# CH5230: System Identification

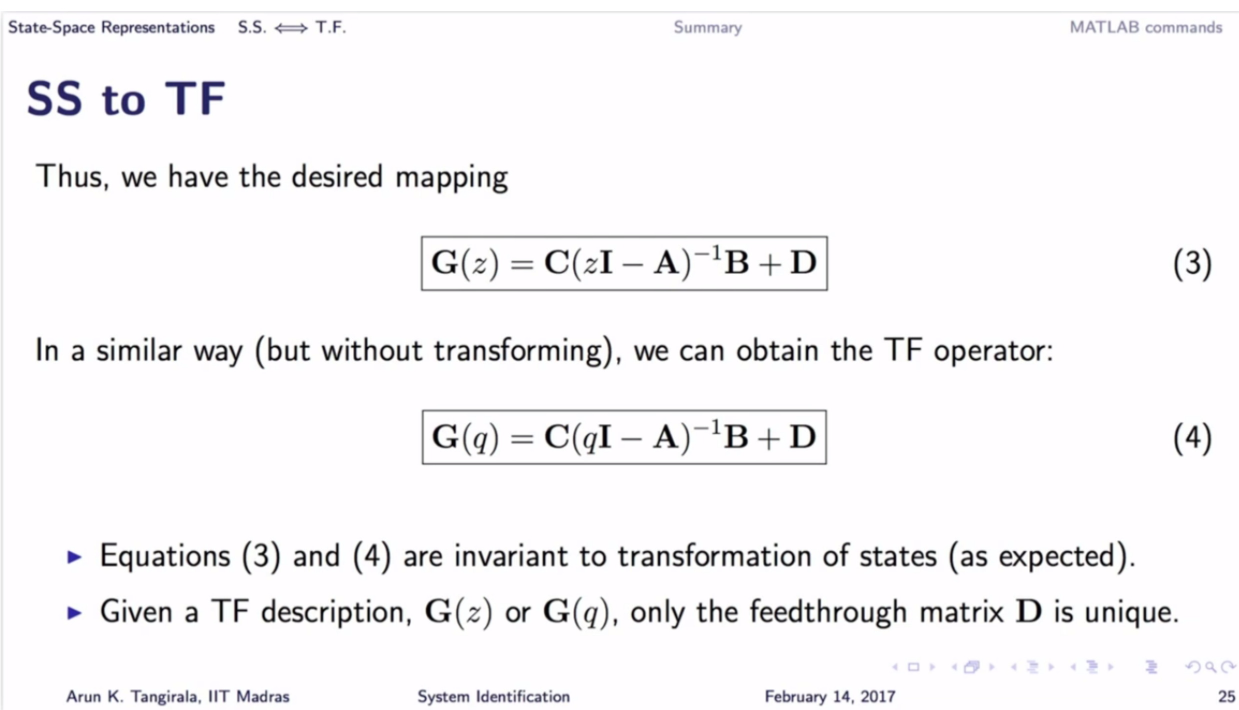
## State Space Representation 4

Very good morning. Okay. So there are only two things that we want to talk about and then we'll close the curtains on state-space models for now. One is to look at the stability state, the stability condition in terms of the state-space model. Earlier we have learned to give stability conditions in terms of impulsive response and then in terms of poles of G. Now it's time to talk of stability in terms of state-space model. And the other thing that we need to look at is, when you have delays I promised to talk about it to show you how delays increase the number of states when you write in the standard state-space form. Then we'll conclude the discussion with why and when we generally look down to state-space models instead of transfer function models.

So let's look at, this as a very simple concept here. The main result to remember is polls of G of z are the eigenvalues of the state-space matrix A. I have gotten myself back to single input-single output system. That is why you see G no longer being bold. Only they say that you become bold when you are together, when you're in unity. Right? When you have many people then you can do all the dadagiri that you want. That's why you become bold, here it is single input-single output, poor thing doesn't have enough strength so it loses its boldness. That is the idea behind the notation also. That is only my theory. Okay. Don't get carried away with that.

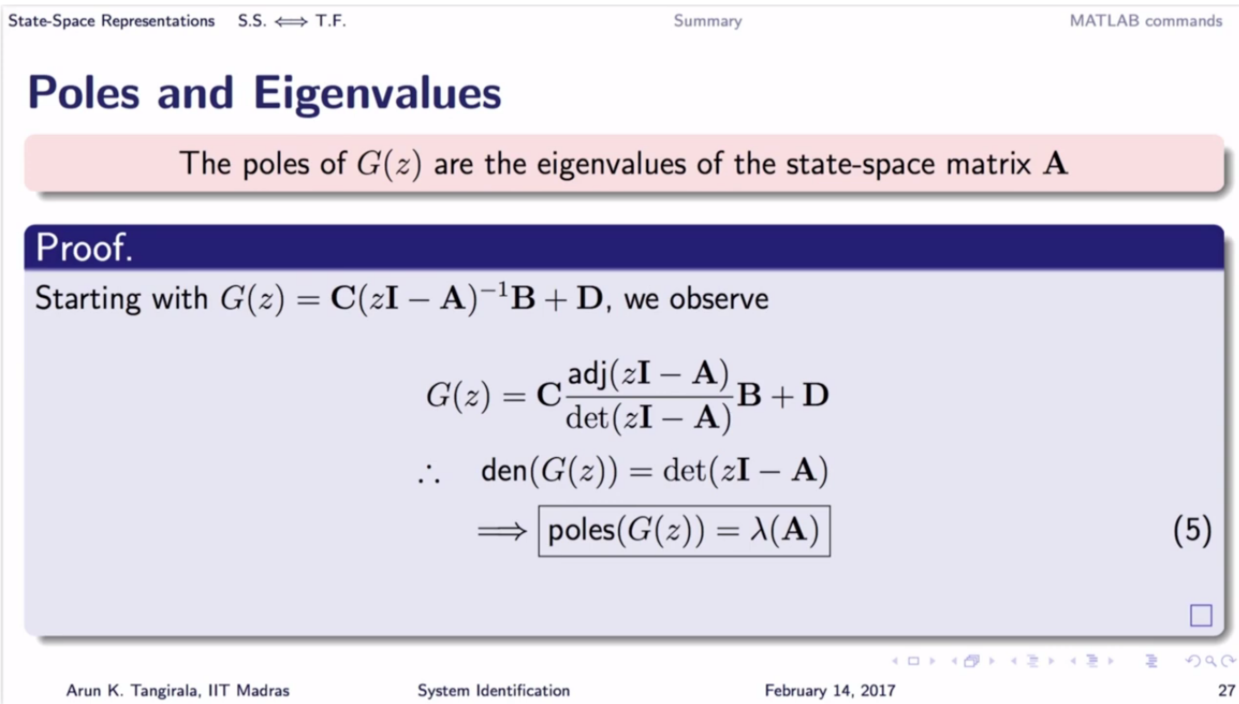
Anyway poles of G are the eigenvalues of state-space matrix. That's fairly easy to see, if you go back to this relation here. Right? You have a z I minus A inverse appearing when you go from state to transfer function.

(Refer Slide Time 01:54)



How do you write inverse of z I minus A, adjoint of z I minus a by determinant. Determinant of z I minus A is nothing but the characteristic equation. And that means and also this tells me that the denominator of G of z is going to be the determinant of z I minus A. Right? That is the basic point here.

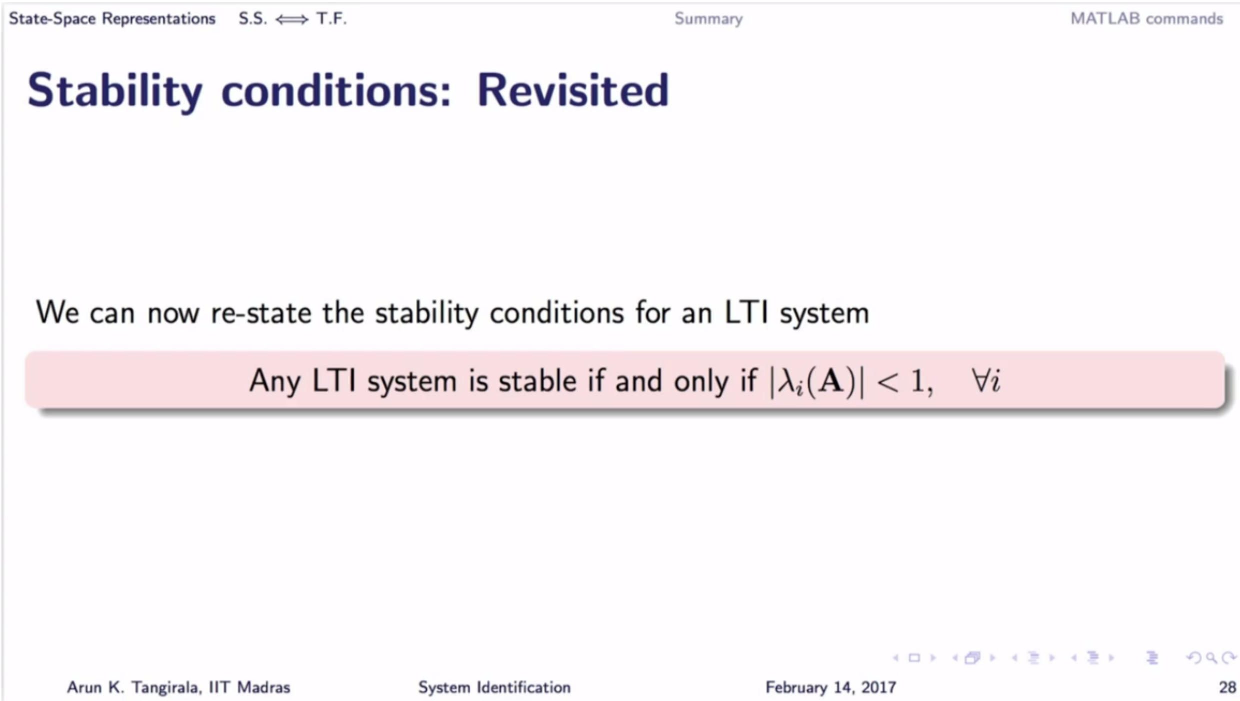
(Refer Slide Time: 2:21)




Which means the characteristic equation of G is the same as the characteristic equation of A. It has to be. We have learned this in many different contexts. Now we are learning it in the context of linear systems. That's all. So the characteristic equation of G is the same as a characteristic equation of A. And we say the roots of characteristic equation of G are poles and from matrix algebra theory we say the roots of characteristic equation are the eigenvalues. So it's just different terminology but otherwise there are one and the same.

Straight away therefore we can state the stability condition as, asymptotic stability condition that any LTI system is asymptotically stable. Although, I don't say asymptotically here. The reason why I don't see the similar article here is, you know, for all practical purposes, although we say, a system can be BIBO stable even if it is not asymptotically stable and so on.

(Refer Slide Time: 3:26)




When can a system be BIBO stable and not asymptotically when then unstable polls are cancelled by unstable zeros but that's just academic. It only happens in classrooms, on the computer screens. That's it. Practically you can't have such an exact cancellation. So, for all practical purposes and an unstable system from an asymptotic viewpoint is unstable. That's it. Don't expect nice zero cancellations and then to construct a BIBO stable system even though it's asymptotically unstable. Even the slightest uncertainty error in your design of the system can lead to the, I mean, violation of the cancellation of zero and pole.

So practically we'll see any system that's asymptotically unstable, he has unstable, itself. So any LTI system is stable if and only if all the eigenvalues of the matrix A are strictly within the unit circle. Same story. You have a similar result in continuous time systems as well. Any continuous time LTI system is stable if and only if the eigenvalues of the continuous time state-space model are all in the left half plain. Same story. Okay. No that's all. So there is not much to discuss there. Once we have established a mapping that is still the story is the same. The second aspect as I said is to learn how to look at stage-space models of systems which have delays. First thing to remember is we have already said every state equation has a built in unit delay it. Right?

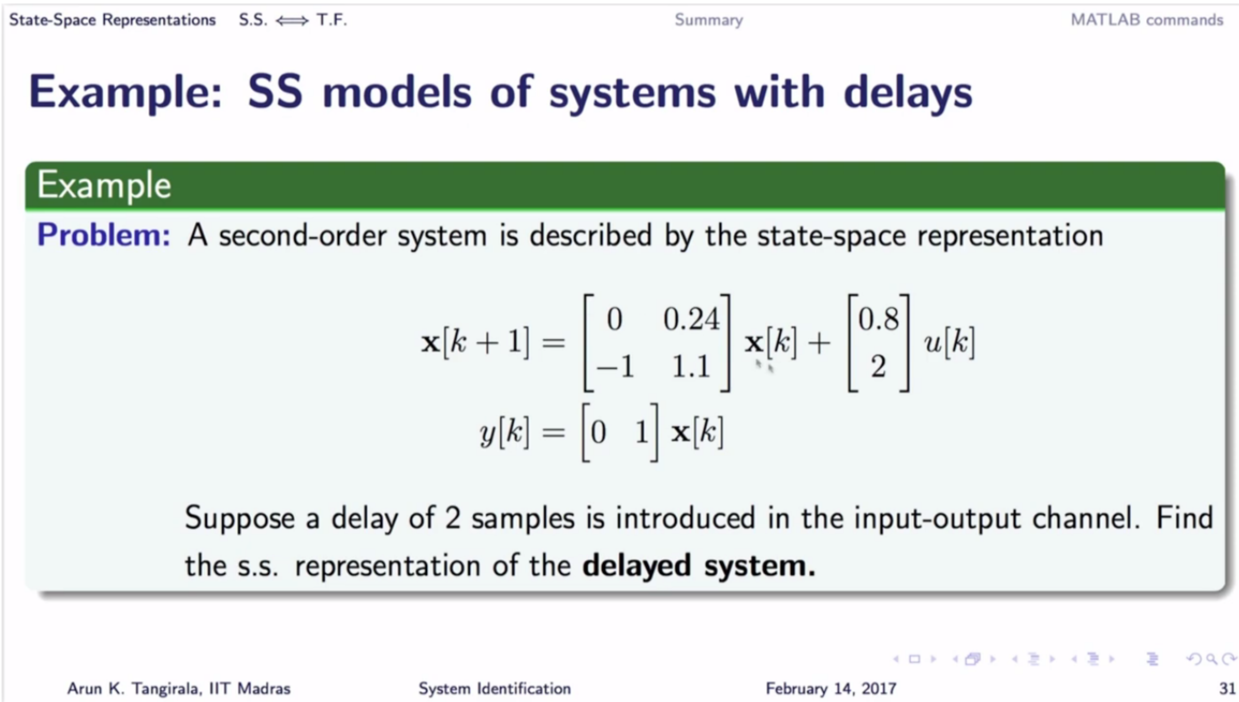
So if there is a unit delay in the system then I don't have to worry whatever stage-space model I get is a state-space model. But if there are delays more then what your standard state-space model has for that system. Then you will have to introduce artificial states to accommodate those excess delays. I'll show you an example. All right. But before we do that from a transfer function viewpoint, if G of z is a transfer function of a system in D additional delays is going to cost result in a multiplicative factor of z to the minus N D, we know that. Right? Or q to the minus N D.

So you should expect N D additional poles are the origin. Provided you have written G of z correctly. Remember we went through an example a couple of days ago where we had a transfer function which is second order and two delays second order from a systems perspective and two delays from the input side. But when you wrote it in the form of G of z, there we saw that the system turn out to be third order. Why did it turn out to be third order? Because there was an additional delay on top of the unit delay that you had and therefore it turned out to be a third order system.

So that was the question that one of the students asked yesterday after the class, know what is ordered here. Should I say that the system is second order or third order? And it was a very good question and the answer to that is there is no unique way of stating the order per say, unless you state that delay also together. If you're looking at it from an input-output perspective with respect to the example that we saw, for that example you would say that the system is second order with two input delays, input-output delay, which is a perfect way of describing it because the order that you're talking about is the order of the characteristic equation. That is off what you will write as a free response.

But when I turned to state-space representation, I would say it's the third order system. That's it. I don't have to mention any delay because we know that there is a unit delay built into the state-space model. I don't have to mention anything because the state space from standard. All I have to say is third order which means there are three states that are required to describe the full system. What is a full system? The input delay plus the system. Let us let us look at an example here as to what happens here.

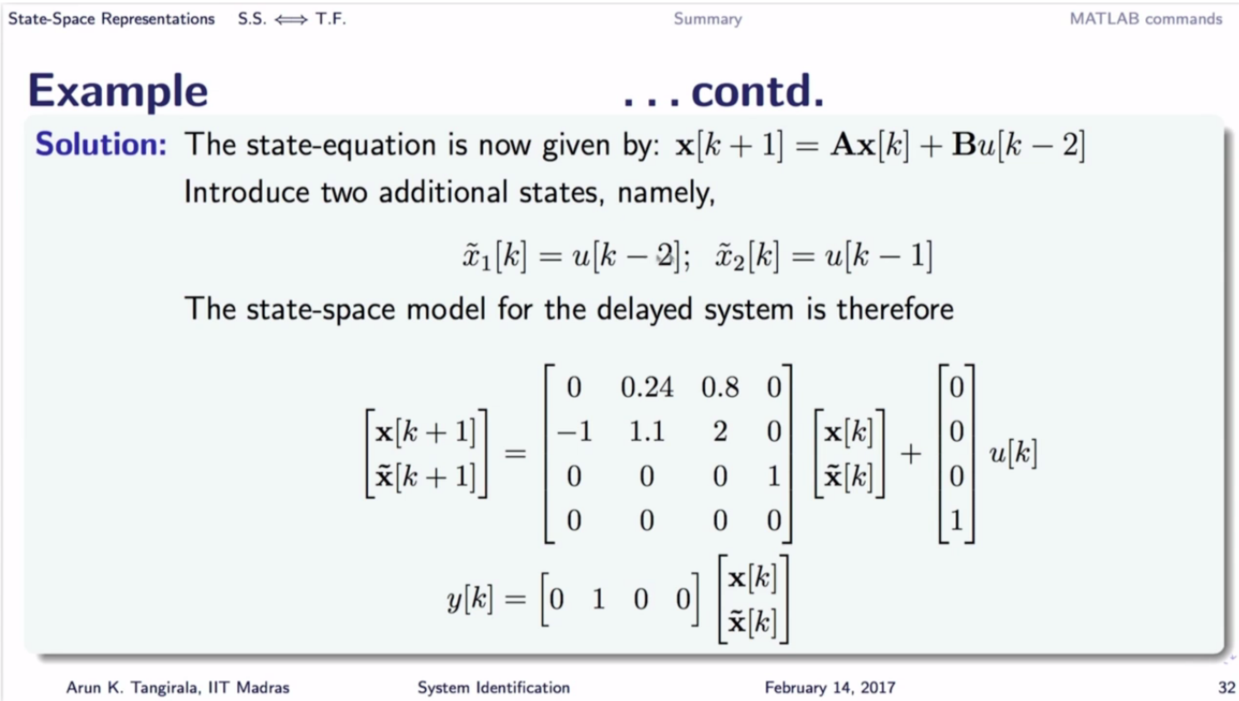
So we have this example. Suppose a delay of two samples is introduced in this system. So this is the system that we saw earlier. Right? We had this conversion but except that the coefficients of B are different.
(Refer Slide Time: 08:18)




So let us take this sample second order system. Now we want to ask if the if there is an additional delay of two samples on top of this, how do a rewrite the state-space model? See there is an additional delay. From a layman's viewpoint all you would do is in place of u k you would have u k minus 2. That's all. But that is not in the standard state-space form. Right. In order to bring that to the standard state-space form we will have to introduce two additional delays.

Okay? Now let's look at that. Here the state equation is now given by this, but this is not in the standard state-space form. And therefore I have to introduce two new states u k minus 2 and u k minus 1. I mean corresponding to u k minus 1 and u k minus 2 I introduce two additional states which denote by a tilde there. Okay?

(Refer Slide Time: 09:89)




So here. Sorry I denote by bar, the state-space model for the delayed system now becomes this. How do I arrive at this? Well, I know that x1 bar k is u k minus 2. Right? So if you look at the state's equation here I have x k plus 1 as A x k plus B u k minus 2 in place of u k minus 2, I would have x1 bar k. one basket. And that is why you would see in our x k plus 1 there you would see an x bar here. So you see here. B u k minus 2. Right? x k plus 1 is a x k plus B u k minus 2 and. Sorry x1 tilde. Here earlier I said bar, I'm sorry. So x k plus 1 is 0.24, 0.8, A matrix remains unchanged. All right? But what you have here is a 1 for the first state of x tilde.

x tilde is a vector, x is a vector. x is your default vector here, that we have already. x tilde is additional states that you have introduced. Right? So when write the state equation here I would have x k plus 1 equals A x k plus B x1 tilde k. And that is why you see a 1 here. Sorry. This part of the matrix remains unchanged. Now you have additional rows in A. Why do we have additional rows in A. Because what we are doing now is, we have introduced two additional states and I have to write state equation for those two additional states as well.

Put together now, my system is described in terms of x and x tilde. And it is my responsibility to write state equations for each of these. So I have x k plus 1, I have x tilde k plus 1. The state equations for x k are already given that come straight away from the problem statement. But we want to rewrite this in terms of the new states that we have introduced. And as I just said, because u k minus 2 is x1 tilde k, we get here x k plus 1, A x k plus B x1 tilde k.

So now how do I write this. Remember, here the way you write this is, you say, well, I have B and then I can also say 0 because ultimately the goal is to x plus x k plus 1 in terms of not only x k but also x tilde k. So this is x tilde k. Right? So you see how the first two rows of A have been modified. Here you have the 0, 0.24, minus 1, 1.1 comes from you're A. And then where does this come from, B. So, if you look at B it's 0.82. But then there are zeros. Remember, because you're going to have to have to zeros. So, the zero is a vector of zeros.
Do you understand?

Now, when you go to x tilde k plus 1, x tilde k plus 1 doesn't depend on any of the other states. It only they just share a shift relation and that is why if you look at x 1 tilde k the state equation for that would be u k minus 1 which is nothing but x 2 tilde k. That's why you have one here. And finally, what would be the state equation for x 2 tilde? Simply u k, which means the rows of A corresponding to x 2 tilde k plus will be all zero and you will see a 1 in the corresponding row of B.

So all we have done is taken that u k minus 2 and absorb this into the matrix A. Why? Because I want to write it in a standard form. That is the central idea behind this augmentation of states methods. And now you straightaway see that the order has increased from 2 to 4. And accordingly you will modify the output equation, the output equation doesn't change. Okay. Except that now you have more zeros coming into your C. The question that you have to ask yourself is should I always, would I always end up augmenting a state-space model with additional number of states like this.

Suppose in this question as a simple thought, a question to ask. Suppose I had asked a question a delay of one sample, an additional delay of one sample was introduced. Would you see an augmentation? So let me actually take you to an example in the book.
I'll just zoom in for your convenience here. So look at this example, this is the example that we just discussed in the former slides. Now suppose I take some other system just to show you that augmentation should be done carefully, that is the number of additional states that you are to introduce, it has to be done with caution. Right?

What kind of caution you how to exercise. Suppose you're given now this transfer function, you're not given the state-space model, you're given this transfer function. And now If I were to look at the state-space model for this. This is a state-space model. Just one of these state-space forms. Nothing has been done. We have just been given a transfer function. And what is the order of the transfer function that we have been given.

It's a second order. And what is it delay? Unit delay. Correct. That we have ascertained, again when you look at the order you have to be careful. Here this system is indeed second order.

(Refer Slide Time: 16:35)

hich has two poles located at $p_{1,2} = 0.8, 0.3$.

A state-space model for this system is

$$\mathbf{x}[k+1] = \begin{bmatrix} 1.1 & 1 \\ -0.24 & 0 \end{bmatrix} \mathbf{x}[k] + \begin{bmatrix} 2 \\ 0 \end{bmatrix} u[k]$$

$$y[k] = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}[k]$$

n additional pure delay in the input-output channel changes the TF operator to

$$G(q^{-1}) = \frac{2q^{-2}}{1 - 1.1q^{-1} + 0.24q^{-2}}$$

he new system is still characterized by two poles. Therefore, additional states shou e introduced to obtain an SS description for the new system.

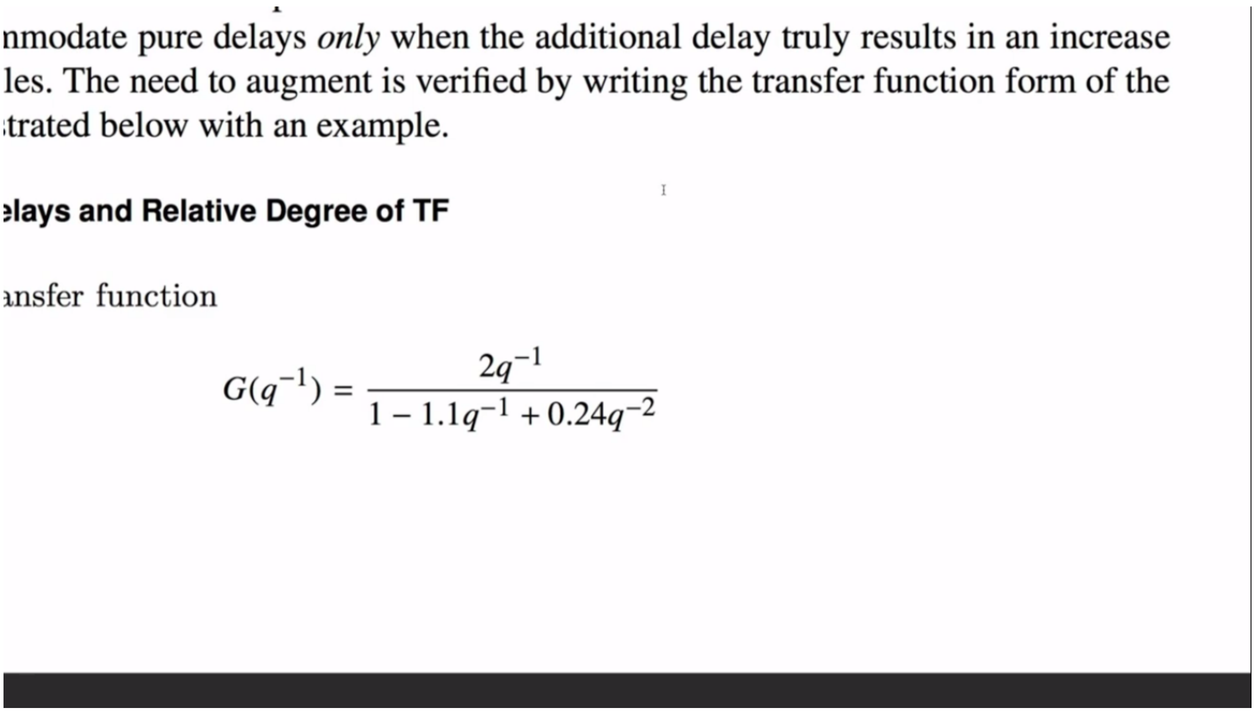
In fact, by a direct inspection of the transfer function, we can write a state-space d on for the new system:

No support to this system and additional delays introduced. Which means now the transfer function is modified in this way. Because of additional delay the numerator has become q inverse square from q inverse. Now the question you should ask is, should they augment. If I look at the state-space form here, should they augment the state-space form with an additional state. What do you think? Yes or no? How will you figure that out?

And that's what this example is trying to tell you. How do you figure out when to augment, when to introduce new states when you have delays and how many to introduce? How will you figure that out?

Remember the number of states request is the order of your transfer function. Same, they're the same. When you write it in terms of G of z, here I am giving you in terms of q inverse of z inverse. So let's look at this transfer function here, without that is when you had only just a unit delay. If you were to write this in terms of G of z, that is not in terms of z inverse or q doesn't matter. What would it look like?

(Refer Slide Time: 18:09)




G of q would be 2 over. This 2 or 2 q? 2q over q square 1.1q plus 0.8. That's it. How many poles does the system have? Two. So it's a second order system. Now go to this system where you have an additional delay. Okay. Let's call this as G1. Although I don't have it in the text as G1 and G2. Let's call the new one as G2. What would be in terms of q? 2 over q square minus 1.1 q plus 0.8. Has the order of system changed? It is not. Still remain second order which means you don't have to augment any new states. You should augment new states only if the order of the transfer function has changed because that is straight away telling you additional states are necessary.

So here you can still rewrite your state-space model. And I'm showing you how you can be that the state-space model with u k minus 1 in place. You don't have to augment. What has changed here? So this is the state space form when there was one delay and now with two delays, what has changed? The B matrix has changed. No new states have introduced. Whereas for the other example that we just discussed, we did have to augment two additional states. Suppose I accidentally did the augmentation. Okay? let's say blindly, because I just remembered. Whenever there is an excess delay I have to introduce a new state. Suppose I did that.

How will I figure out whether I made a mistake? Well, that is where the concept of minimal realization comes about. And there is a clear definition of what is a minimal realization state-space model and Kalman had given this definition long ago among many-- State-space model is set to be of minimal realization if it is both observable and controllable.
We have not talked of observability yet. I've just briefly mentioned how to, I mean, at least qualitatively we're discussed observability but not quantitatively. Later on we will learn the definitions of observability and controllability. What that statement means is, Kalman government is trying to say is, "If you have more states than necessary in your system you will not be able to infer that."

You not be able to observe them. You can have a hundred additional states but none of those states you will be able to infer from the measurements, because they are just extra candidates there. They were not supposed to be there. They have just come like you know in many Bollywood songs today it is 1:100 ratio songs. Okay. There is a heroine, then there are 10,20, 100 and there hero, there's-- because they don't want to be left alone. We think we are actually very, we're progressing but actually we're being very, even more conservative than ever before.

So there you have these additional candidates. They are not? Minimal legalization is hero and heroine only. For the movie. So here also if in this example that I am showing you on the screen right now, you could augment blindly and you will find that that new state-space model is not that of minimal realization. How will you find that out? Well in MATLAB there are ways to do that, you can construct an observability matrix and you can look at the rank of the observability matrix. Theory says if they observability matrix is rank deficient. Then you have a loss of observability. And the rank of the observability matrix is the order of the minimal realization.
So you can quantitatively check how many states were required and how many additional states you have introduced by mistake. [22:45 inaudible] not supposed to be there. So we will, as I said talk about observability and controllability later on. Okay, so the summary is, now when I have delays there is a chance that you will have to additionally augment the system with some states, whether you have to do it or not is best determined by writing the transfer function form and see if the order has changed for now.

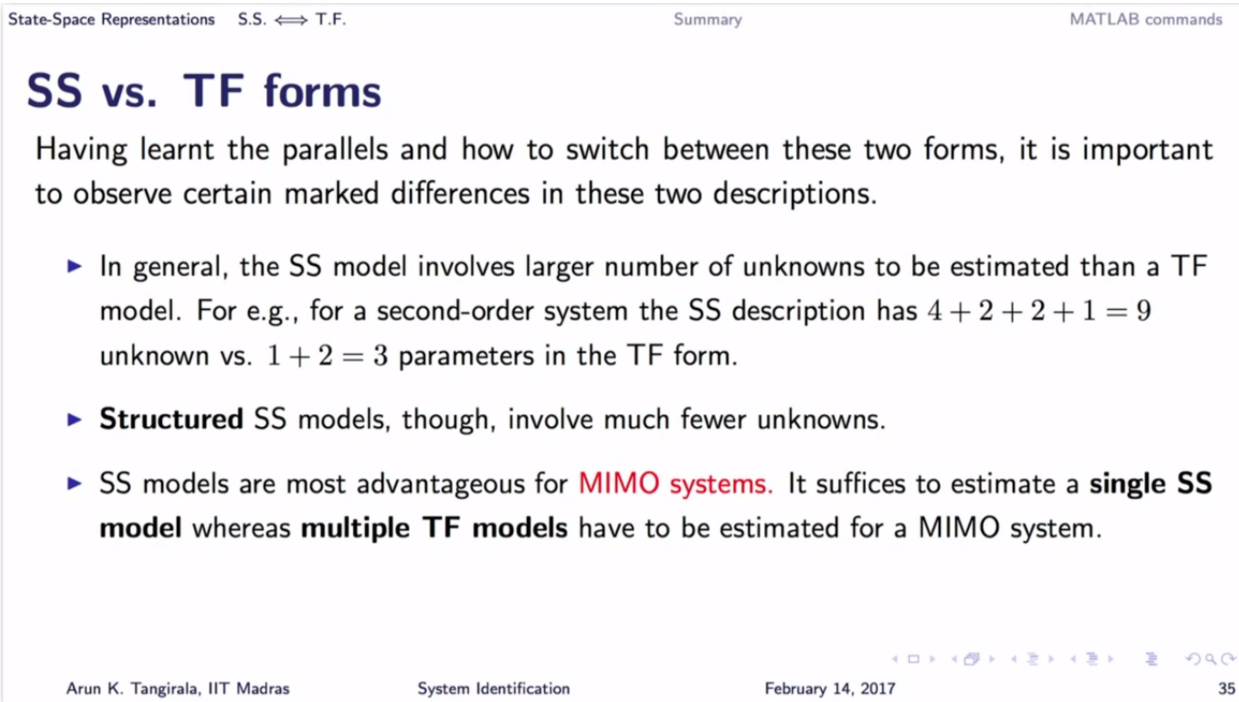
For that's the best way to do it. And it remains in fact a very good way of checking whether I need to augment or not. So simply convert this state-space model to a transfer function model and now in fact you should do this for this example. Go back to this example. The slides are there on the on the course website. Take this example, write the transfer function form. All right. And ask, what is that delay, existing delay in the system? What is the order of the system? And now if I introduced two additional delays, does the order the transfer function increase to four. If it does increase to four then it warrants an augmentation of two more states.

So just do this exercise and you will be convinced. Okay. So very quickly let me talk about the state-space verses transfer function forms and then we will wind-up. So when do we go, when do we seek state-space forms and when do we seek transfer function forms. And we have discussed one aspect already of identifying state-space models, which is that typically there are more number of unknowns, then a transfer function form for an unstructured state-space model. But as you go and impose structure, the number of parameters come down. Suppose I just want to identify, still at unstructured state-space form, very well knowing that there are more number of unknowns. Is it a recommended approach? The answer is yes. There are at least two or three different reasons, why I want to first of all even think of a state-space model. Forget about unstructured and structured. One is, when I have joined state estimation and identification problem, which is impossible for a transfer function model to handle.

We know very well, because the answer function model simply cannot handle hidden variables. So whenever I'm looking at soft sensing or joint state estimation identification problem, where the state estimates are also useful to me and as well as the model. Then there is no choice you have to turn to state-space models. That is one situation. The other situation is where I just want a first-cut model for a system without having to break too much sweat.
You will learn later on, when you come across subspace identification methods, that unstructured state-space models can be identified without solving an optimization problem. You can still get an optimal estimate of an unstructured state-space model using simple linear algebra methods rather than solving an optimization problem. So numerically these are more efficient and therefore if you just want some state-space model, like for the liquid level system or whatever system that you want to have. Don't worry about, no number of unknowns and so on. If you have enough data points you're fine. Then the unstructured state-space model is a natural choice because of the algorithms that are available for estimate.

The third reason why we want to look at stage-space models are multi variable systems. Okay.

(Refer Slide Time: 26:21)




So for multi variable systems, that is multi-input, multi-output systems. If I have let's say, a four by four, four input-four output system. I would be looking at identifying 16 transfer functions. Then I have to break my head on figuring out what is a delay, what is the order of each of those. It becomes a mess. Whereas I still identify a single state-space model [26:43 inaudible] when it comes to multi variable and single C so system, doesn't matter. All of them work with the same single state-space model, the dimensions of the matrixes may change but you don't have to seek a new algorithm, you don't have to break additional sweat. It is only the computational burden may increase. That's it, you may need more data points, but otherwise you don't have to keep figuring out what is the order of the individual subsystem and so on. So these are the three top reasons why you want to look at state-space models despite some demerits with them. All right. So just remember that. Let me just, these are just a summary of what we have discussed.

(Refer Slide Time: 27:22)

# MATLAB commands

| Purpose | Command |
|---|---|
| State-space objects w/o and with delay | `ss, delayss` |
| Absorbing delays into SS models | `absorbDelay, delay2z` (old) |
| Extracting data from SS objects | `ssdata` |
| Converting SS to TF and vice versa | `ss2tf, tf2ss, ss, tf` |
| Eigenvalues, pole and zero | `eig, pole, zero, pzmap` |
| Gain calculation | `dcgain` |
| FRF (Bode plot) | `bode` |
| Simulation and response computation | `dlsim, impulse, step, initial` |
| General LTI system responses | `ltiview` |



These are the MATLAB commands that will come handy particularly when you solve assignments and not for the quiz. So for example state-space objects, there is a similar set of MATLAB comments that are available for the input-output models. Please look up the course notes in the interest of time.

I'm not going to talk about that today but at some other point I'll bring up those. So here you have SS there you have the TF. And you can incorporate delays in input-output models by using this IO delay property. Every object, LTI object that you create in MATLAB has certain properties. You have to remember that. In fact you can examine those properties using get command. Right? Suppose G D is the transfer function object or state-space object. You can say get G D, list all the properties. And some of these properties will include IO delay, input delay, output delay, sampling interval and so on.

And if you want to retrieve the stage space matrixes from a state-space model you can use SS data and then you have the standard SS TF, TF to SS. SS to TF expects you to give A B C D and TF to SS expects you to give numerator and denominator but TF and SS is what I prefer more because all you have to do is pass on the state-space object to TF, it will give you a state-space model and likewise for SS. You can do the analysis using eig or pole, 0 pzmap, dcgain. I have shown you this already, bode also have shown. And of course I've shown you LTI view. dlsim is a nice command for simulating LTI systems to arbitrary inputs. Your custom user defined inputs. Impulse, step will give you the standard impluse and step responses, initial gives you the response to non-zero initial conditions. Particularly useful for state-space, simulating state-space models. So these are the relevant MATLAB commands and that's it. So we really see you.