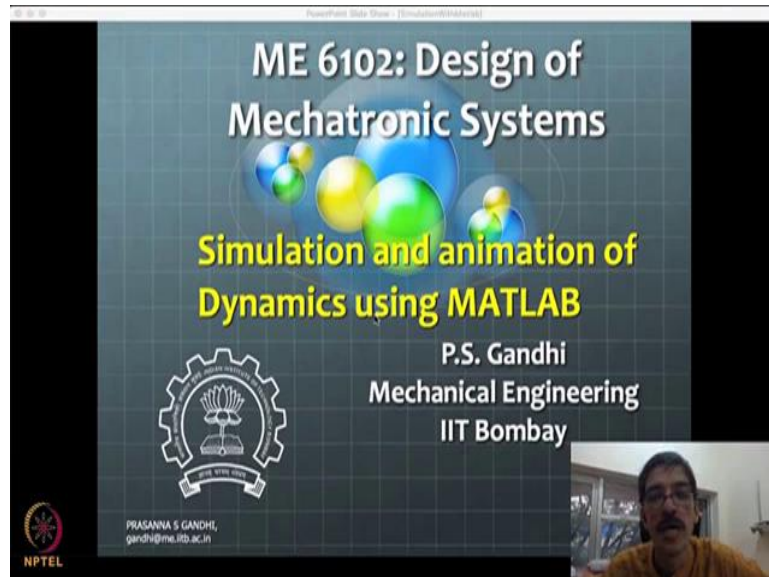


Design of Mechatronic Systems
Professor Prasanna S. Gandhi
Department of Mechanical Engineering
Indian Institute of Technology Bombay
Lecture 30
Fundamentals of Simulation of dynamics using MATLAB

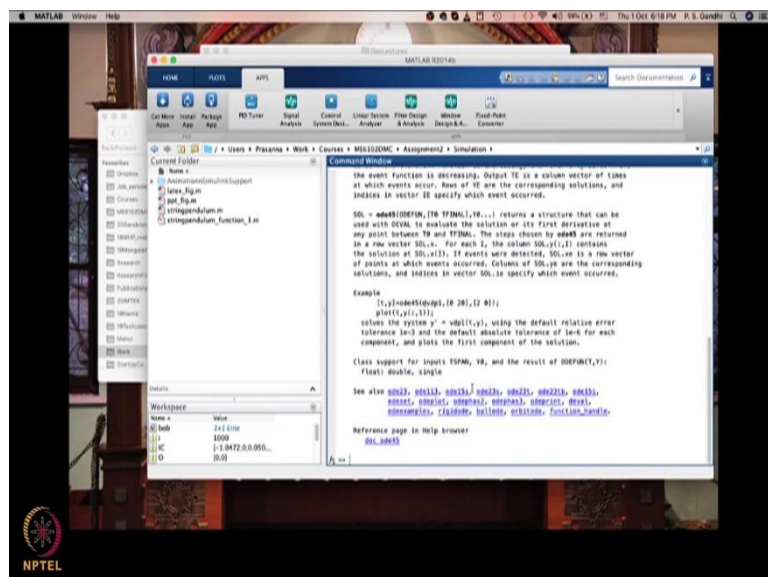
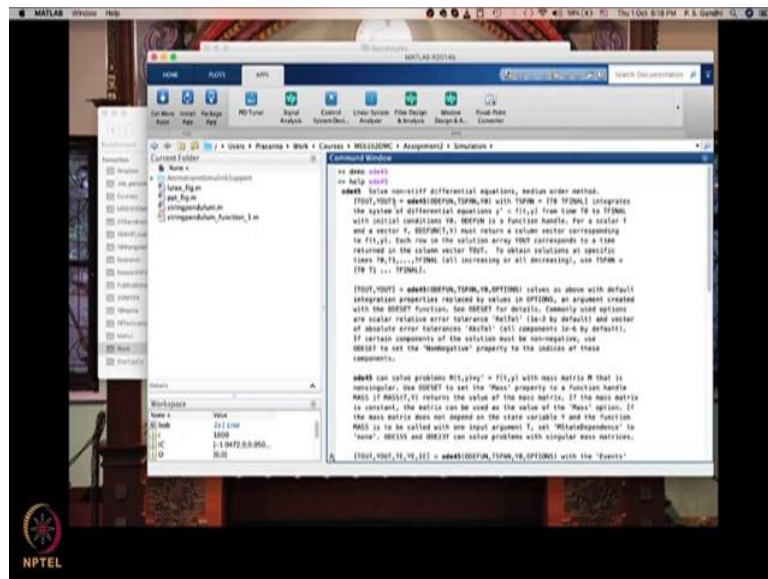
(Refer Slide Time: 00:15)



So, today's class we are going to look at simulation and animation of dynamics using MATLAB for mainly ordinary differential equations. How do you simulate ordinary differential equations and do their animation? Very interesting way, we can do very nice stuff using MATLAB function called ode45.

(Refer Slide Time: 00:42)

The slide has a dark blue background with a grid. The title 'Dynamics Representation for Simulation' is at the top in white. Below it, text explains that equations to be simulated are read from the help of the ode45 function in MATLAB. It lists three points: 1) Ordinary differential equation solvers in Matlab solve systems in the form $\dot{x} = f(x, t)$. 2) The state x is a vector and t is time. 3) All equations must be converted to this form. An example system $m\ddot{x} + c\dot{x} + kx = 0$ is given. A small icon of a person thinking is next to the text 'Think how can you convert it in the form needed above'. The state vector is defined as $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$. The derivatives are shown as $\dot{x}_1 = x_2$ and $\dot{x}_2 = \frac{-cx_2 - kx_1}{m}$. A final note states that the function will give these derivatives in the vector when the state x is given as input. A small video inset of the professor is in the bottom right corner. The NPTEL logo is in the bottom left.



So, before going to, through these like, so we need some representation of a system for the simulation. But I would say before going through this part, you may look at actually the MATLAB help on ode- function. So, for that maybe I will show you first what it produces and then like maybe we can come back.

So, we will go to this window and look at, see, this is, these are command I have showed here, help ode45. And it gives all these kind of non-stiff differential equation and so you just kind of read through all these things and then maybe we can come back and then whatever I am talking here will make much better sense to you.

So, let us come back here. So, you can read through this help. You can pause and read through or you can have your own MATLAB software and go through. So, let us come back now to the simulation here or PPT here. Now, so for this you can see now that this MATLAB ode function

will be able to solve the Ordinary differential equations which are represented in the, in this specific form. This form is like this, where this x here is a vector function and its only first derivative is here is equal to f of x, t and f is some general non-linear function of x and t .

So, and again this is a function of x as a state coming up here. So, whatever state it is a function of it is giving the derivative of the state up here that is what is a meaning of this equation. And this equation is what we need to convert all our systems to this form of equations. And we will see how we do that by an example. So, if you have a system of the kind simple spring mass damper system so $m\ddot{x}$ $c\dot{x}$ and kx are three terms is equal to 0. So, this is not in this form, because this has two derivatives up here. So, we need to kind of convert to this form, we need to do something.

So, what do we do? Think about this. What can be done to get into this form? Pause for a while here and see what comes to your mind and then we can see the next. What we can do is we introduce this additional states into the system. Since this is double derivative here and I want only single derivative of here, but many different kinds of variables can be possible.

So, I define these additional variables here x_1 and x_2 are the two states. They are defined such that when x_1 is equal to x and x_2 is equal to \dot{x} , so when I take derivative x_1 dot it will be equal to \dot{x} here and \dot{x}_2 will be equal to \ddot{x} . And that is how I can get this state derivative here.

Now, I want to represent the state derivative in the, in this form here. So, how do I represent that? When I take \dot{x} , you can see that \dot{x}_1 is equal to \dot{x} . Now, I cannot use x variable, because I have to convert, I have already transform this system into these two additional variables. So, I will use now \dot{x} I know is to be defined as, is defined as x_2 . So, \dot{x}_1 is equal to x_2 will form my first equation.

And then \dot{x}_2 is equal to again this expression comes from this equation. And here again, I cannot use x , I have to substitute them with either x_1 or x_2 which is done here. So, this is how I can get this function defined. So, this is my f_1 here in this equation, and this is my f_2 in this equation and this function is now ready for coding or for doing the ode solver.

(Refer Slide Time: 05:18)

Ex: Spring pendulum system

Equations to be simulated

- Ordinary differential equations
- Spring pendulum system: Pendulum considered as rigid body, l as unstretched length of spring and x is deformation. Spring is nonlinear with total spring force

Damping is considered to be there in θ direction as well as in x direction. Rigid pendulum has mass m and radius of gyration r . Equations of motion are given by

$$m\ddot{x} - m(l+x)\dot{\theta}^2 + k_0x + k_1x^3 - mg \cos \theta + B_x\dot{x} = 0$$

$$[mr^2 + m(l+x)^2]\ddot{\theta} + 2m\dot{x}\dot{\theta}(l+x) + mg(l+x)\sin \theta = 0$$

PRASANNA S GANDHI,
gandhi@me.iitb.ac.in

NPTEL

So, now, we will see these for, applied for the spring pendulum kind of a system example. So, again the ordinary differential equations we simulate which were, which will come for the spring pendulum system, I will show you what these equations are in a minute. But just to give settings like this is a pendulum is considered as rigid body and it has its own inertia of rotation about the z -axis and radius of gyration or mass moment of inertia will be coming from the radius of gyration is given as r . So, mass moment of inertia is mr^2 for this bob around CG, axis passing through CG and z -axis, perpendicular to this page.

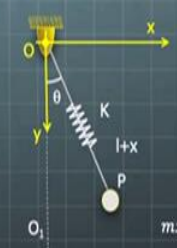
Then the spring is like considered as a stiffness, non-linear stiffness. Why, because typically like for the actual example or actual practical case, this can be a rubber like a string and rubber as it stretches, its stiffness starts increasing. So, it has some kind of a cubic non-linearity which will be increasing the stiffness as the stretching happens. So, that will, that is also incorporated in the equations. So, considering this non-linear spring and important other thing is this L is unstretched length of this spring. This unstretched length is L . And then beyond that the stretch x is deformation that happens.

With this kind of setting these are the equations of the system that are obtained. So, now you can see there are two equations which are second order equation, $m\ddot{x}$, equation in the direction of this L . It moves like up and down like that here. And then like there is a equation in the direction of θ . So, $\ddot{\theta}$ direction, θ is another kind of degree of freedom. So, θ is a degree of freedom, x is another degree of freedom and you have this two degrees of freedom system with the two differential equations governing that dynamics.

(Refer Slide Time: 07:41)

Ex.: Spring pendulum system

How to represent in the form required by ODE solver
Define vector \mathbf{x}



$$\mathbf{x} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \theta \\ x \\ \dot{\theta} \\ \dot{x} \end{bmatrix} \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} y_2 \\ y_4 \\ \ddot{\theta} \\ \ddot{x} \end{bmatrix}$$

Get derivative $\dot{\mathbf{x}}$ from the equations of dynamics
And substitute everything in terms of state variables y_1, y_2, y_3, y_4

$$m\ddot{x} - m(l+x)\dot{\theta}^2 + k_0x + k_1x^3 - mg \cos \theta$$

$$[mr^2 + m(l+x)^2]\ddot{\theta} + 2m\dot{x}\dot{\theta}(l+x) + mg(l+x) \sin \theta$$


PRASANNA S GANDHI,
gandhi@me.iitb.ac.in

NPTEL

Ex: Spring pendulum system

Equations to be simulated

- Ordinary differential equations
- Spring pendulum system: Pendulum considered as rigid body. l as unstretched length of spring and x is deformation. Spring is nonlinear with total spring force



Damping is considered to be there in theta direction as well as in x direction. Rigid pendulum has mass m and radius of gyration r . Equations of motion are given by

$$m\ddot{x} - m(l+x)\dot{\theta}^2 + k_0x + k_1x^3 - mg \cos \theta + B_x\dot{x} = 0$$

$$[mr^2 + m(l+x)^2]\ddot{\theta} + 2m\dot{x}\dot{\theta}(l+x) + mg(l+x) \sin \theta + B_\theta\dot{\theta} = 0$$

PRASANNA S GANDHI,
gandhi@me.iitb.ac.in

NPTEL

Now, the question is how do we convert them into our form, which, will you remember the form, is $\dot{\mathbf{x}}$ is equal to f of \mathbf{x} and t . So, again now we need to define like new vector of four variables. So, what is the variable of your choice? Here since we have these two degrees of freedom x and θ , naturally we use them in the, their derivatives in the, in defining this vector. In the similar way as we did for the simplest pendulum system.

So, $\dot{\theta}$ we say as y_2 variable, θ as y_1 variable, then y_3 is x , y_4 is \dot{x} . And now if we see the derivative of this state, it is having like this $\dot{y}_1, \dot{y}_2, \dot{y}_3, \dot{y}_4$. Now, \dot{y}_1 by definition is $\dot{\theta}$ here and $\dot{\theta}$ by definition is y_2 , so this becomes y_2 . Like that y_2 dot by definition becomes $\ddot{\theta}$, $\ddot{\theta}$ and \dot{y}_3 becomes y_4 and \dot{y}_4 becomes \ddot{x} . Now, this $\ddot{\theta}$ and \ddot{x} can be obtained from these previous equations of

dynamics that you saw here, these equations of dynamics. This will give you the \ddot{x} and this equation will give you $\ddot{\theta}$.

So, how do you obtain them by using moving all these terms on the other side and then dividing that by m , you will get \ddot{x} . Moving all these terms on the other side and dividing them by this term, you get $\ddot{\theta}$. And then we substitute them here and our function f_1, f_2, f_3, f_4 is ready for simulation in MATLAB. Now, how do we code them? So, this is first function, like we have to code this function. And how to use this function in the ode45 command is what we will see actually on the, in the program.

(Refer Slide Time: 09:49)

Ex.: Spring pendulum system

Equations to be simulated


- How to develop code function file for ODE solver

```
function [ dy ] = stringpendulum_function_1( t,y )
%UNTITLED4 Summary of this function goes here
% Detailed explanation goes here
% These parameters should be actually in a different file and defined as
% global variables to be shared (since this is small simulation we can
% afford to spend time in trivially defining them every time)
m=0.03; rm=0.007; l=0.2; ko=15; k1=150; g=9.81;
zeta=0.002;
Btheta=2*zeta*sqrt(m^2*g*l^3);
Br=2*zeta*sqrt(m*(ko+k1*y(3)^2))*0;
dy = zeros(4,1);
dy(1) = y(2);
dy(2) = (-2*m*y(2)*y(4)*(l+y(3))-m*g*(l+y(3))*sin(y(1))
-y(2)*Btheta)/(m*(l+y(3))^2+0.4*m*rm^2);
dy(3) = y(4);
dy(4) = (m*(l+y(3))*y(2)^2-ko*y(3)-k1*(y(3)^3)+m*g*cos(y(1)))/
(m*(l+y(3))^2+0.4*m*rm^2);
```

NPTEL

Ex.: Spring pendulum system

- How to represent in the form required by ODE solver
- Define vector \mathbf{x}



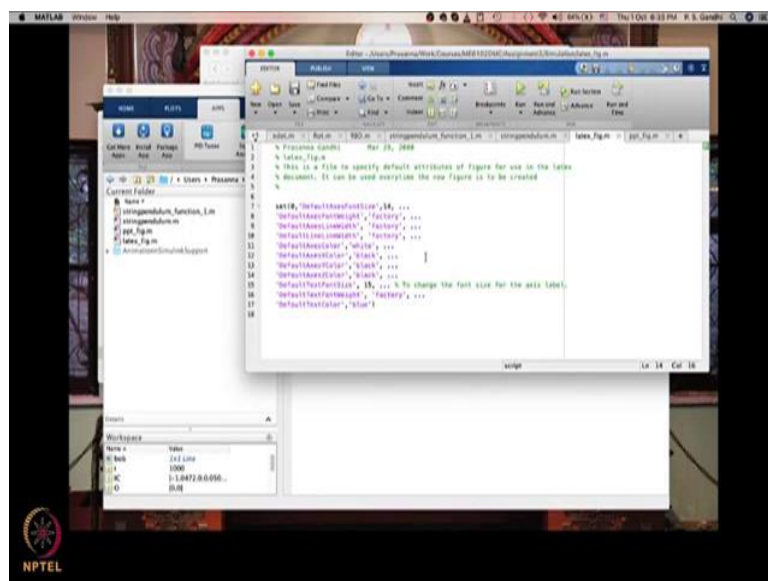
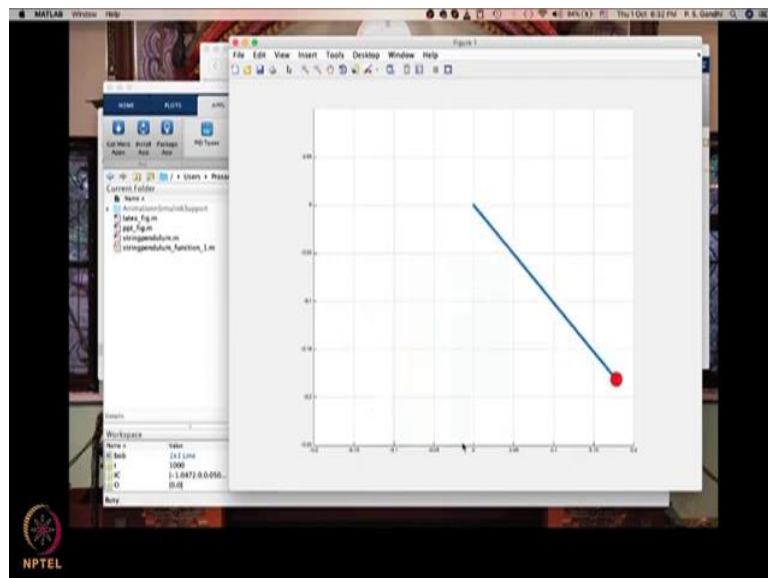
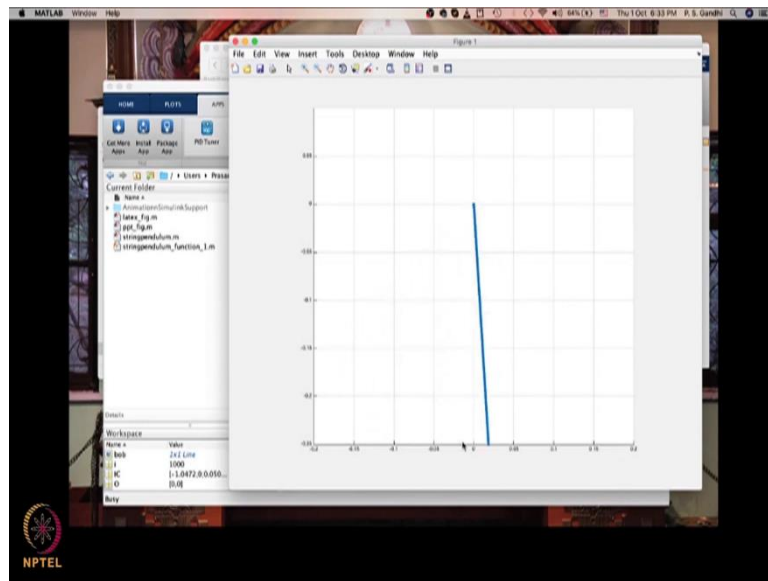
$$\mathbf{x} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \\ \dot{y}_4 \end{bmatrix} = \begin{bmatrix} y_2 \\ \dot{\theta} \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$

Get derivative $\dot{\mathbf{x}}$ from the equations of dynamics
And substitute everything in terms of state variables y_1, y_2, y_3, y_4

$$m\ddot{x} - m(l+x)\dot{\theta}^2 + k_0x + k_1x^3 - mg \cos \theta = 0$$

$$[mr^2 + m(l+x)^2]\ddot{\theta} + 2m\dot{x}\dot{\theta}(l+x) + mg(l+x)\sin \theta = 0$$

NPTEL



So, player for, so now how do we develop this code. So, I would suggest like you do not see this code first. You just go ahead and like write your own way this code. Assuming that this state is given, this vector is given, so if the function f you have to write for which the arguments are t and y . So, you are given y as a vector quantity. That is we have to keep in mind that it is coming in argument as a vector quantity and then you know this y_1, y_2, y_3, y_4 and from this you need to produce like their derivatives $\dot{y}_1, \dot{y}_2, \dot{y}_3, \dot{y}_4$ by using this part and these equations as we explained.

So, this is how you will write a code. And once you write a code then so pause it for now here, write your own code and then like when you come back then you will find like your mistakes or your understanding is missing somewhere that will open up like that becomes like a firm understanding within you. So, do not go to the next part unless you have written the code yourself.

Now, we will go to the next slide and see this code. So, here now I am defining all these terms in some kind of fashion, whatever some kind of a physical quantities and then writing, like so I am assuming like that for this function y is coming as an argument in vector form. So, y_1, y_2, y_3, y_4 will be my four states, y of 1, y of 2, y of 3 and y of 4 like that. And then so these, how it is used here. And then I need to produce a derivative.

So, dy , derivatives of these states, so dy_1 is equal to y_2 , dy_3 is equal to y_4 and dy_2 is, as I explained from this formula now wherever there is a θ you substitute corresponding y and then you can kind of get your equation coded. So, this is the equation for θ and this is the $\ddot{\theta}$ and this is the equation for \ddot{x} . This is how like you write this code and this function ends here. Once this function is written now that is see how we can go ahead and simulate it.

So, let us switch back to the MATLAB window. Now, here we have this function written in MATLAB. I will show you already different function here. This code gives you the function written as we have seen the slide. And now you can see all these arguments and functions written properly and you can write whatever comments that you want to write. How to use this function?

Now, once you write this function then let us save it. You use it in the ode command like this. So, for ode command, you need to define these initial conditions. Initial condition in this case are taken θ is equal to minus π by 2 and x , some deformation is given to the x to begin with

some spring is deformed by some amount and then time is defined for which you want to produce this output.

And then you can see that this time and initial conditions are going as other arguments to this `ode45` function along with the function that you have written. So, this will output the state evolution y in the same order, y is in the same order as you have coded this function. It will be output here and then time will come as a...

So, this is like a time trajectory of all the states that is coming up here. And then you have the animation creation up here. So, you define, create some figure and in that figure you draw these lines and this bob of the pendulum is drawn as a simple point or a circle with the size of the circle big enough for us to visualize it nicely.

So, that is how like these two, these things are plotted and you plot it, delete the plot and again replot it for the next update of the time like that you can do in the loop and create this animation. There are many different other ways also of creating animation. You may find some more links on the YouTube or the MATLAB website and you can create different, different kind of ways animations can be created. So, if you run this file now, you will find it produces this nice animation of the pendulum.

So, you can see that this pendulum is oscillating and at the same time it is having the vertical or the radial direction stretching and unstretching happening for the spring. So, of course, here we are not modeling it as a string. So, for example, like if the compressive force becomes too high like string cannot get compressed, but spring can get compressed.

So, if it is a rubber band like a string then it will just simply not get compressed further, but we are not like capturing that part here. So, if the spring force is always positive like there is some stretching in the spring always there then you will have no issue like this will be realistic kind of a simulation happening in the system.

Then I would like to kind of give you a little bit more attention to this other, I will close this window, part of the files. This *latex.fig* is a file that is written for setting the global variables of figure that can be controlled in a nice way. So, as to create a figures which are good for the latex document or their manuscript or paper that you write. So, this file will be of immense help if you run this file, then and then use figure command in MATLAB to plot any or plot command in MATLAB to plot any figures. Then all the settings will be done nicely for say latex kind of a document.

And you have other file where PPT underscore pip kind of name is given. This file when you run and plot your any of the figures then this will create a setting which is good for PowerPoint presentations. So, you may use these files in a way to, for your future activities as well. This is very interesting nice way to kind of create environments for you and you do not have to worry about like somebody pointing that where pics are looking so small, I cannot see them and those kind of things will not come into picture when you start presenting with this using each of these files. So, we will come back to our presentation now here.

(Refer Slide Time: 17:19)

Ex: 2R manipulator

Equations of dynamics have a form

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{11}\dot{q}_1 + c_{12}\dot{q}_2 + \frac{\partial V}{\partial q_1} = \tau_1$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{21}\dot{q}_1 + c_{22}\dot{q}_2 + \frac{\partial V}{\partial q_2} = \tau_2$$

$$\frac{\partial V}{\partial q_1} = (m_1 l_1 + m_2 l_1) g \cos q_1 + m_2 l_2 g \cos(q_1 + q_2)$$

$$\frac{\partial V}{\partial q_2} = m_2 l_2 g \cos(q_1 + q_2)$$

What to do when such

NPTEL
PRASANNA S GANESH,
ganesh@me.iitd.ac.in

So, one more case we need to discuss here before we close is the case of systems where you have the couple dynamics coming up here. A couple in the sense that it has a double derivatives couple. So, you have this $d_{11} \ddot{q}_1$ and $d_{12} \ddot{q}_2$ coming in the same equation. So, far we had systems where like only one double derivative was there in one equation. There are now these two different quantities double derivatives are coming. How do you handle such case? So, how do you basically now represent this system in the form that is required for the ode solver to have?

Like you remember that form, vector \dot{x} is equal to f of x and t . So, we want to kind of get the system in that form. Other quantities will not bother us because they are all coming as \dot{q} only. There is no any double dot terms anymore here. So, how do you handle such case, how do you do, what do you do, how do you transform. Think about that. And then you go to the next slide. Pause here and then go to the next slide.

(Refer Slide Time: 18:31)

Ex: 2R manipulator

Consider matrix form of equation and get required state derivatives:

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{11}\dot{q}_1 + c_{12}\dot{q}_2 + \frac{\partial V}{\partial q_1} = \tau_1$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{21}\dot{q}_1 + c_{22}\dot{q}_2 + \frac{\partial V}{\partial q_2} = \tau_2$$

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau$$

$$\ddot{q} = D(q)^{-1}[\tau - C(q, \dot{q})\dot{q} - g(q)]$$

This can be further coded
Now using matrix inverse
multiplications

PRASANNA S GANDHI,
gandhi@me.iitb.ac.in

NPTEL

Now, here what we see, what we do here is basically convert the system into matrix form. This is like a matrix vector form. This is a matrix here. This is a matrix here, with g is a vector of this ∂V by ∂q_1 and ∂V by ∂q_2 . So, that g forms a vector, τ forms a vector of τ_1 and τ_2 . So, what we do here is to get \ddot{q} as isolated like the vector here, \ddot{q} consists of \ddot{q}_1 and \ddot{q}_2 . So, this is D matrix that is multiplied here. So, we need to transfer these terms on the other side as we did for other cases also, but now instead of dividing we take an inverse of D matrix here.

$D(q)$, this is typically a function of variables of the system. Not dot variables like \dot{q} here, only q . So, this $D(q)$ inverse multiplied with your transfer terms here, these are now forming a vector here. So, that will give us \ddot{q} . And once we obtain \ddot{q} , our job is easy for us to kind of code this for the ode solver. So, you go through, like you can solve such examples also then very easily. With these we can solve n degrees of freedom robotic manipulator equations and simulate them in MATLAB for whatever purposes, that makes and control purposes that you want.

This is how like this leaves you pathway for like multiple degrees of freedom system simulation by using ode45 solver. There are like some more issues that may come up regarding similarities or some cases where your slopes are very high or your derivatives are becoming very high, those cases we need to handle separately to make sure that there are some kind of other solvers called step solvers that can be used. So, these are like, that is a kind of a future pathway for us to get to.

For now, you can like look at many different parameters that ode function gives you to play around with and change to look at setting, different settings of solving. So, you can play around

with that. That is all given in the MATLAB help. We will not get into that here. But that is a kind of a forward path to look for.

(Refer Slide Time: 21:05)



So, you can also see the demo ode45 in MATLAB and like see, play around with different things in the ode function to get a good grasp of like what you want to do. So, we have already seen the animation code. So, we will stop here for now and you enjoy coding with this background, many different systems and assignments and things like that. For your future research also this may be of quite help. So, thank you and bye for now. We will stop here.