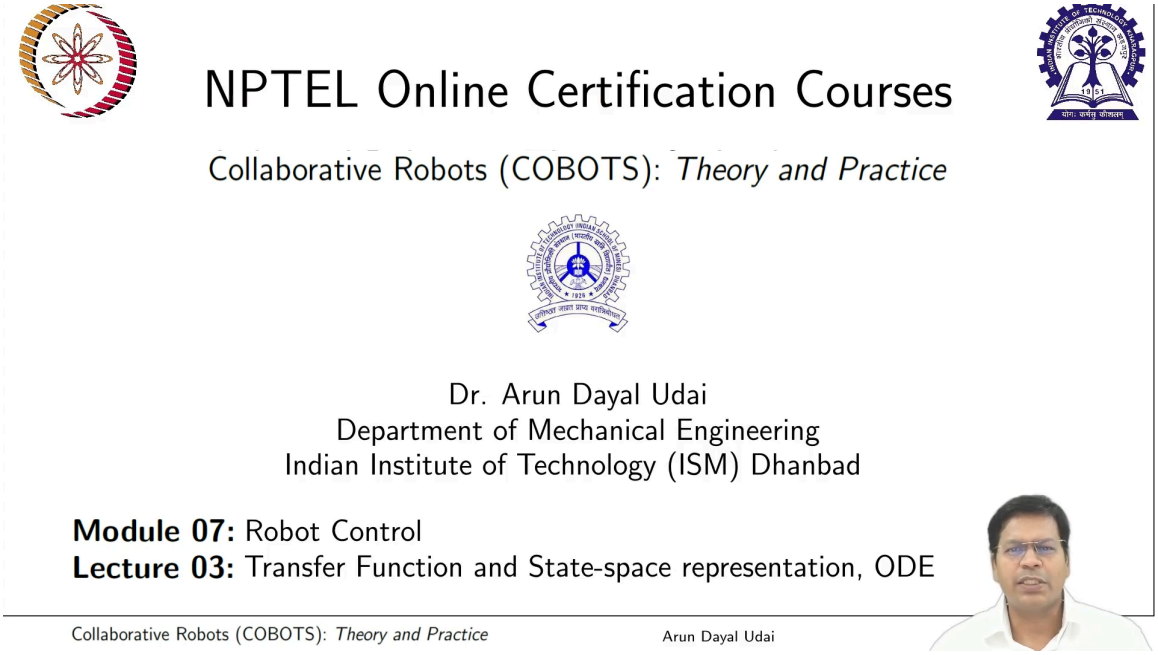
NPTEL Online Certification Courses
COLLABORATIVE ROBOTS (COBOTS): THEORY AND PRACTICE
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 07
Lecture: 30

Transfer Function and State-space representation, ODE



Hi, in this last class, I discussed the behaviour of second-order linear systems. We saw how poles affect the behavior of your system. So, in today's class, we'll discuss two additional tools that are quite commonly used in the context of controls. Those are transfer Functions and State-Space representations of such systems, and we'll see how these tools can be used to understand the behaviour of your system at hand. So, let us start.

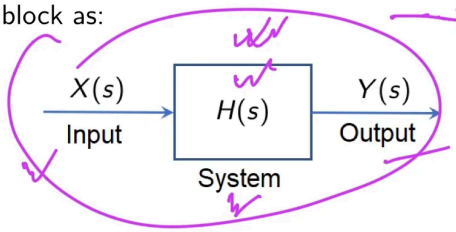
Transfer Function Representation

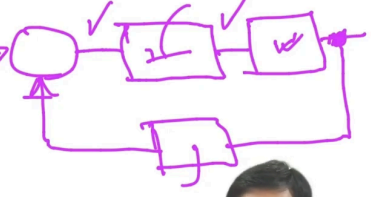
The transfer function $H(s)$ is the linear mapping of the Laplace transform of the input, $X(s) = \mathcal{L}\{x(t)\}$, to the Laplace transform of the output $Y(s) = \mathcal{L}\{y(t)\}$

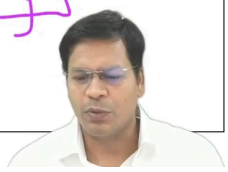
$$\Rightarrow H(s) = \frac{Y(s)}{X(s)}$$

The term is often used exclusively to refer to *linear time-invariant* (*LTI*) systems.

Represented using block as:







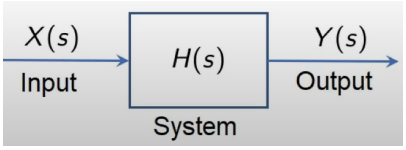
So, let us look at the transfer function representation first. So, the transfer function H(s) is a linear mapping of the Laplace transform of the input, that is, X(s), to the Laplace transform of the output, that is, Y(s). So, Y(s) is the Laplace transform of Y, which is a function of time. Similarly, X(s) is the input Laplace transform function of X(t). So, The transfer function is defined as Y(s) by X(s), that is the output Laplace to the input Laplace function.

$$H(s) = \frac{Y(s)}{X(s)}$$

So, this term is quite often used extensively for referring to linear time-invariant systems, that is LTI systems. Linear systems, but they are time-invariant. Time invariant means the properties of the system doesn't change. Let us say you have a system like a spring mass damper system only. So, if it is fmx double dot bx dot plus kx, so m that is the mass damping ratio, and the stiffness of the spring doesn't change with time. So, that is what the characteristics of that system should not change with time. So, those are linear time-invariant systems.

So, using this input and output can be represented like a block. So, you have a transfer function that goes here. This is your output and this is your input. So this type of block diagram representation is very, very commonly used to represent your control

architecture also. Let us say if it is a simple PID control or any sort of control. So you can represent each of the blocks with having one transfer function within. Let us say you have a closed-loop system. So here goes your input. You have a comparator. So you have a controller over here. This can be a controller. A PID controller. This gives an output to the robot. The robot gives you an output, and finally, that output is sensed. Then you have a feedback line that comes here and goes here. So, this is your comparator. So, you can have a signal that comes here, which is the desired input. This is the output that is sensed. So, you have a transfer function that is over here. That is the feedback transfer function. Similarly, you have a controller transfer function that comes here, and this is your robot transfer function. So, every stage has an input and an output, and each block represents one transfer function. So, this is a very common tool that is used to represent the whole of the system sometimes. It is not just a control system; it can be just an open loop system also. So, your system itself can be represented like this, and overall, your system may have a system transfer function that looks very much like this,



And over here in this transfer function, you have numerator, you have denominator just like this. And maybe you have. This can state the behavior of your system also. We'll see how it can do that.

Transfer Function of Spring-Mass-Damper system

For a typical spring-mass-damper system $f(t) = m\ddot{x}(t) + b\dot{x}(t) + kx(t)$

Taking Laplace on both the sides we get: $F(s) = ms^2X(s) + bsX(s) + kX(s)$
assuming zero initial condition, i.e., $x(0) = 0$ and $\dot{x}(0) = 0$.

$$\Rightarrow G(s) \equiv \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k} = \text{Open loop transfer function}$$

$$\Rightarrow G(s) \equiv \frac{(1/m)\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

where, $\omega_n = \sqrt{\frac{k}{m}}$ and $\xi = \frac{b}{2\sqrt{mk}}$ are *natural frequency* and *damping ratio* of the moving block.

The two poles of which are $s_{1,2} = \frac{-b \pm \sqrt{b^2 - 4mk}}{2m} = \xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1}$

NOTE:

- *Poles* are roots of the characteristic equation formed by equating the denominator
- Equating the numerator to zero would yield *Zeros*.
- Poles and Zeros affect the peaks and amplitudes of the forced and natural respons

Collaborative Robots (COBOTS): *Theory and Practice* Arun Dayal Udai

So yes, let us start with a simple system, the Spring-Mass-Damper System. We have used this quite often now because this represents very analogous to our own control system that we are looking for, that is the robot control system that also is a second-order system, but that is not that linear.

$$f(t) = m\ddot{x}(t) + b\dot{x}(t) + kx(t)$$

So, it can be linearised to say almost near to this okay. So, that is the reason I am using this quite often. So, yes most of the mechanical system should be very near to this. If not exactly this and there are terms which are analogous to m, v and k which are put here. So that is the reason this is quite commonly used.

And yes, the taking Laplace on both the sides you have seen with the initial condition if they are 0 that is, x0 is 0, and x dot 0 is 0. That means the initial position and velocity are at 0. So, if you take Laplace, what you should get is Laplace of f(t) is F(s). Similarly, m is a constant. So, Laplace of x double dot t is s square X(s), and x dot t if you take Laplace, it becomes s X(s). So, every time you take Laplace, there is a term that comes here that says it has X0 at the end or X dot 0 at the end. So, because those terms are 0, you are just remaining with s x(s) over here. k times of x(t) will become k being constant. It will go here, and you have x(s) that come here.

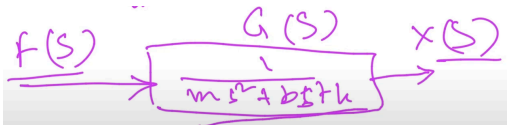
$$F(s) = ms^2X(s) + bsX(s) + kX(s)$$

So, taking x(s) common in the whole of this expression, so x can be taken out. So, you can see a relation which is very much like this.

$$G(s) \equiv \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + k} = \text{Open loop transfer function}$$

So, if you take their ratio, it gives you gs. That is the transfer function. So it says 1 by ms square plus bs plus k. So this is the Open-loop transfer function of a Spring-Mass-Damper System. Where the output X(S) and the input F(s) are related by this. So, your block would say something like this. So you have 1 by ms square plus bs plus k. So, you have an input that comes here, which is F(S), and you have an output that goes here, which is X(S). So, output by input X(S). will give you the transfer function, which is G(S), as shown here (above equation). So, this is what is meant by a transfer function.

Let us see how this represents the behavior of your system. You can clearly see you have something in the denominator. You are very familiar with this already, so this was the characteristic equation of your spring-mass-damper system, and the roots of this we used to analyse the behaviour of your system. So yes, this can alternately be put like this.



So, we are not very much concerned with this formula now. But yes, this also includes the terms, which are very much the same. Only the constants will differ. So it is s square and s over here in the denominator.

$$G(s) \equiv \frac{(1/m)\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

So, this also gives you a similar characteristic equation, and this will also take you to the same place.

So yes, there are two poles here. Those are S1 and S2, which are given by this.

$$s_{1,2} = \frac{-b \pm \sqrt{b^2 - 4mk}}{2m} = \xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1}$$

You have already seen that, and the discriminant over here will basically tell you about your roots, whether they are real and imaginary, both parts are there, or purely real, or purely imaginary, or equal, or it can just be zero. So, it all depends on your roots and discriminant. That will govern your system behaviour also. So, roots can also be expressed like this.

So, yes, the poles are nothing but the roots of the characteristic equation formed by equating the denominator to zero. So, this is your denominator. You take the roots of that; those are known as the poles. So, equating the numerator to zero gives the zeros. So, the roots of the numerator are known as zeros. Poles and zeros affect the peaks and the amplitude of the forced and natural response of these systems.

So, this is how a simple spring-mass-damper system is, and this is your transfer function, and the denominator over here is the characteristic equation, and the roots of this basically govern how your system is going to behave. Naturally, as well as when acted upon by any external force. So, you have seen the behaviour already using this. So no need to discuss much. So, the idea is to come up with the transfer function of your Spring-Mass-Damper System. So, that is what is observed here.

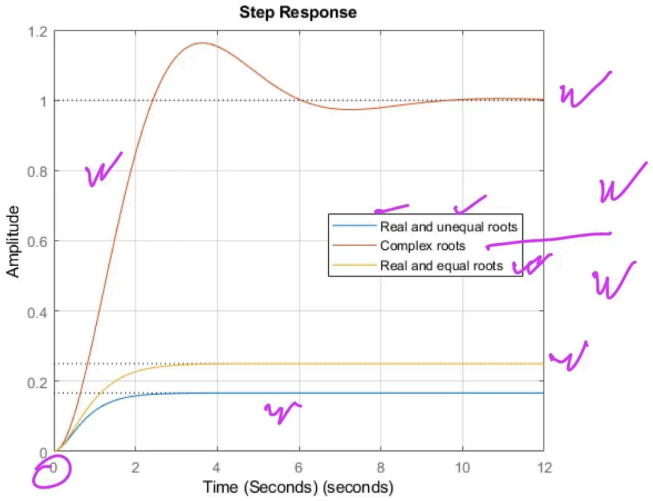
Demonstration: MATLAB® code

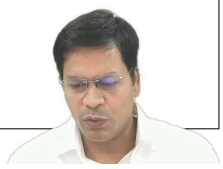
```
%% Response of a spring-mass-damper system with external force
% Defining the systems transfer functions
numerator=[1]; denominator1=[1 5 6]; denominator2=[1 1 1];
 denominator3=[1 4 4];
system1 = tf(numerator,denominator1);
system2 = tf(numerator,denominator2);
system3 = tf(numerator,denominator3);
step(system1,system2,system3);
xlabel('Time (Seconds)'); ylabel('Amplitude');grid on;
```

Using MATLAB code, you can also analyse your system by simply expressing it as a transfer function. So, now, you have a transfer function with the numerator as 1 and the denominator simply as 1. This is the polynomial definition that goes in the denominator.

So, in the case of the system that we have already seen, it has 1, 5, and 6. So, you see, it is m s squared plus 5 s plus 6. So, you can just see it is b squared minus 4 m k. So, m here is 1. So, it is b squared, that is 25, minus 4 into m into 1, which is 24, so it is 1. So, you see, you have a discriminant of this that tells you it is greater than 0, so you have two roots, both unequal and real, meaning it is an overdamped system. You have seen the behaviour already, so that is put here in the denominator. The second system is 1, 1, 1, that is, s squared plus s plus 1, and the third one is the one that we have already analysed all three of them. So, it is 1, 4, 4. So we already know how the behaviour should turn out to be.

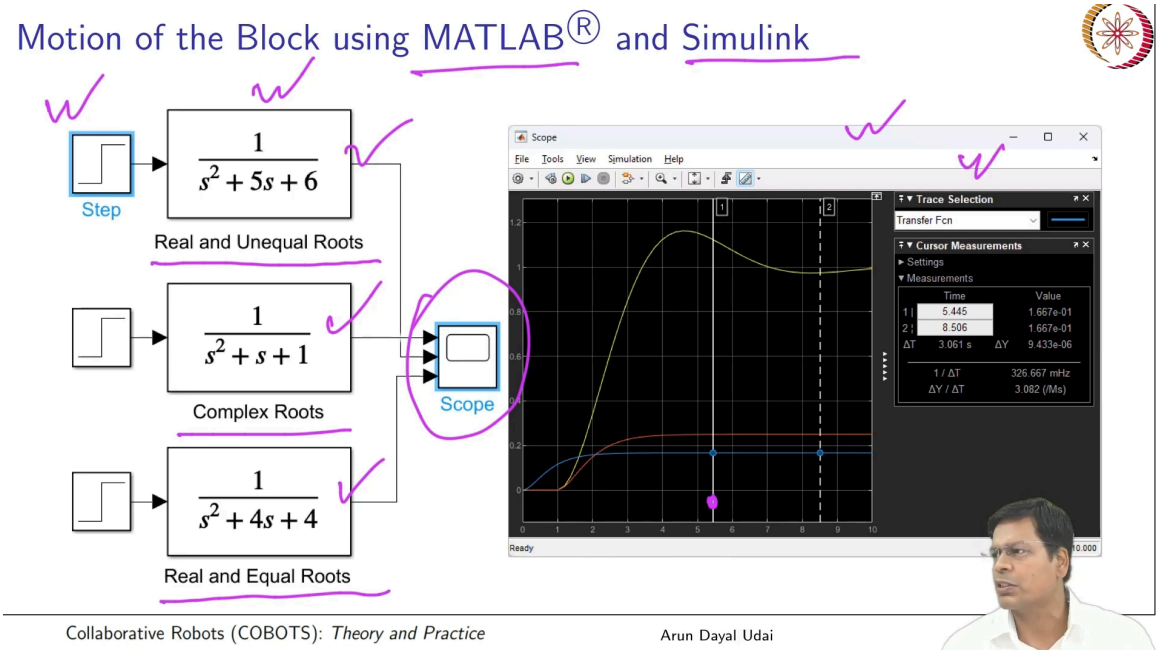
So, I am defining my system transfer function as numerator and denominator. MATLAB allows you to define it in this way. tf is the function that I am using here. So yes, these are my systems. That is system 1, system 2, and system 3 with different functions with different denominators over here with the same numerator that goes here, and I have observed the step input behaviour of these systems. So, for the step input, systems 1, 2, and 3 are plotted here. So, this is just the label. So they are plotted here.





So, let me just see; it is something like this. So, you have a system with real and unequal roots. So, that is coming here. This is an overdamped system, and the next one has real and unequal roots. Real and equal roots that is your critically damped system that comes here, and then you have complex roots. So, the roots are imaginary. All the roots are in the left half of your complex plane. So, you have already seen that. So, this is your behaviour for that. So, this is how it comes.

So, this is what a transfer function is, and that can be used to analyse the behaviour of your system. So, based on the input and time, that goes from 0 to 12 seconds over here. So, this is the behaviour you see.



The same can be studied even using MATLAB Simulink. That is a symbolic tool that MATLAB has. So, it supports a block diagram-based approach to programming. So, you have a step input that is fed to the block with a transfer function defined like this. This is the first system, this is the second system, and this is your third system. So, all of these you have already done. So, this is a real and unequal root, this is a complex root, and this is a real and equal root. So, you see, similar kinds of plots can be observed by using a scope that has three inputs. So, you see, you can observe it.

So, each and every time instant, you can check the values by sliding this. So, this shows you the values. So, at this moment, what are the values of all the plots? So yes, this is an additional tool that is supported in MATLAB to check the values. So, this is how it can be done.

Fundamentals: Ordinary Differential Equations (ODE)

An ordinary differential equation (ODE) contains one or more derivatives of a dependent variable, y, with respect to a single independent variable, usually time t.

- A higher-order differential equation of p^{th} order, $y^{(p)} = f(t, y_1, y_2, y_3, \cdots, y_p)$ is represented as p first-order differential equations.

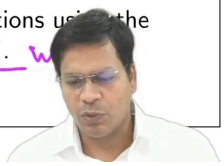
The set of p first-order differential equations are:

$$y_1 = y$$
$$y_2 = y'$$
$$\vdots$$
$$y_p = y^{(p-1)}$$

This results in a system of p first-order equations

$$y_1' = y_2$$
$$y_2' = y_3$$
$$\vdots$$
$$y_p' = f(t, y_1, y_2, y_3, \cdots, y_p) \leftarrow ODE$$

- This representation is quite useful for numerical solution of differential equations using the well developed first-order equation solvers, e.g. *ODE* Solvers in MATLAB®.



Before we move further to state space representation, let me just familiarise you with Ordinary Differential Equations (ODE) and the kind of thing we are going to use them for. So. let us just discuss this. So, an Ordinary Differential Equation contains one or more derivatives of a dependent variable, that is, y, with respect to a single independent variable, usually time t, in the case of physics-based systems like ours, like a robot.

So, this is how it is defined. So your system normally should look like this.

$$y^{(p)} = f(t, y_1, y_2, y_3, \cdots, y_p)$$

So, that is the p^{th} order derivative of y, which is here. So, that is equal to, it is equal to a function of time and y1, y2, y3, and yp. What are those? So, they are basically derivatives of y. A higher-order differential equation of p^{th} order can now be represented as p first-order differential equations. There is a reason for it why we are converting this higher-order differential equation to a set of first-order differential equations. If the highest order is y_p, that is, the p^{th} order, then you will have p first-order differential equations formed.

So yes, the p first-order differential equations look like this. So it is these are different variables now, and these are derivatives of y. This is not a derivative. This is the first

derivative and the p minus 1 derivative. So, y1 is equal to y, y2 is equal to y dash. Similarly, there should be y3 equal to y double dash, and so on, until yp is equal to the y p minus 1^{th} derivative.

If you substitute this into your original differential equation, this results in a system of p first-order equations. If you split them all, you should get something like this. So, y1 dash is equal to y2. So, it is just this one that is put here.

Similarly, y2 dash is equal to y3. So, that was your Second term, so y3 is nothing but that will come here. y3 should be equal to y double dash, and y double dash means that is y2 dash, which can be obtained from here, so that comes here.

Similarly, the y3 dash will be equal to y4, and so on and so forth, till finally, you will get the y_p dash. y_p dash, so you should get your original function which was here. So, this is your ordinary differential equation that comes here.

So, this is how you can split your system into a set of systems. So, that means you have first-order equations that represent independent variables. So, those can be substituted, and they can be finally put like this. Why is this done? Because this representation is quite useful for obtaining numerical solutions of differential equations using the well-developed first-order equation solvers. So, MATLAB already has a set of ODE solvers: ODE4, ODE5, ODE5, ODE4. Those are nothing but Runge-Kutta solvers, and many other types are already defined. So, that is nothing, but they are solvers for first-order differential equations. So, in order to solve an equation which is like this, you have to convert it into a set of first-order equations.

So now, you can club them all together and use ODE solvers to solve them. So, this is the beauty of having expressed your system as a set of equations like this.

Fundamentals: Ordinary Differential Equations (ODE)[2]

- *ODE* is solved by starting from an initial state, and setting the period of time.
- At each step the solver iteratively applies the ODE solver algorithm to the results of previous steps.
- *ODE* solver returns a vector of time steps $t = [t_0, t_1, t_2, \cdots, t_f]$ as well as the corresponding solution at each step $y = [y_0, y_1, y_2, \cdots, y_f]$.
- Any number of coupled *ODE* equations can be specified as:

$$\begin{bmatrix} y_1' \\ y_2' \\ y_3' \\ \vdots \\ y_n' \end{bmatrix} = \begin{bmatrix} f_1(t, y_1, y_2, \cdots, y_n) \\ f_2(t, y_1, y_2, \cdots, y_n) \\ f_3(t, y_1, y_2, \cdots, y_n) \\ \vdots \\ f_n(t, y_1, y_2, \cdots, y_n) \end{bmatrix}$$

[2] https://in.mathworks.com/help/matlab/math/choose-an-ode-solver.html

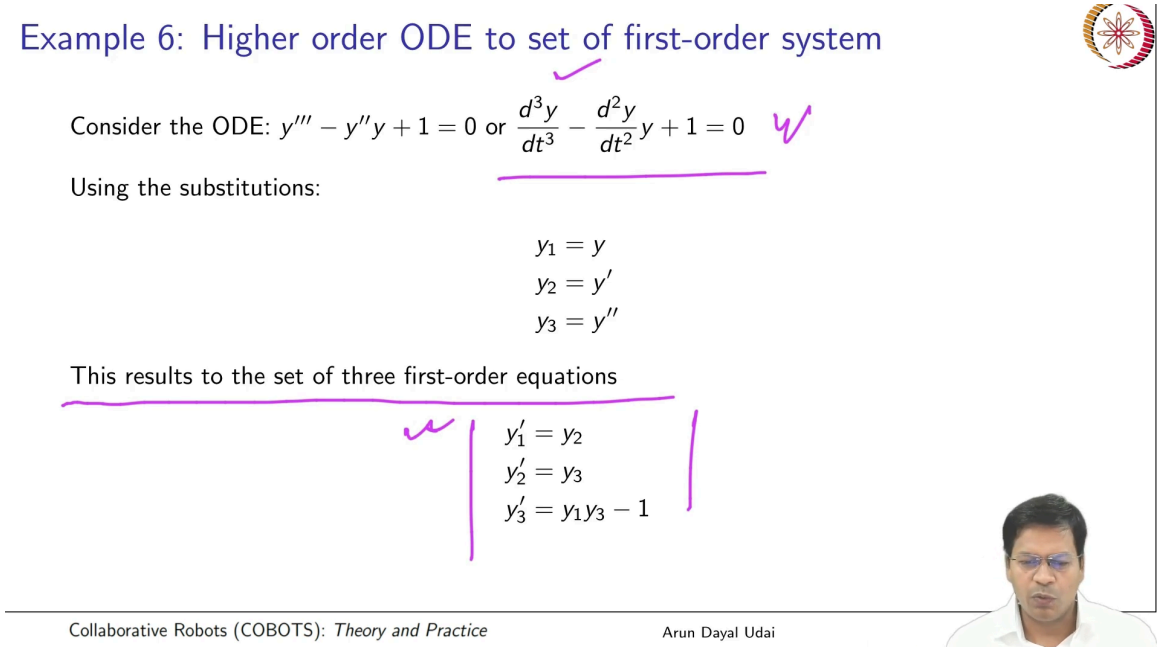
So, ODE is solved by starting from an initial state, that is, what are the states at the initial condition that is the boundary condition, and it sets the period over which you have to get the solution, and at each step, the solver iteratively applies the ODE solver algorithm to the results of the previous step. So, it iterates from the initial step to the next to the next, and finally, it gives you a vector of time steps. Time steps, that is, t is equal to t0, t1, t2, t3, and at each of these time steps, you also get the value of y. So, that is y0, y1, y2, y3, y4, y5, and so on, till all the values of y. So, all the y that you saw in the previous slide are obtained at every time step. That means you have got all the variables and all the derivatives of y. So, that is what is obtained. So, if it is acceleration, you get velocity and position at every instant of time for any system, like a Spring-Mass-Damper System.

So yes, a number of coupled ordinary differential equations can be specified and solved simultaneously in MATLAB like this.

$$\begin{bmatrix} y_1' \\ y_2' \\ y_3' \\ \vdots \\ y_n' \end{bmatrix} = \begin{bmatrix} f_1(t, y_1, y_2, \cdots, y_n) \\ f_2(t, y_1, y_2, \cdots, y_n) \\ f_3(t, y_1, y_2, \cdots, y_n) \\ \vdots \\ f_n(t, y_1, y_2, \cdots, y_n) \end{bmatrix}$$

So, they can be packed like this. These are the independent functions of t and the derivatives of y that go here. So, they are all have different functions. They are all

coupled ODE equations that have the same input variables and one output variable. So, not basically the input and output variable. Basically, these are the derivatives. Got it. So, this is how it can be expressed. So, this is a set of first-order equations.



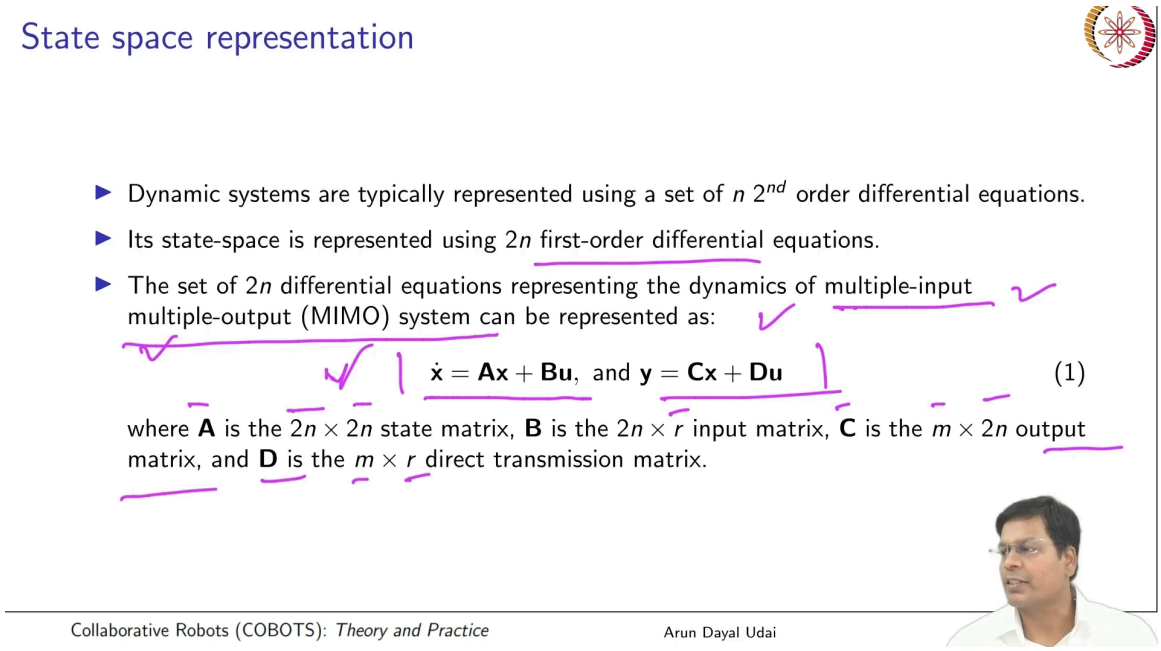
So now, just take one example of ODE and see how we can convert it to a set of first-order equations. Let us say you have a system which is like this:

$$\frac{d^3y}{dt^3} - \frac{d^2y}{dt^2}y + 1 = 0$$

it is a third-order equation, so d³y by dt³, it is with respect to time, d²y by dt², and you have y plus 1. The system can be represented like this in a compact manner. So, prime here, y single prime is basically dy by dt. The double prime is d²y by dt², and y triple prime is equal to d³y by dt³. So, this is how they are.

So, now I will just use the substitutions for y, y prime, and y double prime. So, for all the derivatives, I have this substitution. So, y1, y2, and y3, I will put them into the original equation that is here, and I can get to this. This results in a set of first-order equations. So, if it is third-order, you see, you got how many equations? Three. So, the number of independent first-order equations will be dependent on the order of your original differential equation of the system. So, if it is 3, you got 3 over here. So, each one of

them is a first-order equation. They can be packed together and put into an ODE solver, and you can get the solution to that. So, we will stop the ODE discussion over here. And we will move further with our system again.



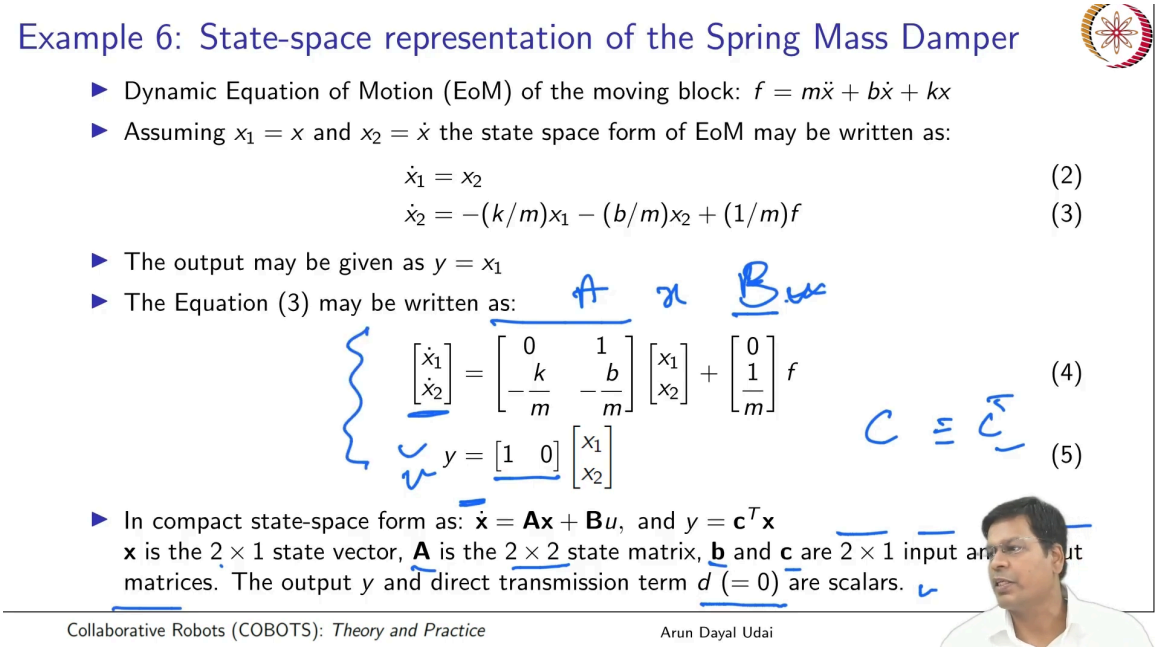
So, let us come back to the State-Space Representation. So, any dynamic system is typically represented using a set of n second-order differential equations. So, if it is a second-order differential equation, the number of first-order differential equations that can be formed out of each one of them should be 2. So, if it is n, it is 2n.

So, its state space is represented using 2n first-order differential equations. So, the set of 2n differential equations representing the dynamics of a multiple-input, multi-output system can be represented in a compact manner. So, this is what is your state-space form. So, multiple sets of equations you have for a multiple-input and multiple-output system, and there were 2n differential equations, first-order equations, that represent your dynamical system. So, each one of them can be clubbed together in a compact manner like this.

$$\dot{x} = Ax + Bu, \; and \; y = Cx + Du$$

So, what are these terms A, B, C, and D? So, the matrix A is a 2n cross 2n state matrix. B is the 2n cross r input matrix. C is an m by 2n output matrix. D is the m cross r direct

transmission matrix. So, each one of them has some significance. We will see by example.

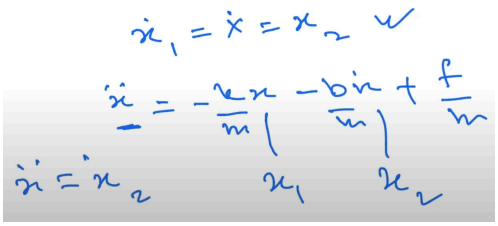


So, let us come back to our old friend, the Spring Mass Damper, once again. So, the dynamic equation of motion this time is the same. So that is the same. f is equal to mx double dot plus bx dot plus kx.

$$f = m\ddot{x} + b\dot{x} + kx$$

Now, I want to express this as a state space representation. So, what should I do? So yes, I will assume the substitutions which are like this. So that is x1 is equal to x, and x2 is equal to x dot. So, what effectively am I doing? So, the first one is x1 dot is equal to, so what it should tell me is, if it is x1 dot, then it is x dot, so it is x2. Got it? So, from these two, you can quickly obtain this. Now, x double dot is equal to minus kx minus bx dot. Right, minus f. So, it was just m, which has multiplied here. So, you can write it as m, m, and m. Got it? So, it should be plus over here.

So, yes. So, now you substitute these again here. So, when it is x, you can say it is x1. So, you write x1 here. If it is x dot, you can quickly write x2 over here, and this remains as a constant.



So, ultimately, x double dot here is basically x2 dot because you see, x dot is x2. So, x double dot is x2 dot, got it? So, that is what comes here. So, you can write x2 and x1 dots, okay? So, these are the derivatives.

Now, the output part, y is equal to x1. So, that is very trivial. So, there is nothing called d over here. That turns out to be 0 in this case. So, that was your output equation. So, equation 3, this one, can now be written as 2 and 3, both of them together in a compact manner. So, I have put these two over here.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} f$$

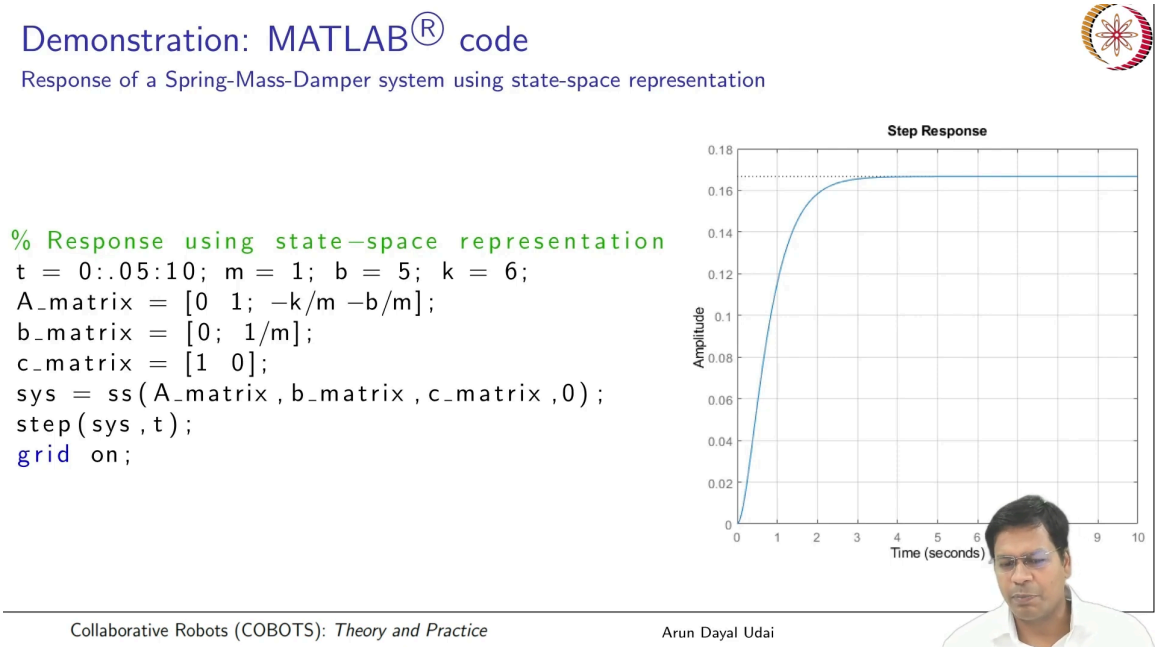
And again, if you see, I have taken x1 and x2 directly over here. So, that is a pairing here. As x1 and x2 are both here. So, that comes in the next column. So, it is minus k by m that is transferred here. Similarly, minus b by m from the equation 3 comes here. And it has both the terms for x1 and x2. Whereas in the first one, it is only x2 with no coefficient just 1. So it is 0 and 1 over here.

Now, the part which is here. So, f appears only in the second. That is x2 dot. So, this turns out to be 0 so it is expressed like this, and y has got just x1 term. So, it can be written as this.

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

1 multiplied by x1 and 0 multiplied by x2. so it is nothing but a simple straight equation. So, that comes here. So, this is your input and output relation. So, those are where the state space forms now.

So now, Comparing this, you can write simply, this one is your x dot, this term is your A, this is your x, again this becomes your B matrix, and you have u that is the input that is f u that comes here. So, exactly f is your u. Got it? So, x is 2 cross 1 state vector. A is your 2 cross 2 state matrix. b and c are 2 cross 1 input and output matrices. The output now is a direct transmission term with d is equal to 0. So, that is scalar because you have no d-term over here. So, there is no direct transmission which is happening. So y is just this. So that is C times of x. If it is bold C capital, it is this. So, this is equivalent to C transpose the way it is written here. So, it is like transpose. So, it is this. Small c transpose. So, both are the same. So, this is how it is converted to state space form. Got it? So, yes, now using this, we can analyse my system.



So, let us just see the Spring-Mass-Damper in state-space representation. I am using MATLAB now to see its behaviour for a step input. So, how is it defined in MATLAB? So, your system, let us say, has m equal to 1, b equal to 5, and j equal to 6. It is an over-damped system. For a time variation from 0 to 10 seconds, I want to do that. So, the A matrix, I have just put it exactly as it appears here (previous slide). So, I have defined my matrices A, b, and c over here and created a state-space system of equations, stored it here in sys. So A, b, c, and d come here. So, a step input to this system with time t varying from 0 to 10 at a step of 0.05 seconds. So, it is plotted. So, it gives me a similar

behaviour to what it should be. The way we have seen earlier in our previous analysis. So, this is how state space can now be used to see the behaviour of a system.

Transfer Function from a State space representation for a SISO System

- Taking Laplace of Equation (1) with zero initial condition

$$s\mathbf{x}(s) = \mathbf{A}\mathbf{x}(s) + \mathbf{B}u(s) \quad (6)$$

$$y(s) = \mathbf{c}^T\mathbf{x}(s) + du(s) \quad (7)$$

where the vectors $\mathbf{y}$ and $\mathbf{u}$ are scalars denoted by y and u, respectively.
Matrices $\mathbf{B}$ and $\mathbf{C} \equiv \mathbf{c}^T$ are vectors of appropriate dimensions, and $D \equiv d$ is a scalar.

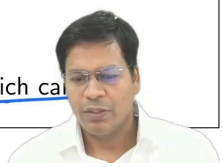
- Using $\mathbf{x}(s)$ from (2) in (3): $y(s) = [\mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d]u(s)$
- The Transfer Function (TF) $G(s)$ is:

$$G(s) = \frac{y(s)}{u(s)} = \mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d = \frac{q(s)}{|s\mathbf{I} - \mathbf{A}|} \quad (8)$$

where $q(s)$ is a polynomial in s.

- The denominator is equal to the characteristic polynomial of $G(s)$
- The eigenvalues of $\mathbf{A}$ are identical to the poles of $G(s)$
- Taking inverse Laplace of $G(s)$, time domain equation may be obtained, which ca
expressed as state-space form.

Collaborative Robots (COBOTS): *Theory and Practice* Arun Dayal Udai



So, how is this all done in the background, basically? The transfer function from state space is obtained for this Single-Input and Single-Output (SISO) System. It has just one input, which is force, and the output is displacement. So for them, this becomes quite trivial. So, can we do that as well? So, let us just see if it is possible.

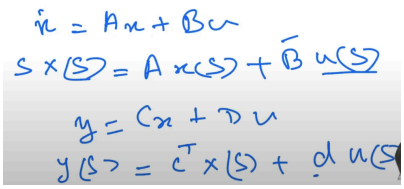
So, how is your system? I will just show you once again over here. Yes, so this is your state-space equation. x dot is equal to Ax plus Bu. y is equal to Cx plus Du.

$$\dot{x} = Ax + Bu, \text{ and } y = Cx + Du$$

Input and output relation. So, taking the Laplace transform of both of these, which are here. So, yes, let me do that here also. So, it is x dot is equal to Ax plus Bu. So, taking the Laplace transform should give me S times x(s) and A times x(s) plus Bu(s).

B is the matrix, and u is in Laplace. So, that will appear here. Similarly, if you convert y now, y was Cx plus Du. So, if you take Laplace, you should get So, y(s) is equal to instead of C; I know both were equivalent, so those constants are put here. It is x(s) here

plus Du(s). If D is absent, this term goes off. Got it? So, this is how it can be written. So, just by taking the Laplace, these two equations are put.



Now, you just take out x(s) from this equation, that is, equation number 6, and substitute it here in this one (equation number 7). What do you get? You get y(s) is equal to c transpose sI, I is a compatible unit matrix. So, if it is second order, its diagonal elements are 1, and the rest are all 0. So, it is a unit matrix minus A whole inverse b plus d u(s). Correct? So, it becomes exactly this. You can try doing it yourself. So you can just take out XS from here, substitute it here, and you will get to this.

$$y(s) = [\mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d]u(s)$$

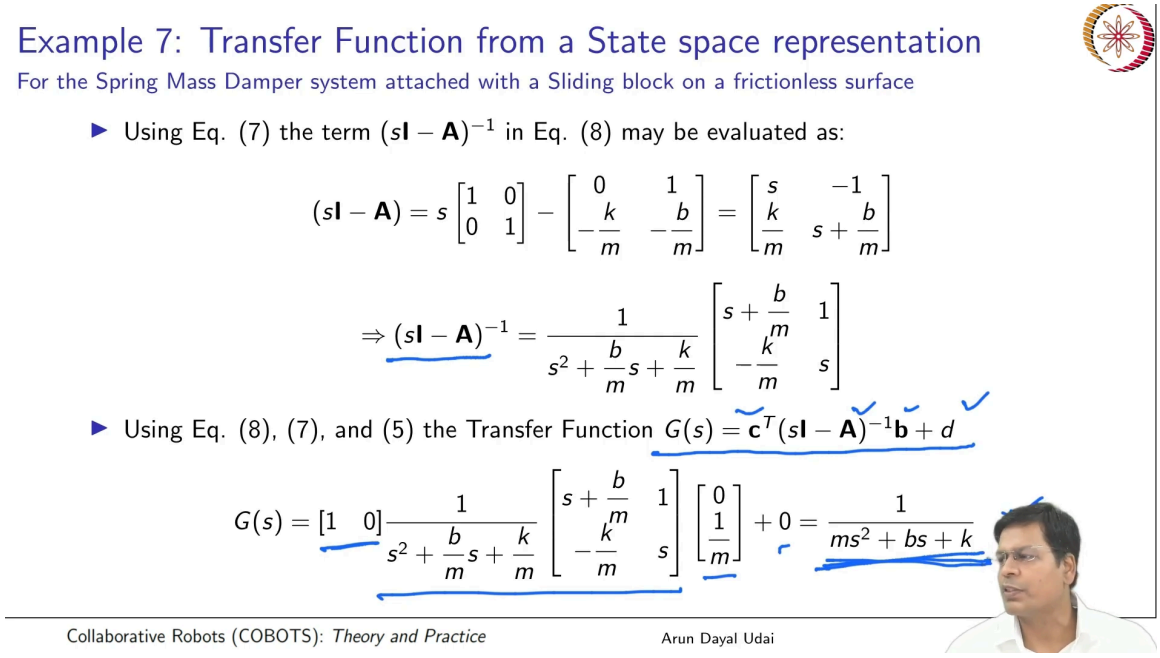
So now, you can relate your output and the input. So, the output was y(s), input is your u(s). So, that is your transfer function. So, that is given as this.

$$G(s) = \frac{y(s)}{u(s)} = \mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d = \frac{q(s)}{|s\mathbf{I} - \mathbf{A}|}$$

So, this is your output, this is your input, so this becomes your output. Transfer function. So, what it is here basically is if you can write inverse if you put inverse over here. So, what it is sI minus A transpose by the determinant of sI minus A. So, if you convert it, you will get some polynomial in s, and the numerator and the denominator will have the determinant of sI minus A. So, this gives your transfer function. So, this is having the numerator with a polynomial in S, denominator as sI minus A. A is this one.

So now, the denominator is the characteristics polynomial of G(s). You already know that. So, the denominator of the G(s) should give you the characteristic equation of this system. So, the roots of this determinant should tell me the behaviour of the system as well.

So, the eigenvalues of A are identical to the poles of G(s). You see. If I equate this also to 0. So, that means. I am finding out the eigenvalues, and the same was for finding out the roots also, they are identical. So, taking the inverse Laplace of G(s), that is the input and output relation. If you take the inverse Laplace of this, you should get the time-domain equation of the system, and again, this can be expressed in the state-space form. So, if you know the time-domain equation, you can convert it to the differential equation and express it by proper substitution in state-space form. So, both ways you can do. You can derive the transfer function from the state-space representation, or you can do it the other way around also. Right? So, both are equally helpful. So, at least in SISO systems, it goes like this.



So, let us again see our own system. How was ours? So, let me just show you once again our original slide. So, this was the system, the state-space representation of your spring-mass damper, you see. You see, what was your A?

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\frac{k}{m} & -\frac{b}{m}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix}f$$
$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

It is the matrix that is shown here. Got it? So it is exactly this one. That is your matrix A. This is your matrix B. And then you have C transpose. That is the C matrix. The output matrix and input matrix they look like this. So, copying that from here, I can now put all those values into this. So, sI, I is because I know it was a second-order system. I is compatible. So, it is diagonal elements 1. So, it is sI minus A. So, this is A that comes here. So, if I do that, I will get to this.

$$(s\mathbf{I} - \mathbf{A}) = s\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}0 & 1\\ -\frac{k}{m} & -\frac{b}{m}\end{bmatrix} = \begin{bmatrix}s & -1\\ \frac{k}{m} & s+\frac{b}{m}\end{bmatrix}$$

So again, taking the inverse is very trivial. So, you just take the inverse. What will you get? You get something like this in the denominator. That is nothing but the determinant of this matrix. Got it? So, that is coming here, and the whole transpose comes here.

$$\Rightarrow (s\mathbf{I} - \mathbf{A})^{-1} = \frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}}\begin{bmatrix}s+\frac{b}{m} & 1\\ -\frac{k}{m} & s\end{bmatrix}$$

So, transpose by the determinant of this. So, the determinant is very trivial. So, you can just find it out.

Now, I can extend it further. I can use this to write the transfer function.

$$G(s) = \mathbf{c}^T(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d$$

So, G(s) is my transfer function. All the values I have put from my previous slide. So, that was here. So A, b, c, all the values come here. So A, you have b and c that go here. d is also here. d is equal to 0, which comes here. So, this is your b. This is exactly sI minus A inverse, so that comes here, and c transpose is here. Got it? So this gives me 1 by ms square plus bs plus k. We have seen this earlier also. Exactly, this should be your transfer function when you just took the Laplace of your given system. So, I got the same result, and again, this is my characteristic equation. So, this is what was expected also. So, the roots of this will tell the behaviour of my spring-mass-damper system. So I can analyse it now anyway.

Demonstration: MATLAB® code

Conversion from state-space to transfer function

```
%Conversion from state-space to transfer function
m = 1; b = 5; k = 6; u = 1;
A_matrix = [0 1; -k/m -b/m];
b_matrix = [0; 1/m];
c_matrix = [1 0]; d = 0;
[num, den] = ss2tf(A_matrix,b_matrix,c_matrix,d,u)
```

Command Window

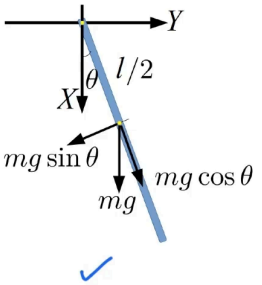
```
>> case10

num =

     0     0     1

den =

    1.0000    5.0000    6.0000

fx >>
```

$$\frac{1}{ms^2+bs+k}$$

So, let us see if MATLAB allows me to do so. So yes, conversion from state space to the transfer function is also possible in MATLAB. So, your system goes here: m, b, k, and your input is here u. A matrix, b, c matrix you can define. You can extract it directly from your earlier slide. A, b, and c matrices. All those are available here. So, you have simply put it here, and now, state space to transfer function. So, straight away, you can get the numerator and the denominator.

Running this gave me this. So, yes, this was your numerator, and this is your denominator. What does this say? It is coefficients, basically. These are coefficients. You see, this is m, b, and k. 1 by m s squared plus b s plus k. So, the coefficient of s squared, that is, m, comes here, b and k. So that is all here. So, this is what can be directly obtained. So, MATLAB also allows you to convert state space to a transfer function format.

Example 8: State-space equations of a Single-DOF Planar Arm



- The dynamic equation of motion of a planar arm is given by:

$$\tau - \frac{1}{2}mgl\sin\theta = \frac{1}{3}ml^2\ddot{\theta}, \text{ or } \ddot{\theta} + \frac{3g}{2l}\sin\theta = \frac{3\tau}{ml^2}$$

- Using the state variables $x_1 = \theta$ and $x_2 = \dot{\theta}$

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{3\tau}{ml^2} - \frac{3g}{2l}\sin x_1 \equiv \frac{3\tau}{ml^2} - \frac{3g}{2l}x_1$$

Linearizing for small oscillations: $\sin x \approx x$ for $x \approx 0$

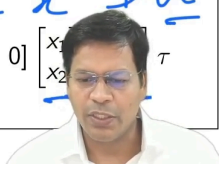
- As there is no input torque τ, $u = 0$.

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix}\tau$$

- The state-space form would be:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$
$$y = \mathbf{C}\mathbf{x} + \mathbf{D}u$$

$$\begin{bmatrix}\dot{x}_1\\\dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\-\frac{3g}{2l} & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}0\\\frac{3}{ml^2}\end{bmatrix}\tau \text{ and } y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}\tau$$



Collaborative Robots (COBOTS): *Theory and Practice* Arun Dayal Udai

So again, let us analyse a simple robot-like, Single-DOF Planar Arm. If there is no torque, it becomes a pendulum, and mind it, it is just a planar pendulum. It is not a conical pendulum, which has 2 degrees of freedom over here. It is just one DoF, one cylindrical joint, that comes here. It can go like this.

So, it has a solid link with mass m. So, the mg force comes here. mg sine theta, you know, is the driving force with l by 2 distance; that is, at the centre of mass, it is acting. So, it is giving you a torque, which is given by mg sine theta into l by 2. So, this is the driving torque, basically.

So, you see, this was your original state-space form.

$$\dot{x} = Ax + Bu \text{ and } y = Cx + Du$$

So, this is what we want to obtain now. So, the dynamic equation of motion of this planar arm will be given as Tau minus this is the external torque.

$$\tau - \frac{1}{2}mgl\sin\theta = \frac{1}{3}ml^2\ddot{\theta}$$

This is the torque that is acting due to gravity. So, the difference of that is actually the driving torque. So, it is nothing but I theta double dot. I is the moment of inertia of this

link. So, that is 1 by 3 ml square. Theta double dot is the acceleration at any instant of time p. So, the same equation can now be written as this.

$$\ddot{\theta} + \frac{3g}{2l}\sin\theta = \frac{3\tau}{ml^2}$$

What is theta double dot is equal to 3 tau by ml square minus 3g by 2l sine theta. So, theta double dot.

So yes, now I can use these substitutions. So, x_1 is equal to theta. x_2 is equal to theta dot. So, this is the first one that, you know, is very trivial. So, the x1 dot is basically theta dot. So, that is equal to x2. This is the first one.

$$\dot{x_1} = x_2$$

Now, the second one. Theta double dot. Theta double dot is x2 dot. So, that comes here. So, that is x2 dot, and that is equal to 3 tau by ml square minus 3g by 12 sin theta.

$$\dot{x_2} = \frac{3\tau}{ml^2} - \frac{3g}{2l}\sin x_1 \equiv \frac{3\tau}{ml^2} - \frac{3g}{2l}x_1$$

So, instead of theta, I can write x1. So, that is coming here for a very small oscillation. Because you see, this is a non-linear type of system. So, it cannot be directly converted like a straight space form. So, this time, you have to do a little linearisation. So, you know, for very small oscillations. Sine x1 tends to x1. So, very near to that if you are making small oscillations. So, this approximation can be done. So, I can directly put, instead of sine x1, I can put x1. So, my new system,

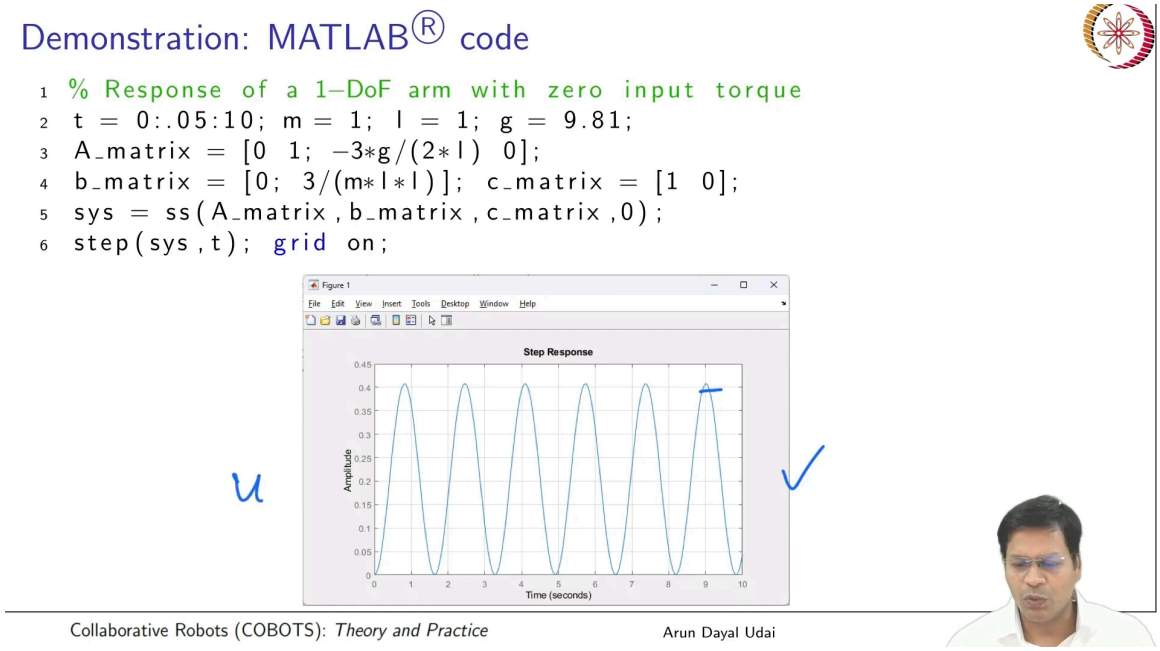
As there is no input torque tau, u becomes equal to 0, and the output relation can be written as y is equal to x1. So, putting them all together here, so it is this.

$$\begin{bmatrix}\dot{x}_1 \\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1 \\ -\frac{3g}{2l} & 0\end{bmatrix}\begin{bmatrix}x_1 \\ x_2\end{bmatrix} + \begin{bmatrix}0 \\ \frac{3}{ml^2}\end{bmatrix}\tau$$

So, it is x1 dot, x2 dot that comes here, and extracting the terms now, so this is your A, this is your A that comes here, Ax, x dot is equal to Ax plus Bu. So, this is your B and u. That is the tau, which is the external torque acting. So, again, the output is y equal to Cx. This is your x and Du. D and u.

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tau$$

So, u is again the input, and there is no D. Got it? So, this is how you have converted your single degree of freedom, planar arm, and dynamic equation of motion to state-space form. Now, it is integrable; it can be analysed, and you can do anything with it. Let us do that.



So now, I will analyse my step input response for this 1-DoF arm with zero input torque. Now, I have defined my A and b, as we obtained earlier in this (previous slide). Exactly, I will put this as A. I know the values here. m is equal to 1. The length of the link is also 1 unit, and g is 9.8 meters per second squared. l is 1 meter. m is 1 kg. So, I have defined all of this here, as we have obtained in the previous slide. So, I will put those values here, and I have got the matrices now.

So, I now have the state space. Using the ss function, I have defined my ss function in sys. So, that is the A matrix, b matrix, b, and d go here, and step input is given and analysed over time. So, this gives me a plot, which is SHM, you see. It is a simple harmonic motion. For small oscillations, it behaves just like a pendulum. Got it? So, this is your system, and this is what is expected because you are not acted upon by any external force, and there is no joint damping, meaning it will keep oscillating forever. So, that is what is shown here as a result.

So, that's all for today, and in the next class, we'll discuss a Robot Joint (DC motor model). So, we'll get very close to what we are actually aiming for. So, we will now start modelling the joint of a robot using a DC motor model.

So, that's all for today. Thanks a lot.