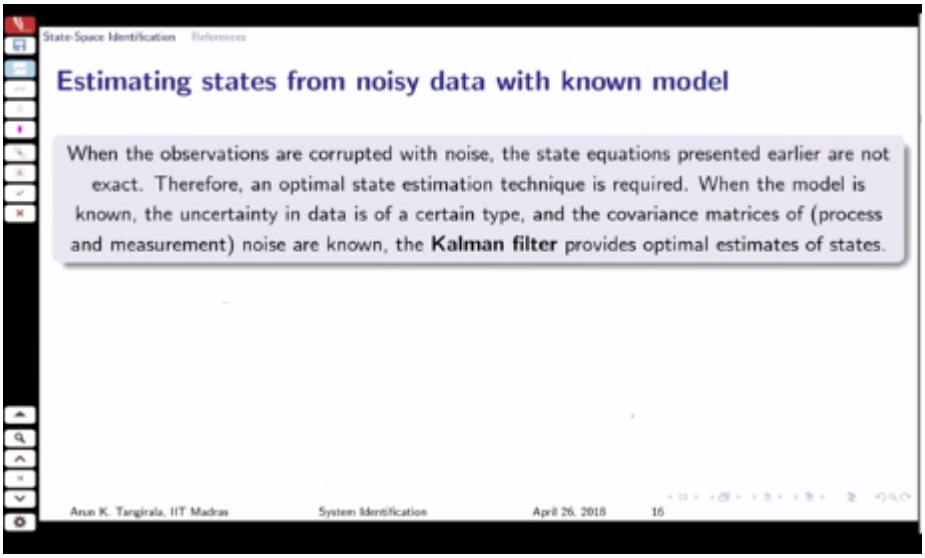
CH5230: System Identification

State-Space/Subspace Identification

Part 3

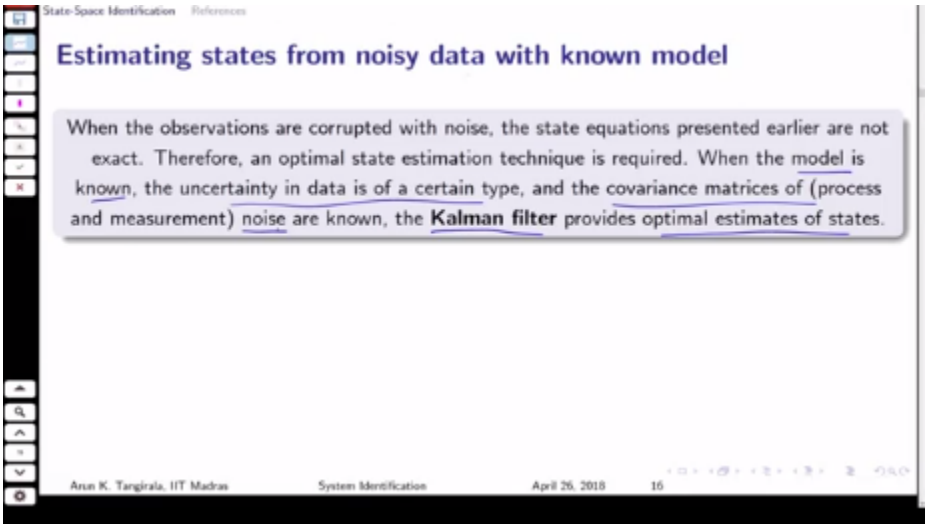
(Refer Slide Time: 00:13)



So, we talked about estimating states from noise free data, now what about estimating states from noisy data with known model. This was the subject of interest in you know in mid 40s, 50s and so on. And ultimately the work culminated into the celebrated Kalman filter. The Kalman filter provides optimal estimates of states from noisy data. Because, now you know, the state equations are not exact. You remember the state equations we had written by y[k] y[k]plus one and so on. We had nicely written exact equations under noise free conditions. But now I cannot afford to do that anymore. Now, I need to turn to an optimal way of estimating. So some kind of approximation has to be done. And I have to get optimal approximations of the states for those equations. And obviously, I will not set up only n equations now.

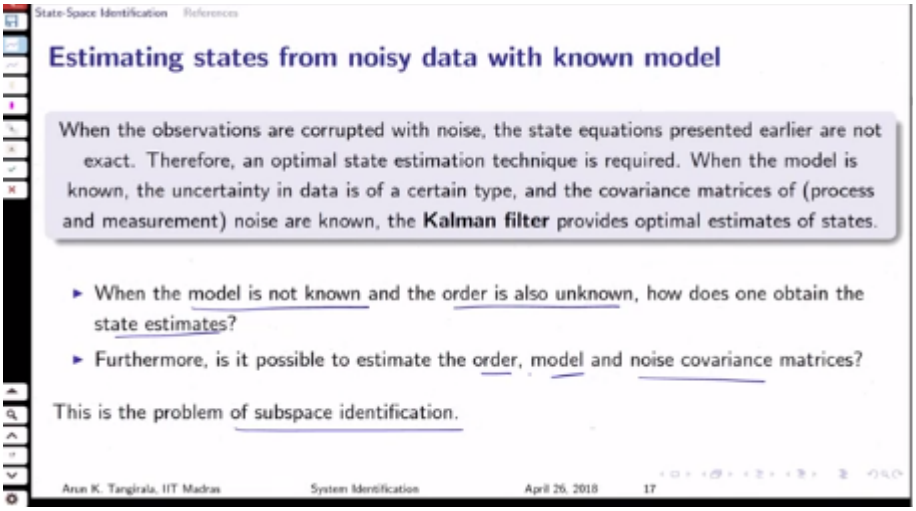
I would like to use as much data as possible to obtain better and better state estimates. So, the Kalman filter assumes that the model is known that's an important thing. And that the uncertainty data is of a certain type namely, Gaussian distribution, and that the noise is stationary and the covariance matrices of the process and measurement noises are also known. And as we will understand through the motivational example, the Kalman filter is nothing, but a Bayesian estimator. There is a prior, there is a posterior. So the Kalman filter is a minimum mean square error estimator of the states.

(Refer Slide Time: 01: 56)



Like the Bayesian estimate. We'll talk about that a bit later. Now just out of curiosity, we want to ask when the model is not known. And order is also unknown. How does one obtain the state estimates? That's the first question we have. And, is it possible to also simultaneously estimate the order, model, and noise covariance matrices. Essentially, this is a problem of subspace ID. Right? We are generally greedy. First we said, "Oh, model, order, all of this is unknown can obtain state estimates." And then I step up my wish and I asked, "Oh, Lord, can I also actually estimate, order, model, and noise covariance matrices?" If the Lord is still in front of you, then you can. If the Lord disappears, then that means you're asking for something not possible, okay? Fortunately, in this case, the Lord will still stay and grant your wish, okay? And that wishes in the form of subspace ID algorithms. But with some limitations, terms and conditions apply.

(Refer Slide Time: 03: 04) Right?



And not in fine print, like many businesses do. In big print only, there is no control on the structure of the model that you will get. Which means you will get a model and estimates of the states in that model description. If you want the model and the state instruments of that model, then you return to constrain state space identification algorithms. Okay.

So, what are the basic ideas here? How do you proceed in the in that case, you will have to construct what is known as an extended observability matrix now. We'll talk about that a bit later, because the extent that you will have to wait for some time, okay? You cannot learn it immediately. And you can show that under noise free conditions is extended observability matrix, the rank of it, it gives you the order, that is the key result on which all the subspace ID algorithms rely. And under noise free conditions, you get only an approximate order. For those of you are familiar with principal component analysis and determining the number of principal components, for a linear data generating process and the noise free conditions the PCA or SVD gives you exact order. That is exact number of principal components. You can say that it gives you the exact number of linear relations. But when you have noisy data, the standard PCA gives you only a rough order you have to use your Heuristics and that can vary from geography to geographical region on this planet. Okay?

That such is the case with subspace ID as well as of now, in 2018, in the month of April. There exists no algorithm, at least in the subspace ID. There have been many attempts but no formal way of identifying the exact order from noisy data in the subspace ID literature. However, in the PCA literature are exists and currently we are exploring now. So we know that iterative PCA gets you the exact order, even from noisy data and principal component analysis literature. So we are trying to now

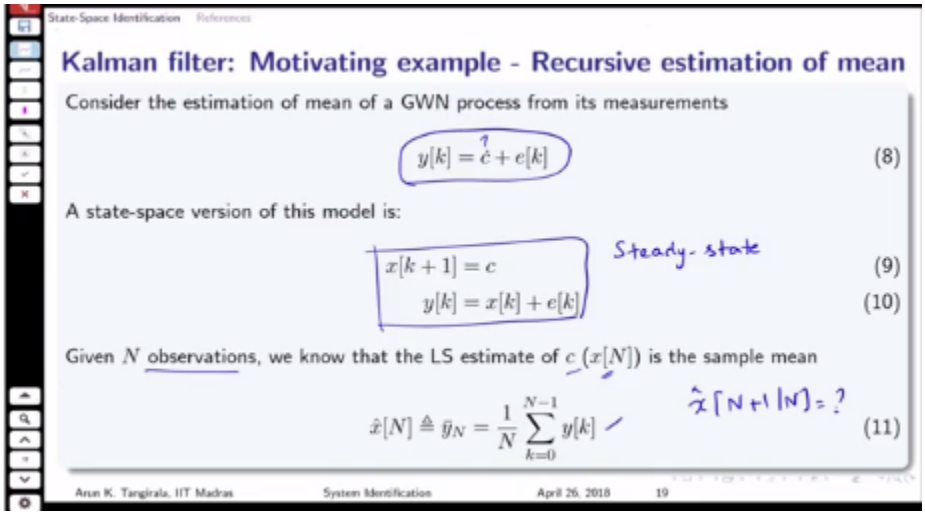
borrow ideas, to see if we can also do the same thing in subspace ID. And I think will be successful pretty soon. So, the answers to the previous question is, yes, it is possible to estimate the order through the construction of an extended observability matrix, under noise free conditions you will get exact order, under noisy conditions you will obtain only approximate order and from that approximate order, with that approximate order you can start low rank approximations of these external observability matrix and from where you can derive the state estimates, the model, everything, which is what we will elaborate on a bit later.

For now, let us understand how do we estimate states from noisy data given a model? Because you can think of it the this problem, you can think of this problem in two parts. Suppose, I give you the model, how do you estimate the states optimally, that is through Kalman filter. Then the only thing that is remaining is to figure out what the model is. So you can think of it in two parts. Although subspace ID algorithm does not necessarily work along these ideas exactly, but implicitly that is what is happening. You're breaking up the problem in two parts and saying, I'll identify the model first and estimate the states or I'll identify the states, estimate the states and get the model and so on.

So let us learn quickly, given a model and when I say model A, B, C, D, and noise covariance matrices and measurements, how does one optimally estimate the states? And we have already said, it is through the Kalman filter under the conditions dictated by the by Kalman in his work. There are, of course, variants of Kalman filter, as you know very well, for any method there are going to be variants. So let's look at a motivating example. Normally, many beginners' think Kalman filter is some mysterious thing that came down from Mars and only a very few can understand it. It very much belongs to this planet. And it can be understood by every individual on this planet, regardless of the geographical region. So let's start with a very simple example. This is our familiar example. You may think that I abuse this example quite a bit, but in fact, I make use of it a lot.

We have seen this in different contexts and we are seeing this again in the context of Kalman filter. What we'll do now is we'll write estate space version of this model. So until now, we have maintained this model in equation eight, but now we have written a state space version of this. What is the problem here? I'm given measurements of y, and I'm supposed to estimate the C. Right, I'm given n observations of y. Instead of working with this so called input-output model or output model. We are now going to work with the state space model where the state is nothing, but the unknown C. That is hidden from me and it's a very nice, special case of a state- space model where the states do not change dynamically. In fact, what do you call this model? It's called a steady state, state-space model. That's why we have the term steady. The states have reached the steady state, the measurements are not. Right? That's why we use the term steady state. So it's a steady state-space model and now, we know that when n observations are given the least squares estimate of this C that means, let us say of X n, you can X n which is the nth, X n or X n plus 1 all of them are C only. So, the least squares estimate of the state X n is nothing but the sample mean. That we have derived before. Correct. Now, let us say that I want to now, I want to estimate what is a state at the next instant. The truth is as fixed, but my estimates can keep changing. You have to remember that regardless of the time at which I'm standing, as far as this process is concerned, the true value of the state is C only. But my estimate of that state can keep changing as I progress as a march in time. If I have marched in steps in time, the state estimate is given by this. The least squares estimate. Now, suppose, I'm going to match I had to n plus one. I want to ask myself what would be the estimate at n plus one? And given the estimate at n, what would be the estimated n plus 1? In this case, they estimate at n plus 1 will be, n plus 1 data is not given, I'm standing still at n, so I'm asking what is x hat of n plus 1 given n. I'm predicting what the state will be. Let us not call it as an estimate. Let us call it as a prediction.

(Refer Slide Time: 10: 23)

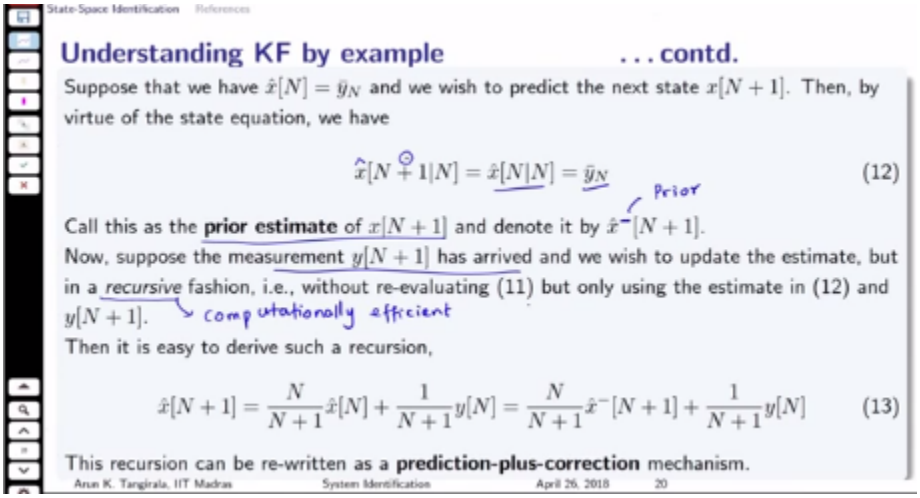


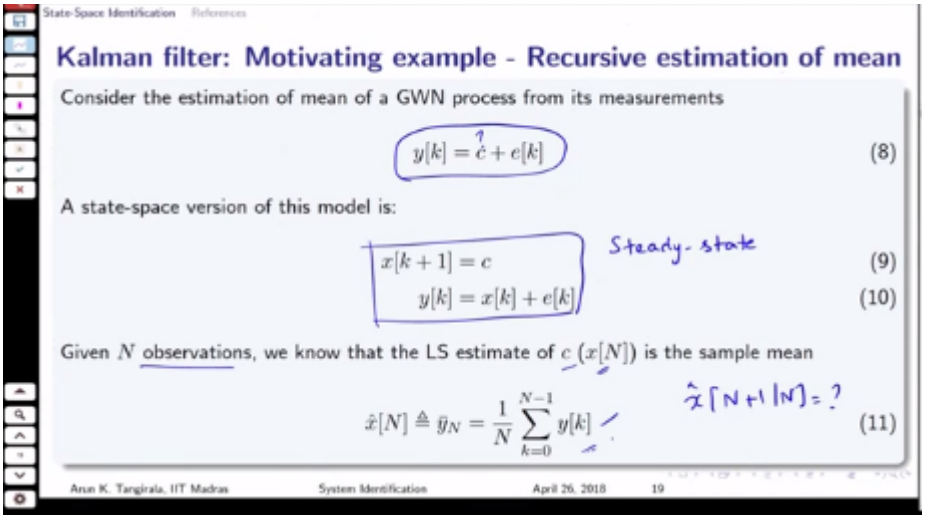
I'm given the state at the nth observation, estimate of the state and want to predict what is the state, at the next instant. What is a prediction? C itself, but remember, we can't, we don't know C, so we'll have to work with estimate of C that is available at n. So we say that x hat of, this should be x hat. X hat of n plus 1 given n, the hat is here. It really was being very naughty, so it's supposed to be here. It's my mistake, obviously. X hat of n plus 1 given n, in nothing but the state that I have in this case, it is x hat of n given n. Whatever estimate I have at Nth observation is the prediction of the state at the next instance, which happens to be y n bar. Now we will call this as a prior estimate of x n plus 1. That means, prior meaning, what?

Why do I say prior with respect to what? Prior with respect to the data that is going to arrive at n plus 1. So, before the n plus one measurement arrives, my prediction of the state at n plus 1 is C -- C y n bar. After now, and we'll denote this as x hat minus. This is usually one of the standard conventions followed in the Kalman or state estimation literature. So this minus here indicates, this is a prior estimate. Before the n plus 1th data point has arrived. Now suppose a measurement y n plus 1 has a right, okay? With a big trumpet sound and so on, and then we wish to update the estimate, obviously, right? Because I may be wrong, I've only predicted what a state estimate will be.

But I want to do this in a recursive fashion. I also want to be computationally efficient. So, this recursive essentially should be read as computationally efficient. Always. Almost always it means that. That is, we don't want to re-evaluate 11. See, ideally what is the state estimate, if I'm given the N plus 1th observation, I will pool all the N plus 1 observations and simply re-evaluate this formula.

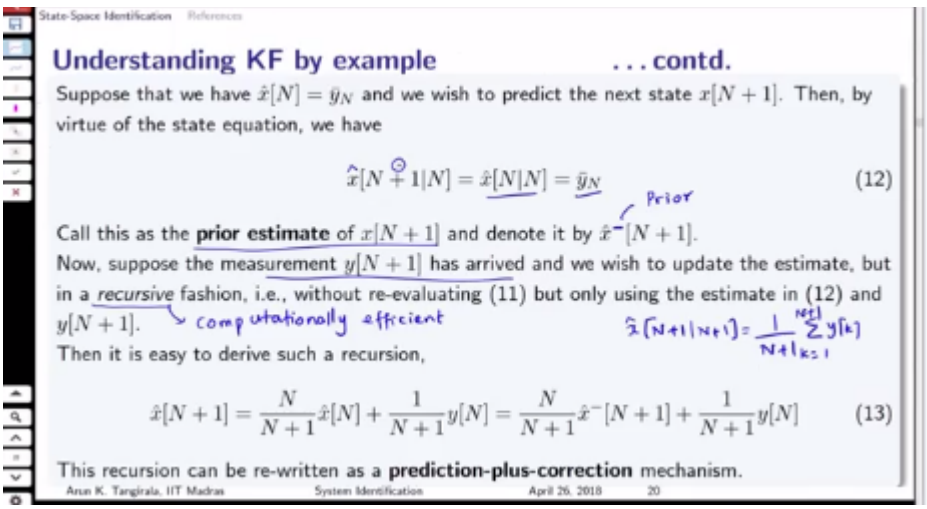
(Refer Slide Time: 12: 37)





So, one way to simply update the state estimate is to say, oh, x hat of n plus 1 given all the information up to n plus 1 is simply 1 over n plus 1 sigma y[k], k running from 1 to n plus 1. Right? That is a Brute-force way. That is natural. But we don't want to do it this way. And this may be okay for, this approach may be okay for this example.

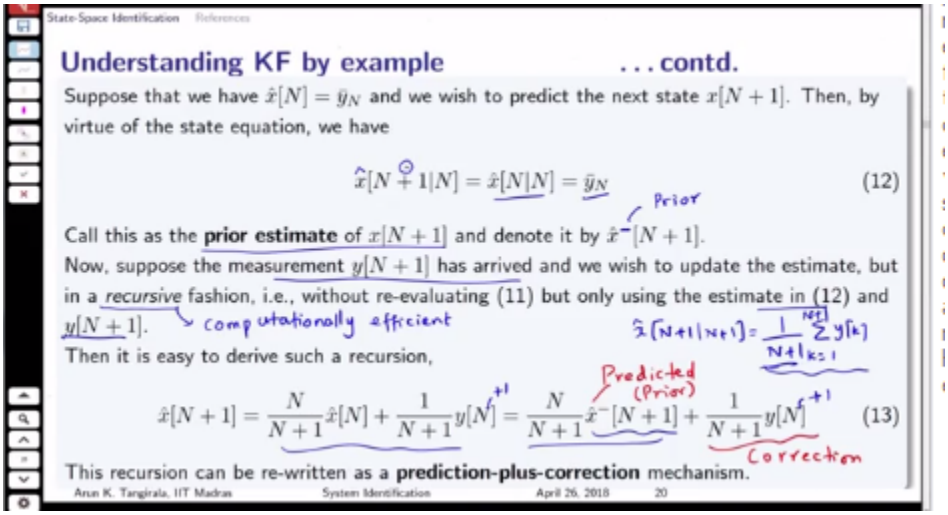
(Refer Slide Time: 13:10)



But in general, re-evaluating the estimation all over is not such a great idea. So, we want to do it in a recursive way that means we want to use already the estimate that I have from N observations and this new measurement, that's all. So how do I do that? I split this equation into two parts, right? One summation running up to N and the other having, sorry, this should be y N plus 1. Unless my summation is running, because my summation runs on 0 to n minus 1. I'm going to correct this to 1 to N, so that becomes easy for you to understand. So I have here because I'm saying y N plus 1 here in my text, I don't want you to have confusions.

So this is y N plus 1, there may be also similar mistakes in the textbook. All I had done is I split the summation into two parts. One summation running from k equals 1 to N. And the other one containing y N plus 1 and there is this 1 over n plus 1. And remember, the summation running from 1 to N can be replaced by N times x hat of N. Essentially, N, y bar of N and that is what exactly I've done. This is standard trick in recursive estimation, and we'll replace x hat of N with this notation. So, we have now a nice expression here. X hat of n plus 1, given n plus 1, that means, now, my updated estimate is a sum of two things. A predicted estimate, this is nothing, but the predicted one, right. This is predicted. Prior, you can say, this x hat of 1, sorry, is the predicted one. Or you can say prior. And then you have a correction, this is your correction. So, it is like I predict in the match that we had yesterday, there was they were predictions being made.

(Refer Slide Time: 15:25)

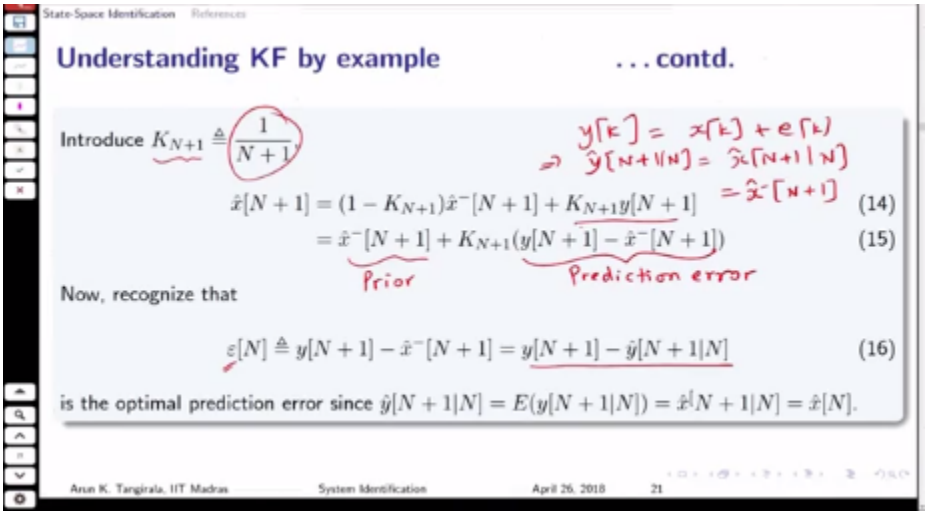


Somewhere in the middle, CSK would lose against Royal Challengers. Okay, then you keep making corrections with the arrival of every score after every ball, you're making your corrections, right? So, there's a prediction followed by correction. But here it's being done in an optimal way. And that is a central idea in Kalman Filter. So, you have, if you introduce this kn plus 1, which is nothing but 1 over n plus 1, I can write the previous equation that I have x hat of n plus 1 as 1 minus kn plus 1 times x hat minus at n plus 1 plus kn plus 1 times yn plus 1. And I simply rewrite it this way.
So, why do I rewrite this way? Because now I see very clearly, this is the prior. And what is this term yn plus 1 minus x hat minus n plus 1, that is nothing but yn plus 1 minus y hat of N… This x hat of, x hat minus of n plus 1 is prior estimate agreed, but it is also the prediction of what the measurement will be. Because remember, our measurement equation is y[k] is x[k] plus e[k].

So, that implies that y hat of n plus 1, given N is x hat of n plus 1, given N, which is nothing but x hat minus. So in other words, this is nothing but the prediction error. So, this is the prediction error. So, it's a nice control. Again you see the duality between control and estimation. What do you do in

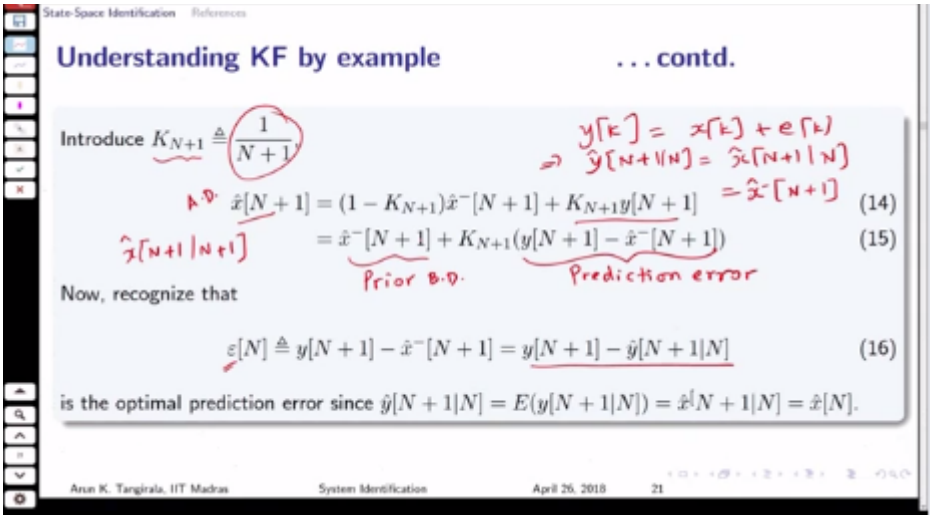
control? You make a move and then you see the response of the process, and you see where the process is, take the measurement, feed it back and compare it to the set point and then make a change.

(Refer Slide Time: 17:29)



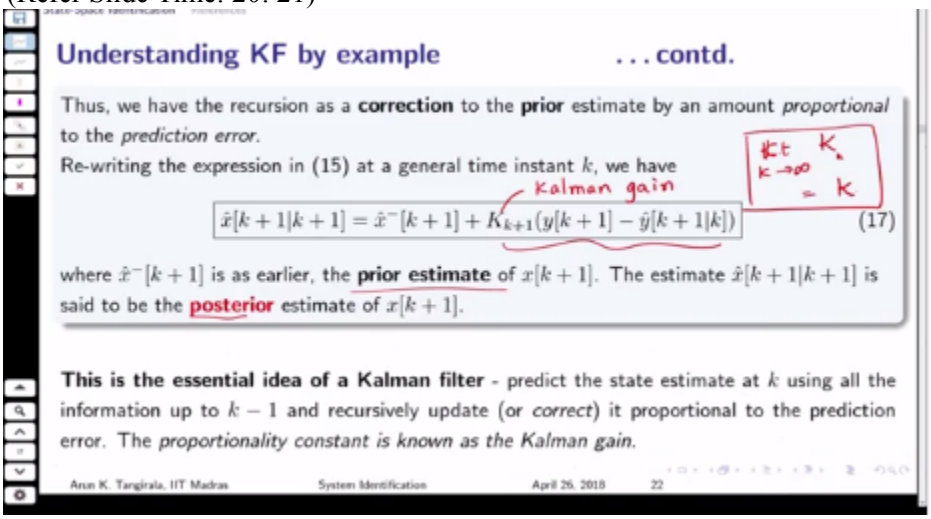
Here, you have a similar situation, you're making a prediction before n plus 1 arrives, after n plus 1 arrives, you look at the difference between the prediction and what the measurement is and make a correction to your state. So, look at the notation. This is x hat of n plus 1, this is x hat minus n plus 1. What is the difference? X hat of n plus 1 is a filtered estimate. It is actually x hat of n plus 1, given n plus 1. It's called a filtered estimator. Remember, we talked about prediction filtering and smoothing. Whereas, this is a prediction or a prior, this is before n plus 1 arrives. So, you can say, this is BD, this is AD. What is BD? Before data. AD is after data. Doesn't this have a Bayesian flavor to it? I have and remember, this estimate is not accurate, x hat minus of n plus 1, is this prediction is not accurate. It will have some error. That error, I mean, therefore, this random variable, it has its own P.D.F. and so on. So that is a central idea in Kalman Filter. What Kalman proposed is two things. He said, of course, "I want optimality, but I also want computational efficiency." And both of these are contained in the Kalman Filter.

(Refer Slide Time: 18:47)



And, in general, if you replace N by k, x hat of k plus 1, given k plus 1 is this or it could be x hat of k given k and then everything else is replaced accordingly. So, the prior estimate is corrected to a posterior by a correction. What is this k here, this is called the Kalman gain. And Kalman showed, how to derive this Kalman gain, given A, B, C, D, and the noise covariance matrices. How do you optimally arrive at this? And if you implement this Kalman filter for a very long time, then you can say at K tends to infinity, unfortunately, I have the same notation for instant and Kalman. But, please think like a computer now. There is a difference between lowercase and uppercase. So, k subscript k becomes a constant for LTI systems, the Kalman gain studies out. Okay, then you have this, Kk plus 1 being replaced by some k, you can think of that as a proportional control of being implement. Okay. Of course, you can ask question, is it proportional or proportional integral and so on? We can ask those questions later on. But the fact is, is Kalman gain studies out for certain class of processes? And all you're doing is a correction. And what does a Kalman gain denote?

(Refer Slide Time: 20: 21)

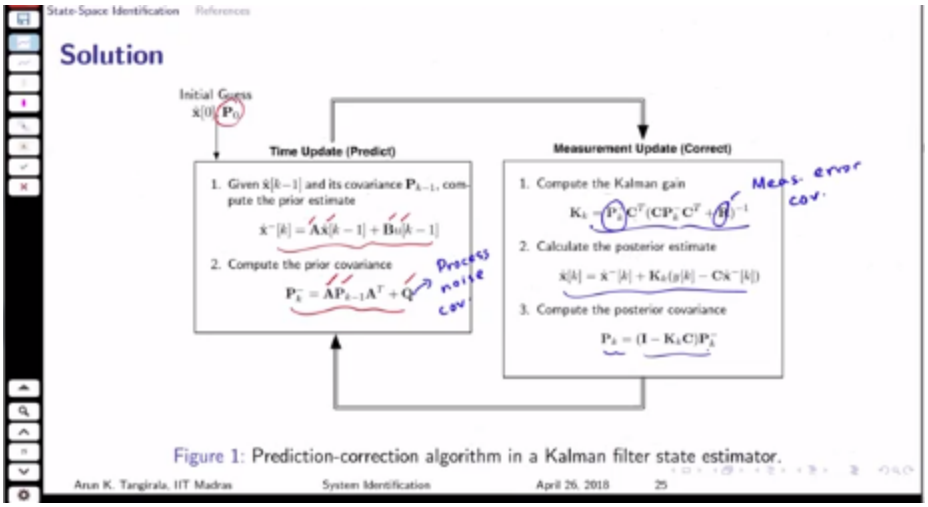


How much importance you're placing to the error in your prediction and intuitively, suppose your measurement had lot of noise. That means yk plus 1 came with a lot of noise or yn plus 1 came with a lot of noise. Will you give more importance to the prior or to the correction value? Which one? If the measurement is coming with a lot of error then you won't really give so much importance to the measurement because, you know, that my collection has to be now done very carefully. So, Kalman gain will be very small. I'm not giving t he expressions here, but I'll use expressions later on. Will be quite small when the measurement error is high. And, is a measurement error is low, so in fact it is relative. It is not just a measurement error. If the measurement error is much more than the error in your prior, that is the correct way of looking at it. It's a relative. If I trust my prior more than the measurement, then Kalman gain will be low. Right?

Sorry. If I trust my prior more than the measurement, then the Kalman gain will be low. If I trust my measurement much more than the predicted or the prior value, then Kalman gain will be high. That is the idea. So this is a problem statement. But I just want to close this lecture with this schematic here for the Kalman Filter. So, you start your initial guess, I've already stated what the problem is. I mean, this is only a formal statement. And that you're given A, B, C, D, Q and R, and you're given data and that there is a prediction-correction step. So, you start with some initial guess and that initial guess, and that initial guess will have some error. You also have to specify the covariance matrix associated with that estimate x hat of zero. So, you kick start the algorithm, and at any step, you first construct the prior. This is the prior. And you also compute the error in the prior. By the way, sorry, this is how you do it. Remember, we know A, we know P, K minus 1, we know Q, therefore, I can update the

prior. Here also, at any stage, I know the previous estimate, I know A, I know B and input. So, I can calculate the prior. And then when I go to the correction measurement update, then first I compute the Kalman gain. How much importance should be attached, and you can see the factors that go into Kalman gain. There is this matrix. See, but more importantly there is P, k minus. What is P, k minus? The error in the prior estimate. It's a measure of the error in the prior estimate. It's a covariance matrix, right. And what is R? Q and R, Q is the covariance matrix, our R1, you can say, and R is R2. So, Q is nothing, but your process noise covariance matrix. And R is measurement error covariance matrix. R and P, k minus will dictate for a given C, what the Kalman gain should be. Right? If R is very high relative to C. Sorry, relative to P, k minus, then the Kalman gain will be low. And then other situations as well, and then you correct the posterior and you also update the error. See, the error prior to data was this, the error post data is this. That is and then you continue. There is an example in the textbook that I would like you to refer.

(Refer Slide Time: 24:11)



So, what we will do in the next lecture is, there are some remarks that we'll go through very quickly in the next lecture. Then we will derive the innovations form and start off with the foundations of Subspace ID, where we will first talk of extended observability matrix and then learn the Ho and Kalman's method. And then move on to the Subspace ID method.