CH5230: System Identification

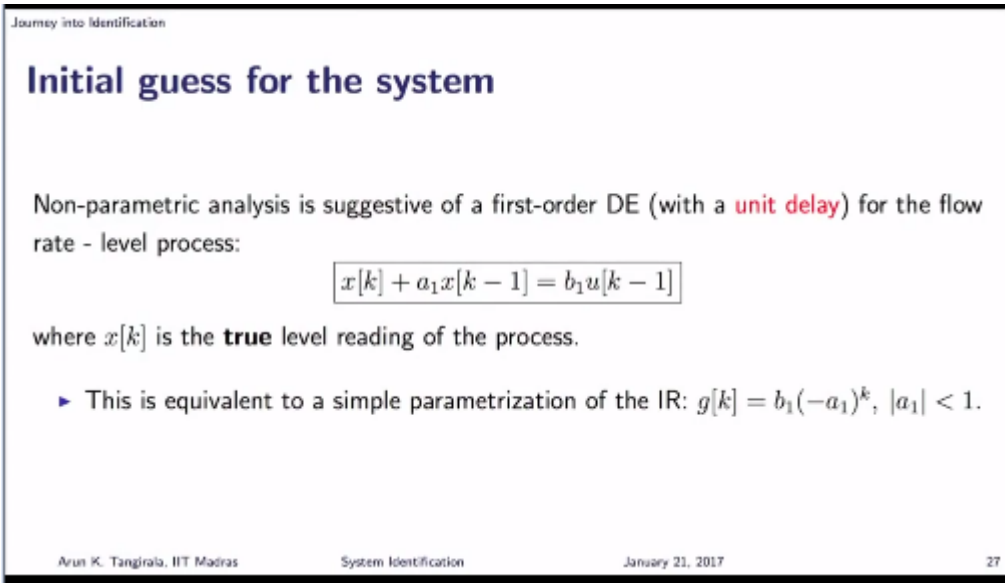
Journey into Identification

(Case Studies) 14

So, let's continue with our case study and hopefully at least close the case study today. So if you recall we are building a model for the liquid level system from experimental data and we have gone through a visual analysis of the input output data. We did some preliminary experiment to generate that rich data that we analyzed. We are continuing to analyze and then we estimated initially non parametric models mainly the impulse response, a step response, and the frequency response models which gave us some vital pieces of information about the system and what I mean by vital is again delay, input output delay, a guess, a reasonable guess for the dynamics, or order of the dynamics and then an idea about the gain, the sign of the gain and so on.

So what now, we have been discussing at least towards the end of the previous lecture is, fitting parametric models now. And we argued why we want to fit the parametric model because the non parametric models they involve typically a large number of unknowns to be estimated. And that's not good or recommended from an estimation viewpoint and therefore we decided to parameterize the responses. What we mean by parameterization is, trying to fit an equation to the curve. And we picked impulse response for that purpose but we could have big step response or frequency response and also assume some equation of curve. Now once you assume an equation of curve, what I have said in the previous classes that amounts to writing a different equation form for the system. And in particular we assume an exponential DK for the impulse response although I didn't show it, I did mention nevertheless that assuming such an equation of curve for the impulse response amounts to assuming a differential equation of first order and that is where we are.

(Refer Slide Time: 2:20)

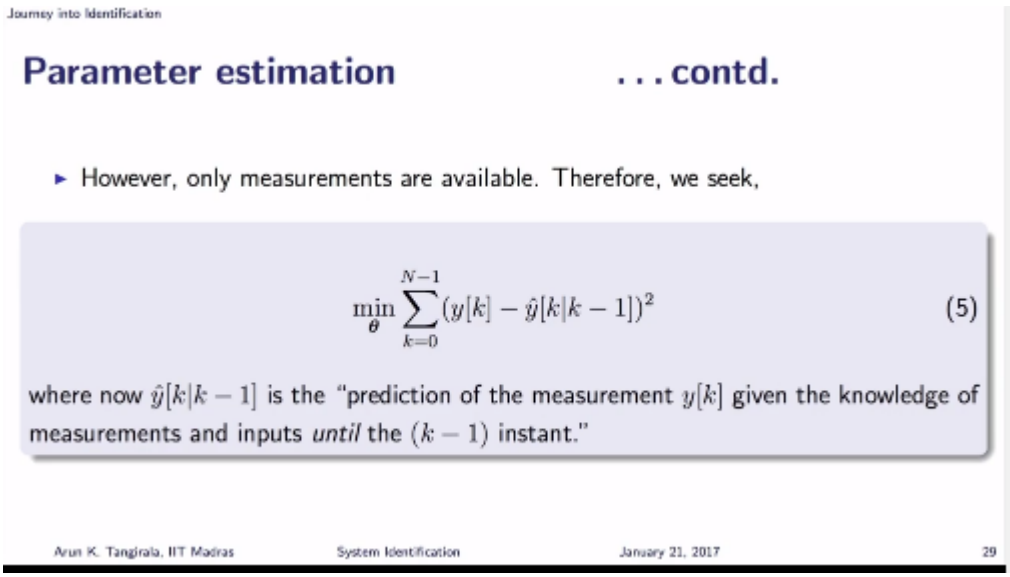


Where x is the noise free response which is our y star, and we have UK minus 1 appearing in the difference equation because as we have already discussed there is one delay between the measured output and the discrete time input. And we also said that given X and U we would measure a and b by minimizing the sum squares errors, right? Because we are although, X is an noise free component, I may have made some modeling error and therefore I may not be able to exactly fit even under noise free conditions it may not be possible to exactly fit a linear model. Therefore, I seek the best

approximate model and one way of seeking the best approximate model is by fitting least squares. Here is where estimation theory comes in.

So this is the least squares estimation approach where we try to seek an approximation x hat of the observed x, if x was given to us. And then drive these knobs that is you can think of parameters as your knobs, you are free to manipulate those and achieve a certain degree of approximation or you know or the sum square error here. But unfortunately, we did not have x. Remember, we do not have access to the noise free input. Instead we have y, therefore we would minimize the approximation error between the observed response and the prediction of the observation. So you should understand there is a distinction between x hat and y hat. So here is our g, the deterministic process being excited by u. And this is our x. They've been using remember, y star also so for the moment both are identical. We have this stochastic component v, corrupting x to give us the observed y. So x hat of k given k minus 1 was the one that appeared in the previous slide that I showed you. Now you have y hat of k given k minus 1.

(Refer Slide Time: 4:47)



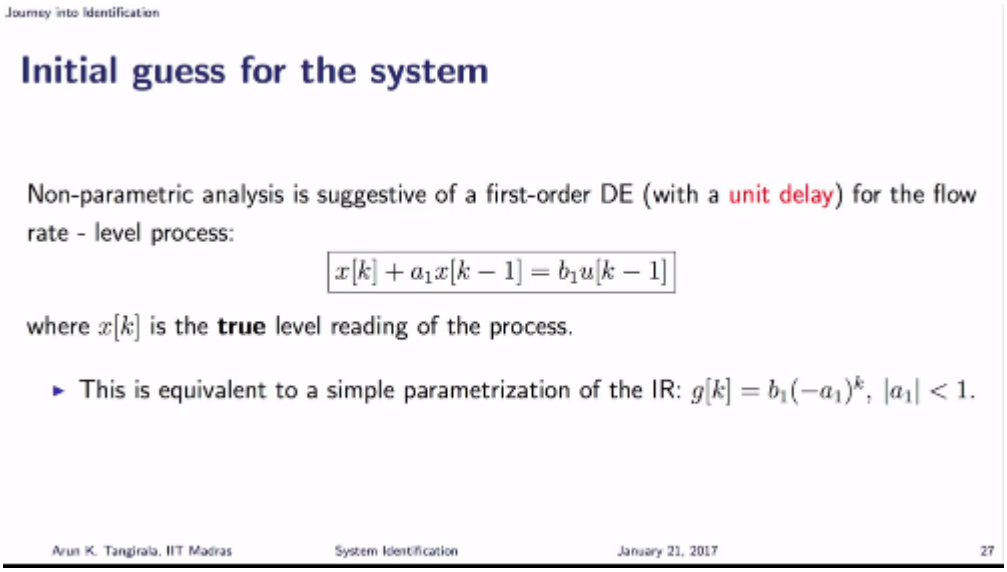
The reason for formulating this problem is I do not have access to x and this is how typically you would estimate parameters. But this is not the only way. You shall learn later on that predominantly this is the approach. But there is another approach which also allows us to estimate parameters where you may not minimize a sum square error like this. Okay, so now you have to be clear in your mind,

why we are solving this kind of an optimization problem, that's because I have access only to y. And what we're trying to do is, we are trying to find a and b or a1 and b1 such that y hat of k given k minus 1. What is y hat of k given k minus 1? It is called one Step Ahead prediction. That's the notation that's conventionally used. What we mean by given k minus 1is given all the measurements up to k minus 1. In fact, strictly speaking I should also include here the parameter. So given the model and the measurements up to k minus 1, the prediction of y at the next instant is y hat of k given k minus 1. What is our theta here? Theta is a vector of parameters that we would like to estimate a1 and b1. So if I were to give you the model and if I were to give you measurements up to k minus 1, the prediction that you would make of y at k is y hat of k given k minus 1. It's pretty much like in a game of chess. If I tell you how your opponent plays and if I give you all the moves that the opponent has made until now in this game, you would make a prediction of what would be the next move that your opponent would make. That's exactly your y hat of k given k minus 1. Of course, now you can extend this concept to two step ahead prediction, three step ahead prediction and so on. We know in a game of chess all such predictions are made, k step you know, p step ahead prediction and so on.

We are particularly interested in one step ahead prediction and we'll talk about y at a later stage. So I assume now that this is the approximation that we are seeking with that is we are seeking a model now even a1 and b1 such that now yk minus y hat of given k minus 1 that sum square error, yk minus y hat k given k minus 1is called the one step ahead prediction error. We are minimizing the sum square prediction errors. So what now is required is the optimizer needs to know how to compute for a guess of a1 and b1? So now you can imagine how the optimization problem works. Just manually it would try out a value of a1 and b1. It doesn't exactly work this way but just as a thought process it would try an a1 and b1 compute a prediction y hat and then compute the sum square error and then see, if there exist another value of a1 and b1 that will get it a lower sum square error and so on. But in order for the optimizer to do that it needs an expression for y hat. If there is a mathematical expression that we need to-- in other words, we have to tell the optimizer for a given guess of a1 and b1. And given all the measurements how it should compute y hat? If you don't tell optimizer, it'll just stay quiet. So that has to be also passed on to the optimizer and then the optimizer does the search. Of course, there are two possibilities always in an optimization problem when it comes to the nature of the solution. One, that you have an analytical solution where you can write the solution by hand and then directly compute the solution. The other is, where you do not have a closed form expression for the solution. Such cases are when you run into non-linear optimization problems where there is no analytical solution, only numerical optimum can be computed and you must have come across these kinds of situations at some point or the other in your courses.

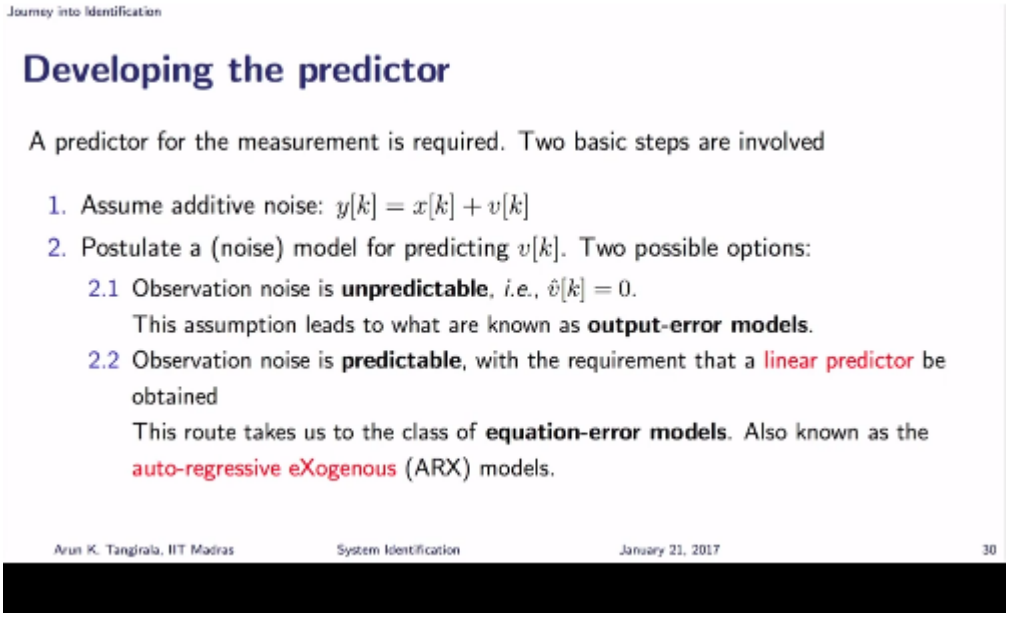
So we will come across these two different scenarios shortly. That is we will run into a nonlinear optimization problem as well as a linear optimization problem depending on what assumptions I make on V. Okay. And that's what exactly where we ended last time. So now as I just now said, we need an expression for y hat so that I can pass that to the optimizer. Now what we have written earlier if you recall is an expression for x that is we have assumed a generating equation for x not for y. In order to compute why we need a generating expression for y from where we will develop expression for y hat.

(Refer Slide Time: 9:45)



So to do that we now need to make the invoke the first assumption which is that of additive noise. Right. Remember now that we have to connect y, I know the at least, I have guessed the equation for x but I don't have an equation for y but I know by virtue of the additive noise assumption y is x plus v. So I invoke that assumption.

(Refer Slide Time: 9:55)



And then there are this two choices that we have. Which is one that we assume noise is unpredictable. And the other is noise is predictable. When it comes to assumptions on vk, we have two possibilities. One that vk is unpredictable. What we mean by unpredictable is, given any amount of past you cannot improve upon the prediction of vk beyond its average. Generally for a given random signal the crudest

prediction is the average. That we know. And when we say that a random signal is unpredictable you cannot improve beyond the average. You can remember it that way for now. For those of you who are familiar with time series modeling probably you know a bit better than this but that is essence of unpredictability. When you make this assumption you are let to what is known as an output error model. Okay. By the way, when we say, a random stationary signal is unpredictable. Typically, we come across this term called white noise. So what we are assuming is that in approach 1 we assume that vk is white noise which is typically denoted by ek that is the notation that we shall use to denote white noise. And wherever you see ek in this course unless otherwise specified you assume it is white noise. What is it mean? It's basically unpredictable and you can say that essentially the prediction of e given any amount of past is always-- well, I assume that ek0 mean strictly speaking it should be mu e, then mean of the signal. But generally we assume this noise sequence is to be of zero mean.

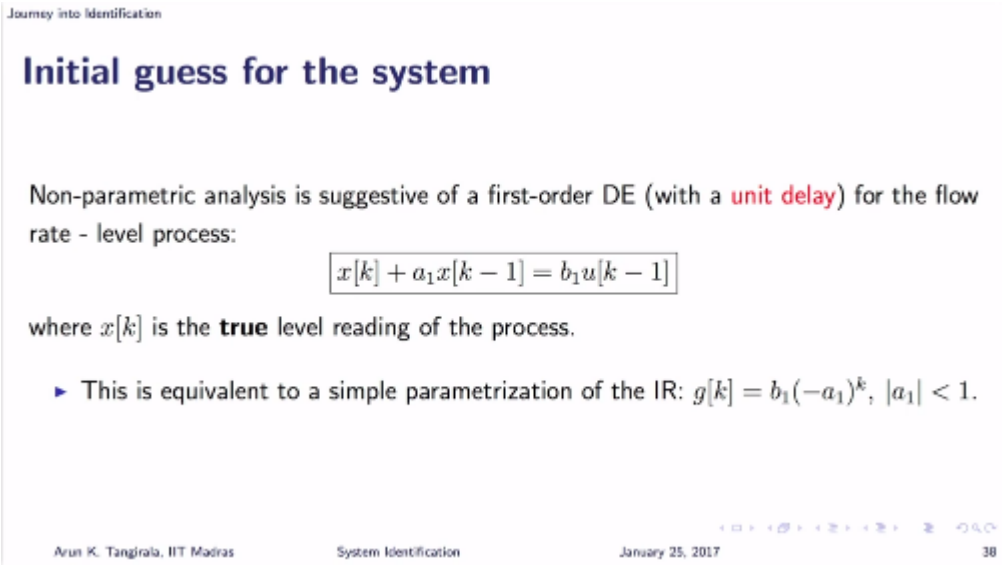
So doesn't matter whether you give me 100 observations of ek or 10,000 observations of the ek, the one step ahead prediction of e is always going to be zero. Well, assuming zero mean or if it is not zero mean then it's simply the mean. In other words, the past has no bearing on the evolution of ek. Is this true for the given process, we do not know. This is where we are going to make. We are making some wild guesses. The other approach is to assume that vk is predictable. Okay. But we will also require that once I write the expression for y hat. Why are we doing this? Because ultimately I want to develop a prediction of y. Depending on the assumptions that I'm making I'll end up with different expression for y hat. When I assume in the second approach, when I assume vk is predictable, I shall also require that y hat which is a prediction of y, be a linear function of a and b. And I'll tell you why later on. Why we are imposing such a requirement. So these are the two different options that we are going to consider. I should also tell you, this is not exactly the way you would work coding system identification but effectively this is what you will be working out in a typical system identification exercise. This is kind of I have peeled a few layers for you and I'm going to the core, when you actually practice, when you fit the model, you may not exactly go to this level of detail but it is important to know this level of detail.

So let's look at what the first approach gets us. Remember, we assume we hat of k, vk to be white noise, which means v hat of k given k minus 1 is 0. Right. Therefore, from this equation that they have y k is y is x plus v, therefore y hat is going to be the sum of their predictions. All right, I'm assuming open loop conditions here. So it's okay, to write this. Right, this now going to be the question that we are going to use to estimate the parameters. In the first assumption, we have assumed vk to be white noise. That means, I assume noise is unpredictable that kind of a noise is entering the system. Therefore, when I follow the first approach, this approach here, this implies that y hat is simply x hat. That is whatever would be the prediction of the noise free component would be the same for the prediction of the measurement. You have to distinguish between prediction of the noise free component which is the deterministic part from the prediction of the measurement.

What is the difference between these two? Yeah, the answer is on the board. What is the difference between the prediction of the noise free component, and the prediction of the measurement? Are they identical always? Correct. So there is v hat. Right. So always you have to remember this, doesn't matter whether you're in system identification or any other signal analysis. The prediction of the measurement is different from the prediction of the deterministic competent. And why is it different?

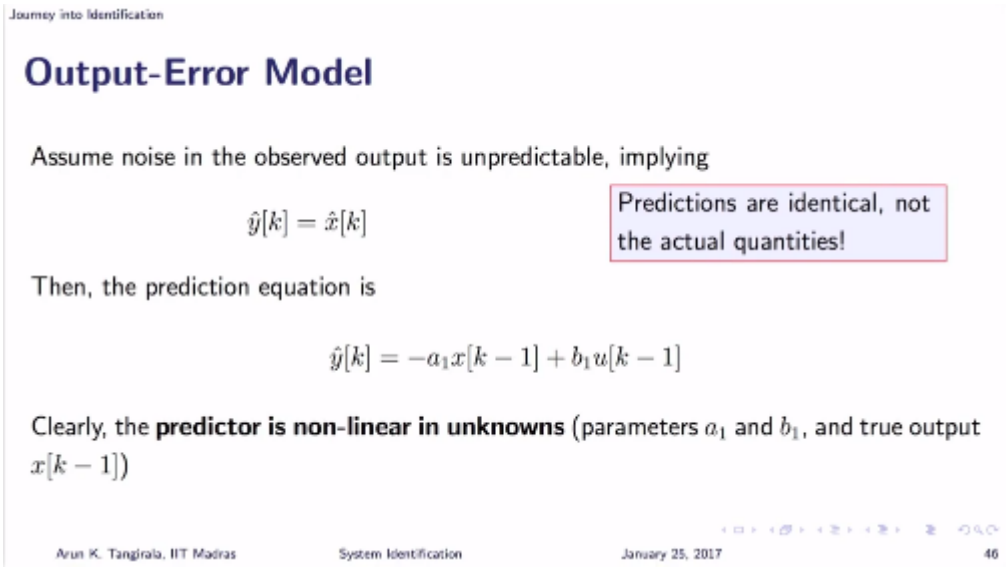
Because there is a noise component and that makes a big difference because we assumed in the first route, first approach that noise is unpredictable, the second term vanishes. And therefore the predictions of the measurement and the noise free term coincide. But this should not make you believe that measurement and x are the same. Okay. Just because y hat equals x hat, it doesn't mean y is equal to x. Be careful, okay. Predictions of these two are identical. Fine, now what does this lead to? Basically, now, I already know x, I already know x, the governing equation for x. It's the first order equation, right. So let me take you back to that equation. This is a question that we have for x.

(Refer Slide Time: 17:08)



If I give you this equation and a and b and I give you information up to k minus 1. That is give you up to the previous instant k minus 1. What would be a prediction of x at k? Input is always known. It doesn't have to be specified. So it's straightforward x hat of k is minus a1 plus xk minus 1 plus b1 uk minus 1. It's very simple, right. That is your prediction. And that's exactly the expression that you see, for y hat a.

(Refer Slide Time: 17:44)



Essentially, what I have done is, replaced x hat with its prediction. Why you have to keep asking why we are doing this. The goal is to come up with an expression for y hat in terms of the parameters. So that I can then optimize the parameters. Now of course, there is still a problem here with this expression. What is the problem? X is unknown, right? So that is where the problem with this assumption is always-- when you--By the way, as I said earlier this is called the output error model structure. Whenever you assume that the noise vk is white noise, the name given to such a model structure is called output error model. That is the white noise directly enters the output and one of the problems with this structure is that the predictor expression is not simple. I still have a problem here because ultimately what I want to do is, I want to write y hat as in terms of knows and parameters. That's all. That's all I need to know. But unfortunately there is an x coming here, which I do not know. However, if you just recall the equation that I showed you x, x itself is a function of a and b and of course, a input, correct?

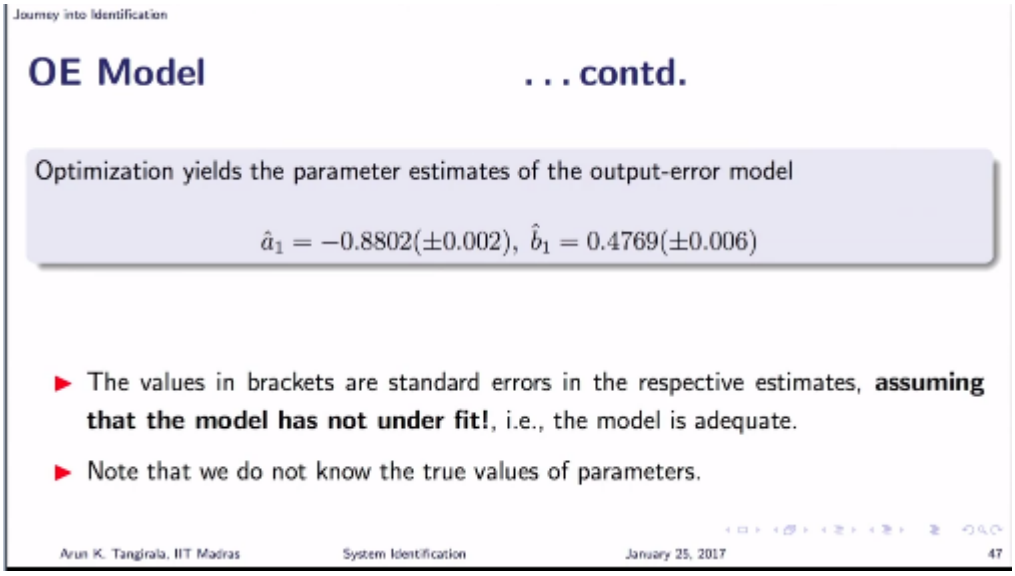
So without going into too many details it is kind of obvious that now y hat is nonlinear function of a and b. If I knew x, input is anyway known. If I knew x then y hat is a linear function of a, correct? But I do not know x. However, I know that x itself is a function of a and b and the input. Input you can leave aside and x is not a straightforward function of a and b, it's some kind of a complicated function, right? Therefore, I can quickly conclude that in this approach I end up with a non-linear, the predictor of being a non-linear function of a and b. What is that function? We learned later on. Don't worry about that. All right? It just suffice as to remember at this stage that whenever I pursue, whenever I assume an output error model structure, the predictor is a nonlinear function of the parameters. That's all. Is it sufficient to remember that. Now therefore, when I tried to estimate a and b by minimizing this optimization problem, I run into a nonlinear least squares problem. Do you know the difference

between a linear least squares problem and a nonlinear least squares problem? Does anyone know the difference? You should be knowing. No? Do you recognize that this is at least squares optimization problem? That part you recognize? Okay. So whenever this y hat which we call as prediction and maybe functional analysis, folks would call it as an approximation whatever, all words are technically equal. And approximation, prediction, estimation they're all equal. Whenever this approximation y hat or prediction y hat is a linear function of the decision variables. What are our decision variables? a1 and b1. Whenever it's a linear function then we have what is known as a linear least squares problem. And whenever it isn't that means whenever y hat is a non-linear function of the decision variables we have what is known as a nonlinear least square problem.

You may ask, why do we observe this distinction at all. Well, there is a very important reason, linear least squares problems have analytical solutions. That means, I can write the solution. I can derive the solution by hand. Whereas nonlinear least squares problems do not have an analytical solution. You need to use a numerical optimizer which will get to some local minimum. Obviously, what we want is a unique solution and a closed form expression so that I can code easily. Always nonlinear least squares problems involve iterative searches for the optimum. Linear least squares problem there is no iterative search. Obviously, a closed form expression and you just code that solution and that's your optimum. You don't have to do anything. Now you understand why we said in the second approach, we will require that y hat be a linear function. We'll come to that. At the moment, what a know is that they y hat in-- when I assume an output error model structure is a nonlinear function of the decision variables. Therefore, I will not have necessarily a unique solution. If they may exist but I will not be able to compute.

Anyway, keeping that in mind we will run the optimizer and that's what I have done. And here are the estimate. So what I have fed to the optimizer is, this measurements that I have ys and the inputs u, and the prediction expression. I have fed all of this to the optimizer, not exactly that way but I have done that implicitly. And the routine has gotten back with these estimates. Optimal estimates. Of course, in MATLAB society tool box. There is a command called OE, which estimates output error models for you. That's what I have used. So as far as the user interface is concerned I only supply the data and I supply the delay and I supply the number of parameters to be estimated. Which depends on what orders I have assumed. That's all I do. If you look at the script, I don't know how many of you have actually gone to my website and looked at the script. If you look at the script that is essentially what I would be doing, nothing more. So these are the optimal estimates of a1 and b1. Do you think they are correct?

(Refer Slide Time: 24:00)



Do you think that these estimates are correct in the sense they are right? They make sense?

Let me give you a minute to think about this. If you don't like my question, you say, you have asked me a weird question. And I'll go on my defence and I'll go back and I will tell you. Well, maybe it's not such a weird question after all. I'm asking these questions because I want to train you into thinking, into raising this question every time you fit a model. Doesn't matter in this course. Anytime you fit a model you have ask, does this model make sense? So if you were to ask this question for this model, what would be your answer? Is it a right question to ask? Okay, by the way you should have asked me what are those numbers in the brackets. They are standard errors. I will caution you not to look at them as of now. There is a reason why I don't want you to look at the errors. Look at the point estimates. Look at the estimate values themselves. First of all, you should ask in your mind, do I know the truth? Do you know the true values of a and b? We don't know. So there is nothing to compare. Okay. So that route is gone. It's closed. How else do I make a sense out of this model? Sorry.

Sign, very good. Look at the sign of the value and tell me. a should be positive, why? Not necessarily. Remember, if you recall in the last class, I had said magnitude of a1 should be less than 1. I've given a way a part of the answer. Is it? So stability is satisfied. That's the first thing you should check. Do you have a stable model for the stable system. You cannot have an unstable model for a stable system and a stable model for an unstable system. That compatibility has to be there, pretty much like you know, your horoscope matching. Seriously. What else? Okay. That's later stage, that will come. We will show you fits on training data. Right. There is one more thing that you can do. So what does this tell me if I were to write the equation, right. So remember, this is a1 and b1, right. So if I write the equation here, I get x k plus a1, xk minus 1. This is the question that we have been that we have postulated for g and

now I have estimates. There is something that we calculated from our non parametric analysis. Without assuming this model, we have calculated something. You can add many things delay but we have already used the delay information. g for gain, time content too, correct. You can use a time constraint as well as a gain, correct. We have calculated these other two pieces of information. Without assuming a model structure, at least, time constant, I assumed a first order dynamic so you can't really say, I didn't assume this but gain is immaterial. Right. That is why students worry about gain all the time. You're already smart. Anyway, so now with that hint can you check if this a1 and b1 makes sense? Is this a steady state equation or a dynamic equation? Steady state model or dynamic model?

You should not think so much, straightaway you should be able to say, it's a dynamic model because it tells me, it describes a transcend behaviour. Given a dynamic model, how do you drive the gain? How is gain defined as? Change in the output to the change in the input. Correct? Already, y and u are deviation quantities but change in the output to the change input at steady state. Correct? So let us say that what we have observed earlier, although when we estimated this step response. A step response was for x only. Right. What was the gain that we found for the system? Do you recall, roughly? Three point something. Correct? So if you go back, gain is roughly 3.7. How do you find out now given a1 hat and b1 hat. How do you find out the gain? What happens at steady state? xk minus 1, exactly. So you plug in that and tell me quickly, whether the gain agrees? At least, it'll not come to exactly 3.7 but it should not be too far. It shouldn't be like 1.2. Because non parametric analysis in that respect is quite reliable. What do you get for the gain? Around 4, okay. All right, so we'll now keep this in mind. Now we'll not look at the standard errors because the standard errors are calculated assuming that we are not under-fit that means, assuming that there is nothing left to be modeled. Only when we are convince that there is nothing left to be modeled we should come back and look at the standard errors. Very often people make this mistake they are not aware of this assumption. Straightaway, the software will compute the standard errors and give it you. The routine assumes that you're not under-fit, over fitting is, okay. The model shouldn't under-fit. What we mean by under-fit is, there should nothing left in the residuals for you to model as the function of the input or even as a function of its past. In other words, your residuals should be uncorrelated with the input and should also be white. They should have white noise like characteristics, which we have not yet check for this model. So let us not therefore, look at the standard errors. We'll come back and look after we're convince that this model has satisfied those two conditions. Let's turn to the other models. So we'll keep this in mind. This is candidate1.

(Refer Slide Time: 31:06)

Equation-Error Model

Predictable observation error with the additional **requirement of a linear predictor** for the **measurement**.

$$y[k] = -a_1 y[k-1] + b_1 u[k-1] + \overbrace{(v[k] + a_1 v[k-1])}^{w[k]}$$

$$\implies \hat{y}[k|k-1] = -a_1 y[k-1] + b_1 u[k-1] + \hat{w}[k|k-1]$$

In order to have a **linear predictor** for the measurement, we require

$$\hat{w}[k|k-1] = 0 \tag{6}$$

meaning $w[k]$ has white-noise like properties.