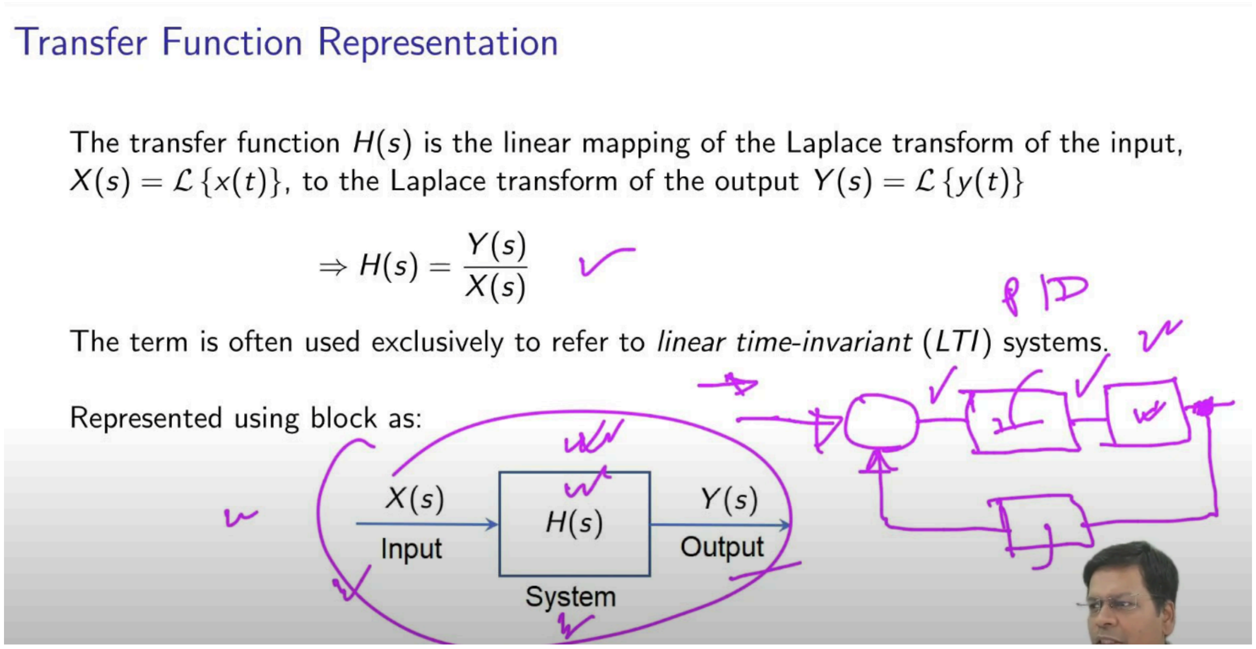
NPTEL Online Certification Courses
Industrial Robotics: Theories for Implementation
Dr Arun Dayal Udai
Department of Mechanical Engineering
Indian Institute of Technology (ISM) Dhanbad
Week: 11
Lecture 45

Transfer Function and State-space representation, ODE

Hi, in the last class, I discussed the behaviour of second-order linear systems. We saw how poles affect the behaviour of your system. So, in today's class, we will 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 will see how these tools can be used to understand the behaviour of your system in 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 $Y(s) = \mathcal{L}\{y(t)\}$

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

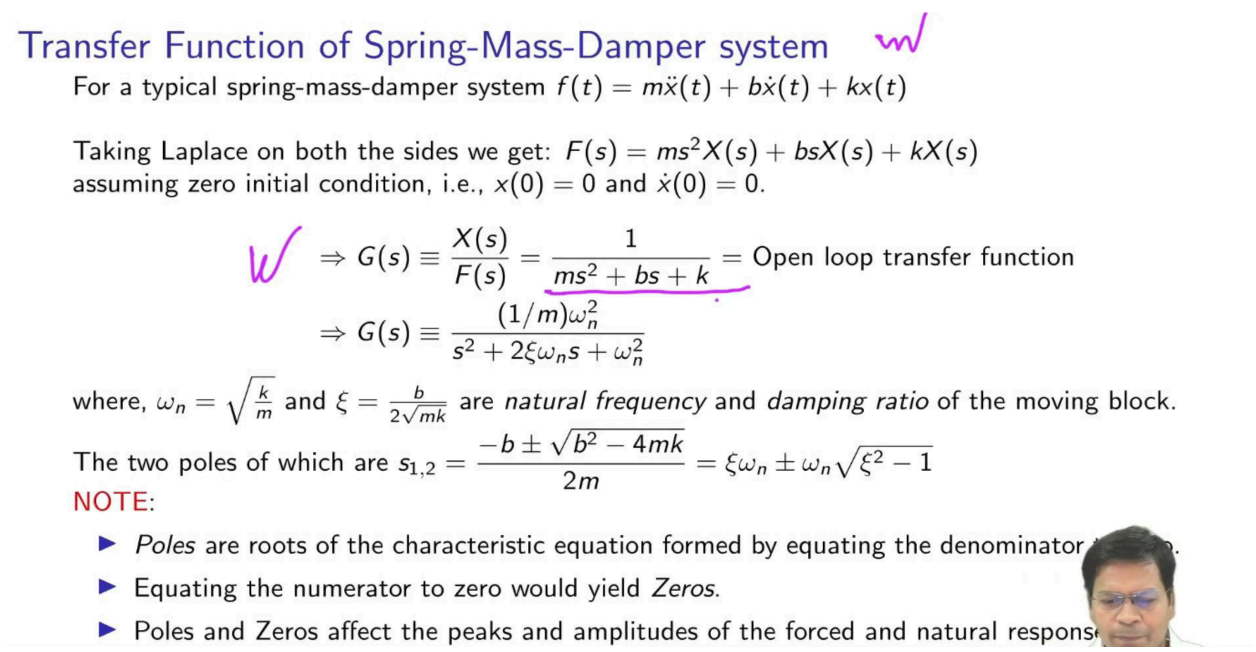
The term is often used exclusively to refer to *linear time-invariant* (LTI) systems.

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.

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

So, this term is quite often used extensively for referring to linear time-invariant systems, that is, LTI systems and linear systems, but they are time-invariant. Time invariant means the properties of the system don't change. Let us say you have a system like a spring mass damper system only. So, if it is f, m, x, double dot, b, x dot plus k, x, so m, that is the mass damping ratio, and the stiffness of the spring doesn't change with time. So, that is why 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, i.e., H(s). 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 c.Omparator. So, you have a controller over here. This can be 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. That is the desired input, This is the output which is sensed. So, you have a transfer function which is over here. That is the feedback transfer function and 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 which is used to represent the whole of the system sometimes. 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 which looks very much like this: and over here in this transfer function. You have a numerator, you have a denominator, just like this, and maybe you have this can state the behaviour of your system. Also. We will 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

So yes, let us start with a simple system: a 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. So, it can be linearised to say almost near to this. So, that is the reason I am using this quite often. So, yes, most of the mechanical systems should be very near to this. If not exactly, this. And there are terms which are analogous to m, b and k which are put here. So, that is the reason this is quite commonly used, and yes, taking Laplace on both sides, you have seen, with the initial condition, if they are zero, that is, x0 is zero and x dot zero is zero, that means the initial position and velocity are at zero. So, if you take Laplace. So, 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 sX(s). So, every time you take Laplace, there is a term that comes here which says it has x0 at the end, or x dot zero at the end. So, because those terms are zero, you are just remaining with X(s)s over here.
Similarly, k times of at will become k being constant; it will go here, and you have is that comes here. So, taking X(s) as common in the whole of this expression, x can be taken out. So, you can see a relation which is very much like this.

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

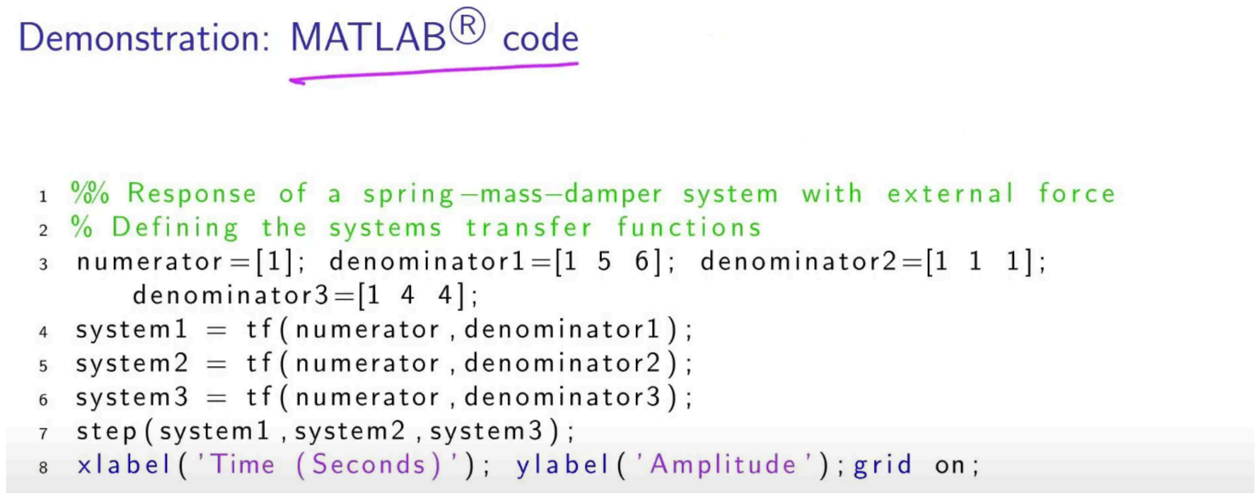
So, if you take their ratio, it gives you G(s). That is the transfer function. So, it says 1 by ms square plus bs plus k. So, this is the open loop transfer function of a spring mass damper system, where the output X(s) and the input are related by this.

$$\frac{X(s)}{F(s)}$$

So, your block would say something like this: you have one by ms square plus bs plus k. So, you have input that comes here, that is, the F(s), and you have output that goes here that is X(s). So, output by input will give you the transfer function, that is, which is here. So, this is what is meant by a transfer function. Let us see how this represents the behaviour of your system. You can clearly see you have something in the denominator. You are very much familiar with this already. So, this was what. 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, this can alternately be put like this. So, we are not very much concerned in this form 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, and the denominator. So, this also gives you a similar characteristics equation and this will also take you to the same place. So yes, there are two poles s1 and s2. That is given by this, you have already seen that and discriminant over here basically will tell how are your roots. Roots are real and imaginary, so both, the parts are there, or it is purely real, or it is purely imaginary, or roots are equal, or it can be just zero. So, it all depends on how your roots are, and that will govern your system behaviour also. So, the roots can also be expressed like this:

$$\xi\omega_{n}\ \pm\omega_{n}\sqrt{\xi^{2}-1}$$

So yes, the poles are nothing but the roots of the characteristics' equation formed by equating the denominator to zero. So, this is your denominator, to take the roots of that that is known as the poles. So, equating the numerator to zero, that is zeros. So, the roots of the numerator are known as zeros. Poles and zeros affect the peaks and the amplitude of forced and natural responses 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 characteristics' equation, and the roots of this basically govern how your system is going to behave, naturally as well as by acted upon by, if it is acted upon by, any external force. So, you have seen the behaviour already using this. So, we need not discuss much. So, the idea is to come out with the transfer function of your spring mass damper system. So, that is what is observed here.

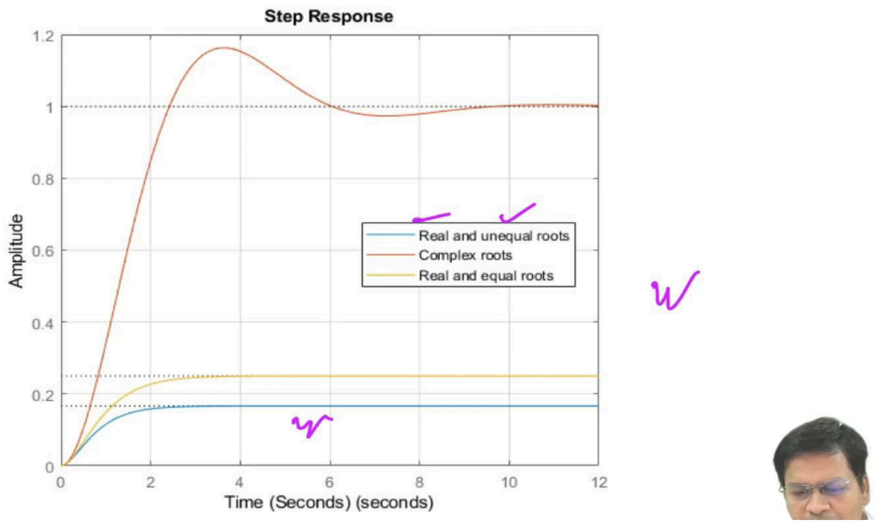


Demonstration: MATLAB® code

```
%% 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;
```

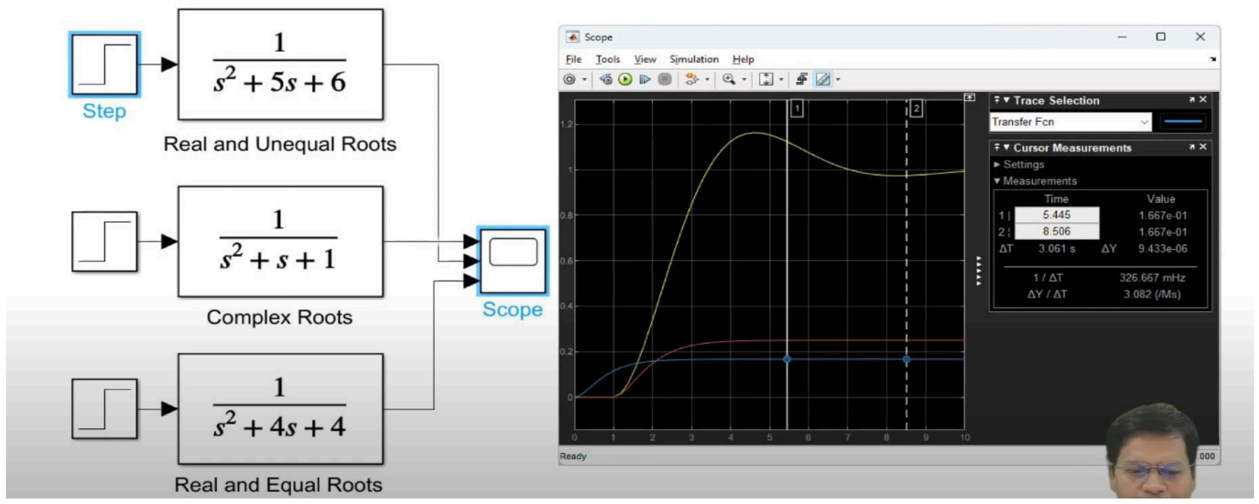
Using Matlab code also, you can analyse your system by simply expressing your system like a transfer function. So, you know, now you have a transfer function with the numerator as one, and the denominator simply is the polynomial definition that goes at the denominator. So, in the case of the system that we have already seen, it has one, five and six. So, you see, it is ms square plus five s plus six. So, you can just see it is b square minus four mk. So, m here is one. So, it is b square, that is twenty-five minus four into m, into one that is twenty-four, so it is one. So, you see you have a discriminant of this that is telling. It is greater than zero. So, you have two roots. Both are unequal, and they are real, also. So, that is an over-damped system. So, you have seen the behaviour already. So, that is put here in the denominator. The second system is one, one, one that is s square plus s plus one. And the third one is the one that we have already analysed. All these three, so it is one four, four. So, we already know how the behaviour should come 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 system, that is, system one, system two and system three 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 one and, two and three are plotted here. So, this is just a label. So, they are plotted here.

MATLAB® Plots



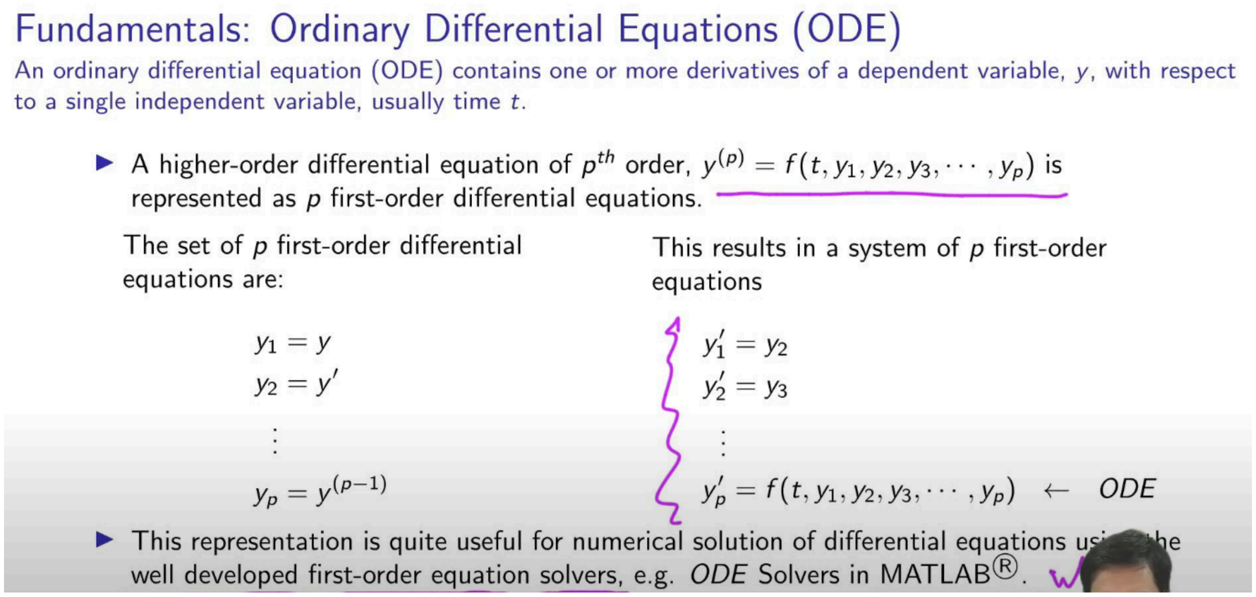
So, let me just see. It is something like this: so you have a system with real and unequal roots. So, that is coming here. This is an over-damped system. And next one is real and in equal roots. Real and equal root, that is your critically damped system that comes here. And then you have complex roots, so 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 is the transfer function, 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.

Motion of the Block using MATLAB® and Simulink



	Time	Value
1	5.445	1.667e-01
2	8.506	1.667e-01
ΔT	3.061 s	ΔY 9.433e-06
1 / ΔT		326.667 mHz
ΔY / ΔT		3.082 (/Ms)

Now, 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 for programming. So, you have a step input which is fed to the block with a transfer function defined like this. This is the first system, this is the second system, this is your third system. So, all of these, you already have done it. 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 which has three inputs. So, you, 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 the plot? So, yes, this is an additional tool which 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:

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

This results in a system of p first-order equations

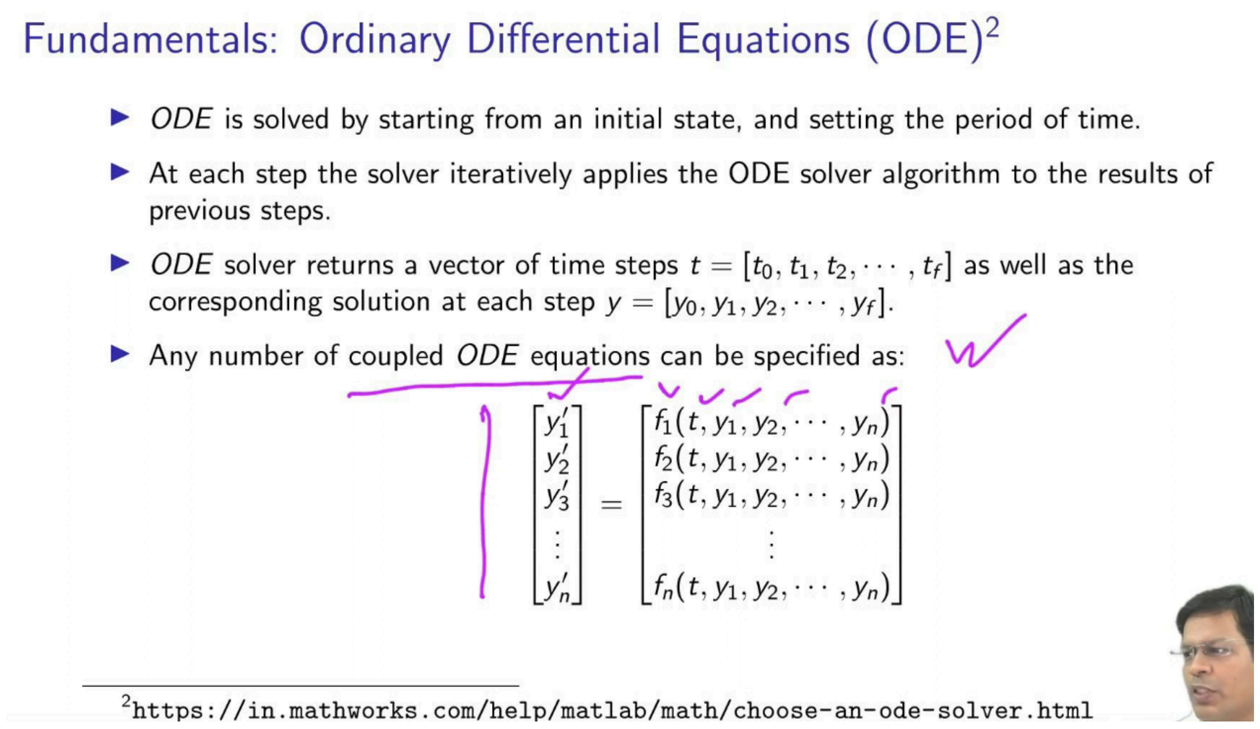
$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ &\vdots \\ y_p' &= f(t, y_1, y_2, y_3, \cdots, y_p) \quad \leftarrow \quad ODE \end{aligned}$$

- This representation is quite useful for numerical solution of differential equations us[illegible]he 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 the Ordinary Differential Equation and the kind of thing we are going to use it 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. In the case of a physics-based system like ours, like a robot. So yes, this is how it is defined. So, your system normally should look like this:

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

That is the pth order derivative of y, which is here. So, that is equal to, and it is equal to, a function of time, and y1, y2, y3 and y_p; what are those? So, they are basically derivatives of y. So, higher order differential equations of pth order can now be represented as p first order differential equations. There is a reason why we are converting this. Higher-order differential equation to a set of first-order differential equations. If it is, the highest order is y_p. That is the pth order. So, you have p first order differential equations will be formed. So, yes, the p first-order differential equations are. They look like this. So, it is. These are different variables. Now, these are derivatives of y. this is no derivative. This is the first derivative and p minus one derivative. So, where y1 is equal to y, y2 is equal to y dash. Similarly, there should be y3 is equal to y double dash, and so on, so forth. Then, y_p is equal to y, p minus one of the derivatives. If you substitute this to your original differential equation, this results in a system of p first-order equations split. 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 it is y2 dash. That can be obtained from here. So, that comes here. Similarly, y3 dash will be equal to y4, and so on and so forth. Then, finally, you will get 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? 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, and ODE4. Those are nothing but parachute 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 a kind of equation which is like this. So,,,,, you have converted it to 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 to a set of equations which are 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:

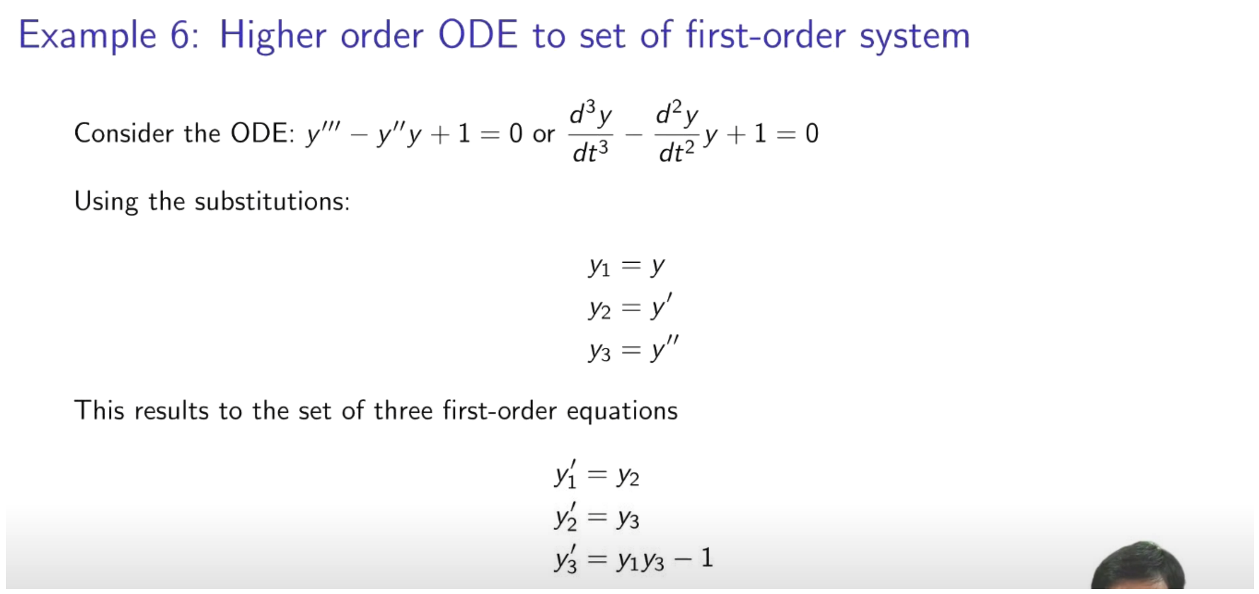
$$\begin{bmatrix} y_1' \\ y_2' \\ y_3' \\ \vdots \\ y_n' \end{bmatrix} = \begin{bmatrix} f_1(t, y_1, y_2, \cdots, y_n) \\ f_2(t, y_1, y_2, \cdots, y_n) \\ f_3(t, y_1, y_2, \cdots, y_n) \\ \vdots \\ f_n(t, y_1, y_2, \cdots, y_n) \end{bmatrix}$$

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

So, what is ODE square 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 iterate. It iterates from the initial step to the next to the next, and finally, it gives you a vector of time steps and time steps; that is, t is equal to c, 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, and so forth, till all the values of y. So, all the y's that you saw in the previous slide. So, that is 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 an acceleration, you get velocity, and you get the 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 can be 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 have different functions. They are all coupled ODE equations, which have the same input variables and one output variable. So, not basically the input and output variables basically, these are the derivatives. Got it. So, this is how it can be expressed. So, this is all a set of first-order equations.



Example 6: Higher order ODE to set of first-order system

Consider the ODE: $y''' - y''y + 1 = 0$ or $\dfrac{d^3y}{dt^3} - \dfrac{d^2y}{dt^2}y + 1 = 0$

Using the substitutions:

$$y_1 = y$$
$$y_2 = y'$$
$$y_3 = y''$$

This results to the set of three first-order equations

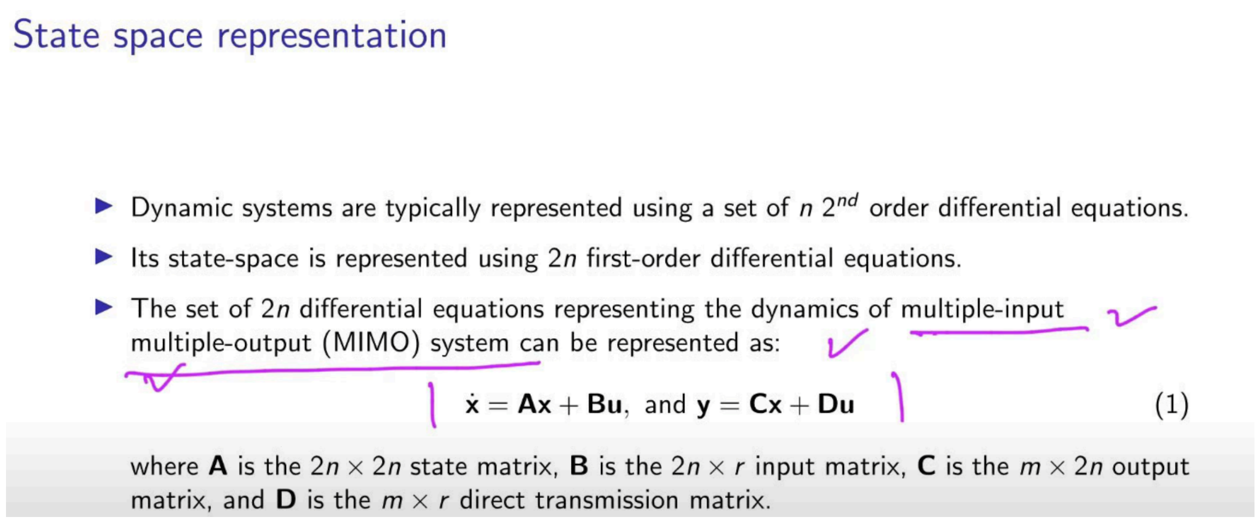
$$y_1' = y_2$$
$$y_2' = y_3$$
$$y_3' = y_1 y_3 - 1$$

So now, just take one example of an 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:

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

It is a third-order equation. So, d cube y by dt cube. So, it is, with respect to time, d square y by dt square, and you have y plus one. So, the system can be represented like this in a compact manner. So, prime here, y single prime is basically dy by dt. Double prime is d square y by dr, and y triple prime is equal to d cube y by dt cube. So, this is how they are.

So, now I'll just use the substitutions for y, y prime and y double prime, so all the derivatives. I have this substitution. So, y1, y2 and y3. I'll put them to the original equation that is here and I can get to this. So, 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 depend on the order of your original differential equation of the system. So, it is if it is three. So, you got three over here. So, each one of them are first order equation. They can be packed together and put to an ODE solver, and you can get the solution to that. So, we'll stop the discussion over here, and we'll move further with our system again.



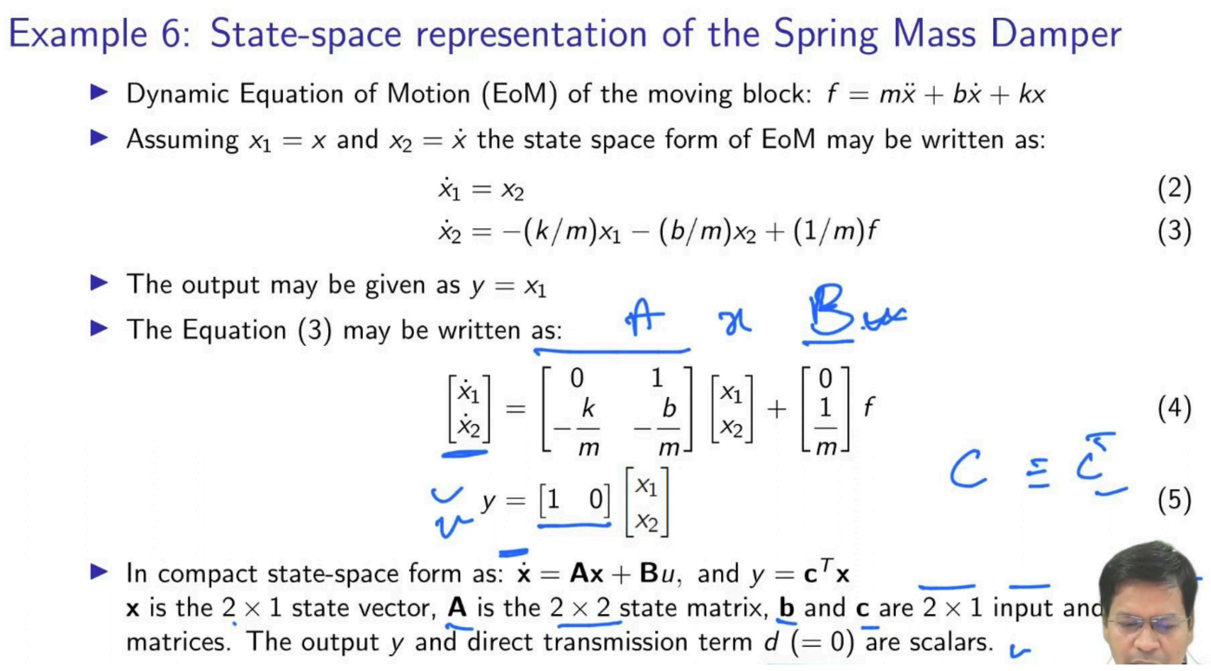
State space representation

- Dynamic systems are typically represented using a set of n 2^{nd} order differential equations.
- Its state-space is represented using $2n$ first-order differential equations.
- The set of $2n$ differential equations representing the dynamics of multiple-input multiple-output (MIMO) system can be represented as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}, \text{ and } \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{1}$$

where $\mathbf{A}$ is the $2n \times 2n$ state matrix, $\mathbf{B}$ is the $2n \times r$ input matrix, $\mathbf{C}$ is the $m \times 2n$ output matrix, and $\mathbf{D}$ is the $m \times r$ direct transmission matrix.

So, let us come back to State space representation. So, any dynamical system is typically represented using a set of n second-order differential equations. So, if it is of the second-order differential equation, the number of first-order differential equations that can be formed out of each one of them should be two. 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 multiple input and multiple output systems and there were 2n differential equations, first-order equations that represent your dynamic system. So, each one of them can be clubbed together in a compact manner like this. 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 m by 2n output. Matrix D is the m cross r direct transmission matrix. So, each one of them has got some significance. We'll see by example.



Example 6: State-space representation of the Spring Mass Damper

- Dynamic Equation of Motion (EoM) of the moving block: $f = m\ddot{x} + b\dot{x} + kx$
- Assuming $x_1 = x$ and $x_2 = \dot{x}$ the state space form of EoM may be written as:

$$\dot{x}_1 = x_2 \tag{2}$$

$$\dot{x}_2 = -(k/m)x_1 - (b/m)x_2 + (1/m)f \tag{3}$$

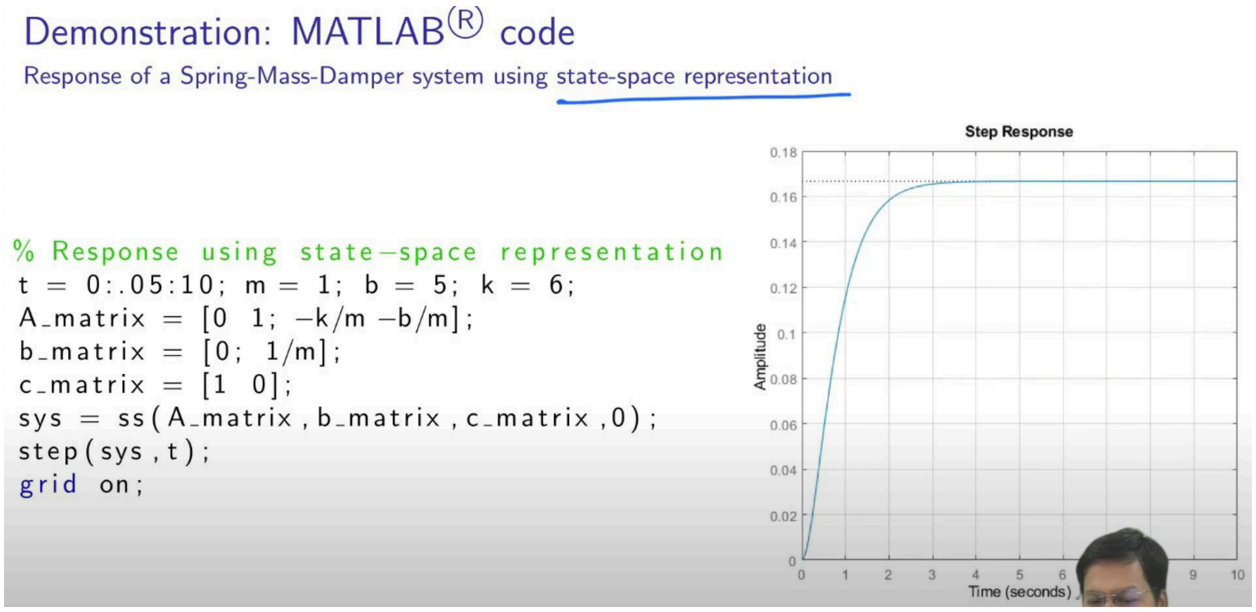
- The output may be given as $y = x_1$
- The Equation (3) may be written as:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\frac{k}{m} & -\frac{b}{m}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix} f \tag{4}$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} \tag{5}$$

- In compact state-space form as: $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$, and $y = \mathbf{c}^T\mathbf{x}$
  $\mathbf{x}$ is the 2×1 state vector, $\mathbf{A}$ is the 2×2 state matrix, $\mathbf{b}$ and $\mathbf{c}$ are 2×1 input and
  matrices. The output y and direct transmission term $d\ (= 0)$ are scalars.

So, let us come back to our old friend, who is the spring mass damper once again. So, dynamic equation of motion. This time is the same. So, that is, f is equal to mx double dot, bx dot plus kx. Now, I want to express this as a state space representation. So, what should I do? So yes, I'll assume the substitution, which is like this: 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 should it tell me? So, if it is x1 dot, so 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 km, 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 a plus over here. So, yes, now you substitute these again here. So, when it is x, you can say it is x1, so write x1 here. If it is x dot, you can quickly write x2 over here, right, and this remains 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. So, these are the derivatives. Now, the output part. So, y is equal to x1, so that is very trivial. So, there is nothing called d over here; that is, that turns out to be zero in this case. So, that was your output equation. So, equation three: this one can now be written as two and three, both of them together in a compact manner. So, I have put these two over here and again, if you see, I have put x1 and x2 directly over here. So, that is appearing as x1 and x2. Both are 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 equation three 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 one. So, it is zero and one 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 zero. So, it is expressed like this: and y, y has got just x1 term. So, it can be written as this: one multiplied by x1 and zero 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 you 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 matrices. B and C are 2 cross-1 input and output matrices. The output now is a direct transmission term, with d is equal to zero. 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, yes, 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 also, we can analyse my system.

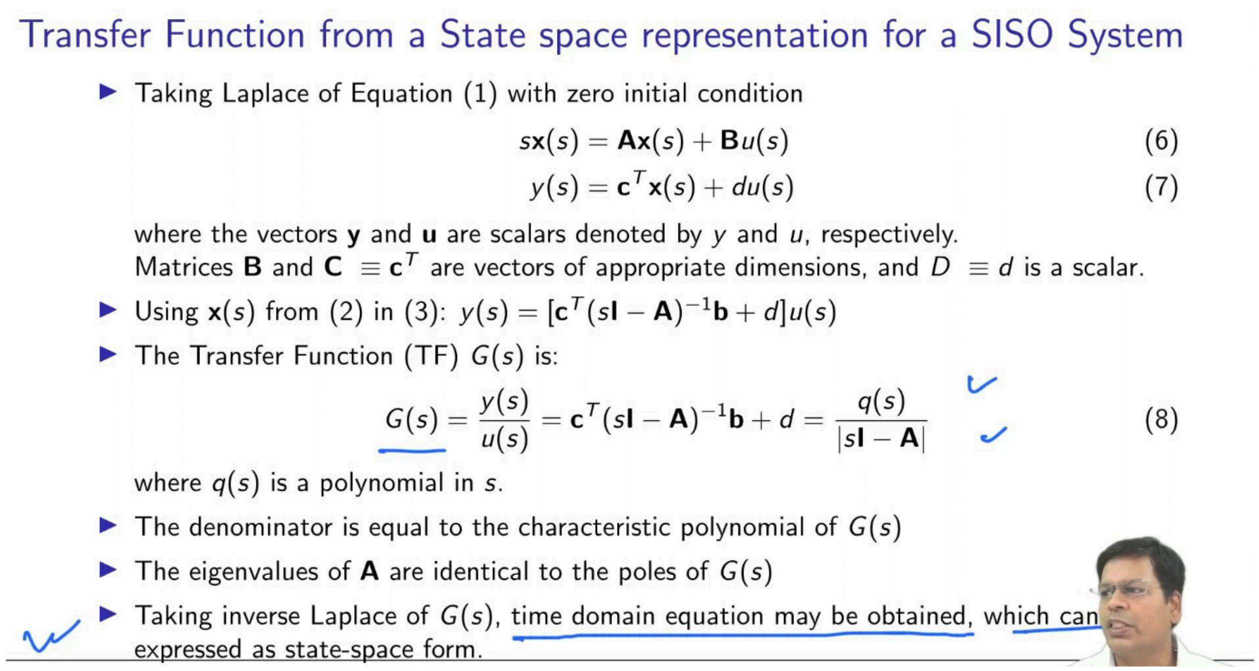


Demonstration: MATLAB® code

Response of a Spring-Mass-Damper system using state-space representation

```
% Response using state−space representation
t = 0:.05:10; m = 1; b = 5; k = 6;
A_matrix = [0  1; −k/m −b/m];
b_matrix = [0; 1/m];
c_matrix = [1  0];
sys = ss(A_matrix,b_matrix,c_matrix,0);
step(sys,t);
grid on;
```

So, let us just see if the spring mass damper is in the state space representation. I am using Matlab now to see its behaviour for step input. So, how is it defined in Matlab? So, your system, let us say it is your- m is equal to 1, b is equal to 5, and k is equal to 6. It's an over-damped system for time variation from 0 to 10 seconds. I want to do that. So, a matrix. I have just put it exactly as it is appearing here. So, I have defined my matrix- a, b and c over here and created a state space system of equations, stored it here in says. So, a, b, c and d come here. So, step input to this system, with time t variation 0 to 10 at a step of 0.05 seconds. So, it is plotted. So, it gives me a similar behaviour, the way it should be right, 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 your system. How this is all done in the background, basically.



Transfer Function from a State space representation for a SISO System

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

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

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

where the vectors **y** and **u** are scalars denoted by y and u, respectively.
Matrices **B** and **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:

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

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

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

The transfer function from state space is obtained using this single input and single output system. It has just one input, that 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 was your system? I'll just show you once again over here. Yes, so this is your state space equation: x dot is equal to Ax plus Bu, and y is equal to Cx plus Du input and output relation.

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

So, taking Laplace 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 Laplace should give me s of x(s) and A times of x(s) plus Bu(s) b is the matrix, and u is in Laplace. So, that will appear here.

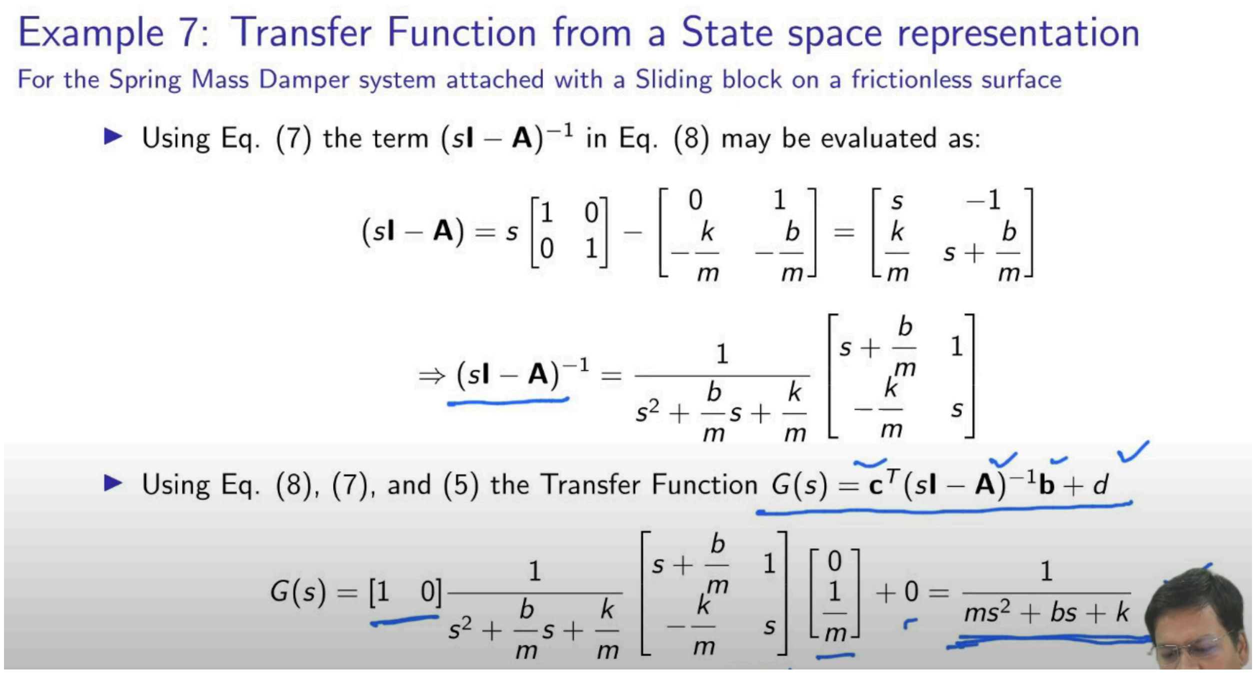
$$\ddot{x} = Ax + Bu$$

$$sx(s) = Ax(s) + Bu(s)$$

Similarly, if you convert y now, so y was Cx plus Du. So, if you take Laplace, you should get is y(s) equal to instead of c. I know it is. Both were equivalent. So, those constants are put here. It is here plus d us, if at all 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 is from this equation, that is, equation number six, and substitute it here in this one. 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, it is diagonal elements are one, rest are on zero. So, it is a unit matrix minus A whole inverse b plus du (s), correct, so it becomes exactly this. You can try doing it yourself. So, you can just take it out from here and substitute it here. You will get to this. 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.

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

So, this is your output, and this is your input. So, this becomes your transfer function. So, what is it here, basically? 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 is this one. So, now the denominator is the characteristics polynomial of is. You already know that. So, the denominator of the should give you the characteristics equation of this system. So, the roots of this determinant should tell me the behaviour of the system as well. So, the eigenvalues of an are identical to the poles of is. You see, see, if I equate this also to zero, that means I am finding out the eigenvalues, and the same goes for finding out the roots. Also, they are identical. So, taking the inverse Laplace of G(s), that is, 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 you can express them by proper substitution to state space form. So, both ways you can do. You can do the transfer function from state space representation, or you can do otherwise, also right. So, both are equally helpful. So, at least in the SISO system, it goes like this.



Example 7: Transfer Function from a State space representation

For the Spring Mass Damper system attached with a Sliding block on a frictionless surface

- Using Eq. (7) the term $(s\mathbf{I}-\mathbf{A})^{-1}$ in Eq. (8) may be evaluated as:

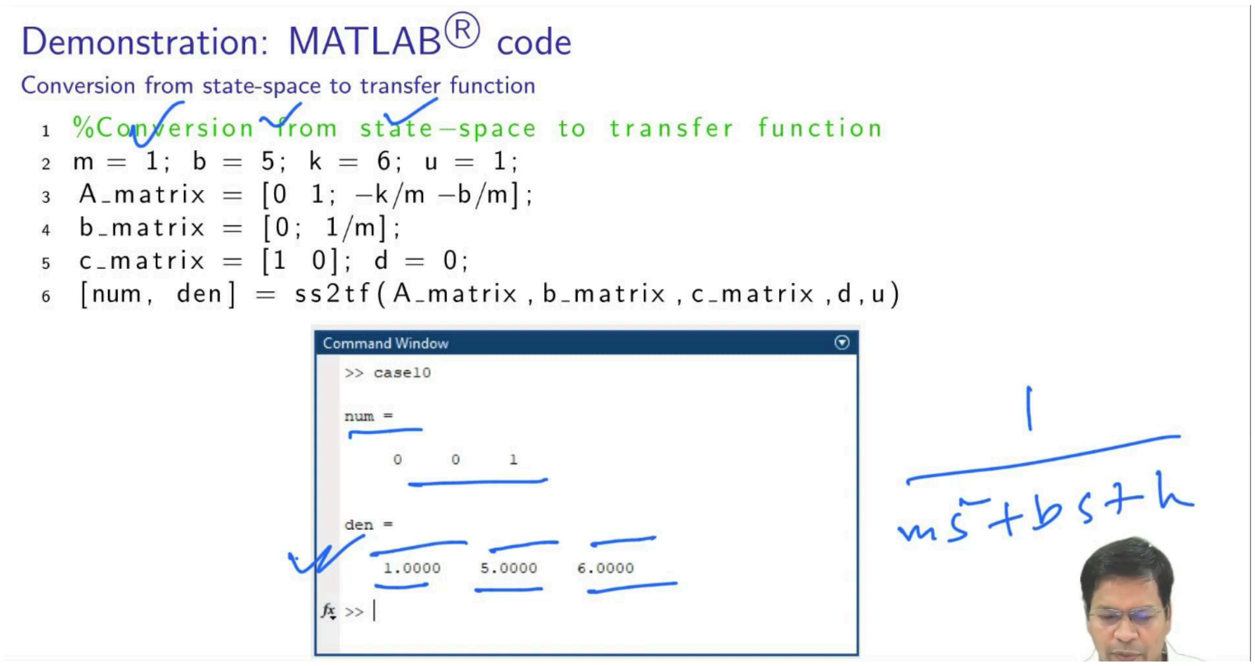
$$(s\mathbf{I}-\mathbf{A}) = s\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} - \begin{bmatrix}0 & 1\\ -\frac{k}{m} & -\frac{b}{m}\end{bmatrix} = \begin{bmatrix}s & -1\\ \frac{k}{m} & s+\frac{b}{m}\end{bmatrix}$$

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

- Using Eq. (8), (7), and (5) the Transfer Function $\underline{G(s) = \mathbf{c}^T(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{b}+d}$

$$G(s) = \underline{[1 \quad 0]}\frac{1}{s^2+\frac{b}{m}s+\frac{k}{m}}\begin{bmatrix}s+\frac{b}{m} & 1\\ -\frac{k}{m} & s\end{bmatrix}\begin{bmatrix}0\\ \frac{1}{\underline{m}}\end{bmatrix} + 0 = \underline{\frac{1}{ms^2+bs+k}}$$

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 state space representation of your spring mass damper. You see? You see what your A was. It is a matrix that is shown here, got it? So it is exactly this one. That is your matrix A. This is your B matrix, and then you have c transpose, that is C matrix, and the output matrix and input matrix. They look like this. So, copying that from here, I can now put all those values to this. So, sI is because I know it was a second-order system I is compatible with. So, it is diagonal elements one. So, it is sI minus A. So, this is a that comes here. So, if I do that, I'll get to this. So, again, taking the inverse is very trivial, so you just take the inverse of what you will get. You get something like this in the denominator: that is nothing but the determinant of this matrix, got it? So that is that is coming here and the whole of the transpose, that comes here. 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. So, is 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 goes here. d is also here. d is equal to 0, which comes here. So, this is your b. this is exactly minus an 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 earlier seen 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 characteristics equation. So, this is what was also expected. So, the roots of this will tell the behaviour of my spring mass damper system. So, I can analyse it anyway now.



Demonstration: MATLAB® code

Conversion from state-space to transfer function

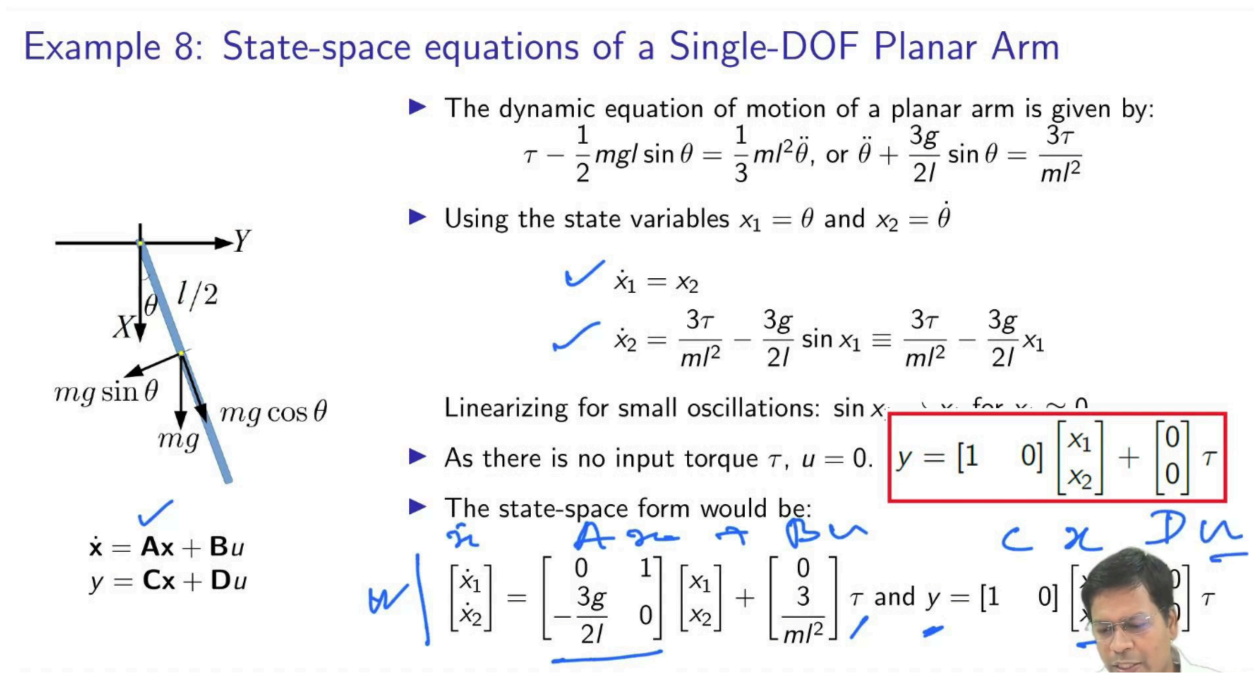
```
%Conversion from state-space to transfer function
m = 1; b = 5; k = 6; u = 1;
 A_matrix = [0 1; -k/m -b/m];
 b_matrix = [0; 1/m];
 c_matrix = [1 0]; d = 0;
 [num, den] = ss2tf(A_matrix,b_matrix,c_matrix,d,u)
```

Command Window

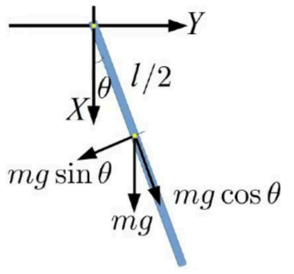
```
>> case10

num =

     0     0     1


den =

    1.0000    5.0000    6.0000

>> 
```

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

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. So, that is here: 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 this says is coefficients. Basically, these are coefficients. You see, this is m, b and k: 1 by ms square plus bs plus k. So, the coefficient of s square, that is, m, that 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



✓

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

- 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: $\sin x_1 \to x_1$ for $x_1 \simeq 0$

- As there is no input torque τ, $u = 0$.
$$\boxed{y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix}\tau}$$
- The state-space form would be:

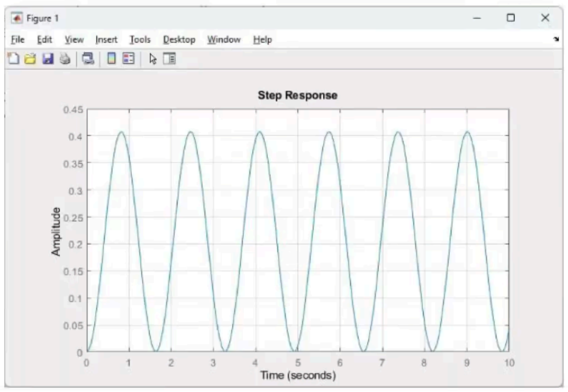
$\dot{x}$ $Ax + Bu$ Cx Du

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

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, and it is just a planar pendulum. It is not a conical pendulum which has two degrees of freedom over here. It is just one dot, one cylindrical joint, that comes here. You can go like this. So, it has a solid link with mass m. So, mg force that comes here, mg sine theta, you know that is the driving force, with l by two 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. 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, and this is the torque which is acted due to gravity. So, the difference of that is actually the driving torque. So, it is nothing but i theta double dot. I am in a 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 t. So, the same equation can now be written as this: what is theta double dot is equal to 3 taus by ml square minus 3 g by 2 l sine theta, so theta double dot. So, yes, now I can use these substitutions. So, x1 is equal to theta, and x2 is equal to theta dot. So, this is the first one that you know where it is. It is very, very trivial. So, x1 dot is basically theta dot, so that is equal to x2. This is the first one. 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 3 g by 2 l sine theta. So, instead of theta, I can write x1. So, that is coming here. for very small oscillation because, you see, this is a non-linear type of system, so it cannot be directly converted like a state space form. So, this time, you have to do a little linearisation. So, for you know, for very small oscillations, sine x1 tends to x1. So, very near to that, if you are making a small oscillation so that this approximation can be done, I can directly put, instead of sine x1, I can put x1. So, my new system, as there is no input torque t u becomes equal to 0, and the output relation can be written as y is equal to x1. So, putting them, or putting them all together here. So, it is this. So, it is x1 dot, x2 dot, that comes here. And extracting the terms now. So, this is you a that comes here: axe, x, dot is equal to axe plus by. So, this is your b and u, that is the tau, which is the external torque which is acting. So, again, the output is y is equal to c x. This is your x, and Du, d and u. So, u is again the input, and there is no d what it. So, this is how you have converted your single degree of freedom planar arm dynamic equation of motion to state space form. Now, it is integrable, and it can be analysed, and you can do anything with that. Let us do that.

Demonstration: MATLAB® code

```
% Response of a 1−DoF arm with zero input torque
t = 0:.05:10; m = 1; l = 1; g = 9.81;
A_matrix = [0 1; −3*g/(2*l) 0];
b_matrix = [0; 3/(m*l*l)]; c_matrix = [1 0];
sys = ss(A_matrix, b_matrix, c_matrix, 0);
step(sys, t); grid on;
```



So now, I'll analyse my that is a step-input response to this one-off arm with zero input torque. So, now I have defined my a, b, as we have obtained earlier in this. Exactly, I'll put this as a. I know the values here. M is equal to one. The length of the link is also one unit, and g is 9.8 meters per second square. L is one meter, and m is 1 kg. So, I have defined the whole of this here as we have obtained just the previous slide. So, I'll put those values here. And I have got the matrices now. So, I now have state space using the SS function. I have defined my SS function in says. So, that is a matrix b. matrix c and d goes here and step input is given and analysed with time. So, this gives me a plot, which is SHM. You see, it is the simple harmonic motion for a small oscillation. It behaves just like a pendulum, got it? So, it is your system, and it is what is expected because you are not acted upon by any external force, and there is no joint damping, which means it will keep oscillating forever. So, that is what is shown here as a result, also.

So, that's all for today, and in the next class we'll discuss a robot joint, a DC motor model. So, we'll go very near to what we are actually intending for. So, we will now start doing the joint model of a robot using a DC motor model. So, that's all for today. Thanks a lot.