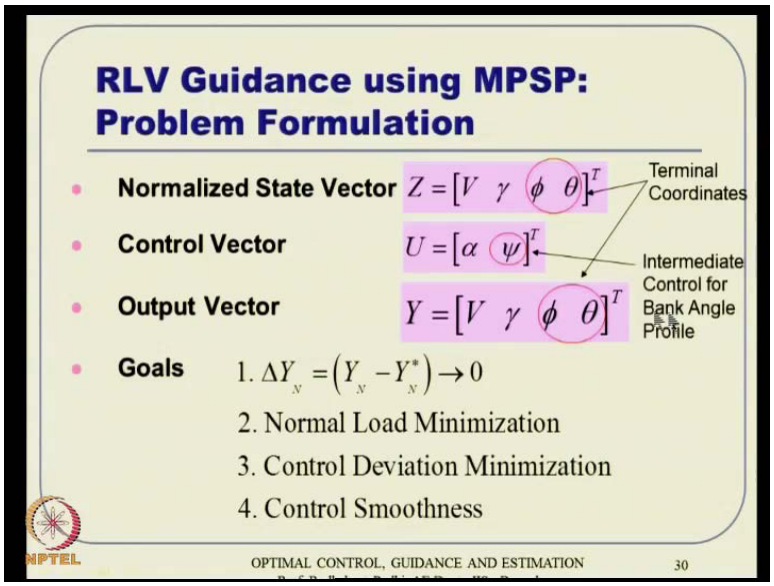
(Refer Slide Time: 48:38)



Alright and ultimately see these these points are meeting at the desire point. This altitude was is at something like 20 kilometers. So, it is all happening at the end and velocity also decide value is meeting you can see this plot line sort of things. So, no matter what happens on the way the the boundary conditions are enforced is is hard constraint. That is the reason why it happens this way actually. Now what about the next formulation which talks about both pitch plain and yaw plain maneuvers actually? That means, alpha and angle attack of alpha as well as back angle both are control variables then what actually? ok.

And even were this part this part of the problem formulation we did not bother much for the for the position coordinates actually wherever coordinate happens happens ultimately is our Path constrains and all has to be met and ultimately I want some sought of position coordinate in terms of height and I mean height only rather in a way. The latitude, longitude was kind of ignored basically.

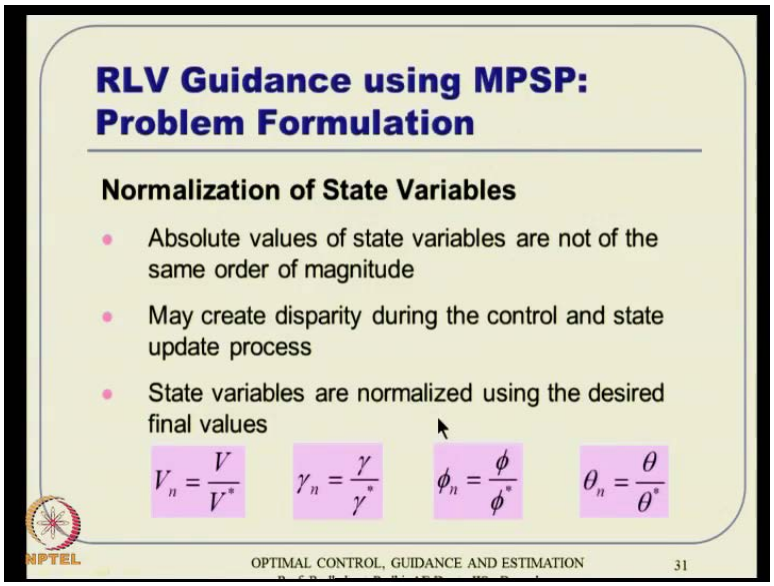
(Refer Slide Time: 49:43)



But, anyway coming to this one because we have back angle maneuvers as well. So, we can think about doing that also. So, instead of V and gamma which essentially talks about h as well because entire formulation is based on e. So, instead of V gamma h only what you talk here is phi theta also basically. So, that is what inter studied here and then all other things remains same actually. So, these are the Terminal coordinates what I am talking about here.

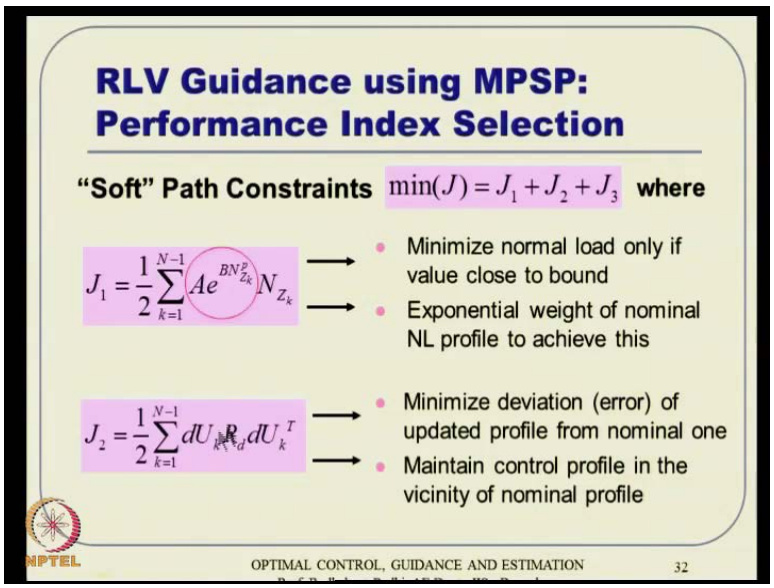
So, what you do it here, if you some little bit interesting thing out here what you can see here is, these this variables you want to control, but they they are not a direct function of sigma basically. So, if you if you consider sigma as a control variable then you have a problem here of problem control of availability starts coming and all that. So, what we thought is we will consider psi is the is the some sought of artificial control variable here and once you get a psi trajectory then, we can think about going to psi dot equation and compute sigma because psi dot is a strong function of sigma here as been compute I mean once your psi trajectory go to this equation and compute sigma based on the dynamic inversion actually. So, that is what you have done there and we can (( )). So, that is that is what is done here.

(Refer Slide Time: 51:04)



And this is intermediate control back angle. And also they it has done some normalization was done because V remember? V happens to be very high large quantity and become a these are angles and gradients which has small quantities like that we introduces this this normalization thing and then operated based on normal variables actually.

(Refer Slide Time: 51:22)

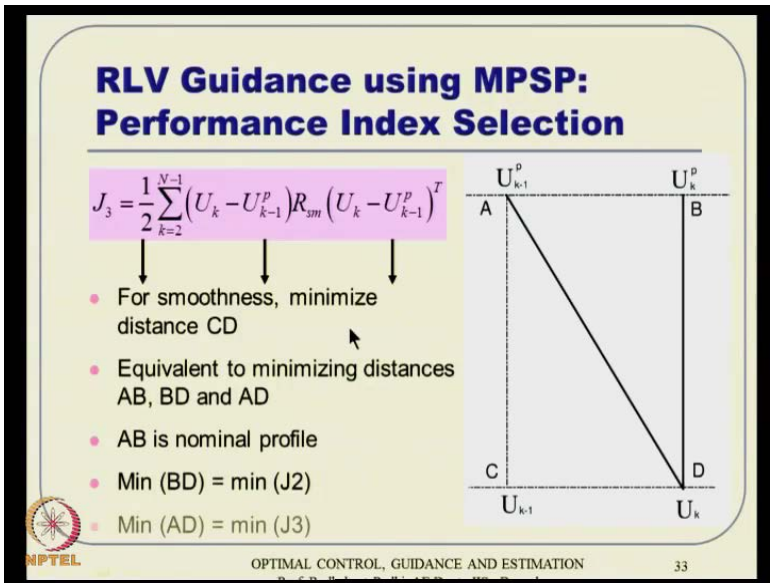


And as I told that J consist of three parts and J 1, J 2, J 3. J 1 contains this normal load factory and remember it operates some some sound factor here which is an experiential

function of the previous normal load at that particular point actually. That means, if there is the the iterations sends if the pervious value has taken it towards the boundary then, the waiting function is increased. That means, the update should happen in the negative. I mean in other direction actually ok.

So, that is our idea here. So, that is the how to address this normal load bound sort of thing and then this J2 happens to this control deviation minimization actually and remember when we talk about deviation minimization if you talk about the reference profile something like zero when we are talking about the control minimization directly actually. So, J 1 is is done to take care of normal load constraints in a soft constraints way and J 2 is done to take care of the alpha boundaries and sigma boundaries things like that and then these are all mentioned here. Essentially this file this fellow is effective only if the value is close to the bound otherwise is a very small value anyway but when it close to the bound that is increased exponential. So, suddenly the source is source is power actually.

(Refer Slide Time: 52:50)

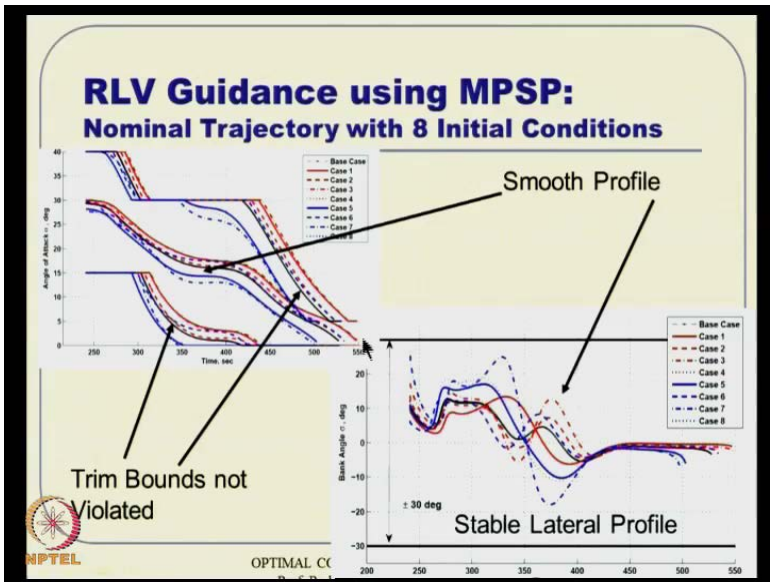


So, this this what is you see here J2 and this is what is done for smoothness actually. Smoothness remember? The derivative is small. That means, the difference between these two should be small, but once you directly formulate that that the cost function becomes or some some sort of a complex function. I mean the algebra becomes messy and things like that. So, what you thought instead of getting this deviation minimum

what how about making this deviation minimum and that deviation minimum? What is this is actually? The previous value of two. No, Never in more point. Previous value of the control. This is what you want, the correct value of the control k minus 1 and U k U k minus 1 and U k they happen to the current values of control is is U k. This is what you want up to update and this is what you want after the updated k minus 1, but these these things are already available. These are numbers actually ok.

But if you want to play around directly then the cost function becomes a kind of a involve it. I mean in kind of involve functions. So, the algebra leads to messy. What you thought is you can you can get away from that where you just having this innovative formulation where you talk about this this minimize. The difference between U k and U k minus 1 whatever it is, it can be minimized by minimizing the difference between U k and U k p. So, the p is value of the control at the same grid point. As well as the difference between U k and U k minus 1 p. So, if I if I minimize these and minimize that and obviously, this has to be usually minimized it in the way that is the philosophy followed here and telling that I mean this this is anyway taken care because we want every deviation minimization here. When you do that; that is already taken care. What you do is, we just need to do this minimization. Also actually this is what is given here for the next part of this alright. So, these are all explained here. I what. I have told already here.

(Refer Slide Time: 54:43)

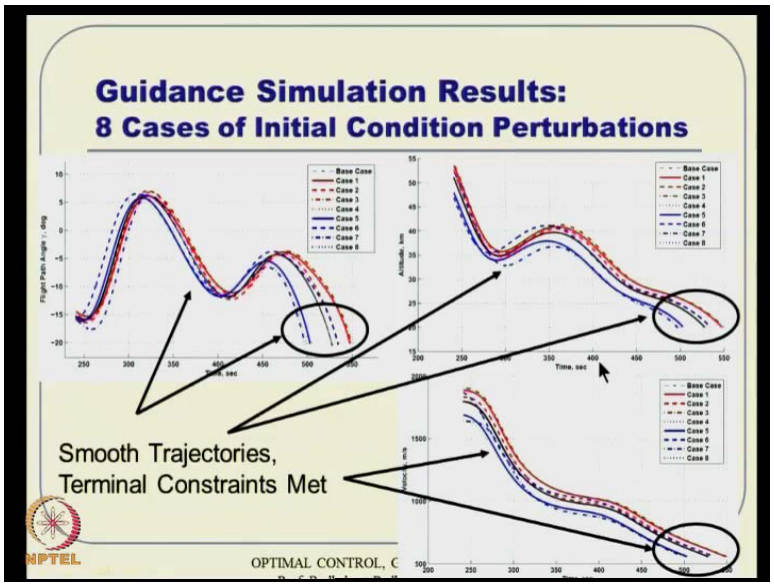


And then essentially the cost function is ready algorithm is ready. These are some of the reasons actually. So, what you see is some height cases we are plotted. So, you can see that for each of the cases all of the arguments that I told you before remains valid and ultimately every all of the cases different different color coding and land coding can you can see and remember if you are concentrating on a that particular line here for the control the history. Then you should also concentrate on that particular line for the maximum down and minimum down because everything happens to be time varying sort of thing actually ok.

Essentially if you concentrate this line then your I mean this is the upper bound and that is the lower bound, the control history like that way what the solution what you got actually like that. So, essentially you can observe that all the alpha solutions angle of a attack solutions happens to be smooth as well as the happens to be somewhat at the middle of the both boundary. Both the bounds actually now what happens the sigma part of it the back angle with we took some pressure minus 30 degree and. In fact, it can be taken much more that in a 30 degree because incase it is a kind of non manned missile basically.

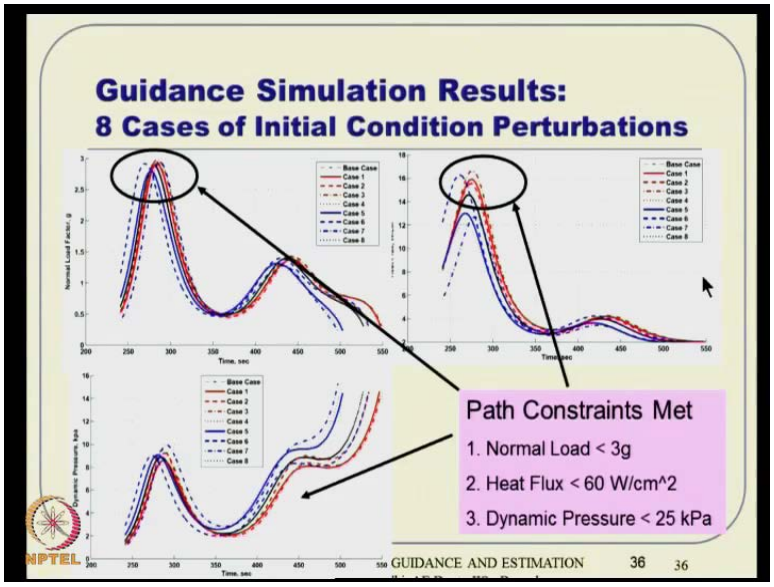
If you if you do not have human sitting there, probably the vehicle can turn little more and roll little more actually. So, anyway so, coming back we put some 30 degree pressure minus 30 degree bounds for our particular experiment. So, I started with different different initial conditions and all ultimately you can see that finally, I mean it is taking there, but also remember the final back angle what you are looking for is happening to be zero actually which is also nil. Ultimately at the end of the reentry one to be some sort of zero sigma altitude sort of things actually. So, these are some of the some of the comments that I have already told we are all smooth profiles and trim bounds are not violated here.

(Refer Slide Time: 56:35)



And and you can also see that the final conditions here are met in a tight conditions and see you can see, but no matter whatever is the initial condition ultimately the end of at the same value. That is why the the final wanted condition is enforced some sort of a hard constraint actually. So, these are the powerful things that you talk about when you use this optimal control based methods and approaches all that actually.

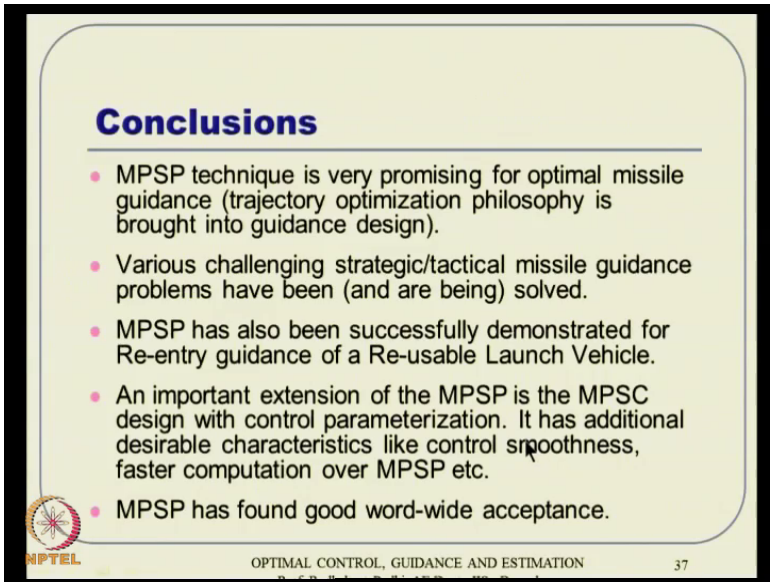
(Refer Slide Time: 56:59)



And you can also plot this is a these other the different things; this non allowed and this kind of dynamic pressure values and all that actually you can see that the all these I mean

this is strictly below 3 g and then this is I mean this constraint is is met and then you can some this the heat flux is also met actually. Dynamic pressure this is the value for a I mean, this is the dynamic pressure. it has to The Bound is about 25 kilo per scale, but what you are getting here is14,15 kilo per scale actually. It fluxes actually 60 watts for centimeter per kilo, but what you are getting is where we low way below. Basically that again depends on the missile and all that actually. So, if you go to higher and higher till teach you then try to come back your energy will be high and hence other things will come up actually this particular experiment that you have done the turns out the heat flux is not a major constraint actually.

(Refer Slide Time: 57:56)



In conclusion for this lecture what it turns out that, MPSP techniques is is the a very promising techniques for optimal missile guidance in particular, but project optimization in general. So, what it does is project optimize in Philosophy is what into the Guidance reason that that is more important actually. So, it can be used in missile guidance, it can be used in any U I V guidance. For example, any autonomous guidance problem, autonomous vehicle Guidance problem that we talk about and formulated based on this particular formulation if it is I mean, able to then you can think about using this.

It will essentially lead to this this optimal trajectories and optimal guidance all that various challenges I mean we have solved various challenging strategic and missile guidance problems. That I will take it through take it through in the next class, next

lecture actually some of these problems and there are still we will solve actually many things you can think about posing the problem in a in a good way then try to extract the benefit of this this method actually. It has also been successfully demonstrated for reentry guidance of a reusable launch vehicle that I have just taken you through and then the some some Extensions like the m p s c and all that with with Control parameterization. It additional desirable characteristics characteristics like smoothness and things like a faster computation and all that actually. So, it is now slowly getting world wide acceptance as well. I have seen many people when you liking this, referring this using in their problem and think like that.

So, I encourage all of you to kind of put your hands on this; the algorithm as well as the code. It is simple to code and and experiment also alright. So, with that probably I will list out some some references. These are the references that you can think about. This first one as I told, the last one is the very first one, but I have taken this earlier problem from these two references. One is already there. The other one in the in the conference version is there. The journal version we are trying to submit as soon as possible actually. So, with that I will stop this lecture. Thank you.