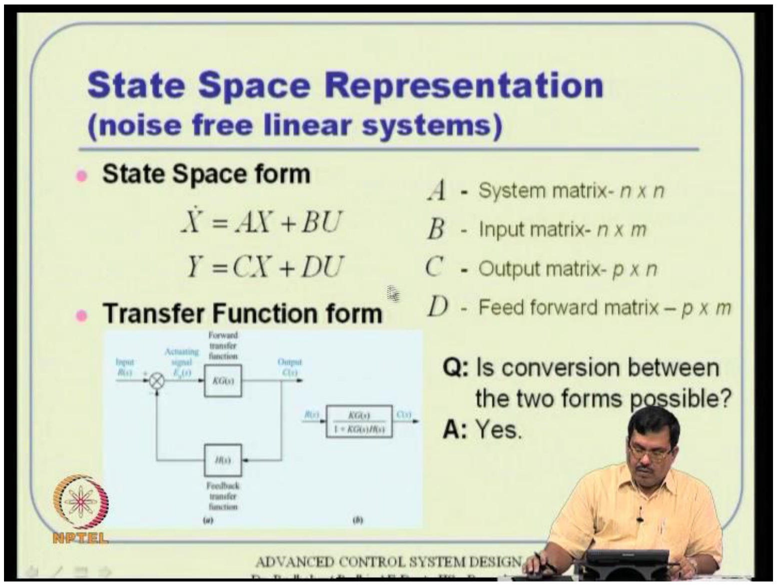
Advanced Control System Design

Prof. Radhakant Padhi
Department of Aerospace Engineering
Indian Institute of Science, Bangalore

Lecture No. # 11
Representation of Dynamical Systems – III

Good evening everyone, we are there in the lecture number 10 today. However, before going there probably it is good idea to overview, give every preview of what we have discussed in previous lecture that is lecture number 1, lecture number 9, because this two are tightly related to each other. So, let us overview quickly what we discuss and there was kind of small mistakes in the slides also, that I will also correct that actually.

(Refer Slide Time: 00:44)



State Space Representation
(noise free linear systems)

- **State Space form**

$$\dot{X} = AX + BU$$
$$Y = CX + DU$$

A - System matrix- $n \times n$
B - Input matrix- $n \times m$
C - Output matrix- $p \times n$
D - Feed forward matrix – $p \times m$

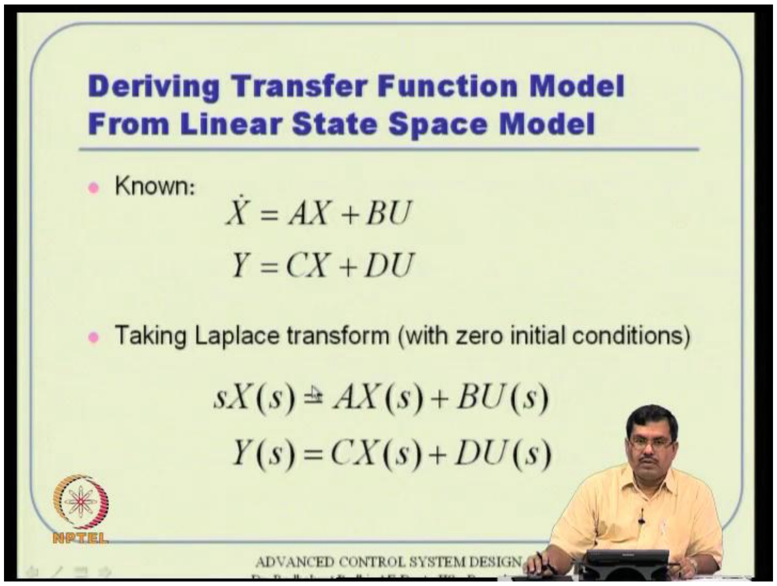
- **Transfer Function form**

Q: Is conversion between the two forms possible?
A: Yes.

ADVANCED CONTROL SYSTEM DESIGN

So, this is what we are talking about how do you covert it from a state space representation to transfer function representations and vice versa when you have a linear systems. So, here it is I mean the; what we discuss last time, when you have a state space representation X equal to AX plus BU and Y equal to CX plus DU and this is what the transfer function form that you are looking out.

(Refer Slide Time: 01:11)



Deriving Transfer Function Model From Linear State Space Model

- Known:

$$\dot{X} = AX + BU$$
$$Y = CX + DU$$

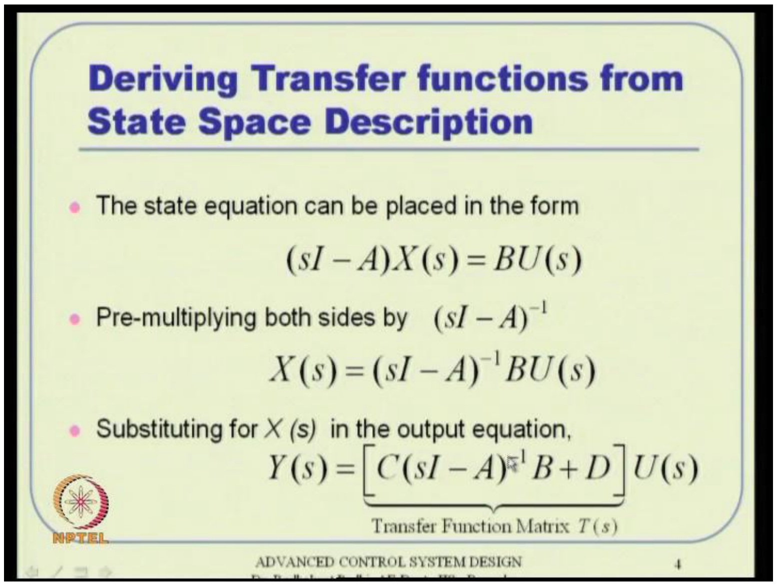
- Taking Laplace transform (with zero initial conditions)

$$sX(s) = AX(s) + BU(s)$$
$$Y(s) = CX(s) + DU(s)$$

ADVANCED CONTROL SYSTEM DESIGN

Looking for, then you started with this equation here and then get it out Laplace transform for both the equations then, eliminated X of s from first equation substituted with care.

(Refer Slide Time: 01:25)



Deriving Transfer functions from State Space Description

- The state equation can be placed in the form

$$(sI - A)X(s) = BU(s)$$

- Pre-multiplying both sides by $(sI - A)^{-1}$

$$X(s) = (sI - A)^{-1}BU(s)$$

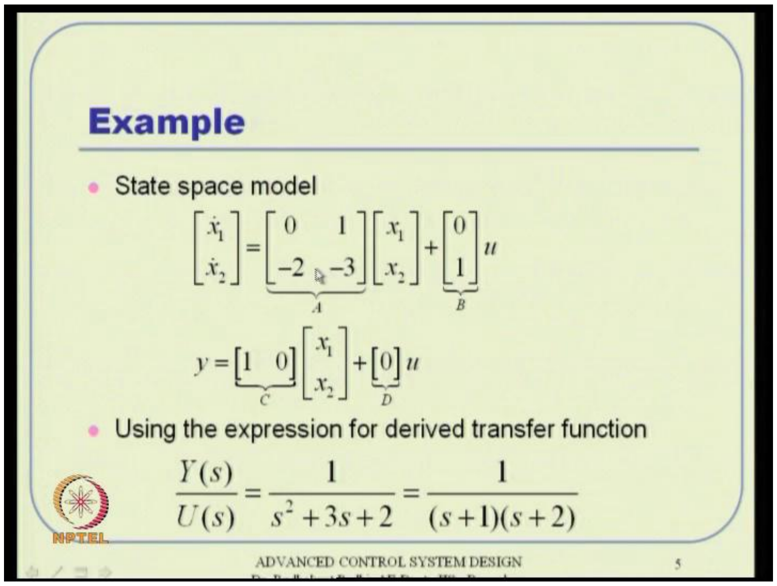
- Substituting for $X(s)$ in the output equation,

$$Y(s) = \underbrace{\left[C(sI - A)^{-1}B + D\right]}_{\text{Transfer Function Matrix } T(s)}U(s)$$

ADVANCED CONTROL SYSTEM DESIGN 4

That is what we get it on there. So, then it turns out that Y of s is that given out this final expression out here. So, this is C into s I minus A inversely plus D that is the transfer function matrix in general we are talking about you know, there are multiple outputs and multiple inputs out here. So, this in general Y is a vector and U is also a vector. So, in principle we cannot talk about Y by U anymore, but some books write that way Y by U and then they give there is a transfer function matrix. Then, there was example so, you carry out that examples.

(Refer Slide Time: 02:01)



Example

- State space model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{B} u$$

$$y = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{D} u$$

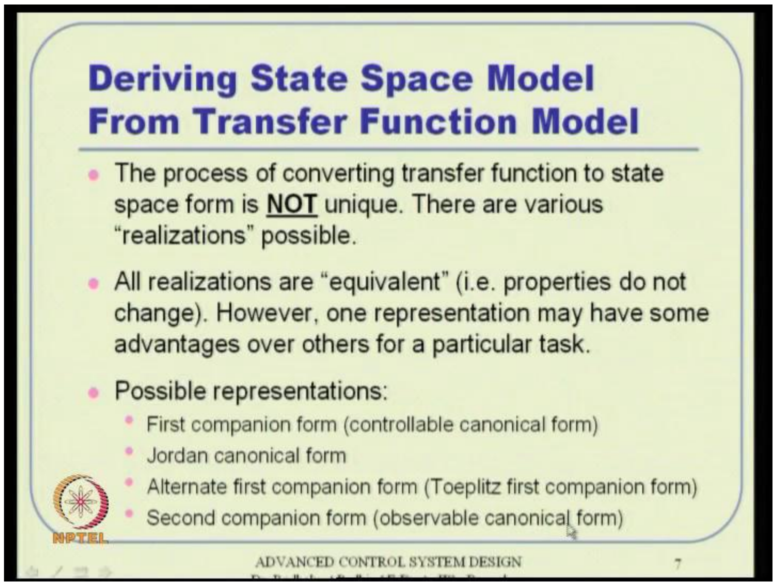
- Using the expression for derived transfer function

$$\frac{Y(s)}{U(s)} = \frac{1}{s^2 + 3s + 2} = \frac{1}{(s+1)(s+2)}$$

ADVANCED CONTROL SYSTEM DESIGN 5

So, that if this is a linear system, two dimensional linear system that is all the transfer function all about it is straight forward derive it from the formula, which we carry out in detail algebra here.

(Refer Slide Time: 02:11)



Deriving State Space Model From Transfer Function Model

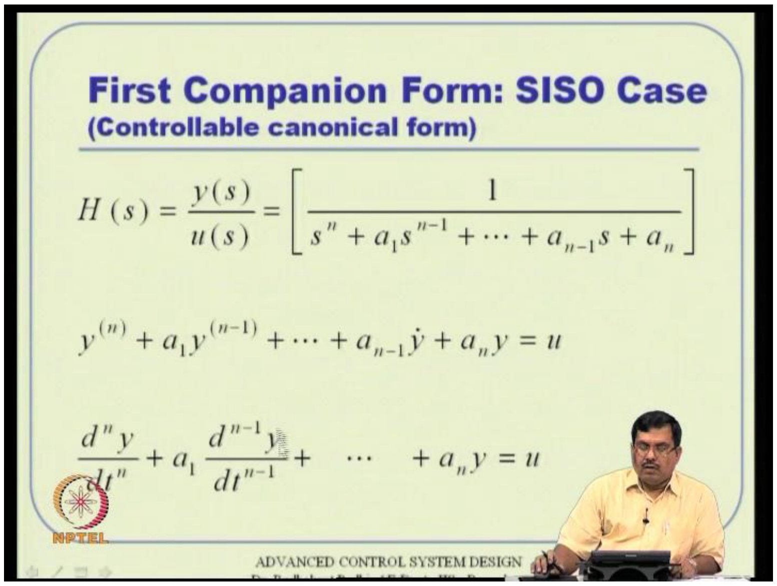
- The process of converting transfer function to state space form is **NOT** unique. There are various "realizations" possible.
- All realizations are "equivalent" (i.e. properties do not change). However, one representation may have some advantages over others for a particular task.
- Possible representations:
  - First companion form (controllable canonical form)
  - Jordan canonical form
  - Alternate first companion form (Toeplitz first companion form)
  - Second companion form (observable canonical form)

ADVANCED CONTROL SYSTEM DESIGN 7

And we also made some points that this converting the transfer function to state space is rather not unique, where is the reverse one is very unique actually. So, we have I mean there are various realizations possible and he also made a note the; that all realizations are equivalent actually. That means, the basic properties do not change however, one representation have some advantage over the other for a particular task actually. So, lot of research has gone into that in around 50's and 60's. How do you get a good realizations actually, the possible representations are that comes to mind is first companion form, but otherwise known as controllable canonical form.

And we have also discuss what is Jordan canonical form last time, then in this particular class I will talk about alternate first companion form which is called as a toeplitz first companion form also, because of the matrix that it pops of actually. And then, will also see what is called a second companion form and otherwise known as observable canonical form. So, for example, if you want to design a controller then, controllable canonical form is a better option. Or as if you want to design a observer or a filter then observable canonical form turns out to be better choice actually.

(Refer Slide Time: 03:36)



First Companion Form: SISO Case
(Controllable canonical form)

$$H(s) = \frac{y(s)}{u(s)} = \left[\frac{1}{s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \right]$$

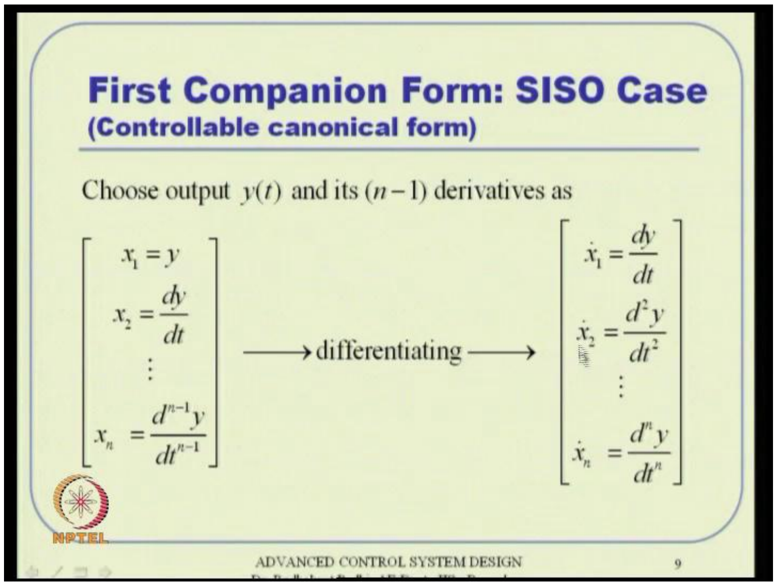
$$y^{(n)} + a_1 y^{(n-1)} + \cdots + a_{n-1}\dot{y} + a_n y = u$$

$$\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_n y = u$$

ADVANCED CONTROL SYSTEM DESIGN

Then we had this; we started with this transfer function Y of s by U of s then we multiplied both sides then if you multiply then that is the differential equation that pops off and that is in a; I mean the same thing that is written in elaborate form. Then, you choose this straight variables x 1 x 2 of how to action that way then differentiate a Landered of the equation actually.

(Refer Slide Time: 03:55)



First Companion Form: SISO Case
(Controllable canonical form)

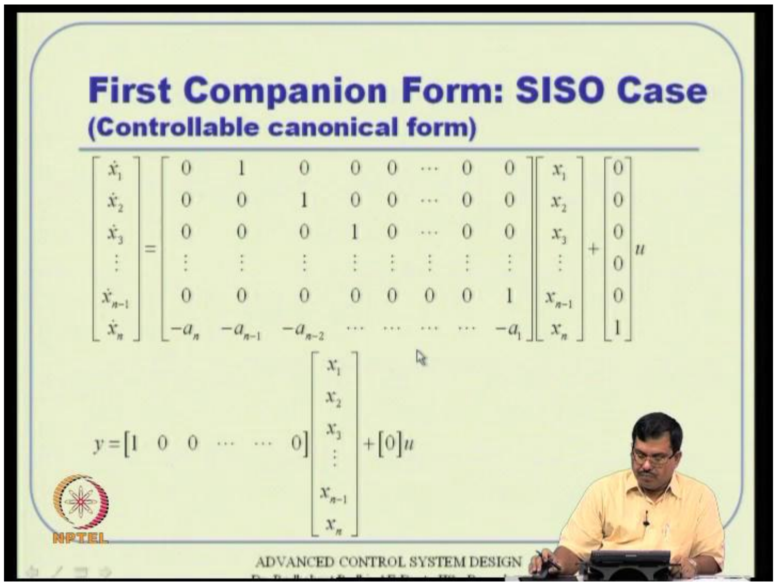
Choose output $y(t)$ and its $(n-1)$ derivatives as

$$\begin{bmatrix} x_1 = y \\ x_2 = \frac{dy}{dt} \\ \vdots \\ x_n = \frac{d^{n-1}y}{dt^{n-1}} \end{bmatrix} \longrightarrow \text{differentiating} \longrightarrow \begin{bmatrix} \dot{x}_1 = \frac{dy}{dt} \\ \dot{x}_2 = \frac{d^2 y}{dt^2} \\ \vdots \\ \dot{x}_n = \frac{d^n y}{dt^n} \end{bmatrix}$$

ADVANCED CONTROL SYSTEM DESIGN 9

So, first n minus 1 equations or differential equations will simply come from definitions, rather and the last one will come from the differential equation that we are having here actually.

(Refer Slide Time: 04:15)



First Companion Form: SISO Case
(Controllable canonical form)

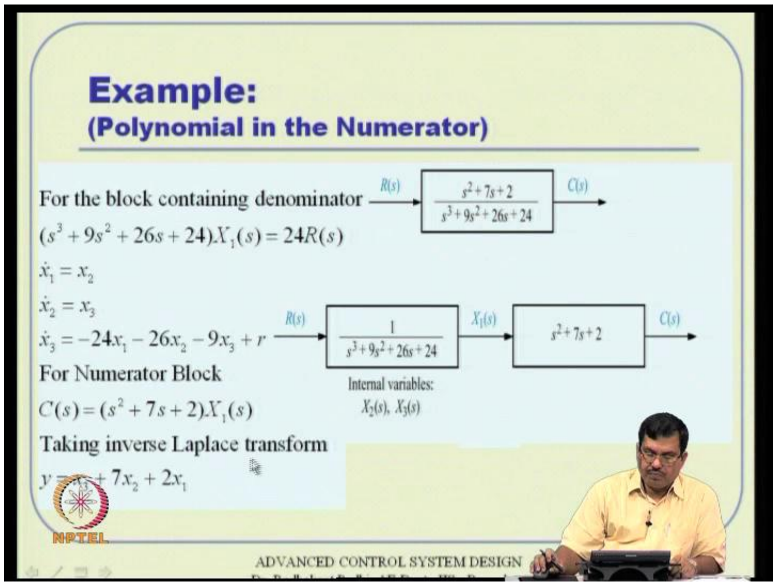
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & \cdots & \cdots & \cdots & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} + [0]u$$

ADVANCED CONTROL SYSTEM DESIGN

That is how we got this first companion form, which look like that way. And because Y is nothing but a x 1 so, the c and d matrix are to be just actually so, that is how we landed out with first companion form. We also have briefly overview this blocked diagram representation, which essentially requires n integrators, because you have an n of a differential equation anyway.

(Refer Slide Time: 04:47)



Example:
(Polynomial in the Numerator)

For the block containing denominator

$(s^3 + 9s^2 + 26s + 24)X_1(s) = 24R(s)$

$\dot{x}_1 = x_2$

$\dot{x}_2 = x_3$

$\dot{x}_3 = -24x_1 - 26x_2 - 9x_3 + r$

For Numerator Block

$C(s) = (s^2 + 7s + 2)X_1(s)$

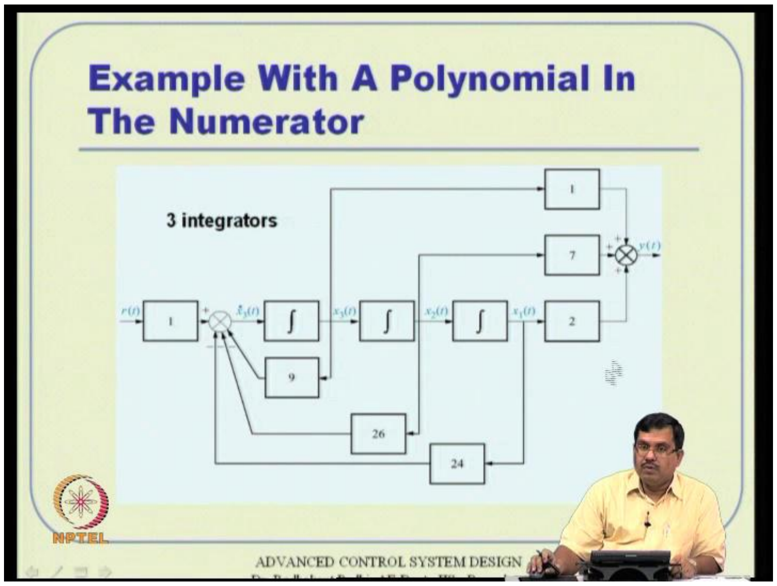
Taking inverse Laplace transform

$y = x_3 + 7x_2 + 2x_1$

R(s)
$\frac{s^2+7s+2}{s^3+9s^2+26s+24}$
C(s)
R(s)
$\frac{1}{s^3+9s^2+26s+24}$
$X_1(s)$
s^2+7s+2
C(s)
Internal variables:
$X_2(s)$, $X_3(s)$
NPTEL
ADVANCED CONTROL SYSTEM DESIGN

So, that is the reason why that we require this n state variables really. So, then we had further examples, where we talked about this transport function what is the there are these third error polynomial so, we have essentially third order system. So, we have defined X 1 to be c and X 2 to be c dot and C 3 to be double dot like that and then, we carried out this are the state space representations or state space equations that way actually. So, then we moved on with a polynomial in the numerator what you do about that then, we decomposed that into two blocks rather first block will give us rather the same differential equation, that we just derive before. And the next one will give; I mean this particular thing will give us the output equation actually.

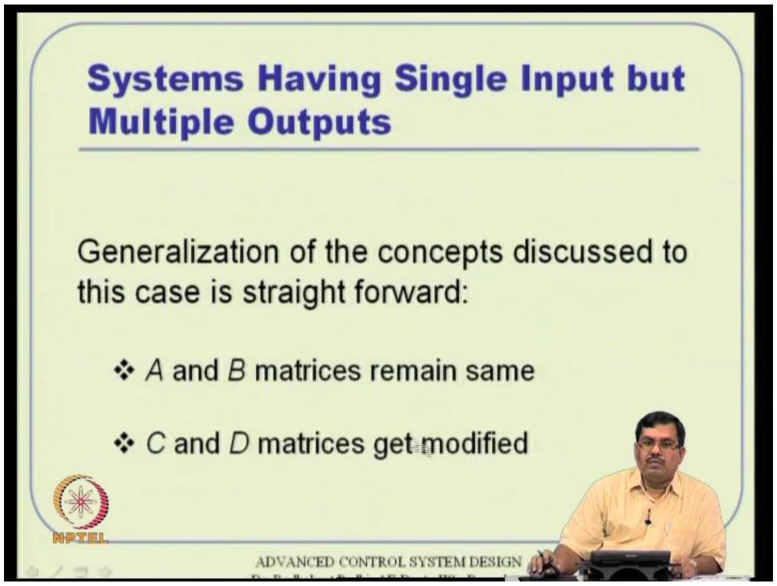
(Refer Slide Time: 05:36)



Example With A Polynomial In The Numerator

So, this is how we arrived it, this kind of forms and all that etcetera. So, this is the realization with this particle of sort it that output equation contains all this 1 7 2 sort of thing actually.

(Refer Slide Time: 05:47)



Systems Having Single Input but Multiple Outputs

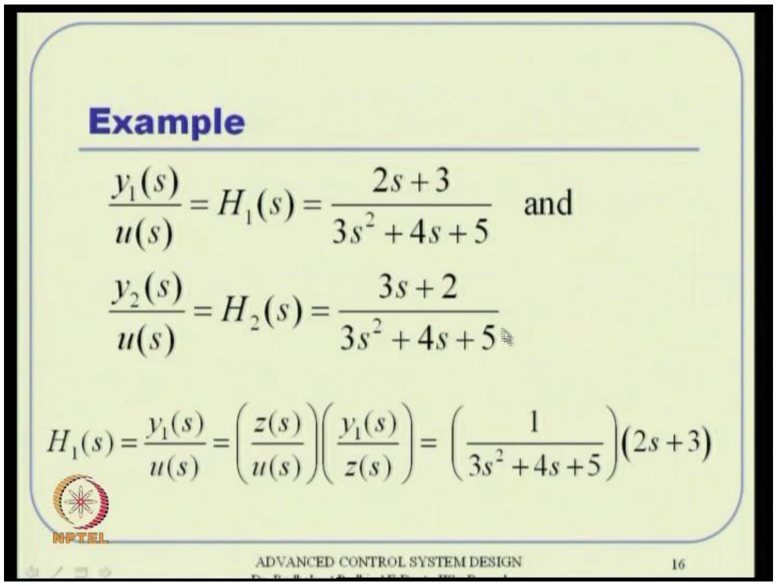
Generalization of the concepts discussed to this case is straight forward:

- ❖ *A* and *B* matrices remain same
- ❖ *C* and *D* matrices get modified

ADVANCED CONTROL SYSTEM DESIGN

So, how do you I mean generalize this to multiple outputs these are all about single outputs anyway. So, if you have multiple outputs then essentially the A and B metrics remains same however, the C and D matrices get modified actually.

(Refer Slide Time: 06:04)



Example

$$\frac{y_1(s)}{u(s)} = H_1(s) = \frac{2s+3}{3s^2+4s+5} \quad \text{and}$$

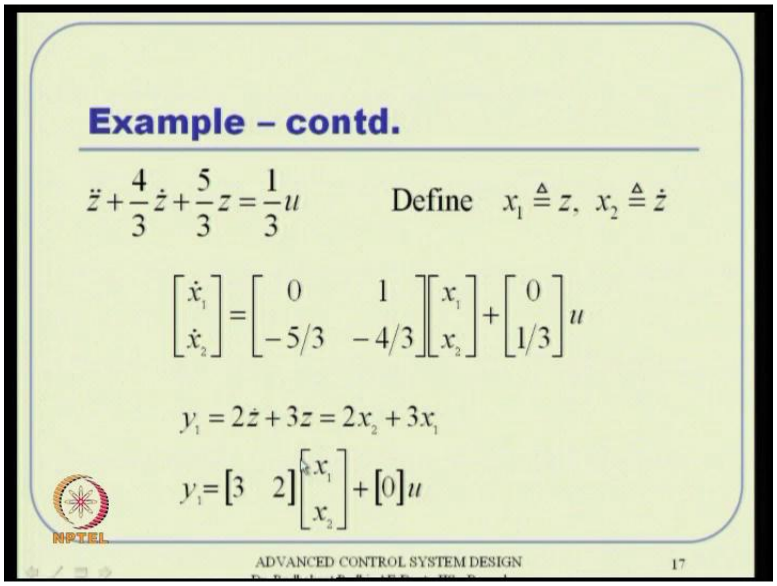
$$\frac{y_2(s)}{u(s)} = H_2(s) = \frac{3s+2}{3s^2+4s+5}$$

$$H_1(s) = \frac{y_1(s)}{u(s)} = \left(\frac{z(s)}{u(s)}\right)\left(\frac{y_1(s)}{z(s)}\right) = \left(\frac{1}{3s^2+4s+5}\right)(2s+3)$$

ADVANCED CONTROL SYSTEM DESIGN 16

Also, solved saw a example to demonstrate that, where this one is the y 1 of s divided by u and this is y 2 of s divided by u this are the transfer functions that we are talking about, you note that this denominator is same. So, here we decomposed this particular this by 1 by u as this two variables that way. So, this 2 s plus 3 falls out outside of this are two essential two cascaded blocks what we are talking about here.

(Refer Slide Time: 06:33)



Example – contd.

$\ddot{z}+\frac{4}{3}\dot{z}+\frac{5}{3}z=\frac{1}{3}u$ Define $x_1 \triangleq z,\ x_2 \triangleq \dot{z}$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}=\begin{bmatrix}0 & 1\\ -5/3 & -4/3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\\ 1/3\end{bmatrix}u$$

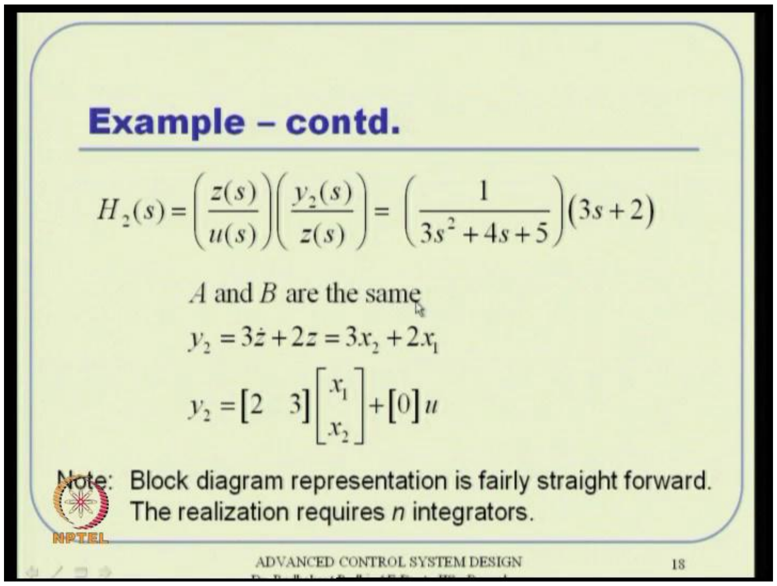
$$y_1=2\dot{z}+3z=2x_2+3x_1$$

$$y_1=\begin{bmatrix}3 & 2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\end{bmatrix}u$$

ADVANCED CONTROL SYSTEM DESIGN 17

And then, the first part of the block gave us the straight representations and the second part of the block will give us, the output equation actually.

(Refer Slide Time: 06:48)



Example – contd.

$$H_2(s)=\left(\frac{z(s)}{u(s)}\right)\left(\frac{y_2(s)}{z(s)}\right)=\left(\frac{1}{3s^2+4s+5}\right)(3s+2)$$

A and B are the same

$$y_2=3\dot{z}+2z=3x_2+2x_1$$

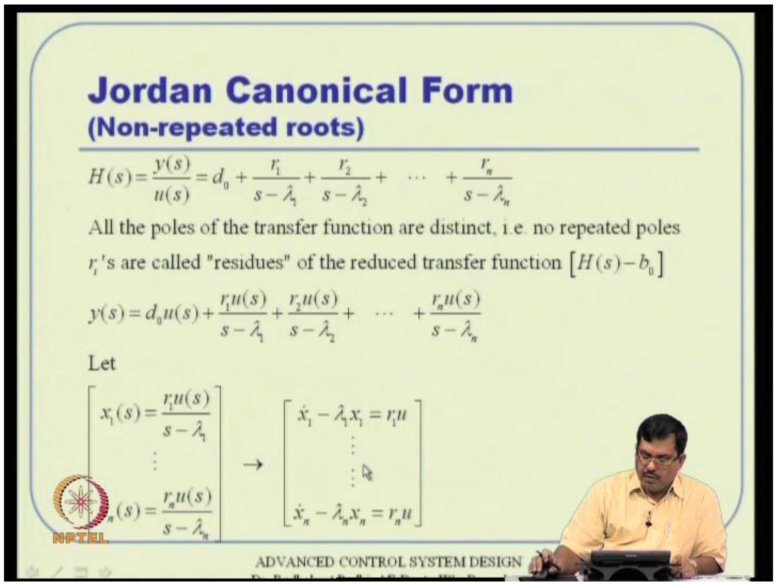
$$y_2=\begin{bmatrix}2 & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\end{bmatrix}u$$

Note: Block diagram representation is fairly straight forward. The realization requires *n* integrators.

ADVANCED CONTROL SYSTEM DESIGN 18

So, that is how we got the output equation for phi 1 and similarly, the output equation for y 2 will be this way. Whereas the straight equations, I mean the n matrices will remain same actually.

(Refer Slide Time: 07:00)



Jordan Canonical Form
(Non-repeated roots)

$$H(s)=\frac{y(s)}{u(s)}=d_0+\frac{r_1}{s-\lambda_1}+\frac{r_2}{s-\lambda_2}+\cdots+\frac{r_n}{s-\lambda_n}$$

All the poles of the transfer function are distinct, i.e. no repeated poles

r_i's are called "residues" of the reduced transfer function $[H(s)-b_0]$

$$y(s)=d_0u(s)+\frac{r_1u(s)}{s-\lambda_1}+\frac{r_2u(s)}{s-\lambda_2}+\cdots+\frac{r_nu(s)}{s-\lambda_n}$$

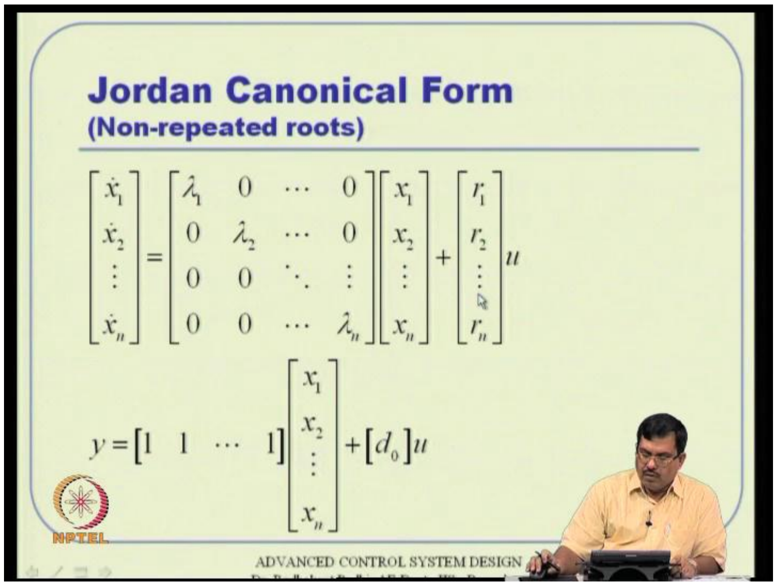
Let

$$\begin{bmatrix} x_1(s)=\frac{r_1u(s)}{s-\lambda_1} \\ \vdots \\ x_n(s)=\frac{r_nu(s)}{s-\lambda_n}\end{bmatrix} \rightarrow \begin{bmatrix} \dot{x}_1-\lambda_1x_1=r_1u \\ \vdots \\ \vdots \\ \dot{x}_n-\lambda_nx_n=r_nu\end{bmatrix}$$

ADVANCED CONTROL SYSTEM DESIGN

Then we also reviewed what is known as I mean we went through detail what is known as Jordan canonical form so, if you have transfer function in general already factor into this form. If it is not a factor then, the first thing we should do this factorization actually. Assuming about this factorization is already available then with a we multiple again this y of s is equal to all that and then we define this very this little expressions as x 1 of s x 2 of s like that up to x of s that is definition. And simply from definition this differential equation pops off like that and hence we have already have (( )) representation of the system dynamics x 1 dot actually.

(Refer Slide Time: 07:45)



Jordan Canonical Form
(Non-repeated roots)

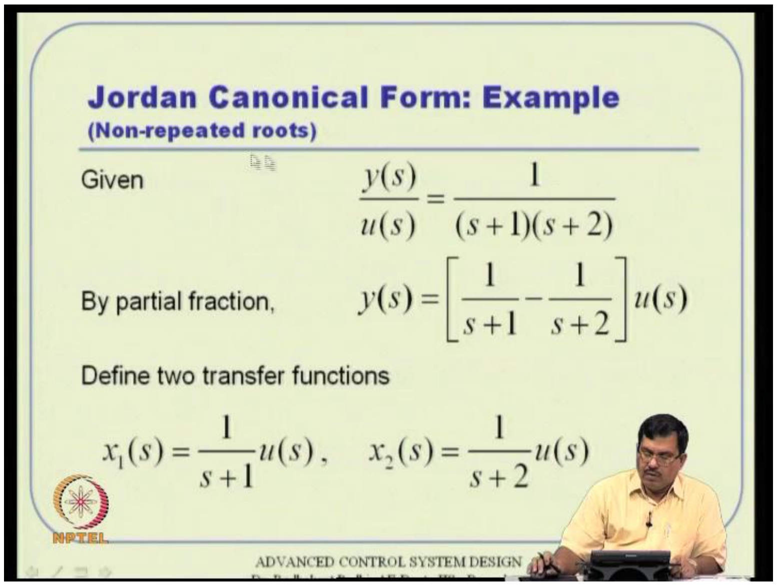
$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \vdots\\ \dot{x}_n\end{bmatrix}=\begin{bmatrix}\lambda_1 & 0 & \cdots & 0\\ 0 & \lambda_2 & \cdots & 0\\ 0 & 0 & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_n\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ \vdots\\ x_n\end{bmatrix}+\begin{bmatrix}r_1\\ r_2\\ \vdots\\ r_n\end{bmatrix}u$$

$$y=\begin{bmatrix}1 & 1 & \cdots & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ \vdots\\ x_n\end{bmatrix}+\begin{bmatrix}d_0\end{bmatrix}u$$

ADVANCED CONTROL SYSTEM DESIGN

That is how we got it. And then y turns out to be all this y is summation of this, I mean this plus this plus this all this expressions. So, but these are nothing but x 1 plus x to s then r into s n like that so, y of s simply turns out to be this x 1 plus x 2 plus up to x n plus d 0 times u. So, this is our C matrix, (( )) D matrix actually and these two are essentially in B matrixes now, this is a Jordan Canonical form.

(Refer Slide Time: 08:13)



Jordan Canonical Form: Example
(Non-repeated roots)

Given $$\frac{y(s)}{u(s)}=\frac{1}{(s+1)(s+2)}$$

By partial fraction, $$y(s)=\left[\frac{1}{s+1}-\frac{1}{s+2}\right]u(s)$$

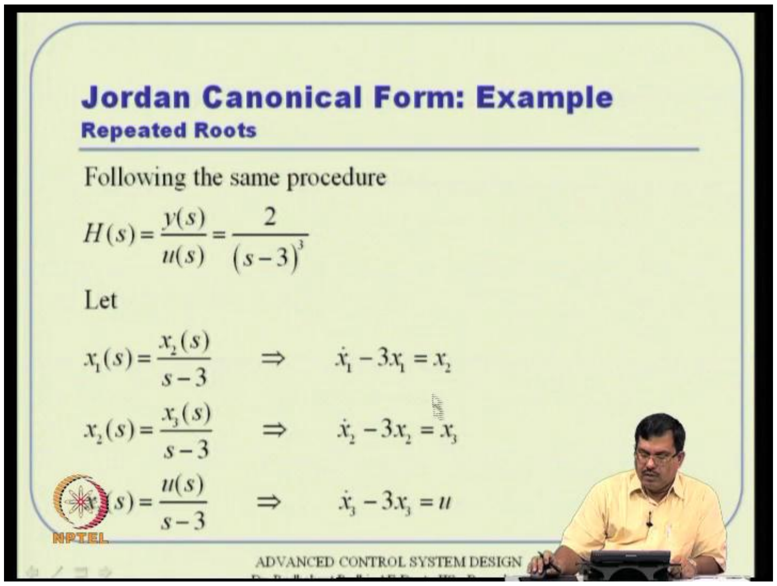
Define two transfer functions

$$x_1(s)=\frac{1}{s+1}u(s),\quad x_2(s)=\frac{1}{s+2}u(s)$$

ADVANCED CONTROL SYSTEM DESIGN

And we also had that (( )) I mean this an example where you have this none repeated roots sort of thing then, we take essential distribution then we teach essential for what we discussed and that linear of with this we say as C D matrix actually d 0 here. So, a this is a that is B and this is A matrix actually.

(Refer Slide Time: 08:31)



Jordan Canonical Form: Example

Repeated Roots

Following the same procedure

$$H(s)=\frac{y(s)}{u(s)}=\frac{2}{(s-3)^3}$$

Let

$$x_1(s)=\frac{x_2(s)}{s-3} \quad \Rightarrow \quad \dot{x}_1-3x_1=x_2$$

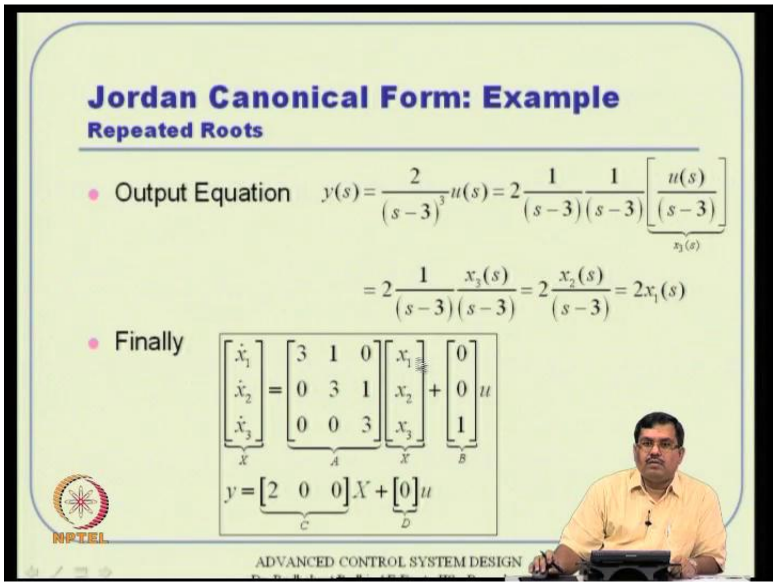
$$x_2(s)=\frac{x_3(s)}{s-3} \quad \Rightarrow \quad \dot{x}_2-3x_2=x_3$$

$$x_3(s)=\frac{u(s)}{s-3} \quad \Rightarrow \quad \dot{x}_3-3x_3=u$$

ADVANCED CONTROL SYSTEM DESIGN

Now, what you do when you have repeated roots that is a special case probably. And then, we define this straight variables that way, as so x 1 is nothing but x 2 by s plus 3 this is a cubic polynomial remember that so, x 2 is x 3 by x plus 3 and then finally, x 3 (( )) of x plus 3. So, again from this definitions this static equations pop up actually.

(Refer Slide Time: 08:57)



Jordan Canonical Form: Example
Repeated Roots

- Output Equation $y(s) = \frac{2}{(s-3)^3} u(s) = 2 \frac{1}{(s-3)} \frac{1}{(s-3)} \underbrace{\left[\frac{u(s)}{(s-3)} \right]}_{x_3(s)}$

$$= 2 \frac{1}{(s-3)} \frac{x_3(s)}{(s-3)} = 2 \frac{x_2(s)}{(s-3)} = 2x_1(s)$$

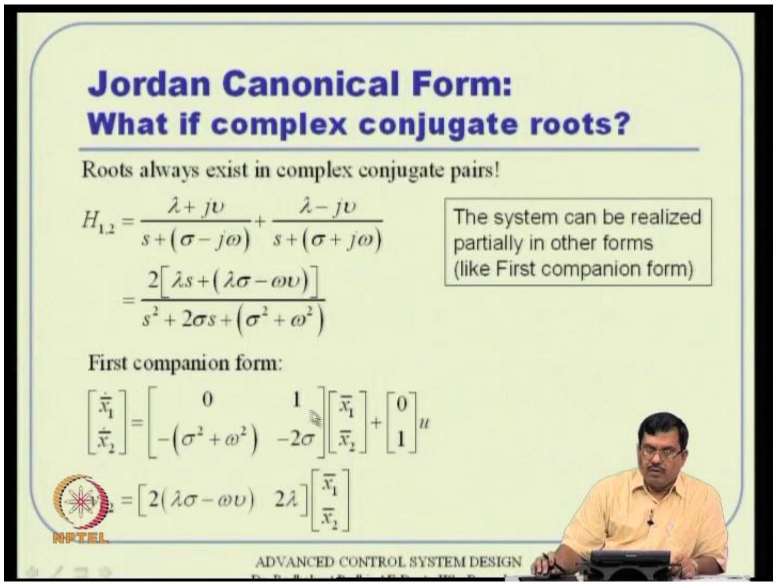
- Finally

$$\boxed{\begin{gathered}\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}}_{\dot{X}} = \underbrace{\begin{bmatrix} 3 & 1 & 0 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{X} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{B} u \\ y = \underbrace{\begin{bmatrix} 2 & 0 & 0 \end{bmatrix}}_{C} X + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{D} u\end{gathered}}$$

ADVANCED CONTROL SYSTEM DESIGN

Then, the output equation is slightly interesting to see here, the output is essentially this entire thing multiplied by u of s. So, then if you analyze that little bit this u of; I mean what you mean hear I can expand; in an expanded form I can put it there. I just write it that a 3 times essentially 1 by s plus three into 1 by s plus three into 1 by s plus three like that. What do we note that, this nothing but a x 3 of s so, if once I put x 3 of phase here then it turns out to be x 3 of s by s plus 3 which is nothing but x 2 of s. And once I put x 2 of s here it is nothing but x 1 of s, essentially y of s turns out to be just two times x 1 of s. So, noting all these that this of the state equations and this is the output equation you can finally, convert all this into time domain. And tell this is mine Jordan Canonical form A and B matrixes and this is the corresponding C and D matrixes actually.

(Refer Slide Time: 10:06)



Jordan Canonical Form: What if complex conjugate roots?

Roots always exist in complex conjugate pairs!

$$H_{1,2} = \frac{\lambda + j\upsilon}{s+(\sigma - j\omega)} + \frac{\lambda - j\upsilon}{s+(\sigma + j\omega)} = \frac{2\left[\lambda s + (\lambda\sigma - \omega\upsilon)\right]}{s^2 + 2\sigma s + (\sigma^2 + \omega^2)}$$

The system can be realized partially in other forms (like First companion form)

First companion form:

$$\begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(\sigma^2+\omega^2) & -2\sigma \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y_{1,2} = \begin{bmatrix} 2(\lambda\sigma - \omega\upsilon) & 2\lambda \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix}$$

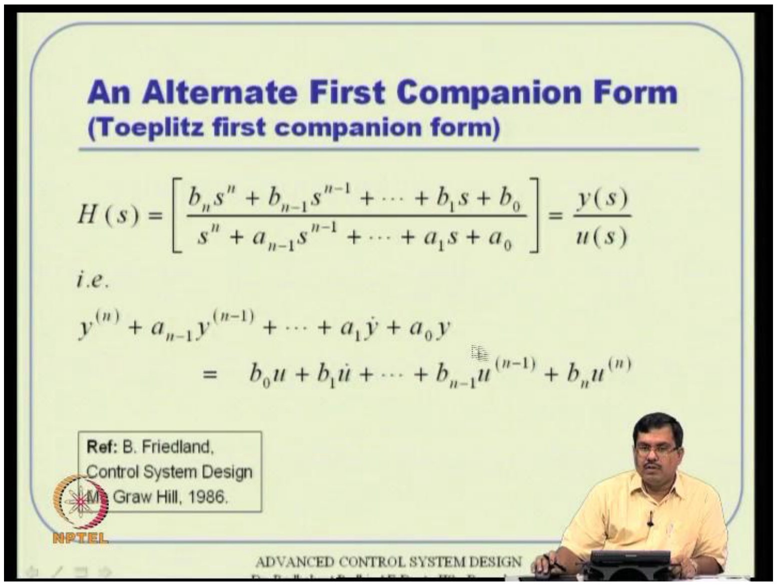
ADVANCED CONTROL SYSTEM DESIGN

So, that is how we got for repeated, up to that probably we studied with a sums like corrections of these lights and all that. Then, we also note that, there can be one more complexity in Jordan Canonical form and that happens when this factorization form the roots become complex actually. And we do not want, this matrices of A B C D elements of these matrixes in complex numbers we do not want, that is not their real realization for real system actually so, what we really want is real number sitting out there actually. So, how do you do that and also this observation comes to our rescue here actually. That if we take; if the roots becomes complex they can always exist in complex conjugate pairs actually.

And interestingly this let say only one particular form we have a something like two repeated roots out here and they are in complex conjugate pair actually. What you see out here and interestingly corresponding to that you will also numerators the complex conjugate pair actually. So, all these nice observations give us a hope actually. So, what do you do, if you just combine them together then the complex parts goes and you are left out with a real transfer function actually. But remember this in a factorization form I cannot write the reminder actually so, what I can do that rest of the transfer functions elements what I watched rest of the factors I will do; I mean the regulars Jordan canonical form.

But this particular cup; I mean this particular conjugate pair roots I will go at least some other companion form like first companion for m for example, so this particular block I will write it in a regular first companion form what we have discussed already before. However, the other things will turn out to be Jordan Canonical form actually. So, that is the way to do so, this is all about like a some overview of the previous lecture so, that we carry on with the further things in this class actually. So, the last lecture we solve this; a first companion form and a Jordan Canonical form.

(Refer Slide Time: 12:08)



An Alternate First Companion Form
(Toeplitz first companion form)

$$H(s) = \left[\frac{b_n s^n + b_{n-1} s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} \right] = \frac{y(s)}{u(s)}$$

i.e.

$$y^{(n)} + a_{n-1} y^{(n-1)} + \cdots + a_1 \dot{y} + a_0 y$$
$$= b_0 u + b_1 \dot{u} + \cdots + b_{n-1} u^{(n-1)} + b_n u^{(n)}$$

Ref: B. Friedland, Control System Design Mc Graw Hill, 1986.

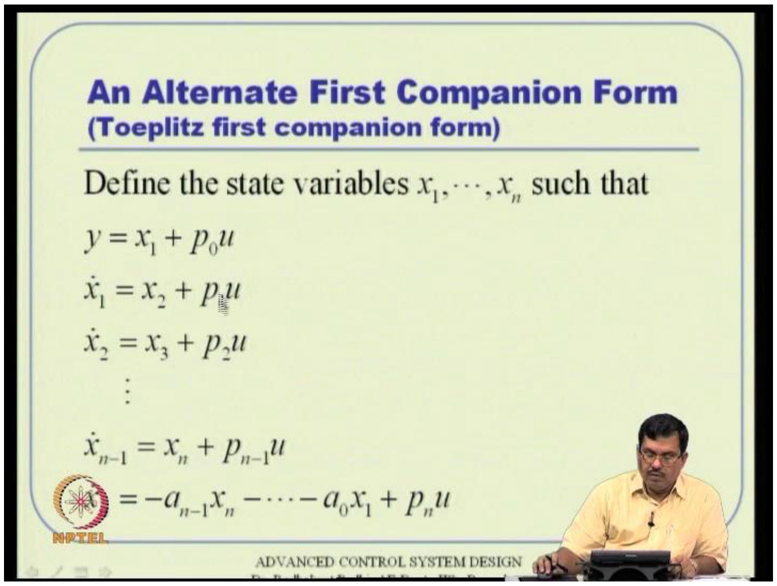
ADVANCED CONTROL SYSTEM DESIGN

I will continue about discussion with further things here with first thing is alternate first companion form or sometimes called as toeplitz first companion form, we will see that why this name pops off actually and this the reference for that what I have taken actually.

So, we will start again with the transform function in this form and then we can again multiply both sets and then write it in the standard differential equation form. So, this is a I mean difference differential equation both in y as well as u actually, this is a very generic thing actually. This proper transfer function what we do not strictly, proper transfer function is the order of the numerator polynomial is less than equal to the order of the denominator polynomial so, in this case you are taking it is like a equal, that is the best pair you can have actually.

In general this order of the numerator is much lesser than the order of the denominator actually so that is a proper strictly proper transfer function rather actually. Anyways once we write this differential equation form for this output and input relationship.

(Refer Slide Time: 13:17)



An Alternate First Companion Form
(Toeplitz first companion form)

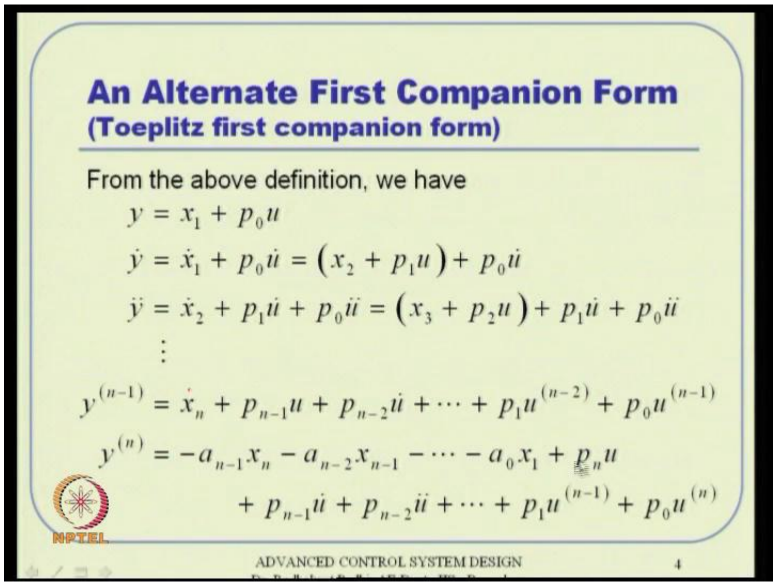
Define the state variables $x_1, \cdots, x_n$ such that

$$y = x_1 + p_0 u$$
$$\dot{x}_1 = x_2 + p_1 u$$
$$\dot{x}_2 = x_3 + p_2 u$$
$$\vdots$$
$$\dot{x}_{n-1} = x_n + p_{n-1} u$$
$$\dot{x}_n = -a_{n-1} x_n - \cdots - a_0 x_1 + p_n u$$

ADVANCED CONTROL SYSTEM DESIGN

Now, let us define this state variables x 1 to x n in such a way so, how do you define let us say output that a x 1 plus p 0 u this is simply a once out definition actually. And we write x 1 dot is x 2 plus p 1 u and x 2 dot is x 3 plus p 2 u actually these are all state equations. Remember if I somehow know this p 0 up to p n then I am done actually, because this is an output equation and these are my state equations already known to me actually in a way. So, it is my task to get this constants p 0 p 1 p 2 up to p n that is the task actually.

(Refer Slide Time: 14:02)



An Alternate First Companion Form (Toeplitz first companion form)

From the above definition, we have

$$y = x_1 + p_0 u$$

$$\dot{y} = \dot{x}_1 + p_0 \dot{u} = (x_2 + p_1 u) + p_0 \dot{u}$$

$$\ddot{y} = \dot{x}_2 + p_1 \dot{u} + p_0 \ddot{u} = (x_3 + p_2 u) + p_1 \dot{u} + p_0 \ddot{u}$$

$$\vdots$$

$$y^{(n-1)} = \dot{x}_n + p_{n-1} u + p_{n-2} \dot{u} + \cdots + p_1 u^{(n-2)} + p_0 u^{(n-1)}$$

$$y^{(n)} = -a_{n-1} x_n - a_{n-2} x_{n-1} - \cdots - a_0 x_1 + p_n u + p_{n-1} \dot{u} + p_{n-2} \ddot{u} + \cdots + p_1 u^{(n-1)} + p_0 u^{(n)}$$

ADVANCED CONTROL SYSTEM DESIGN 4

So, we will go back to the definition start from y probably so, y is nothing but x 1 plus p 0 u and then we take first differentiation let us say so that is y is y dot a nothing but x 1 dot that is simple times p 0 remember p 0, p 0 is a constant number so, it turns out to p 0 times c u dot actually. But x 1 dot where definition is nothing but x 2 plus p 1 u so, this is coming from here so I can substitute that x 1 dot here. And then, retain this p 0 times u dot so, the that is what I leave with actually here. Now, let us go to y double dot one more derivate out here, start with this expression rather so, this is x 2 dot plus p 1 times c u dot plus p 0 times t u double dot so, that is what it is actually.

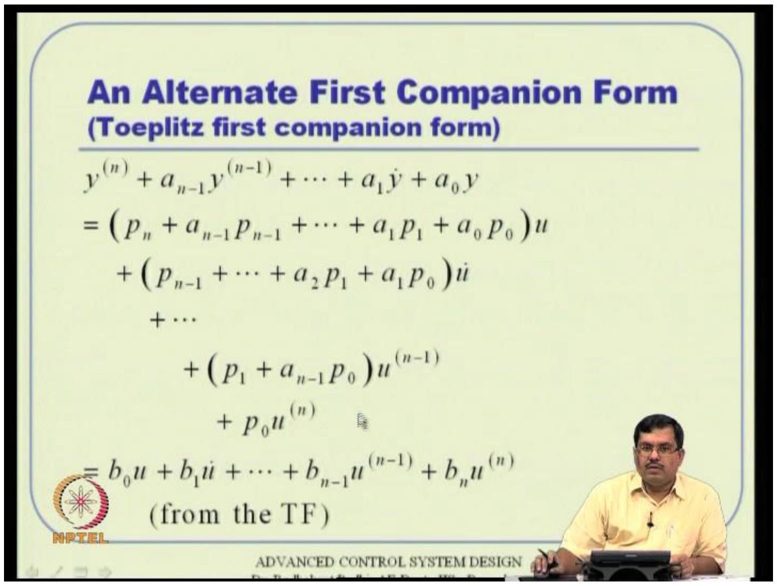
Then x 2 dot is again I will put again go back to the set of definition substitute from here x 2 dot is nothing but x 3 plus p 3 here. So, x 3 I mean that x 2 dot is a x 3 plus p 2 u plus this two I will keep it as actually. And note that when you have a y there is a u when you have y dot there is a u dot expression there is a y double dot there is a u double dot expression so, that is how it will pop out actually. So, I will carry this sequence of operation and then I will land up with y n minus 1 of derivative which will turn out to be like this, simply by observation purpose I can write something actually. Because if I say this y this is a x 1 plus this y dot this is x 2 plus p 1 plus p 0 u dot then y double dot is x 3 plus p 2 u plus p 1 u dot plus p 0 u dot u double dot.

So, I mean if you see that there is trend out here there is a y double dot starts with x 3 then there is p 2 p 1 p 0 there is a u, u dot u double dot actually. So, containing with the trend I can probably generalize this and then write it and I can also verify that actually. So, this is; this will turn out to be like this a small dot mistake probably there, is a mistake I can correct that there, is a dot out here it is an dot actually. So, Anyway so that is the just what it is; so, I will go back to I mean from here I will stop and I will substitute this; I will take one more derivative and carry out this algebra properly actually.

So, y n remember that y n will come from this definition actually, y n I will take everything to the right hand side and all this derivatives are available to me now so, I will substitute all these. And then, I will carry out with a; this algebra you can probably sit down with the equations and then try to see what is going on here actually. (No Audio From: 17:11 to 17:16) Probably I mean there is a small correction here probably this a this dot is a I mean this is not probably, not necessary that is why it actually. If I see why double dot there is a; if I see why double dot which starts with x 3 if I say why dot starts with x 2 so, if I see y n minus so derivative starts with x n there is know the dot is not necessary.

So, I will carry out and then I will put this when I take one more derivative x n dot will pop up here and then going back to the equation this is my x n dot, this is the simple definition part of it. And then I mean this section (( )) this expression will come into the right hand side alright. And then I will combine all this an long n expression actually, the last equation. So, up to that we have to slightly be patient to see what is going on actually. Now, I am ready, because I am ready with all this definitions y dot y double dot up to y to the power and the derivative. So, then I will go back to this equation what I started with that is that what my objective anyway. So, this entire equation is available to me so, I will just simply login all this expressions simply actually, I have all these expressions anyway.

(Refer Slide Time: 18:31)



An Alternate First Companion Form
(Toeplitz first companion form)

$$\begin{aligned} &y^{(n)} + a_{n-1} y^{(n-1)} + \cdots + a_1 \dot{y} + a_0 y \\ &= \left(p_n + a_{n-1} p_{n-1} + \cdots + a_1 p_1 + a_0 p_0\right) u \\ &\quad + \left(p_{n-1} + \cdots + a_2 p_1 + a_1 p_0\right) \dot{u} \\ &\quad + \cdots \\ &\quad + \left(p_1 + a_{n-1} p_0\right) u^{(n-1)} \\ &\quad + p_0 u^{(n)} \\ &= b_0 u + b_1 \dot{u} + \cdots + b_{n-1} u^{(n-1)} + b_n u^{(n)} \end{aligned}$$

(from the TF)

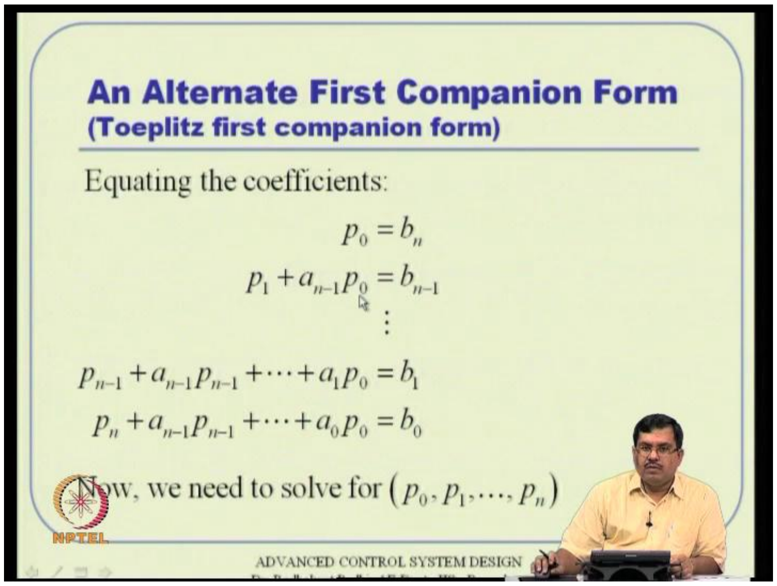
ADVANCED CONTROL SYSTEM DESIGN

So, once I plug in; this is the left hand side and the right hand side part of it, this right hand side all this expressions whatever available all, these things will also come to right hand side and think like that. What I essentially mean to say is you just plug in on to these things and try to simplify these terms and interestingly it turns out it will just retain u, u dot u I mean this powers of u actually. Once you substitute for all these variables, let us say if you can; I mean you can do this algebra here, this term; all this term that is right here that will try to kind of a; I mean this try to help you in simplifying this expression. It will have you; I mean it will lead us to some expressions like this actually.

But, this expression what we see in the first line that is nothing but this expression also that is an transfer function anyway. So, this transfer function if; I mean this expression is equal to that expression is also equal to that expression, this expression comes from the transfer function actually. So, what I do now I mean if these two have to be equal, these expression has to be equal then I can equate the coefficient actually so, whatever if I have you then that is equal to be 0 then this coefficient what I have that is equal to p 1 and similar thing all the coefficients I can equate to; and then I land up with set of equation actually.

So, if I go in a little reverse sort let us say so, this is due to the n power actually. So, what you are have; nth power actually so, what we have actually nth derivative not really cover, but this is also nth derivative here actually. So, you have this p 0 equal to v n actually then, the next equation this; what you have p 1 plus this expression is nothing but b n minus 1 like that actually.

(Refer Slide Time: 20:33)



An Alternate First Companion Form
(Toeplitz first companion form)

Equating the coefficients:

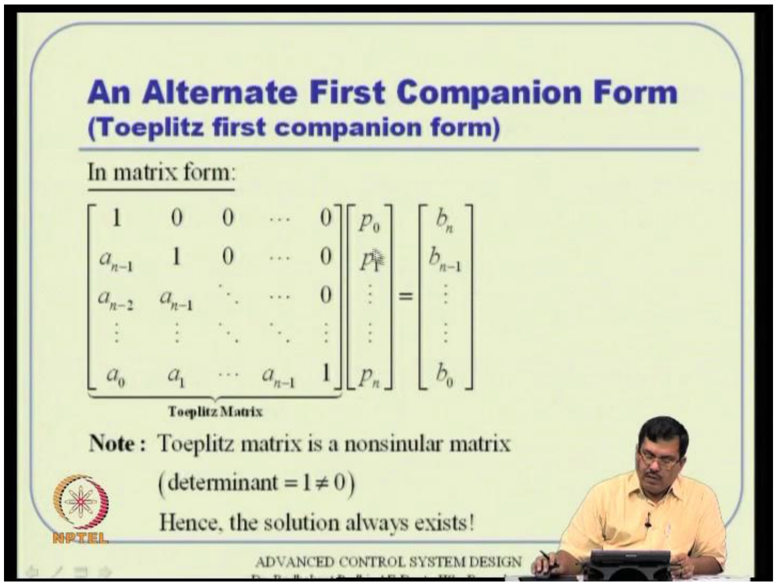
$$p_0 = b_n$$
$$p_1 + a_{n-1}p_0 = b_{n-1}$$
$$\vdots$$
$$p_{n-1} + a_{n-1}p_{n-1} + \cdots + a_1 p_0 = b_1$$
$$p_n + a_{n-1}p_{n-1} + \cdots + a_0 p_0 = b_0$$

Now, we need to solve for $(p_0, p_1, \ldots, p_n)$

NPTEL
ADVANCED CONTROL SYSTEM DESIGN

So I will constitute a I mean self construct a set of equations. Now, this set of equations if you see this has this has given me sufficient number of equation that means if you count number of equations, it is actually 1 to n plus 1 that means what I have is n plus 1 equations and I also have n plus 1 variables starting from p 0. p 0 p 1 p 2 up to p n that is n plus variables so, when n plus 1 set of equations for n plus 1 variables actually so, I can essentially solve this. So, if I want to solve it in a vector matrix form, these equations I will first put it in vector matrix form and that is what I will do actually first equation is simply p 0 equal to p n so that is one everything 0 is equal to p n. The second one is a n a n minus 1 p 0 plus 1 times p is equal to p n minus 1.

(Refer Slide Time: 21:25)



An Alternate First Companion Form
(Toeplitz first companion form)

In matrix form:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_{n-1} & 1 & 0 & \cdots & 0 \\ a_{n-2} & a_{n-1} & \ddots & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_0 & a_1 & \cdots & a_{n-1} & 1 \end{bmatrix}}_{\text{Toeplitz Matrix}} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} b_n \\ b_{n-1} \\ \vdots \\ \vdots \\ b_0 \end{bmatrix}$$

Note : Toeplitz matrix is a nonsinular matrix ($\text{determinant} = 1 \neq 0$)

Hence, the solution always exists!

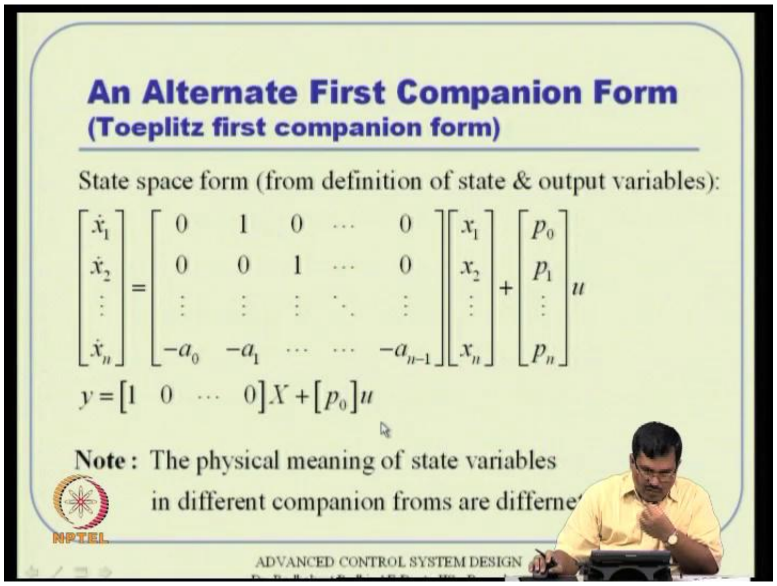
ADVANCED CONTROL SYSTEM DESIGN

So, that is how it is actually n minus 1 p 0 plus p 1 is equal to p n minus 1 that to I can construct this actually. So, this is the set of equation and here that is why this name toeplitz comes, because this is particular if you observe this matrix this is lower triangular matrix essentially. And these specific form what you have is called as toeplitz matrix actually. The interesting property out here is actually not necessarily only aimed lower triangular matrix, but the diagonal elements are all one so that is the difference actually. This is a lower triangular matrix, all other elements can arbitrary numbers let us say, but the diagonal elements have to be all one actually.

Now, as you know if I really carry out this determinate evolution of this any triangular matrix, the determinaties is nothing but product of diagonal matrix actually. And since all diagonal element are just one out here so, the determinant are guaranteed to be one. And hence it is always guaranteed to be non 0 that means no matter whatever values by a; for a 0 a 1 up to n minus 1. Whatever values I have; this matrix is guaranteed to give me a (( )) I mean this equation is guaranteed give me a solution, because this matrix is guaranteed to be non singular actually, solution always exists.

So, then I can always say not this matrix I not get a solution or this is already you can say I can very well use (( )) and elimination is already in the triangular form actually so, gauss and elements that are quicker to get that anyway. So, this set of equation essentially give me the solution hence I am done, because once I know this as I told in the beginning from the simple definition. We have already constructed the straight phase form actually, y is already known to us C matrix is given from here and D matrix is just p (( )). And then, this points of equation x 1 dot to x n already gives us the form that we desire actually. So, that is combining them altogether after I get the solutions p 0 to p n.

(Refer Slide Time: 23:27)



An Alternate First Companion Form
(Toeplitz first companion form)

State space form (from definition of state & output variables):

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_0 & -a_1 & \cdots & \cdots & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} X + \begin{bmatrix} p_0 \end{bmatrix} u$$

Note: The physical meaning of state variables in different companion froms are differe

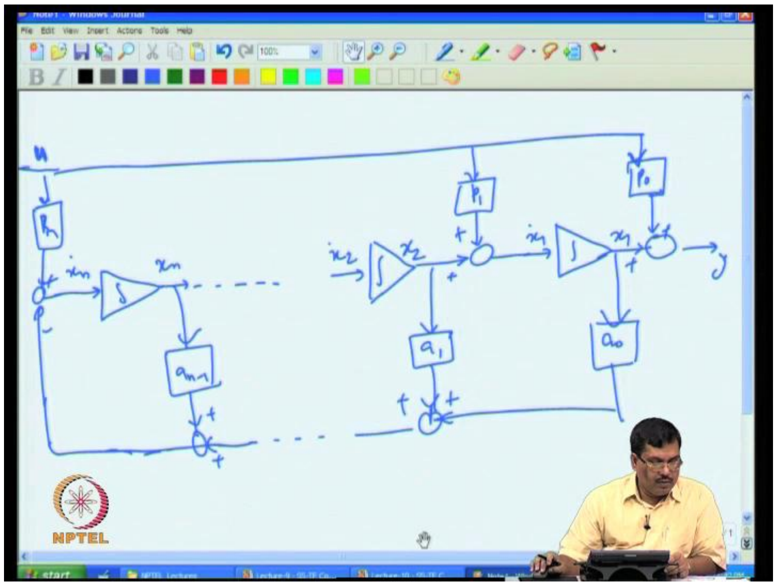
ADVANCED CONTROL SYSTEM DESIGN

I can simply write it, this differential equations in state phase form to be something like this actually. And it interestingly turns out that this is also like a first companion form, if you remember the first companion form this is all super diagonal elements just 1 1 1 and then we have this last row some numbers. And then, the slide difference is this first companion form had all 0 0 0 in the last element was something, but here it is all something or other is there some numbers are there, all over the director actually. Other than that I think all the things remain same so that is why it is also called an alternate first companion form or sometimes is called is a toeplitz first companion form. Now, even though there is first companion form if you write regular first companion form that we discuss or controllable canonical form rather that is called that way.

Then, the state variable; I mean the matrices can be fairly similar that means they are very close to each other the appearance wise actually. How are the numbers corresponding to that will be different as well as the physical meaning of the state variables will be different in two different forms. That is in general that is true if you really utilize the system in form one and form two then obviously these systems; I means this definitions of state variables are different, depending on the form you are selecting actually. So, that should that you keep it always in mind actually.

Now, if you really want to do a block diagram realization, let me do that quickly out here this block diagram realization turns out to be rather toeplitz straight forward here. (No Audio From: 25:09 to 25:15) What we have here actually, we will go back to this set of equations and try to construct something here actually. Just remember this set of equation y equal to x 1 p 0 u where x 1 dot is we are having that kind of a thing actually. So, maybe so for my benefit let me do this way, this is so remember, this is (( )) sorry so, these are the set of equations that you want to represent in block diagram form actually. So, probably I will start with some y definitions let us say this is x 1plus p 0 u.

(Refer Slide Time: 25:58)



So, I will start with some integrators which will give me let say, this will take x 1 dot it will give me let say x 1 that is the form I am looking. And then, remember there is y what I am talking about y is nothing but this x 1 plus p 0 u basically. So, x 1 is already available so I have to get a p 0 u somewhere actually so, this is plus, this is plus and then I will talk about something like a multiplication by p 0. And there is a u, this is a u out here control term so, let me take it all the way and then multiply that and then pass it through actually so that is what I will get y essentially, this is a our; like a that we are done actually. Now, we will come back to this x 1 dot is nothing but x 2 plus p 1 u so, I will go that way. Then, this is a; x 1 dot is available here so I have to construct this x 1 actually x 1 dot out here.

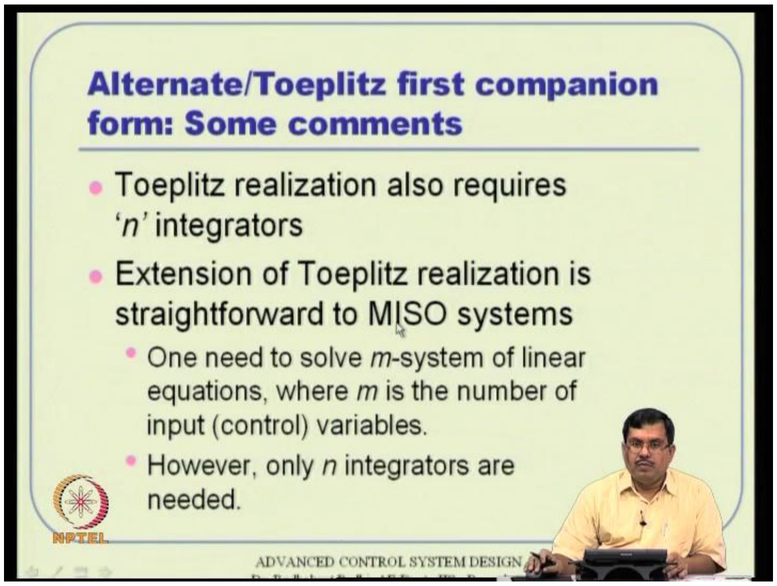
So, then I will talk about p 1 it is pass through p 1 block, u is already available actually so, this is p 1 block will come through that. And then, there is something like x 2 basically right so, if you see this expression one more time, this is x 2 plus p 1 actually here. So, I got this p 1 u then I have to construct which a; I have to take this x 2 out here actually. And similarly, I can construct see if x 2 then obviously it has to pass through integrator actually then I will get x 1 x 2 dot rather here, it is not double dot x 2 dot actually. So, that way I will be able to construct, keep constructing the top portion of it and then if I come back to this side of the story, I have to ultimately construct something like x, this has to start with let say x n.

And then obviously the input has to be x n dot then if it is x n dot remember that it is from the definition point of view that x n dot is all this actually. So, these are all; these entire thing has to be realized actually, up to x 1 minus 1 dot there is no problem, but this entire thing has to be realized. So, what I will do? I will take this x 1 is available to me so I will pass bit through this 0 out here and x 2 is available to me out here so, I will take it there and pass it through this a 1 out here. So I will construct this things actually that way. And then, I will have summation blocks actually all over, I will have summation blocks allover and then I will take it a; this one and then it will go and if you talk about there is a final summation blocks somewhere which will connect it.

And this will give me negative sign and this is; this will come to some sort of p n, this is p n and this will come connect that actually, this is positive side so, this will construct x n dot actually. So, remember this even x n I have to take it through a n minus 1 and then I have to construct some sort of summation block out here, because it is science actually. So, if you go back to this and try to correlate this x n dot is nothing but p n time u plus summation of all this is with an negative sign actually so that is what we have done actually, p n time u it has come from this side and then summation of all this that we are talking about it actually.

So, that is why; that is how we construct a block diagrams (( )) actually so, it is not that difficult to realize block diagrams actually. And lets carry on move forward with that so, this is what is called a; so a toeplitz form actually, in block diagram we are drawn actually anyway. Now, some comments out here if you see this differential equation or this block diagram also or whatever you have here essentially they represent an integerators anyway, that is given in this differential equation actually so, that is a; that is easy to say actually.

(Refer Slide Time: 30:31)



Alternate/Toeplitz first companion form: Some comments

- Toeplitz realization also requires '*n*' integrators
- Extension of Toeplitz realization is straightforward to MISO systems
  - One need to solve *m*-system of linear equations, where *m* is the number of input (control) variables.
  - However, only *n* integrators are needed.

ADVANCED CONTROL SYSTEM DESIGN

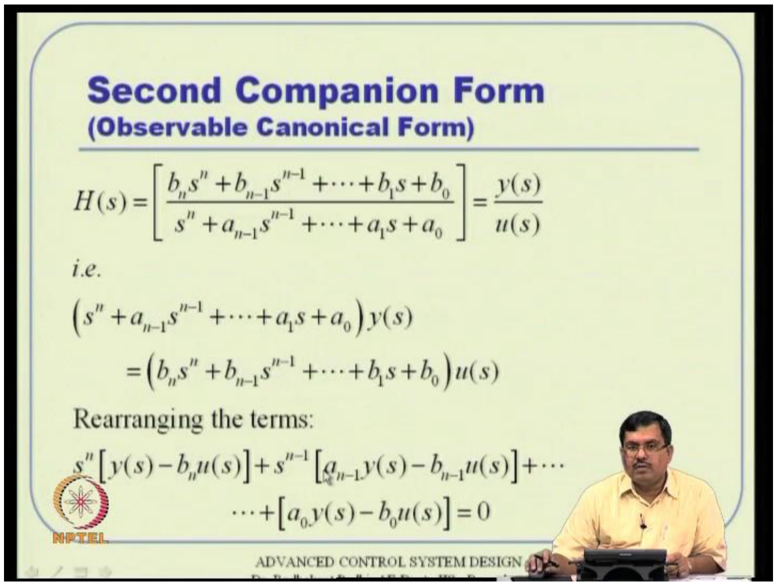
So, no matter what realization you talk about if the transfer function is of nth order denominator polynomial then, you certainly nailed a number of integrators like n actually so, you need an integrators to realize that actually. Now, extension of this why this particular form is kind of interesting is because, if you have MISO system that multiple input, but single output system. Then, what you; I mean the extension is fairly straight forward, because you can always go back to this form of definition. So, you start writing another set of equation, lets a say for an example like an always write, this is like p 0 u 1 plus q 0 u 2 so, this is like p 1 u 1 plus p q let us say q 1 u 2 like that actually.

So, this is q 2 u 1 plus q 2 q 2 like that so, I will set a construct another set of a; I mean another set of a constants so, parameters so that I need to solve actually. So, the extensions and the analysis will be very straight forward, I will land up with a two sets of equations like that or both the sets will lead me to this toeplitz matrix and all that where solution is guaranteed to exist actually. All these things are very nice to; I mean this particular form is very nice to extend to multiple input, but single output systems actually. So, I mean going by the argument that we have just discussed with essentially one need to solve m system of linear equations, where m is the number of inputs in control variables actually, if you have m control variables then you have m such systems matrix equations it is all for actually.

However, if you know the number of states are not changing actually, that means we still need an integrators actually. And remember, if you remember the last class that we reviewed in the (( )) I mean in the first a couple of minutes back in this lecture as well. If you have this multiple output case, but single input case that going by the first companion form that was easy to generalize actually. Only the output equations were changing number of states were remaining constants and the same that means we still need that integrator actually. So, I mean the message here is we have this multiple input, but single output we still need an integrator and the reverse case where we have a single input, but multiple output we can still realize this system with an integrator actually.

Or the all the ball game changes when you have MISO system that is multiple input, multiple output system then, that is no more true, we will just discussed we have a few comments later in this lecture actually.

(Refer Slide Time: 33:22)



Second Companion Form
(Observable Canonical Form)

$$H(s)=\left[\frac{b_n s^n + b_{n-1}s^{n-1}+\cdots+b_1 s+b_0}{s^n+a_{n-1}s^{n-1}+\cdots+a_1 s+a_0}\right]=\frac{y(s)}{u(s)}$$

i.e.

$$\left(s^n+a_{n-1}s^{n-1}+\cdots+a_1 s+a_0\right)y(s)$$
$$=\left(b_n s^n+b_{n-1}s^{n-1}+\cdots+b_1 s+b_0\right)u(s)$$

Rearranging the terms:

$$s^n\left[y(s)-b_n u(s)\right]+s^{n-1}\left[a_{n-1}y(s)-b_{n-1}u(s)\right]+\cdots$$
$$\cdots+\left[a_0 y(s)-b_0 u(s)\right]=0$$

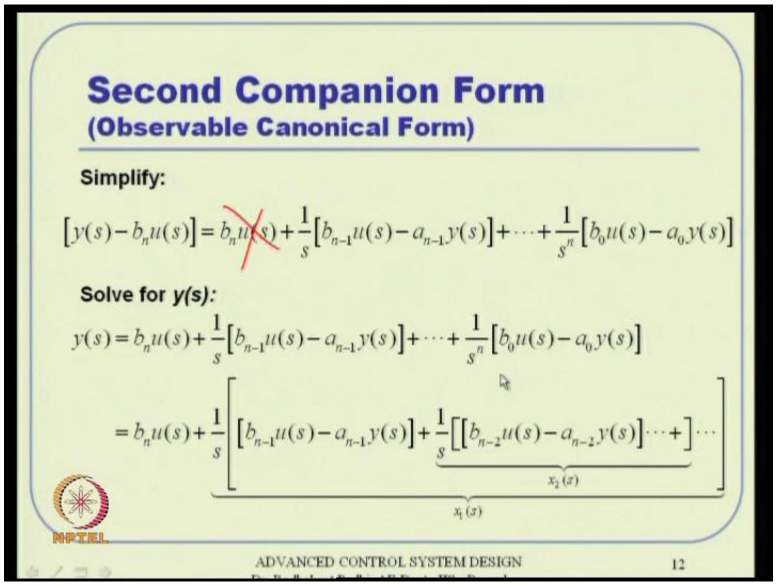
ADVANCED CONTROL SYSTEM DESIGN

Before moving at with the generalization of a MISO thing, let us talk about the other form that is available to us actually and that is a second companion form or observable canonical form. And this particular form as I told you in the beginning is useful when we have to design some sort of an observer or a filter actually, that is some observability and all this properties are nicer in this form actually. So, how do you go about and do that so, we start with this same transfer function, in generic transfer in general that is again the same order. So, we have this proper transfer function what we start with; and then we again multiply this two sides and then we talk a will; now, start looking at this equation in Laplace domain itself in a slightly different way actually.

How do you do that now, let us arrange this terms this way actually now, I take all the terms to the left hand side and I will start taking the powers of s n, s n minus 1 like that actually so if I take these two terms this two entire term this expression goes left side. So, s 2 the power n is here so. I just take that one common then y of s is minus b n time u of s so that will pop off here actually. So, I am writing everything so this kind of polynomial of in terms of s actually so, s to the power n multiplied by some polynomial and s to the power n minus 1 multiplied by some polynomial like that actually, this is the same equation anyway actually.

Now, after doing that, I just want to extra this first term actually, this first term is nothing but everything to write inside divided by x 2 the power n actually. So, when I start doing that then x 2 the power n is the highest one so, the first term will I mean pop of a 1 by s actually just next term will be 1 by a square actually.

(Refer Slide Time: 35:16)



Second Companion Form
(Observable Canonical Form)

Simplify:

$$\left[y(s)-b_n u(s)\right] = \cancel{b_n u(s)} + \frac{1}{s}\left[b_{n-1}u(s)-a_{n-1}y(s)\right] + \cdots + \frac{1}{s^n}\left[b_0 u(s)-a_0 y(s)\right]$$

Solve for *y(s)*:

$$y(s) = b_n u(s) + \frac{1}{s}\left[b_{n-1}u(s)-a_{n-1}y(s)\right] + \cdots + \frac{1}{s^n}\left[b_0 u(s)-a_0 y(s)\right]$$

$$= b_n u(s) + \frac{1}{s}\underbrace{\left[\left[b_{n-1}u(s)-a_{n-1}y(s)\right] + \underbrace{\frac{1}{s}\left[\left[b_{n-2}u(s)-a_{n-2}y(s)\right]\cdots+\right]}_{x_2(s)}\cdots\right]}_{x_1(s)}$$

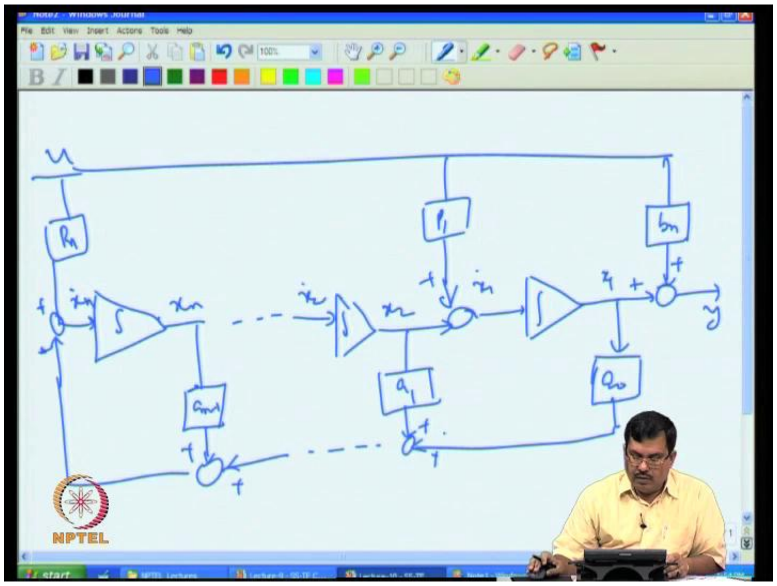
ADVANCED CONTROL SYSTEM DESIGN 12

And that is what is happening, (( )) this double type actually. So, if we to solve this equation here these entire expression here, you just take it a I mean if you want to extract this expression first from rest of the things. So, you are taking everything to the right hand side dividing by x to the power n actually. So, this term is not required at this point of time actually. So, then this 1 by s I mean multiplied by that next term will be 1 by a square multiplied by that particular term what we have here, I mean the next term what we have here and things like that actually. Now, let me analysis this expression a little bit carefully so, this is actually if I see this expression now, from here I will try to; I am just trying to extract what is y of s actually.

So, this expression; this b n times of u of s will come right hand side from here onwards, not here actually. So, this is what it is plus 1 by s into all this terms plus 1 by s into another term, 1 by a square in that term all the way up to a actually. So, if I start taking 1 by s common from rest of term I can do that, because they are all higher powers of s actually first is 1 by s next is 1 by s square next is 1 by s q like that actually. So, if I take 1 by s common here for all the expressions then, if I take next set of equations I can take again 1 by s common for next set of equations like that. So, I am just interpreting this in a nested way actually so, 1 by s into multiplied by everything else.

Then, the next term onward it is still one ways multiplied by everything rest of the terms like that so, in the nested way I am interpreting that. Now, I will start defining terms actually so, what I am defining? I am; I start defining this entire what I see here there is nothing but, let say x 1 is x 1 of s is the entire term what I see next in the bracket is nothing but x 2 s like that actually. So, if I start doing that so, let us; I mean with this expression let us try to draw the block diagram again then, we will that picture will start emerge a little bit more actually. So, remember that this is the expression that we will put it in a block diagram form actually so, let us draw this I mean start doing that actually.

(Refer Slide Time: 37:53)



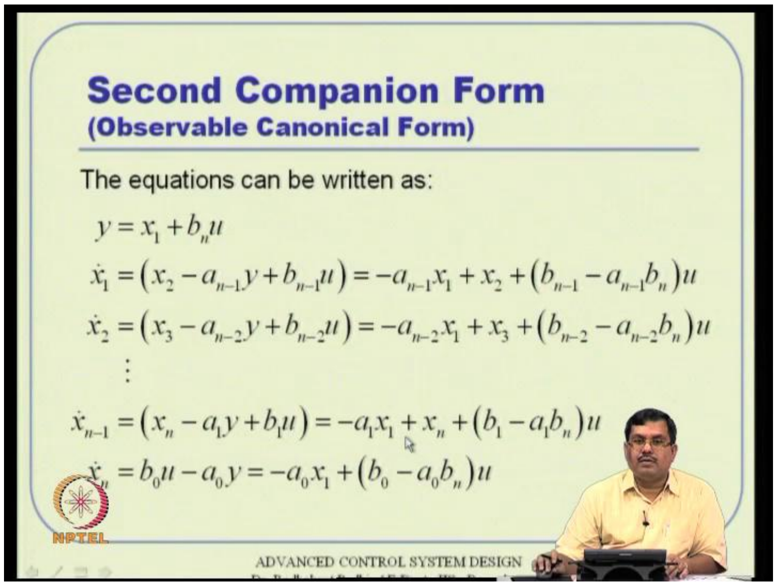
So, what we have here is essentially like a again that I will put that top side as some click the d u signal is available so, wherever I want I can get it. And then this is like again this a; I will put that s b n and then I will take it as addition of two signals then I want to construct y out here actually so, let us see what is y out here actually. Y is nothing but b n times u of s what you see here plus x one of s actually, the entire term remember is x 1 actually. So, b n times u of s is already available to me this u this is b n so that is available then, what I have is simply x 1 here actually so, obviously this is x 1, so if I pass it through a I mean an integrator then this has to be x 1 actually.

But remember this x 1 dot is nothing but if I go back to that expression basically so, this x 1 dot (( )) I mean if you see that this rest of the term is x 1 remember there is an integer 1 by s turns integration actually. So, if I just take out the integration then, the rest of the term is nothing but x 1 dot actually, if I take this term as x 1 dot then pass it through integration 1 by s then I will get x 1. So, this entire expression what I see is nothing but this term plus this two of x actually, whatever I have this is a x 2 of s essentially. So, like that I can; let us say I can construct it quickly so, what I have is; let us x 1 dot. So, I have this a; it is a p 1 terms then it will go there and then it will have this; and then this is a this will have x 2 and obviously there is a integration there then that will have like x 2 dot like that actually.

This series is anyway same for us so, the things if you have one serious of equations you will have x 1 dot pass it through integration, it will have x 1 and under then x 2 dot pass through integration x 2 like that actually anyway. So, if I see this then ultimately I have to look forward is; can I realize this x n dot that is where it will x n dot integration passing through actually. Then this; I mean this particular thing will give me what is going on here actually so, this I will take it as sum a n minus 1 and then it will pop up the something like this. And then this loops on the top side will pass it through p n and here is 1 summation out here this plus thing this will become everything else become a minus thing here.

And then, I will proceed through this case actually, this is a whole chain of equation out here actually in a way. So, if I this is how; now, x 1 I have a 0, a 0 turns out to be like this and then x 2 when I have this is; this will be like I will take like a 1, this is a 1 out here so these two guys will be you can summit up here like that actually, you can turn out the way like that. So, rest of the things probably you can fill it up all in a written sort of thing, this is the type of above diagram that we talking about here actually. If you look at this equations essentially, what are you getting? I mean either this equations right here or through the above diagrams out here, whatever way you can interpret that way. Then, what you are getting here is something is very straight forward.

(Refer Slide Time: 42:01)



Second Companion Form
(Observable Canonical Form)

The equations can be written as:

$$y = x_1 + b_n u$$
$$\dot{x}_1 = (x_2 - a_{n-1} y + b_{n-1} u) = -a_{n-1} x_1 + x_2 + (b_{n-1} - a_{n-1} b_n) u$$
$$\dot{x}_2 = (x_3 - a_{n-2} y + b_{n-2} u) = -a_{n-2} x_1 + x_3 + (b_{n-2} - a_{n-2} b_n) u$$
$$\vdots$$
$$\dot{x}_{n-1} = (x_n - a_1 y + b_1 u) = -a_1 x_1 + x_n + (b_1 - a_1 b_n) u$$
$$\dot{x}_n = b_0 u - a_0 y = -a_0 x_1 + (b_0 - a_0 b_n) u$$

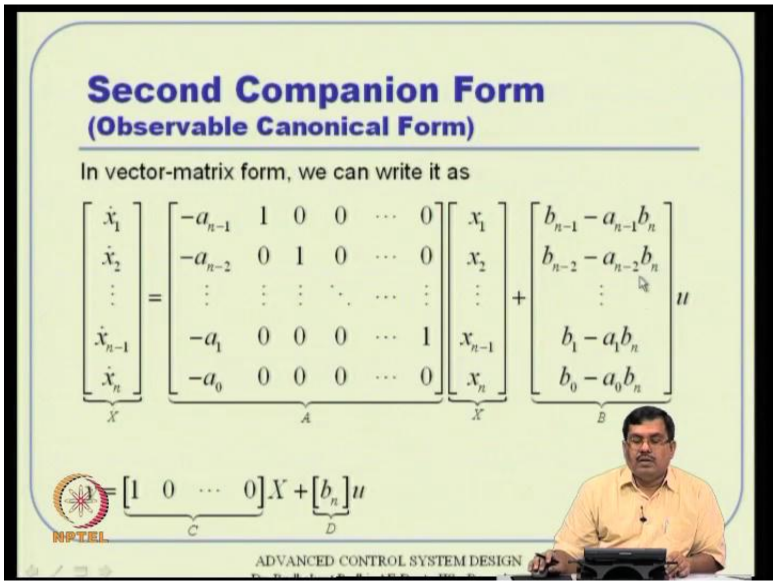
ADVANCED CONTROL SYSTEM DESIGN

So, y equal to x 1 plus v and u that is fine, then x 1 dot is essentially x 2 this is this is the x 1 dot remember that the entire expression that is there. And then this expression out here this expression is nothing but x 2 actually entire thing actually, this is x 2 of s so that is a; so, if I have x 1 dot I have to get up for these term as this term and this term actually essentially. So, x 1 dot is nothing but x 2 minus a 1 by plus b n when b n minus 1 here actually so, x 1 dot is this 1 minus this plus x 2 basically, that is what I have it here x 2 first I write x 2 and then these terms are there actually.

So, then a I go ahead and try to simplify this expression so, I mean this; if I substitute y, where y is already available to me then I can expand this expression this y is nothing but that; so, this will turn out to be that minus a n minus 1 into x 1 that is the term what you are looking for then, n minus 1 into b n term actually so that is the term that you are looking for actually. So, these; I may so just by putting this expression for y whatever we have, we will be able to expand it that way actually. So, what are you learning off we are y expression that way we are having x 1 dot x 2 dot all the way up to x n minus 1 dot popping out that way actually. And x n dot turns out to be simply the; that actually because x n dot if you see this equation, this is probably easy to see from the block diagram this is x n dot actually.

So, what you see as x n dot out here, this is a x n dot term here and this x n dot term this x n dot out here what you see is nothing but this two terms actually p n times u then minus times are all these expressions actually. So, what you get here is actually like x n dot is all these two expressions b 0 times u minus a 0 times y actually. So, anyway so you can looking back to that I mean this; what I mean is wall this expressions available to you now actually. Because y is again I will substitute that that way whatever expressions I have here and then, if I substitute that that is how I land up with. So, what you what observe here is a nice form out here see x 1 dot contains x 1 and x 2, x 2 dot contains x 1 and x 3, x n minus 1 contains x 1 and x n like that actually.

(Refer Slide Time: 44:49)



Second Companion Form
(Observable Canonical Form)

In vector-matrix form, we can write it as

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{n-1} \\ \dot{x}_n \end{bmatrix}}_{\dot{X}} = \underbrace{\begin{bmatrix} -a_{n-1} & 1 & 0 & 0 & \cdots & 0 \\ -a_{n-2} & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ -a_1 & 0 & 0 & 0 & \cdots & 1 \\ -a_0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix}}_{X} + \underbrace{\begin{bmatrix} b_{n-1} - a_{n-1} b_n \\ b_{n-2} - a_{n-2} b_n \\ \vdots \\ b_1 - a_1 b_n \\ b_0 - a_0 b_n \end{bmatrix}}_{B} u$$

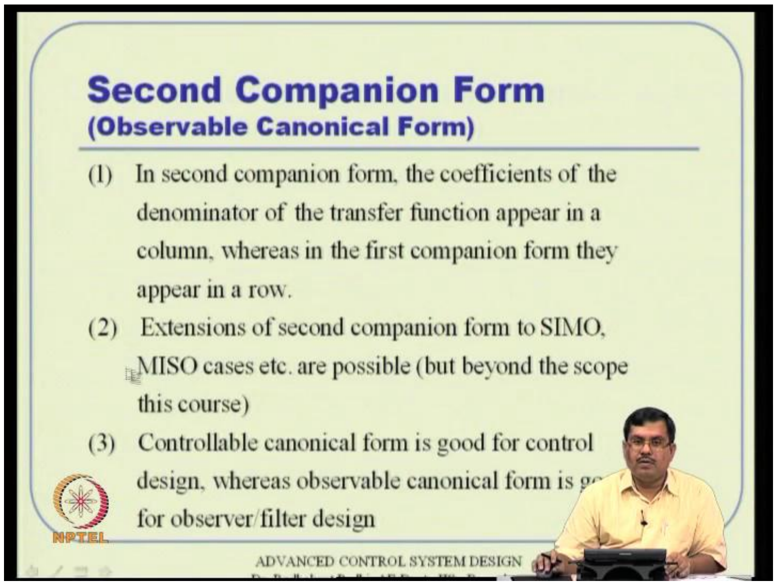
$$y = \underbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}}_{C} X + \underbrace{\begin{bmatrix} b_n \end{bmatrix}}_{D} u$$

ADVANCED CONTROL SYSTEM DESIGN

So, then you can try to put it that in the matrix vector, matrix forms sort of thing what it turns out here because all this dots will contain x 1 actually. So, that means this column vector is non empty these are all these column vectors will pop out instead of a row, what you get is a entry scene first column rather actually. That is the difference between observable canonical form and controllable canonical form here actually. And then, this correspond B matrix is full lecture because everything is function of u also basically so, all this early entries will have some numbers all the way actually.

And then simply from definition y is nothing but x 1 plus b n times u, but will give you the C and D matrix directly basically so that what is happening actually. So, essentially in this observable canonical form we will be able to realize this A B and C D matrix that way actually. The difference is in controllable canonical form we had row of entries, in observable canonical form we have a column of entries in the; a matrix actually. And these are all so nicely related to this like, if you see this controllability observability check that will sorry, little bit late we will see that this a; this I mean appearance of element come handy actually. For calculation of these; for checking this conditions are necessary like the rank of the matrix are number actually, we will see that later what happens.

(Refer Slide Time: 46:16)



Second Companion Form
(Observable Canonical Form)

(1) In second companion form, the coefficients of the denominator of the transfer function appear in a column, whereas in the first companion form they appear in a row.

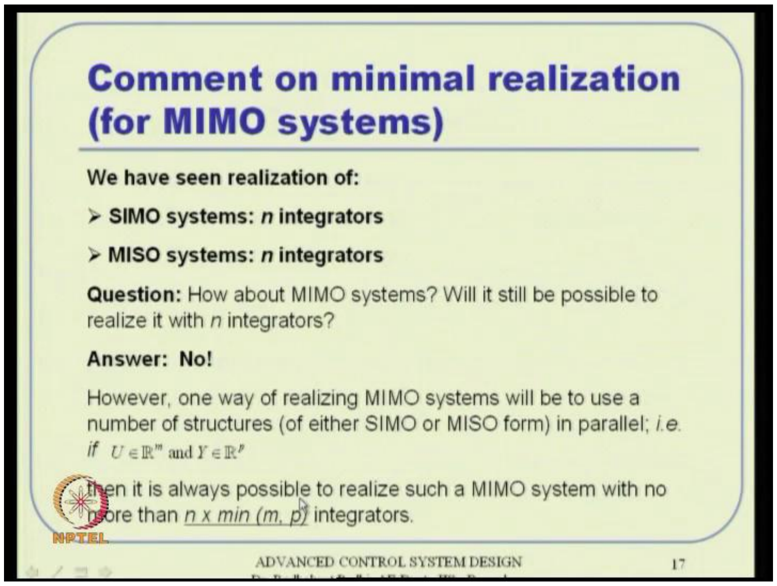
(2) Extensions of second companion form to SIMO, MISO cases etc. are possible (but beyond the scope this course)

(3) Controllable canonical form is good for control design, whereas observable canonical form is go for observer/filter design

ADVANCED CONTROL SYSTEM DESIGN

So, this is the observable canonical form actually. Now, this first point I have already talked that they appear in a column instead of a row, an extension of the second companion term to single input multiple output case are possible. This particular form what we have talked about, is possible to extend that for single input to multiple output as well as multiple input to single output cases etcetera, essentially these two. For these we will not study in detail these are probably beyond the scope of these particular course actually. I will give a reference at the end where people are interested they can see at this for the details and all that actually. So, as I told in the beginning the controllable canonical form is good for control design whereas, the observable canonical form is good for observer to filter design actually.

(Refer Slide Time: 47:07)



Comment on minimal realization (for MIMO systems)

We have seen realization of:

- **SIMO systems: *n* integrators**
- **MISO systems: *n* integrators**

Question: How about MIMO systems? Will it still be possible to realize it with *n* integrators?

Answer: No!

However, one way of realizing MIMO systems will be to use a number of structures (of either SIMO or MISO form) in parallel; *i.e.* *if* $U \in \mathbb{R}^m$ and $Y \in \mathbb{R}^p$

then it is always possible to realize such a MIMO system with no more than $n \times \min(m, p)$ integrators.

ADVANCED CONTROL SYSTEM DESIGN 17

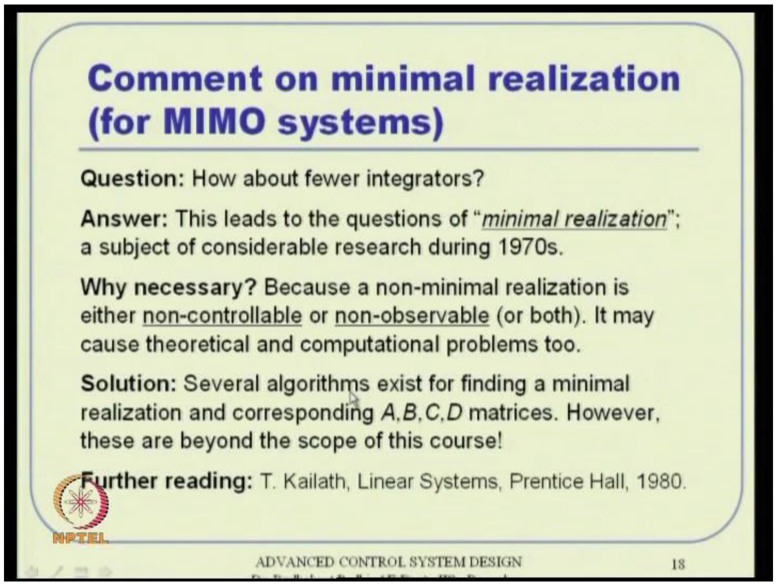
Now, some final comment sort of thing so, what you have seen you have seen both single input single output systems I mean the SIMO sort of thing that is the; that is why we started all these realization. And we essentially started with nth order polynomial in the denominator and we saw that they are equal and integrate. However, there are realizations possible that was single out single input and multiple output systems, we can still realize the system with an integrators and for multiple input what single output systems, we can still realize the system with an integrator also basically.

Now, the question is what about this MIMO systems, multiple input multiple output systems will it be possible to realize that as well with an integrators. Now, remember that that is not a straight forward answer actually and unfortunately it turns out the answer is really know, that means it is still not possible to do that with the integrators along actually. Now, then what the I mean immediate answer that comes to mind so, one way to realize this MIMO systems will be used the number of structures like whatever forms you see like well c is either SIMO form or MISO form, will use a number of them in parallel.

And especially, if u belongs to this m dimensional vector and y belongs to this y dimensional vector that means there are m elements of u and p elements of y then, it is always possible to realize such a MIMO systems in something like n into; or n is a number of state variables in SISO systems basically that is, what it is? N into minimum of this two basically that either I will take the minimum of this dimension of u and dimension of p. And then if I construct n; I mean remember for one of that form I will choose, I will tell you whatever form that is as I mean that gives me the minimum number.

And that is either I will choose something like form which is extendable to m kind a multiple output case easily, all multiple input case easily that who is using that particular form; I will simply I mean it is obvious that you need n into that particular number minimum of these m into p that mean integrators to realize this actually. so that means state variables are necessary to realize this MIMO systems in general. However, there is a problem out there actually, because that particular thing turns out to be slightly over actually it I mean we probably do not need that many.

(Refer Slide Time: 49:48)



Comment on minimal realization (for MIMO systems)

Question: How about fewer integrators?

Answer: This leads to the questions of "*minimal realization*"; a subject of considerable research during 1970s.

Why necessary? Because a non-minimal realization is either non-controllable or non-observable (or both). It may cause theoretical and computational problems too.

Solution: Several algorithms exist for finding a minimal realization and corresponding *A,B,C,D* matrices. However, these are beyond the scope of this course!

Further reading: T. Kailath, Linear Systems, Prentice Hall, 1980.

ADVANCED CONTROL SYSTEM DESIGN 18

So, the question is it possible to do it with a smaller number of integrators fewer integrators and the answer is the yes you can do that actually. So, that essentially leads to the question of minimal realization or sometimes minimum realizations some books write that way. So, what I mean is like this is we can certainly do, the question is can we do it even with fewer integrator. Then, however why is it so important than why is it necessary? Why are you interested in even looking at the question actually? And the answer to answer to b that because a non minimal realization. Suppose you take more number of integrators then necessary then, it can either lead to some saying some very odd situations like a non-controllability, non-observability or both actually that means either you cannot design a controller or you cannot design both actually.

So, in addition to this numerical complexity remember this that also turn into comes to picture, because they say the number of integration integrators essentially the number of state variables that you need. And they remember the number of state variables essentially is n then the dimension of a matrix turns out to be a by a n by n. So, if you have extra number of state variables, the dimension of the A matrix B matrix and all that the stars are expanding in a big way actually. So, if you have instead of three states let us say five states then you have instead of a 3 by 3 matrix will talk about 5 by 5 matrix.

And essentially the number of elements goes from 9 to 25 so, it is a lot of competition actually so, we can actually; I mean can we avoid that too many computation that is one thing actually. But more important case is we will also simultaneously loose controllability and observability which is even worse actually. So, it may essentially cause theoretical and computational problems, I can just explain that means if you have more number of the dimension of the problem increases the dimension of the various matrixes increases. And then, we are learning up with doing too many competitions and necessary for whatever control design controllability analysis and everything actually.

What more than that these are more serious issues for examples, if the system is not controllable then; there we cannot design a controller no matter what method you use actually. Similarly, that the realization turns out to be non observable then we cannot design an observer or we cannot design a filter actually and these are more serious issues actually. So, there are a solutions, there are a people who thought in 1960s let 60s and 70s as well and there are several algorithms exists for finding minimum realization. That means start with probably a little higher order and then you try to eliminate this number of variable that are not necessary. And you are essentially looking for some variables which are linear combinations of other variables actually so that you can eliminate them.

So, these algorithms do exists for finding minimal realization and then the corresponding A B C D matrixes as well, but all these details will not sorry, in this particular course actually. Anybody who is interested for further reading, they can always refer to this book it is a very seminal book anyway, this is a written by Thomas Kailath linear systems very seminal book for linear systems actually. So, anybody interested in doing further and further studies they can probably read this book actually. Very nice book what then it is also mathematically rigorous that means, if somebody is not very strong in mathematics and think like that this book may not be that much readable actually. So, with that small comment I can probably stop here for these lecture.

Thanks a lot for the tension, next class we will move further with some different topic actually. So, assuming that this conversation between transfer function and state phase and vice versa is possible in a variety of ways rather forms also, not discuss. Anybody interested, as I told it, they can always refer this Thomas Kailath book actually. Thanks a lot.