

Process Control-Design, Analysis and Assessment
Professor Raghunathan Rengaswamy
Department of Chemical Engineering, IIT Madras
Model Predictive Control-Part-2

We will continue with our 2nd lecture on model predictive control. In the last lecture I introduced the basic idea of model predictive control, contrasted that with the PID control idea that we have been teaching in this course. I made 3 important points about model predictive control. Number-one, model predictive control uses an explicit model, number 2 there is this notion of horizon which is introduced in model predictive control which is usually not seen in PID control. And number 3, while in PID control, the control law, it is an analytical equation, in model predictive control, the control itself is our solution to an optimisation problem.

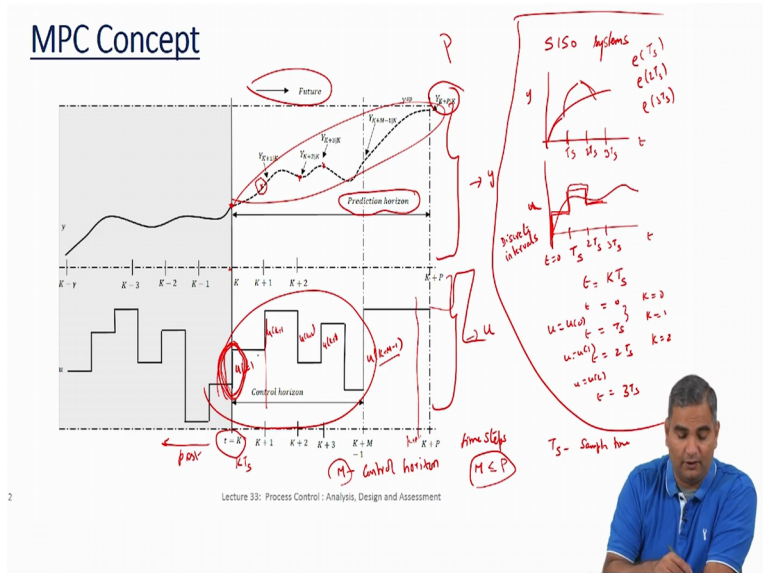
So these are the 3 major differences between the standard PID controller and model predictive control. As I said before, model predictive control can be derived from single input, single output systems and model predictive control can also be derived for multiple input multiple output systems. In fact one of the advantages of model predictive control is the formulation is pretty much the same, nothing changes but when you go from SISO formulations to MIMO formulations. That is a big advantage.

The other advantage which we will not see as much of this course at which is of real relevance is that when we go from linear systems to non-linear systems, the theory that underlies control, pretty much remains the same. In fact the formulation is absolutely the same. In one case you will be solving optimisation problems within models, in the other cases you will be solving optimisation problems with non-linear models. In that sense once you understand optimisation theory quite well, whether you are designing a controller for a linear system or non-linear system, really does not make that much of a difference when it comes to model predictive control.

Contrast that with traditional control approaches, for non-linear control itself, there are so many different techniques that you can think about, each of which comes with its own theory. So if you take the other approaches, you have to kind of keep learning different ideas for each type of population. Here however, in optimisation formulation, directly gives you a solution to this problem and once you understand how to solve optimisation problems well, then whether you are doing control for linear or non-linear system, does not matter quite a bit.

Okay, so with that what I am going to do is we are going to go and then look at a picture which is very very important, this is a picture that is useful for MPC quite well. And I will explain the idea of MPC using that picture. And then we will talk about how to actually formulate each one of the ideas that is there, that we learned from this picture.

(Refer Slide Time: 3:14)



So the idea of MPC is the following. This picture I am going to kind of explain each one of this. So 1st let us look at SISO system, right, so we will at SISO Systems. And you will see how easily we can extend this idea to MIMO systems and so on. So, in this picture if you break it into 2 halves, the 1st or the top half is plotting the control output Y and the bottom half is plotting the manipulated input U . So that is the 1st thing to notice. If the x-axis you have time steps, okay, so I am going to use this notion.

This is slightly different from the continuous time formulation, that we have been seeing till now. And I will explain what this means. 1st I will scroll of this conceptually and then we will talk about how you build the model for this and all that later. So, x-axis represents the different timestamps and, the 1st half of the y-axis, Y is plotted, the 2nd half for the bottom half of the Y axis, I am plotting U . And then we have to denote what current time that we are at and so T equal to K is the current timestamp that we are at.

So, anything in this direction is the past, things that have happened and everything in this direction is the future, things that are waiting to happen will happen. Okay, now when we, remember when we did all of these plots for even Y , have shown you many such plots, so I have said Y goes like this, maybe U goes like this and so on. So, when we do this, when we

talk about continuous time formulation, at every time there is an input and that is a corresponding output. So that is how we think about this continuous time formulation.

However if you think about controllers that are operational now and in fact controllers were computers through the computation for moving the actuator, the control moves are not realised at all times like this instantaneously changing. What happens is that we make this control moves at pre decided times, that is how most of the current controllers are implemented. So this is a different from what we have been seeing till now. So, till now we have looking at U being a continuous variable with respect to time, that is continuously changing with time and correspondingly Y also continuously changes with time.

Now U is in our control, right, we are making changes. So basically what happens in systems nowadays, is that let us say this is Time T equal to 0, we can arbitrarily chose anytime where this controller starts to work or it starts operating. Now, what we do is we basically figure out what is the control move that I need to take at time T equal to 0. So I make that move. And then basically in both systems, what I do is I hold this to this value. So, see the difference between the PID kind of pictures that we have been seeing and discrete nature of the control move that I am talking about for the 1st time, right.

So physically what you are going to say is if there is a tap that you are opening as a control valve, so what you do is that time T equal to 0, you open it little bit and then hold onto it. And then what you do is you decide the next sampling time, okay. This is called TS , next sampling time at which you are going to think about the next control move. So you could make the control move up or down, let us say you make a control move app, then what you do is you hold this till $2 TS$ and the 2nd instance of the sampling time and then maybe you make a decision and then you hold like this and then you make another change at $3 TS$ and so on.

So the changes that you make are at discrete intervals, this is something that is different from what we have been seeing till now. Because what we have been seeing till now is looking at this as continuous functions of time. But for the 1st time we are saying that these are changes that are made at discrete intervals of time. No, in reality whenever you do computer-controlled digital control nowadays, you basically make these discrete control moves.

So, from picture of U versus T , there is nothing that changes. If you have continuous control, you keep changing U at all times with respect to time. In discrete you have to choose this sampling time and then you keep making these changes at the sampling times. Now you will

notice obviously, if we make the sampling time smaller and smaller, you will make the system more and more continuous, right. So if TS is 1 second, then you basically making a control change every 1 second but if the TS is changing every 1 millisecond for example, then it will be a lot more continuous when you plot this in the scale.

Nonetheless, there is absolutely no difference. Basically what we are going to think of as this type of file for you as now these discrete steps. And the time at which you make these discrete steps T , we call this as $K TS$. So K equal to 0, T equal to 0, so that is the 1st time we make the 1st move. Then K equal to 1 at T equal to TS , so the 2nd time we make a move, K equal to 2, T equal to 2 TS and so on, the 3rd time we make a control move. So, if you ask what will happen to the U between 0 and TS , so between 0 and TS , whatever is a move I made at time T equal to 0, U_0 will be the value of you in this interval.

And after TS , U will be U , the move I make at 1 and after this it will be U move I make at 2, till 3 TS and so on. So this is how you can actually get value for U at any time, it need not be only at the sampling time, you can get a value at any time, however the changes are made only at those sampling instances, right. So that is the key idea when you go from continues to discrete. Now, let us look at what will happen to this picture, Y , Y will react to this but Y is a physical variable, so Y will keep changing based on these control moves you make.

So, if you are looking at the height of the tank, so height of the liquid in the tank, you will notice that whenever you open and close the tap, the height will start changing, going up and down. And we know for each of these changes, the behaviour is going to be kind of let us say if it is a first-order behaviour, you know if it is a step, the first-order behaviour will go something like this. And then if for a while it is going to stay and then once I make the next change, the thing is going to change like this and maybe it will change like this and so on.

So, Y will keep changing like based on the changes that are made in U . And while Y keeps changing, in MPC flavour, what we assume is also that we take a measurement for Y not continuously but had the same time instants that I do my control moves. So the measurements also are available at only those times however between these times the measurement, the actual physical variable will be having certain dynamics based on the standard when whenever we change U step, what will happen to Y , that will keep happening.

But you are only measuring at times TS_2 , TS_3 , TS_n and so on. So, this is how you go from a continuous framework to discrete framework where everything is happening at all times,

okay. So that is not changing. But the control moves are made only at discrete time intervals TS_2, TS_3, TS_n so on. And the measurements are also made at TS_2, TS_3, TS_n , so on. So, basically what this means is that when you compute an error between Y and Y setpoint.

Since you are not measuring throughout the interval, you have errors that you can compute at T of TS is a time at which you can compute an error, which will be the measurement at Y of TS, Y setpoint - measurement at Y of TS . Similarly you can compute an error at $2 TS$, error at $3 TS$ and so on. So again see the difference, in the previous PID case, we said we will keep comparing the error at all times, that is how we got the integral and differential and so on. And here the errors are also computed at discrete time intervals because the measurements are made at the time intervals and you are comparing the measurements with the setpoint at discrete time intervals, okay.

So this part is very important, 1st step in understanding MPC. And this is another difference between PID and MPC. However you can actually implement PID in a discrete form also and you will see that in your lap was, that goes with this course. MPC is mostly talked about only the discrete framework. So, to summarise discrete framework does not change anything physically, this is that U and Y a continuously changing. But you make these changes in U which is in your control only at discrete intervals. So basically this will be the U profile, time profile.

In response to that, Y will be doing its own dynamical thing, you only measure the Y at TS_2, TS_3, TS_n , so on. Okay, now we have understood this, let us come back to this picture and explain what this picture means. So what this picture says is, I am at a current time T equal to K , so we now write time as $0, 1, 2, 3, 4, K, K + 1, K + 2, K + M - 1, K + P$ and so on. So, what basically this means is that when I write time T equal to K , it is actually $K TS$, where TS is the sampling time that we talked about.

Okay, when I say $K + 1$, that is $K + 1 TS$, when I say T equal to $K + 2$, that is $K + 2 TS$ and so on, okay. So since we do not want to keep repeating this TS everywhere, we have simplified this and then say at time equal to TS . Now the MPC idea is the following. At time T equal to K , I have already got, let us measurement value here, okay. And I am planning to make a control move and I am planning to make a control move at time T equal to K . So this is the control move that I am deciding what I should make.

So I want to decide this control move in such a way that when I get measurement at $K + 1$, $K + 2$, $K + 3$ and so on up to a certain horizon which is what we talk about, this is into the future and I have a horizon. And what I want to do is I want to make a control move in such a way that when I get these measurements in future at time $K + 1$, $K + 2$, $K + 3$ and so on, I want the error between these measurements and the setpoint to be as small as possible, okay. So that is the basic idea.

So the key difference between PID and MPC is that in PID basically you might be interested in this error, not that this measurement is already there, I cannot do anything about this. We might say this error between Y measurement at this point and Y setpoint, we want that to be a small value, if something PID controller might want. However in MPC, what we are saying is, we are not only looking at what measurement I will get one-time step ahead, I am going to be looking at measurements then I will get P time steps ahead, okay.

So, if we start from T equal to K , the 1st measurement I will get one time step ahead is $K + 1$, okay. This given K basically means I am sitting at K in very simple terms, right. So all the computations are being done by the time is T equal to K TS. Okay, now instead of just saying I want Y of $K + 1$ given K to be close to setpoint, what I am saying here is I want to Y of $K + 2$ in given K , given K meaning I am at time K , also to be close to setpoint, all the way up to Y of $T + P$.

So, K predictions into the future, I want the errors between those predicted measurement values and the setpoint to be as small as possible. So if that is as small as possible, then the actual measurement will be close to setpoint, okay. This is the prediction horizon. So we will keep coming back to this, you will understand this better once we do the formulation. But this picture is very very critical to understand. So if you understand this picture, the Delhi becomes easy as we described the other things. So, let me say this again.

So, time T equal to K is where we are. We assume that the measurement has already come. I am trying to make a control move, for now I am just saying at time T equal to K . And basically I want to make this control move in such a way that all the measurements in the future or predictions that I make in the future. All those values are as close to setpoint as possible because I am trying to do this setpoint tracking here. Now, so this would be Y_{K+1} , given K , I am showing this as actual measurement. And once we use a model, you will see some minor differences there, we will come back to that later.

But as of now, whatever measurements I get at $K+1$ given K , $K+2$ given K , $K+3$ given K and so on up till $K+P$ given K have to be as close to setpoint as possible. Now you might be wondering how can you do this because you are saying whatever measurements I get at $K+1$, $K+2$, $K+3$ come all the way up to $K+P$ are as close to the setpoint as possible. So the key idea, this is where the model comes into picture. So, what we are going to do is sitting here we are going to predict what this Y will be.

So this Y is actually what is called \hat{Y} which is the prediction for the output. So sitting at this point, what I am going to say is, if I use a model and predict, if I make a change in U , how will the Y change over a period of time. And then if I say I am going to see the difference between that prediction and the actual setpoint and is up to times $K+P$ I will make this predictions and see the difference between the setpoint and these predictions. And somehow I am able to minimise that error, then the actual system in terms of the measurements itself will be close to the setpoint.

For example if there is absolutely no model plant mismatch, then the predictions will be actually what the measurements will be and they will be as close to setpoint as possible. Even if there is some mismatch between the model and predictions, even then there is a good chance that the actual measurements or the actual system output will be as close to the set point as possible, they may implement the controller.

So the notion of the model comes your where I am going to make predictions into the future, okay. And then enforce that the future predictions are as close to the setpoint as watchable. Okay, now that part of the picture is done, let us come to this part. The key thing to notice is this P , which is called the prediction horizon, which is how many predictions into the future, many points, right, that you are going to predict into the future to do this comparison between Y the point. So, that is the prediction horizon.

Sincerely to the business of predicting, so you might ask why should I make a plan for only time T equal to K . So I am going to assume that while I am sitting at T equal to K , I am also going to say that I will plan a set of input moves, I will just say there is this control move that I make only at T equal to K . So what I am going to say is, okay, if I say this control move is U_K , I am also going to plan for it U_{K+1} , I am also going to plan for U_{K+2} , I went to also plan for U_{K+3} and the last move I made to plan is U_{K+M-1} , okay.

So notice that the control move that we are planning start from U_K , U_{K+1} , U_{K+2} , all the way to U_{K+m-1} . Whereas the predictions are for $K+1$ all the way up to $K+P$. So, this is an important idea to remember. The reason why we start with K for the input, but $K+1$ for the output is because wherever you make a change, I have said there will be some time delay, in fact if I make a change right now, I will see the effect not right now because that is a static system. So, if I make a change now, I will see a change, that is a static system.

But we knew, we know that most of the systems are dynamic, so if I make a change now, it will take some time for the change to be realised. So if I make a change at U_K , Y_K cannot be changed, right, because it is not instantaneous. But, hopefully Y_{K+1} can be changed, okay. So that is the basic idea here. So, that reason why what we will do is once I get a measurement of Y_K , I will use that information also somehow. And then what I will do is I will calculate U_K because this U_K once I implement, that can have an effect on K , Y_{K+1} . U_K cannot have an effect on Y_K because instantaneous effects are usually not possible. Okay.

So what we now say is okay I do not want to make one move but I want to make a move plan. Now what is a move plan. Okay what is a move plan? I am going to make plan up to M values so that is this M is called the control horizon. Okay, and how is this M values, U_K , U_{K+1} and if it says U_{K+1} , U_K is 1, U_{K+1} is 2, U_{K+2} is 3, U_{K+M} and -1 is $K+M$, right. So there are M values that you have from U_K to U_{K+M-1} . And this is $K+M-1$ because I want to have M values and call that the control horizon, okay.

Now notice that this M , if it is P , that basically means I make a control move plan till $K+P-1$. And I know this $K+P-1$ move that I make will have an impact on $K+P$. So in some sense, M , the maximum value it can take is P , okay. So, then $K+P-1$ will still have an impact on $K+P$. If you take plan for control moves beyond this, it does not matter because if I am going to plan control moves beyond this, these are not going to have an effect on $K+P$ because these thereafter $K+P$.

So when I am trying to only minimise up to here, there is no point in try to make control moves beyond this point. So the M is less than equal to P is a general constraint then you can see will happen in MPC. However, in general we keep this M much smaller than P , okay. So basically what we are going to do, we are going to make a plan for control moves, so the idea is if I make a particular plan like this, how would that change this Y , if I am able to model this?

Then what I am saying is I can tune this plan in such a way that these predictions are as close to Y setpoint as possible, this is the basic idea of MPC. So, I am going to make a future input move plans and I am going to choose them in such a way that predicted output in relation horizon is as close to that point as possible. So how do we get this mathematically is the question that we have to answer. But conceptually this is what basically happened. And one last thing here, since I am explaining MPC in detail in using this slide.

The way actually this will be implemented is if you are at time T equal to K, you will do this move plan and then you will get UK, UK +1, UK + to come all the way up to UK + -1. And what you will do is you will implement this control more at UK. And then you will wait just like I said here and you will come back here. I am able to the same computation but now starting at T equal to K +1 because K +1 becomes the current instance. And then you will compute UK +1 again and then implement it and then keep doing this, okay.

So though you use a move plan at this time and compute all of this, you will not stay with this move plan till you come here. What you will do is when you go to the next time instant, you will make the next move. The reason why you will do this is the following. So, I gave you this example in the last lecture while I said why would I ever think of a move plan. That is because if I want to get to someplace in 5 years, then I have to decide that I will do certain things. And till today but maybe you might plan for a year and a half into the future, so that in 5 years you will kind of get to where you have to be.

Right, so you continuously plan. So let us say I plan today saying that I will do this for the 1st one month and then I will do this for the next one month and so on. And let us say one month later if I finished a lot more than what I planned or other things, the environment has changed, so what you plan might not be as relevant anymore. So what you will do is you will replan again for 1.5 year period, one month from now and then you will do that for a month and then at the end of the 2nd month you might again look at where you are and re plan again, right.

So while you are planning for both you know, 15-18 months, what you do is you implement your 1st month plan and then at the end of it you do not stick to the plan that you made 2 months back because everything changes, right. So there is an environment which is equivalent to disturbances in control and so on. So, what you thought today might not be exactly the same one month down the line. Not because your thinking has changed, your

thinking might still be the same but the world is moving so fast, things are changing that you will be forced to re-evaluate everything.

Right, the same idea is used in control. While you make all of these calculations into the future while you are at time T equal to K . At $K + 1$ you still have to re-evaluate your whole plants and then see what is the best at $K + 1$ because you have more information in terms of the surrounding systems and so on at this point. So, the only question then that begs an answer is if you are going to redo this at every time K , $K + 1$, $K + 2$ and so on, why do this move plan at the 1st instance itself. It is a question which is a fair question to ask and in a more advanced course I will explain this in lot more theoretical detail.

In this class all I am going to say is this, if you make a plan and if you do not have let us say a horizon of planning. Then typically what you are going to do is going to be very aggressive with your plan, right. So again going back to the same idea, if you say I want, what do I have to be 5 years down the line, then I am going to plan today what I have to be, for I want to be in 5 years time. Then what I am likely to do is, okay, there is so much difference between where I am and where I will be, where I want to be 5 years down the line, so I have to do so many things.

So you might make very very aggressive plan for the 1st month and then when you get to the 2nd month, it looks as if you have only one month to get everything done, though you have a lot of time to get things done, right. So if you make plans only for the present, then this was thinking in terms of spreading the work out over a period of time does not come in. So at every move you make you might be very aggressive. So you might be all over the place, you might say today, oh I am very deficient in this, have to really focus on this, and spend a lot of time and then one month down the line you suddenly say oh I am deficient in this, so you might be all over the place.

Which is that it might be an aggressive controller from a control viewpoint. So the idea really is to temper yourself and then say, look I have bit of time to get to where I want, so how do I distribute the work that I need to do or the things that I need to do in a more rational, both stable manner, so that I get to where I want to be 5 years down the line. So, having the horizon like this for the input variable makes the controller more stable, less aggressive and still at the same time get to where you want to get to 5 years down the line.

So that is the reason why you want to have a move plan every time. And the reason why you want to keep doing the plan again and again, replanning is because what the situation was here in terms of either your goals or your understanding of your ability which is a model or external factors. Right, you might think IT is the greatest thing 5 years down the line, you know a month later IOT is the greatest thing and so things keep changing from the environment viewpoint.

So if you are here and you can actually make a move plan again and not stick to the plan you made one month back, then it makes really no sense for you to stick to the previous plan, you might always recompute. No, us actually can see if this computation takes a long time, then you might say, okay, once I make a plan, then let me wait and then make plan only much later than simply follow through this. But if the computations are handlable or tractable, then you might actually keep making these computations again and again and keep re-evaluating what control moves love to make. So that is the basic idea in terms of MPC, how do we do this all of this mathematical is a question that we need to answer.

(Refer Slide Time: 31:17)

MPC Structure:

```
graph LR; A[Higher level objective] --> B[MPC]; B --> C[PID loops]
```

In summary, the key aspects of predictive control are:

1. The control law is a solution to an optimization problem
2. Model is explicitly used in optimization
3. Notion of horizon (errors in future)

3

Lecture 33: Process Control - Analysis, Design and Assessment

So now one another thing that I just want to mention here. Because we have gone MPC and kind of talked about MPC as being a panacea for many things, I just do not want to leave the people taking this course with the idea that PID loops are done and dusted with. Absolutely not true, because these PID loops are very very critical for maintaining the basic regulatory functions of any plant and particularly with respect to safety and so on. So, typically how MPC is implemented is the following, right. I just wanted to give you this idea.

So if you have a plant, it might have several PID loops already, okay. So these are PID loops, let us say that already there in the plant. Now when you want to implement an MPC, you are not going to come here and then select me remove all the PID loops and implement and MPC, that is not how MPC is implemented. So, when you want to implement MPC, what you do is the following. So you say look I have some higher-level objective, okay, for the plant, I am going to pass that objective onto the MPC and based on this objective this MPC is going to manipulation the setpoints of these PID loops, okay.

So, you might think of this as a joint cascade structures that we talked about in terms of the cascade control loop. Here there is one MPC which is going to manipulate the setpoints of all the lower-level PID loops. So the way the setpoints are figured out is by maximising some objective, okay, which is a higher-level objectives. So the outputs are actually related to the higher-level objectives and the model that MPC uses is between the setpoints of these various PID looks. And how this setpoints of these PID loops affect the output.

Right, so, if I think about this and MPC controller takes let us say some higher level objective and then gives out the setpoints for this PID loop. So, which goes into the process, right and the process give me an output and that is compared with whatever I desire and the MPC again gives setpoints to the PID loop. So this is a general structure that is used. It really does not matter. So what the MPC is manipulating as U are the setpoints to the PID loops. And the output is or outputs are related to some higher-level objective, which is what is shown in this picture. There are several advantages implementing MPC like this.

Number-one is that all the advantages that you have got out of implementing PID, you do not take it out and then remove it because the PIDs are still in place. So, that is number-one advantage. So whatever work you have done till now are all in-place and of course if there are some safety critical loops that require your safety-related attention, it is very difficult to bring that in an MPC framework because your plug-in located in each individual loop and then say whether the objective is to do with safety, the objective is to do with the performance and so on.

So all of those kind of details can be relegated to a lower level PID control loop or loops. And MPC can really focus on getting something more than what the PID loops are doing a terms of higher-level performance objective. So that is one reason why the PID loops are there. The other reason and even more importantly is that whenever we implement Controllers, we worry about the reliability of these controllers. And if we were to remove all of these PID

loops and then make them all and MPC loop, so in other words there are no setpoints, there are direct manipulated variables of this PID loops themselves, which are being set by the MPC.

Then let us say the idea of MPC is to implement this on large number of control loops. Let us say I implement an MPC on let us say 40 loops, then what happens is if this MPC fails for some reason, right, or decommissioned for some reason for a short period of time, then all of these 40 loops go out of control, okay. That is something that is a very poor design from reliability viewpoint. You do not want this to happen. However if you were to implement MPC this framework where MPC is giving setpoints to the individual loops, then even if the MPC is decommissioned or taken off for a while, the lower-level loops are still going to operate as they have been operating with the setpoints that they have, okay.

So in that sense it is much better to do the structure from reliability viewpoint. Now, when we actually do this MPC lab part of this course, in all cases we might not look at setting up this problem where there are lower-level loops and MPC operates at a higher level. You might show examples of actually MPC directly manipulating, the many plated variables corresponding to kind of SISO loops. And that kind of implementation also does happen if we have a very small MPC implementations with 2-3 control variables and 2-3 manipulated variables.

It might be better to do an MPC rather than multivariable transfer function based controller design. So that also happens but when we really talk about very large scale MPC, then this is the approach that is generally part of. So, when you see the lab portion where the actual manipulated variables are being changed with an MPC, think of that as an actually a larger multivariable control problems that are SISO 3 by 3, 4 by 4, 5 by 5 and so on. So, do not look for underlying SISO loops and all that.

But what I want to say is both of these are possible, you can think of MPC for smaller blocks of multivariable control. Or you can think of MPC as large solution which is going to coordinate several underlying PID control loops.

(Refer Slide Time: 37:53)

Advantages of MPC

The advantages of MPC over traditional control techniques are:

1. Can be applied directly to linear and nonlinear systems
2. Easy to implement for univariate and multivariate systems
3. Ability to include constraints and cost minimization
4. Intuitive tuning parameters

4

Lecture 33: Process Control - Analysis, Design and Assessment

So the advantages which we have already talked about. It can be directly applied to linear and nonlinear systems, this is something that we already talked about. I am not sure how it is going to happen because we have not looked at the mathematical formulation yet, but this is something that we can easily do. It is easy to implement for univariate or single input single output and multivariate or multiple input multiple output systems. It is very very simple, direct extension, which we will see once we understand how this formulation is done.

The ability to include constraints which is an important aspect, I talked about U , that is a manipulated variable U itself having some bounds within which it has stopped rate. Not only that, the rate of change of U could also have bounds within which the changes have to operate. So we can include all of those quite easily into the optimisation formulation. This is very difficult to do if you are looking at PID where we talked about just including a clipping constraint, if you have to include constraints.

And the last one is the tuning parameters are intuitive, I have not explain what those are but in this 1st level undergrad course, my idea is to get you familiar with this notion of MPC and not think of MPC as something very complicated. I just wanted to explain to you that it is very very simple, very easy to move to multivariable control using MPC. So these tuning parameters and detailed description of tuning parameters, I might not be able to do in this course but once you have this basic understanding, when you look at the other material which is little more advanced, then you will be able to easily understand some of these ideas.


(Refer Slide Time: 39:42)

Summary - Differences between PID and MPC

PID	MPC
<ul style="list-style-type: none">• PID control algorithm is prescriptive in nature, and same for any process except for choice of PID parameters• PID controls the current output close to its set-point at a given time• Control law is an analytical solution	<ul style="list-style-type: none">• MPC uses an explicit model of the process• MPC includes the effect of current action on future outputs in its formulation through defining a future horizon of interest• Control law is a solution to an optimization process

5

Lecture 33: Process Control - Analysis, Design and Assessment



So, we have always talked about this in detail at the beginning of this lecture also I started by saying this. The 1st difference is that MPC uses an explicit model of the process and the notion of horizon is something that MPC does, which is very different from PID. And the control solution itself is a solution to an optimisation problem and not direct analytical expression, though in more advanced analysis of MPC, one could see that actually you can come up with some analytical expressions.

But you have to figure out in which region you are operating and with analytical expression to use and so on. So, that is for more advanced ideas in MPC. As of now one can think of this as solution to an optimisation problem and because it is a solution to an optimisation problem, we can include all kinds of constraints that we want an optimally solve for the control move, which would be more difficult to do if you are working with PID control.

So, basically the 1st picture and the whole explanation, if you are able to understand it really well, then the other aspects of MPC formulation become rather simple to explain, which is what I will try to do in the next lectures that I am going to teach in this course, thank you.