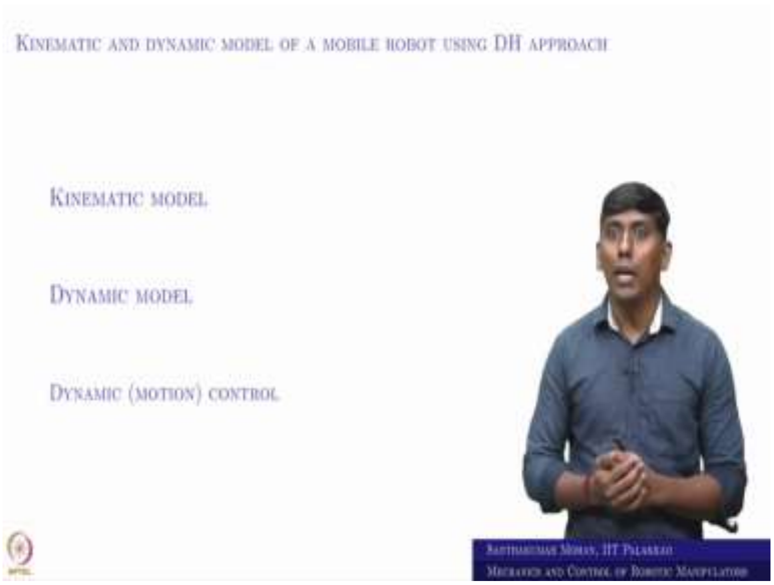
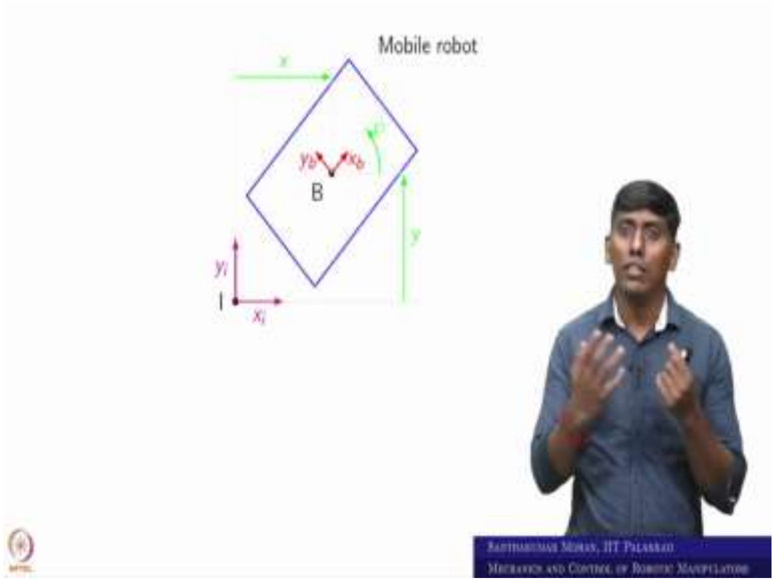
Mechanics and Control of Robotic Manipulators
Professor Santhakumar Mohan
Department of Mechanical Engineering
Indian Institute of Technology, Palakkad
Lecture 45
Kinematic and Dynamic Models of a Mobile Robot Using DH Approach

Hi, welcome back to Mechanics and Control of Robotic Manipulator. This particular lecture is like slightly different because we are taking the mobile robot and we are like trying to derive the equation using what we have learned so far. So, what we have learned the robotic manipulator is a; you call multibody approach then the Denavit Hartenberg approach we have used for even deriving the kinematic and dynamic level. The same philosophy we are trying to use it for the mobile robot that to like land based mobile robot how that would be work here.

(Refer Slide Time: 00:47)





So, in that sense what we are trying to intend here. So, the kinematic and dynamic model derivation of a land based mobile robot using Denavit Hartenberg approach. So, in that sense we are trying to see how to derive a kinematic model that to up to velocity level then we will come back to the dynamic model then we will end with the dynamic motion control.

So, if that is the case what we can see. So, the mobile robot I can show as a small you can say rectangular box because it is a land based so, it can be plotted as a small you can say planar object. So, now even we can like take it in a mechanics point of view it is a planar joint. So, planar joint means it is like two, translation and one rotation would be possible.

So, now if I assume that there is an inertia or you can say inertial frame which I call I, so, this inertial frame I, I consider as the reference can I like a refer the mobile robot. So, obviously yes, but in this case, there are n number of points within the mobile robot how we can like choose. So, for that we assume that there is a body frame which is associated with the body or the mobile robot, which is consist of xb yb and zb are the coordinates or you can say coordinate frames.

So, now can I like define it yes, we can do this simple you can say the spatial description manner where we can consider the rotation about you can say rotation of B with respect to I we can write as a rotation matrix of B with respect to I and the position vector of B with respect to I we can write and then we can come back with the transformation matrix and then so on so, we can do it, but if that is the case, so then you can see like the derivation of what we have come up with here is not possible because that is straight away everything is a rigid body basis, but we talk about what you call Denavit Hartenberg.

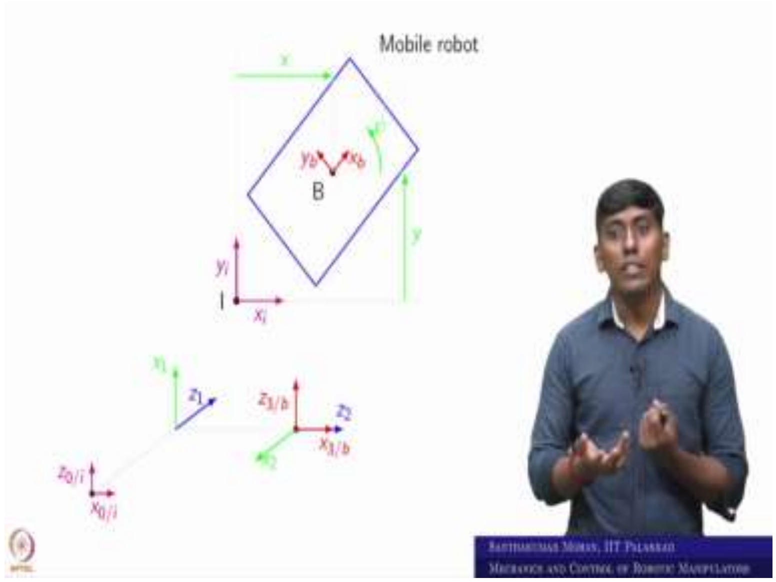
So, there are several constraints are coming. What are the constraints? So, you can see like Denavit Hartenberg say that every link should be a binary link in the sense only two joints are possible for one body, but in this case it is like three joints that to like one you take one body, so, ground to one then one to one like that it keep goes but here it is not like that.

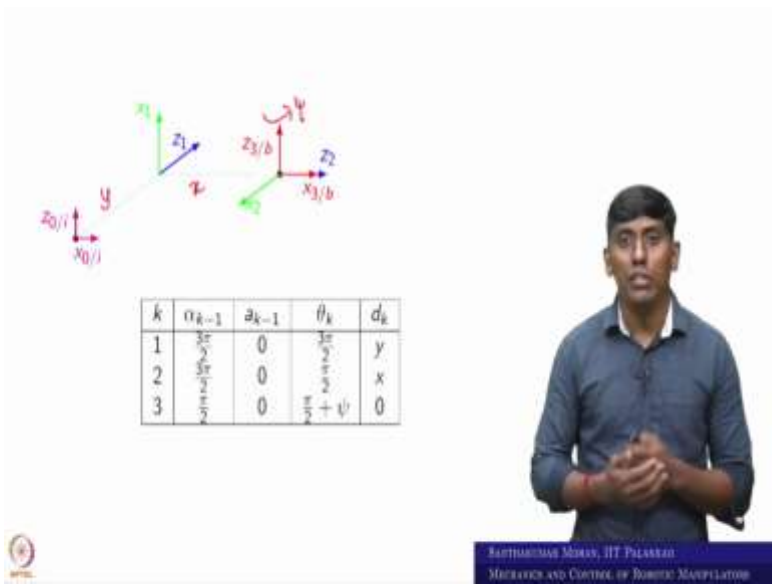
So, second consideration every joint supposed to be one DOF joint but it is a planar joint. So, obviously you have to do some kind of modification for the frame arrangement. So, we cannot take it straight forward like RB with respect to I and PB with respect to I that may not work in Denavit Hartenberg approach.

So, for that what we are trying to do we are trying to bring the coordinate what we need to know. So, here you can assume that inertial frame to body frame which is like we can write as a position vector which is nothing but x and y because it is in a plane in addition to that the zb you can see and zi are, you can say parallel, but the xb and xi are like non parallel that angle what we call heading angle or you can say yaw angle whatever you can call it.

So, which is like come as a psi so now if you talk about the coordinates or the variables in terms of Denavit Hartenberg so, in the sense here there are three joints. So, one is actual like x translation joint y translation joint is the second one and the z axis rotation is a third joint. So, now, we want to write it this in the Denavit Hartenberg approach or for that first we have to draw the frame arrangement.

(Refer Slide Time: 04:14)





So, how we can do the frame arrangement I can see that there are two translations, and one rotation is coming. So, I can assume that. So, in order to make it their life comfortable or life is easy, I assume that y axis translation as the first joint which is z1 and the x axis translation I take it as a z2 which is second and the third axis eventually it is end up with the body frame which is I am considering as the z3.

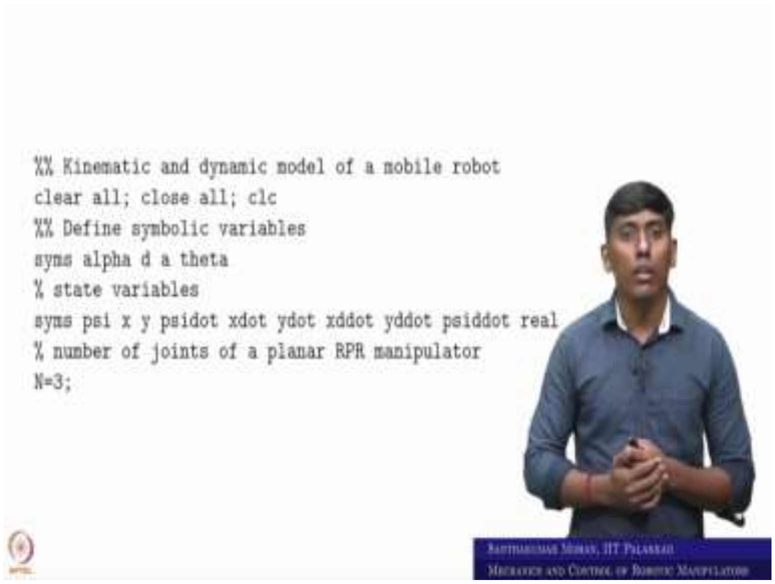
So, z3 is like a coincident with the B so these are the three joints which are active. So, now what I am trying to see from the inertial frame, which I assume that this is the inertial frame from the inertial frame, how this goes one to another. So, now I want to derive as a Denavit Hartenberg approach. So, for that what we need to know. So, you need to know like where the x axis coming so, the x axis I can like make it the first x axis which is x0 or xi can be written in that way so where it would be the perpendicular to the plane containing z0 and z1 this is what we learned. So, in that case what you can draw you can like draw this line.

So, which is like the plane containing you can draw. So, in that sense you can see this is the, you can say forward direction of x0 or xi. So, once you obtain the same way you can go for x1. So, the plane containing z1 and z2 the perpendicular to that so, it can come upward or downward I took it as upward direction as x1. So, similar way you can go z3 and z2 are a plane containing on the screen. So, now the plane piercing probably the x2 are coming out what can be x2 I have taken coming out as the x2, now x3 I can choose as per my own because there is no proceeding link. So, I can choose as simple where it is parallel to x naught.

So, in the sense x3 and xb are parallel. Now, coming to the screen where we can like see the same thing what we derived I put it here just to derive the Denavit Hartenberg parameter. So, here there are three variable y x and psi. So, now based on this what I can derive I can derive the Denavit Hartenberg parameter we can like see again alpha k minus 1 in this case alpha 0. So, alpha 0 is like rotation about x0 where z0 parallel to z1. So, you can see this it is like rotate 270 degrees that is why it is 3 pi by 2. Further you can see that the theta, theta k in the sense theta 1 here. So, you can see like x1 and x0 are like you can see minus 90 or 270 degree rotation about z1.

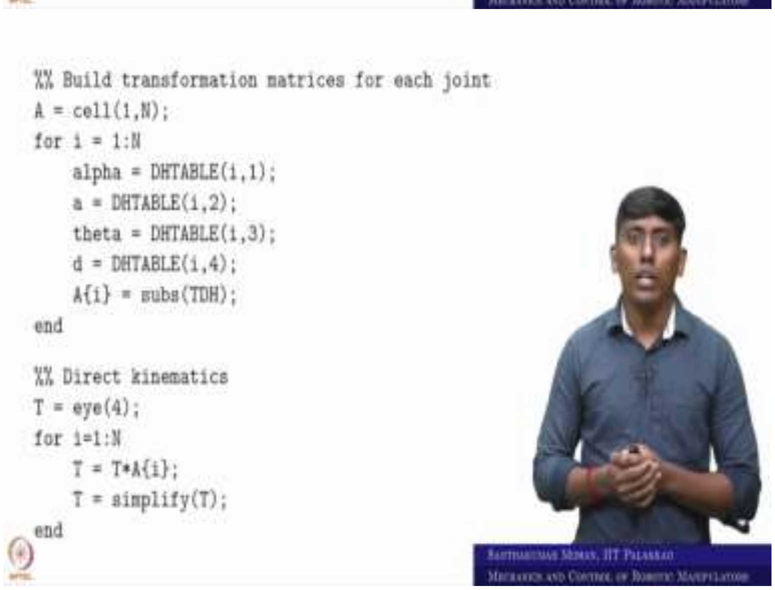
So, like that you can like derive all other you can say joints where 2 and 3 once you derived what you can do you can like go back you can say MATLAB or you can write it in the arm matrix and you can derive the individual transformation matrix and then you can like multiplied you can say post multiply and you will get the final kinematic model then you can go with the you can say angular velocity and the linear velocity then you can get the differential kinematics. So, for that what we are trying to do we are trying to do the MATLAB base which is we have seen in the very second week of the course. So, that is what we are trying to show here.

(Refer Slide Time: 07:28)



So, in the sense we are trying to derive the kinematic and dynamic model together. So, first we are writing the general DH parameter which is alpha d a theta. So, further based on this particular system, so, it would be having psi psi dot psi double dot then x x dot x double dot y y dot y double dot which we have written here. So, y double dot we have written as y ddot or you can say d dot so similarly, x double dot I have written as x ddot psi double dot like written psidd dot like that and here how many joints so there are three joints excluding ground are excluding 0 it is like three joints.

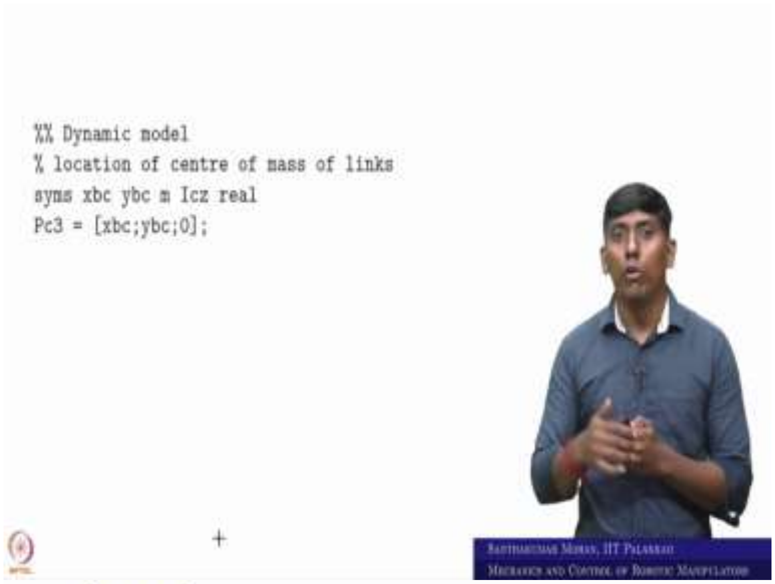
(Refer Slide Time: 08:10)

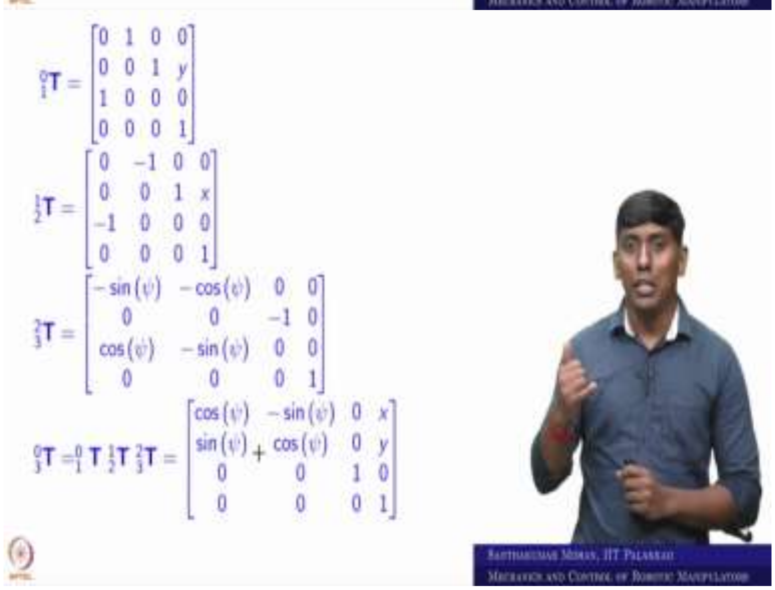


```
%% Transformation matrices
T01 = simplify(A{1})
T12 = simplify(A{2})
T23 = simplify(A{3})
%% Transformation matrix of the end frame
% with respect to base frame
TON = T % output TON matrix
%% Position vector
p = simplify(T(1:3,4)) % output ON position
%% Orientation vectors
n = T(1:3,1) % output xN axis (normal vector)
s = T(1:3,2) % output yN axis (sliding vector)
a = T(1:3,3) % output zN axis (approach vector)
%% end
```

```
%% Velocity kinematics
R01 = A{1}(1:3,1:3);
P01 = A{1}(1:3,4);
R12 = A{2}(1:3,1:3);
P12 = A{2}(1:3,4);
R23 = A{3}(1:3,1:3);
P23 = A{3}(1:3,4);
```

```
%% Angular velocity propagation
w0 = [0;0;0];
w1 = R01'*(w0) ;
w2 = R12'*(w1) ;
w3 = R23'*(w2) + [0;0;psidot] ;
% end-effector angular velocities w.r.t. base frame
w03 = R01*R12*R23* w3

%% Linear velocity propagation
v0 = [0;0;0];
v1 = R01'*(v0+cross(w0,P01)) + [0;0;ydot];
v2 = R12'*(v1+cross(w1,P12)) + [0;0;xdot] ;
v3 = R23'*(v2+cross(w2,P23)) ;
% end-effector linear velocities w.r.t. base frame
v03 = simplify(R01*R12*R23*v3)
```





So, based on that we can write the DHTABLE which we have obtained there so, we have incorporated that DHTABLE then we can like see what you call the general Denavit Hartenberg which is we call nonstandard one that is we have returned it here. So, then you can see like we are building the transformation matrix for each joint we are substituting individual DHTABLE row wise substituting into the individual transformation matrix then that would be you can say freezing it into a cell.

So, here the cell, which is A so, A is consist of three sub cells. So, that is what we call A of i within the braces. So, based on that what we can do we can do the direct kinematics, direct kinematics in the sense forward kinematics which is nothing but post multiply with the further matrices there is T01 first then T12, T23 like that you can keep extending. So, that is what we have done here based on the, for loop you can see like this is like doing it. So, once this is done what you can do is so, you can like first identifies the individual transformation matrix T01 then T12, T23 after that you are trying to find out what is T03. So, T03 here I have written as T01.

So, then you can see the position vector and the orientation vectors all coming into a picture. Once this all done what we can do we can go for velocity kinematics for that you need to know individual rotation and position vectors. So, that is what we have obtained here. So, once we obtained this then we can use the angular velocity propagation.

So, which is like straight forward here you can see the first two joints, or you can say linear joint then it is very simple. The third joint is like rotary that is also we have done. The same extension we can do it for the linear velocity propagation. The first two joints are linear joint you can see that y dot and x dot is added here and then we can go cross.

So, similar way we can find the end effector velocity and angular velocity and linear velocity. So, once these done we can go the same way for the angular acceleration propagation and what you call velocity propagation before that, I just want to plot what is the individual matrices and what is the final end effector convey position with respect to base which is fulfilling. You can see the position vector x and y and the orientation which is like psi if you take the tan inverse then you can see psi is the case which is the orientation angle about the z these all like fulfilled.

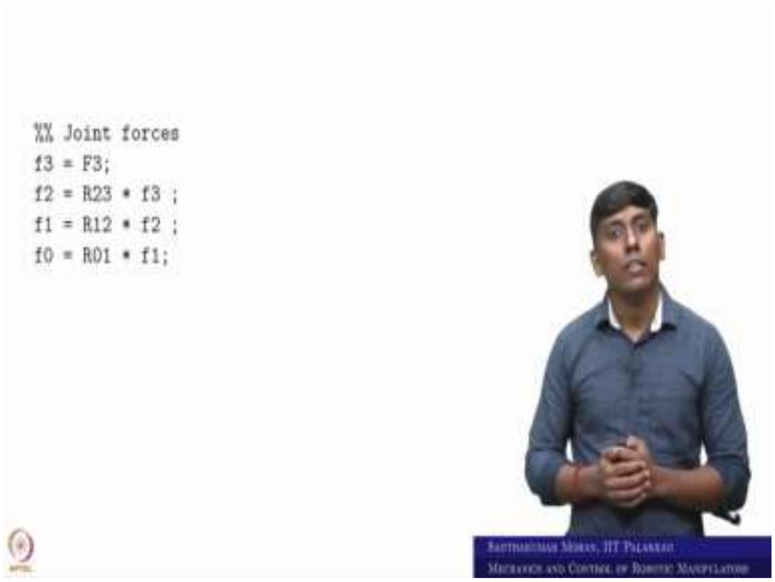
(Refer Slide Time: 10:41)

```
%% Dynamic model
% location of centre of mass of links
syms xbc ybc m Icz real
Pc3 = [xbc;ybc;0];

%% Angular acceleration vectors
al0 = [0;0;0];
al1 = R01'*(al0) ;
al2 = R12'*(al1) ;
al3 = simplify(R23'*(al2 + cross(w2,[0;0;psidot])) + [0;0;psiddot])
al03 = simplify(R01*R12*R23*al3)
```

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

```
%% Linear acceleration vectors
a0 = [0;0;0];
a1 = R01'*(a0+cross(al0,P01)+cross(w0,cross(w0,P01))) ...
+ [0;0;yddot] + cross(w1,[0;0;ydot]);
a2 = R12'*(a1+cross(al1,P12)+cross(w1,cross(w1,P12))) ...
+ [0;0;xddot] + cross(w2,[0;0;xdot]);
a3 = simplify(R23'*(a2+cross(al2,P23)+cross(w2,cross(w2,P23))))
a03 = simplify(R01*R12*R23*a3)

%% Linear acceleration of centre of mass of links
ac3 = a3 + cross(al3,Pc3) + cross(w3,cross(w3,Pc3));
%% Inertial forces of the links
F3 = m*ac3;
I3 = diag([0;0;Icz]);
N3 = I3*al3 + cross(w3,I3*w3);
```

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators



So, now what one can see we want to derive the dynamic model. For deriving dynamic model what you need to know, you need to know the angular acceleration, the linear acceleration and the centroidal linear acceleration. So, we will do the forward propagation for finding the centroidal linear acceleration you need to know the centroid and location.

So, here there is only one body exists. So, although we have taken three virtual links, but only one physical link is there, so in the sense the centroid, we consider only that third body where the B frame and the centroid is like away, we assume xbc and ybc is the; you can see x-axis and y-axis coordinate.

So, based on that, we can like derive the, what you call the linear velocity propagation in the forward. So, in this sense, first we calculate the angular velocity. So, the angular velocity you can see like you have you can see the cross product which is like coming as what you call simple gyroscopic effect and then you can see the angular acceleration of that particular joint. So, like that, you can do it. so, now coming back to the linear acceleration.

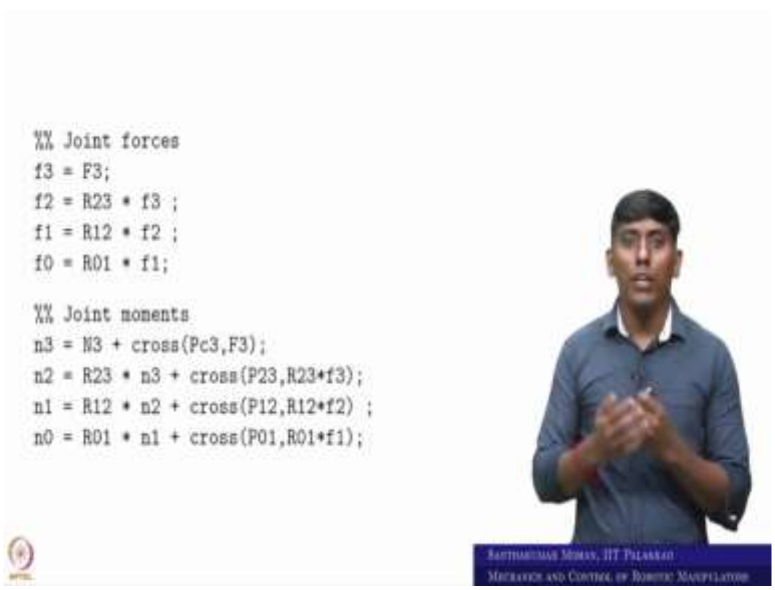
So, linear acceleration is like straight forward where you can like do it a1, a2, a3. So, once you obtain then you can like go back the centroidal linear acceleration here the third body, so that is what we can like to do it here. So, now, I hope once you calculate the centroidal linear acceleration. So, what one can like expect the inertial force, the inertial effects, you need to like come back.

So, the inertial effects what you have, so here you have inertial force and inertial moments. So, what are the inertial force and moments so, F3 which is like m3 multiply with the ac3, where m 3 here only one body, which is like you can see, which is just m and here we assume that it is in a plane. So, in that sense, it is having only centroidal moment of inertia that too z axis only second moment of inertia only there the product of inertia we assume to be 0.

So, then you can see the inertial moment we can calculate once you calculate then you go across the joint forces, the final joint does not have anything else because there is no end effector added in the sense F3 would be simple inertial force, whereas n3 would be small n3 would be capital N3 plus this what you call f3, which is acting on the centroid, that would be generate the couple and when we come backward.

So, you f2, f1 and f0 can be calculated in this way, because it comes from all the cases since there is no body physical body. So, that is why it is just a transpose, or you can see just the rotational operator added.

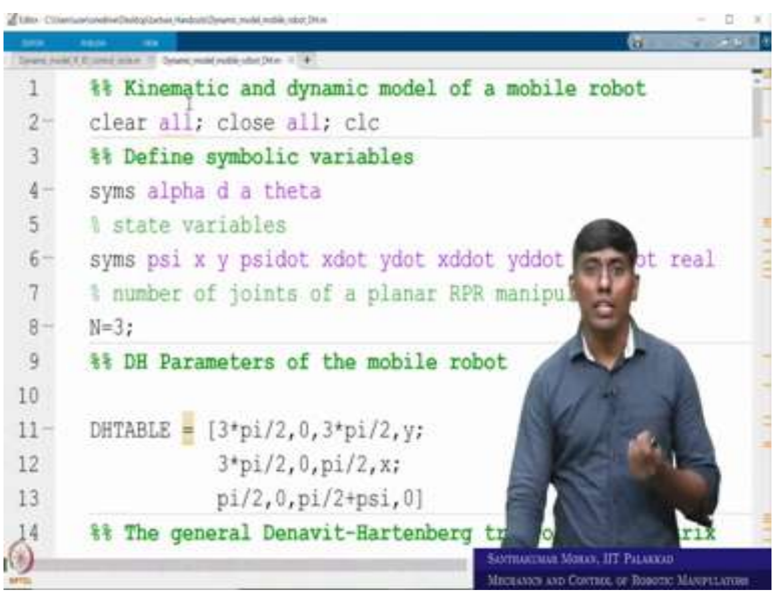
(Refer Slide Time: 13:27)

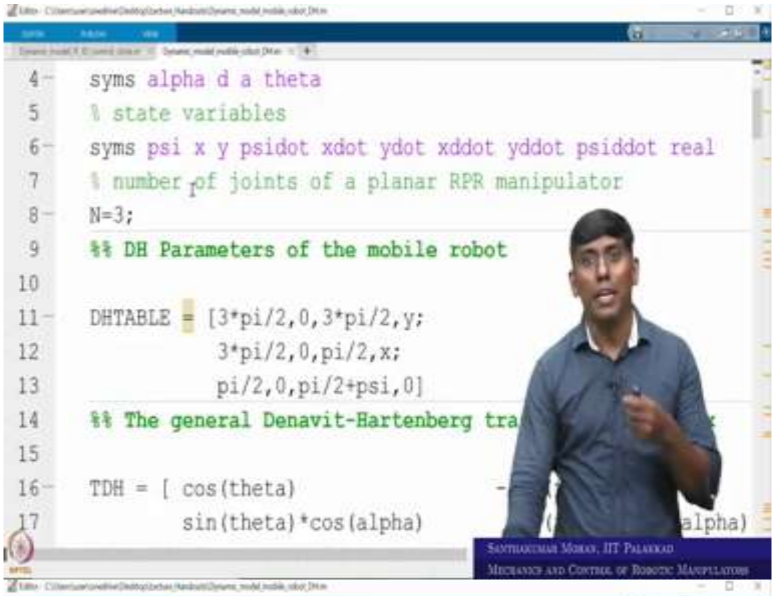


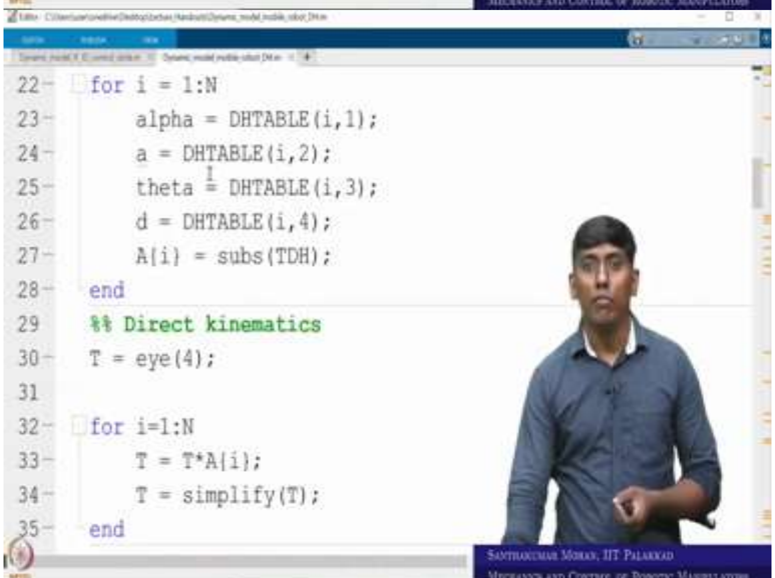
So, the similar way you can come to the joint moment. So, n3, I said small n3 is a capital n3 plus the moment are the couple generated by the inertial force, then you can come backward n2, n1 and n0, where f0 and n0 are the shaking force acting on the ground of the body. Similarly, shaking moment acting on the ground based on the body motion.

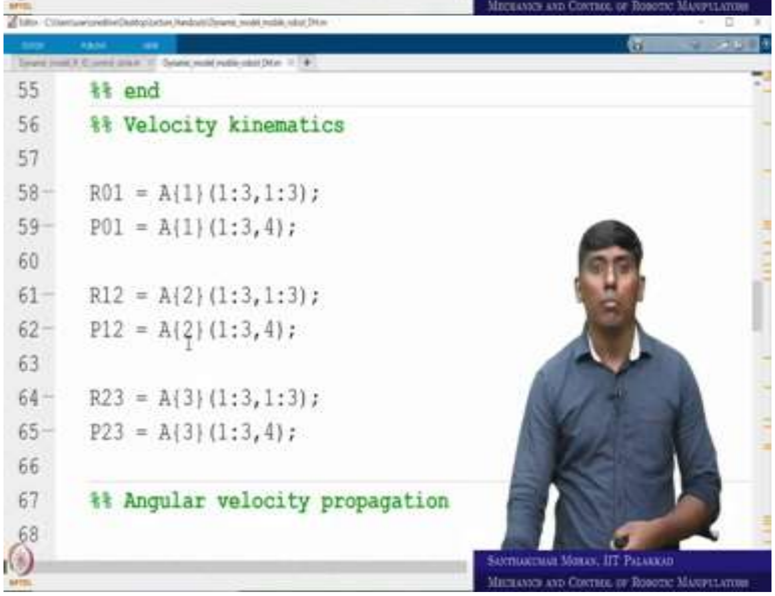
So, now we will go to the input vector. So, here we have to like to get the first two joints are like linear joint then the forces the third component of the force would be equal to tau1 and tau2 the tau3 is like the rotational. So, moment third component would be tau3 but what you can write Fx, Fy and Mz you are to write, but here the tau 1 is equal to Fy and tau 2 equal to Fx.

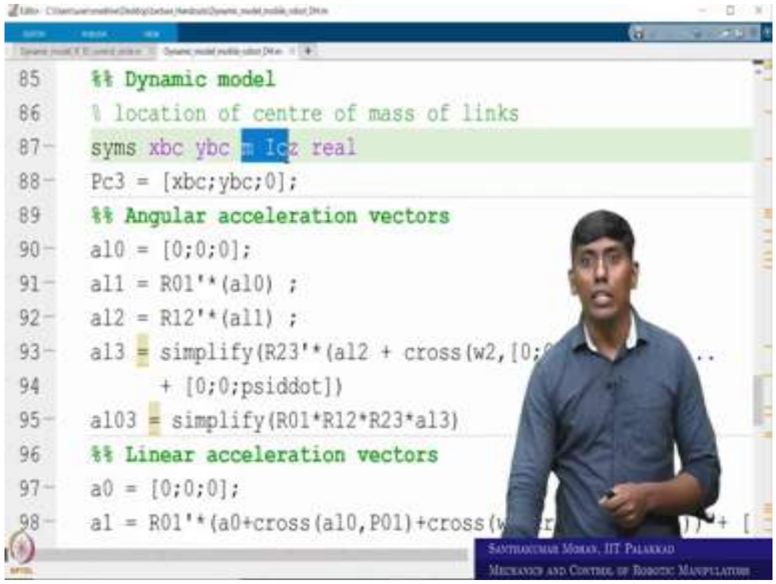
(Refer Slide Time: 14:19)

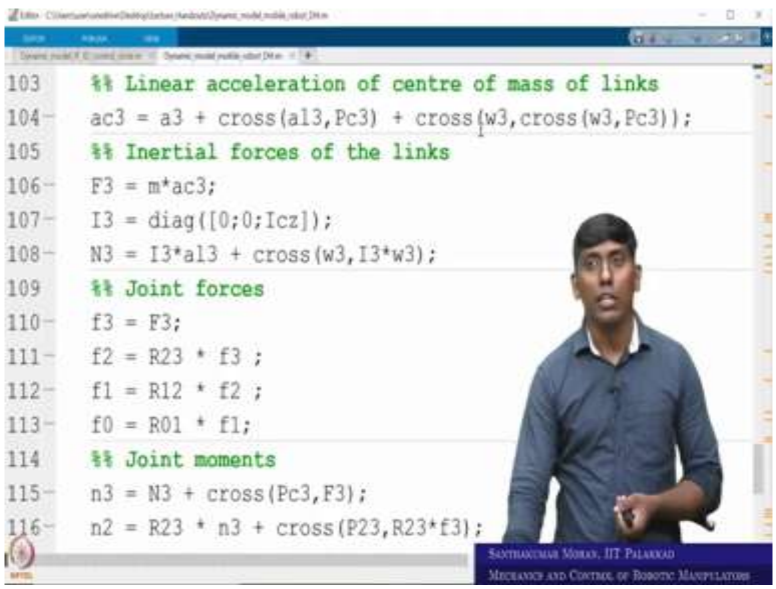



syms alpha d a theta
% state variables
syms psi x y psidot xdot ydot xddot yddot psiddot real
% number of joints of a planar RPR manipulator
N=3;
%% DH Parameters of the mobile robot
DHTABLE = [3*pi/2,0,3*pi/2,y;
3*pi/2,0,pi/2,x;
pi/2,0,pi/2+psi,0]
%% The general Denavit-Hartenberg tra
TDH = [cos(theta)
sin(theta)*cos(alpha)
alpha)
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


for i = 1:N
alpha = DHTABLE(i,1);
a = DHTABLE(i,2);
theta = DHTABLE(i,3);
d = DHTABLE(i,4);
A{i} = subs(TDH);
end
%% Direct kinematics
T = eye(4);
for i=1:N
T = T*A{i};
T = simplify(T);
end
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


%% end
%% Velocity kinematics
R01 = A{1}(1:3,1:3);
P01 = A{1}(1:3,4);
R12 = A{2}(1:3,1:3);
P12 = A{2}(1:3,4);
R23 = A{3}(1:3,1:3);
P23 = A{3}(1:3,4);
%% Angular velocity propagation
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators





So, we have to like modified before modifying you can see like the dynamic model we can write it in a state-space form in this way, where we can use the simple MATLAB command equations to matrix, where we can like take the coefficient of this x double dot y double dot, and psi double dot and you can see like I have interchanged already this is Fx this is Fy and this is Mz the same way we can like make it n vector where this multiply with this would be the coefficient and M that would be subtracted that would be the other effort.

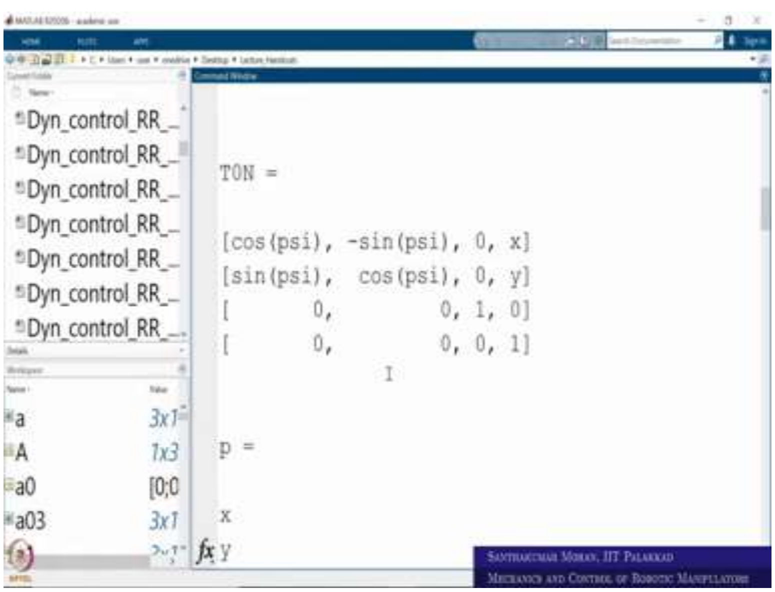
I hope now you are like clear so we can like go to the MATLAB and then you can like see. So, now we come back to the MATLAB window. So, you can like see this is what we have seen. So, initially just to make comfortable everything is like clear, then we are like defining the individual transformation matrix you can say variable alpha d a theta, then you can see the state variable depend on the system.

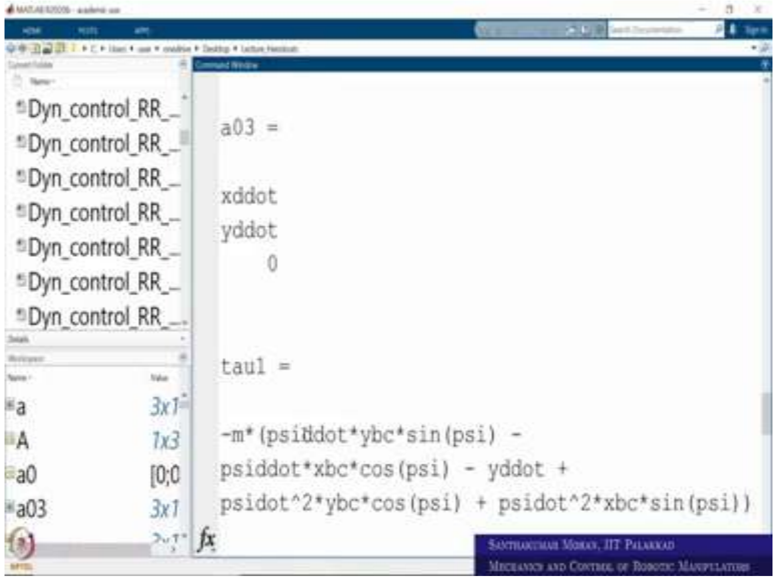
So, here you can say psi, x, y, psi dot, x dot all those things are coming and as per the DHTABLE we have written all the cases whatever we have obtained based on the frame arrangement, this is the DHTABLE then we can like get the non-standard then we can like come back with what you call the individual transformation matrix.

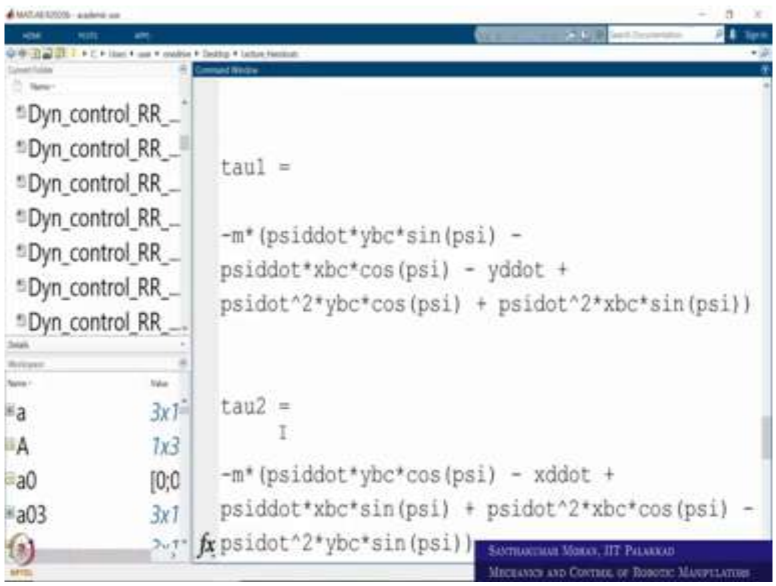
Then you can get the final direct kinematics and I want to show this then you can see like the final position vector and orientation vectors all I am trying to show then coming to the velocity kinematics for the individual rotation and position vector supposed to be known. Then the angular velocity propagation then the linear velocity propagation we have done then we are coming back to the centroidal linear acceleration for that we are coming back this.

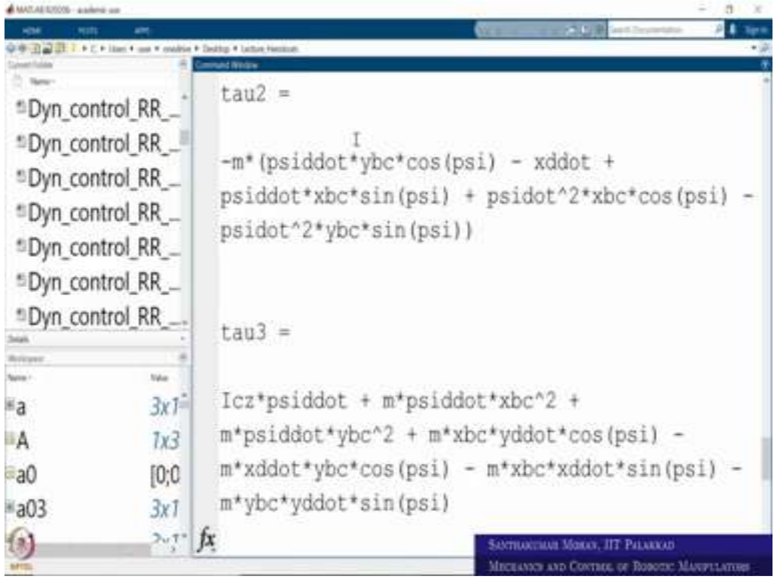
So, here we have added the, you can say inertial effects. So, then we are coming to the angular velocity propagation. So, then the linear velocity propagation then the linear acceleration of the center of mass then we are coming back to the inertial force and moments. So, here only you can see you F3 and N3 would be coming, then the joint forces we have calculated as such, then the joint moments, then we can like get the tau 1, tau 2, tau 3 then we can rewrite the M and n v like this.

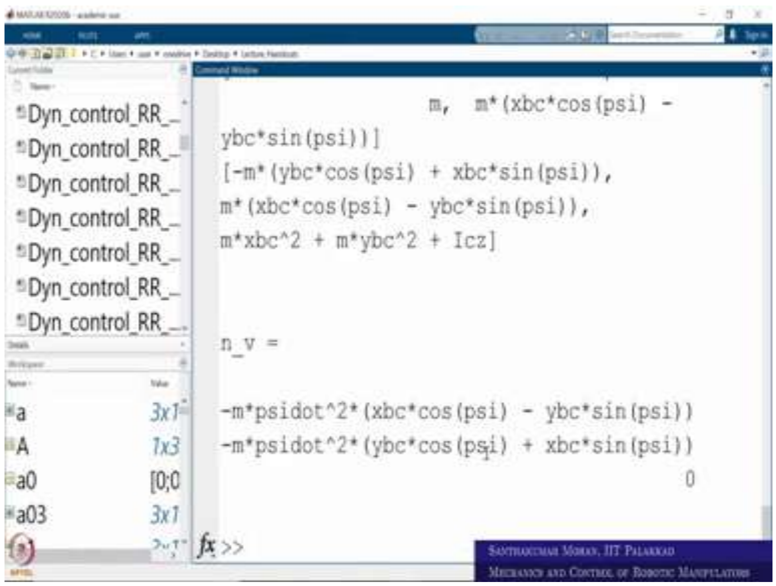
(Refer Slide Time: 16:57)




Command Window
Dyn_control_RR_
a03 =
xddot
yddot
0
tau1 =
-m*(psiddot*ybc*sin(psi) -
psiddot*xbc*cos(psi) - yddot +
psidot^2*ybc*cos(psi) + psidot^2*xbc*sin(psi))
a 3x1
A 1x3
a0 [0;0
a03 3x1
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


Command Window
Dyn_control_RR_
tau1 =
-m*(psiddot*ybc*sin(psi) -
psiddot*xbc*cos(psi) - yddot +
psidot^2*ybc*cos(psi) + psidot^2*xbc*sin(psi))
tau2 =
-m*(psiddot*ybc*cos(psi) - xddot +
psiddot*xbc*sin(psi) + psidot^2*xbc*cos(psi) -
psidot^2*ybc*sin(psi))
a 3x1
A 1x3
a0 [0;0
a03 3x1
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators



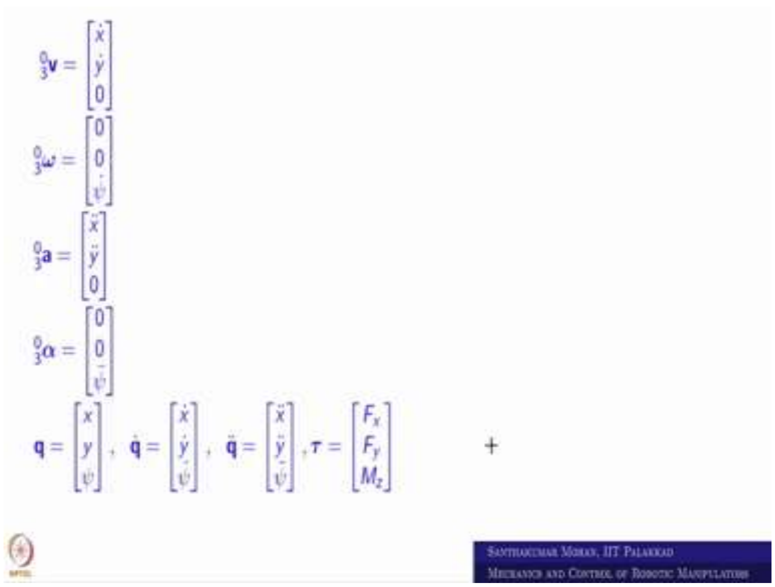


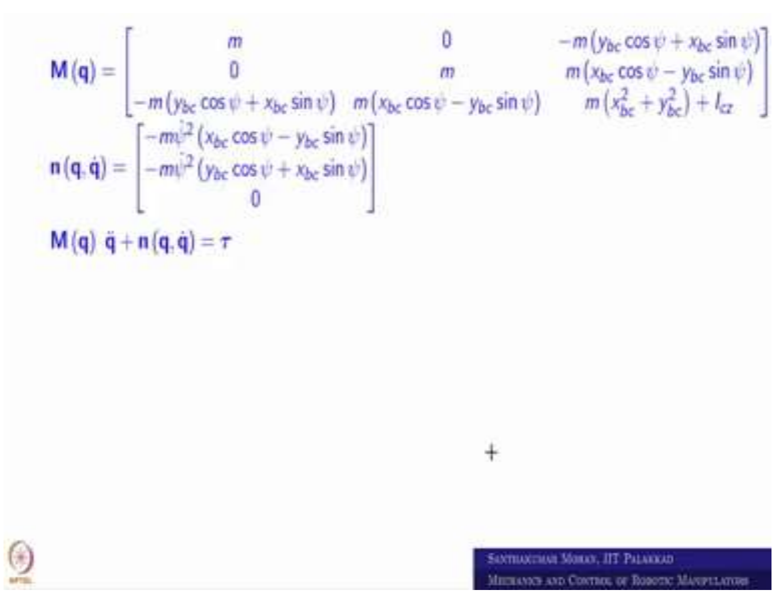
So, now if I run this, so what I can do, you can see like if I run this, so I will get a benefit. So, just to show that I am just skipping to the, what you call MATLAB window. You can see like these are the output which we are like obtained. So, you can see like this is the individual transformation matrix.

This is a DHTABLE and this is the individual transformation matrices and this is the final end effector matrix with respect to base frame and this is the position vector and this is the you can say normal sliding and you can see approach vector and this is the final or you call end effector angular velocity with respect to base and this is the end effector linear velocity with respect to base and this is the end effector angular acceleration with respect to base.

And this is the end effector linear acceleration with respect to base. So, like that, you can like get it then you can get the tau 1, tau 2 and tau 3 these are the three values which we needed, but this is equal to Fx, this is equal to Fy and this is Mz. So, then we derived the, you call inertia matrix and then we have like get the other effects or other vector which is like you; even you can do it in Newton Euler approach with the rigid body method the same equations, which will get the same thing we derived it here. So, now we will go back to the MATLAB window itself. So, where we are trying to show the dynamic model so for that, I will just show the slide then we can get understand.

(Refer Slide Time: 18:15)





$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} m & 0 & -m\left(y_{bc}\cos\psi + x_{bc}\sin\psi\right) \\ 0 & m & m\left(x_{bc}\cos\psi - y_{bc}\sin\psi\right) \\ -m\left(y_{bc}\cos\psi + x_{bc}\sin\psi\right) & m\left(x_{bc}\cos\psi - y_{bc}\sin\psi\right) & m\left(x_{bc}^2 + y_{bc}^2\right) + I_{cz} \end{bmatrix}$$

$$\mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} -m\dot{\psi}^2\left(x_{bc}\cos\psi - y_{bc}\sin\psi\right) \\ -m\dot{\psi}^2\left(y_{bc}\cos\psi + x_{bc}\sin\psi\right) \\ 0 \end{bmatrix}$$

$$\mathbf{M}(\mathbf{q})\ \ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) = \boldsymbol{\tau}$$

$$\begin{aligned} F_x &= m\ddot{x} - m\ddot{\psi}\left(y_{bc}\cos\psi + x_{bc}\sin\psi\right) \\ &\quad - m\dot{\psi}^2\left(x_{bc}\cos\psi - y_{bc}\sin\psi\right) \\ F_y &= m\ddot{y} + m\ddot{\psi}\left(x_{bc}\cos\psi - y_{bc}\sin\psi\right) \\ &\quad - m\dot{\psi}^2\left(y_{bc}\cos\psi + x_{bc}\sin\psi\right) \qquad (1) \\ M_z &= -m\ddot{x}\left(y_{bc}\cos\psi + x_{bc}\sin\psi\right) \\ &\quad + m\ddot{y}\left(x_{bc}\cos\psi - y_{bc}\sin\psi\right) \\ &\quad + \ddot{\psi}\left(m\left(x_{bc}^2 + y_{bc}^2\right) + I_{cz}\right) \end{aligned}$$

Santhakumar Mohan, IIT Palakkad

Mechanics and Control of Robotic Manipulators

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}\ \left[\boldsymbol{\tau} - \mathbf{n}(\mathbf{q},\dot{\mathbf{q}})\right]$$

+

Computed-torque control

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\left[\ddot{\mathbf{q}}_d + \mathbf{K}_p\tilde{\mathbf{q}} + \mathbf{K}_d\dot{\tilde{\mathbf{q}}}\right] + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}})$$

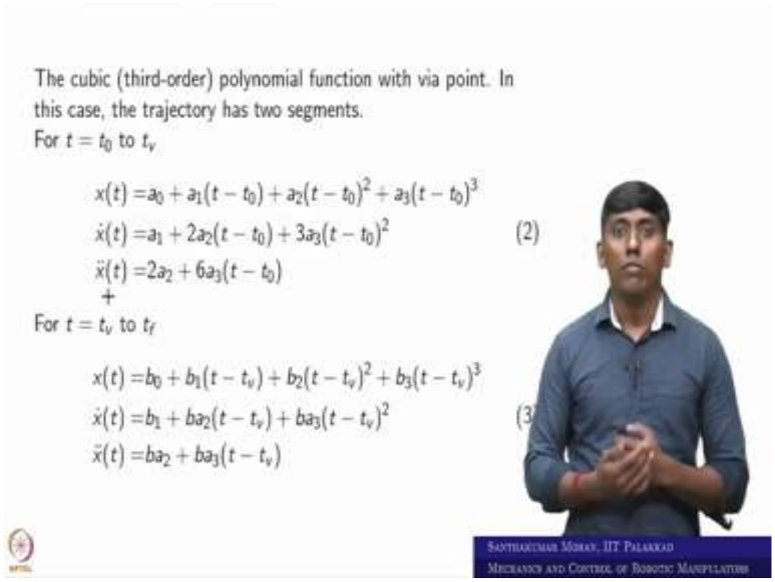
$$\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q},\quad \dot{\tilde{\mathbf{q}}} = \dot{\mathbf{q}}_d - \dot{\mathbf{q}}$$

PD control

$$\boldsymbol{\tau} = \mathbf{K}_p\tilde{\mathbf{q}} + \mathbf{K}_d\dot{\tilde{\mathbf{q}}}$$

Santhakumar Mohan, IIT Palakkad

Mechanics and Control of Robotic Manipulators



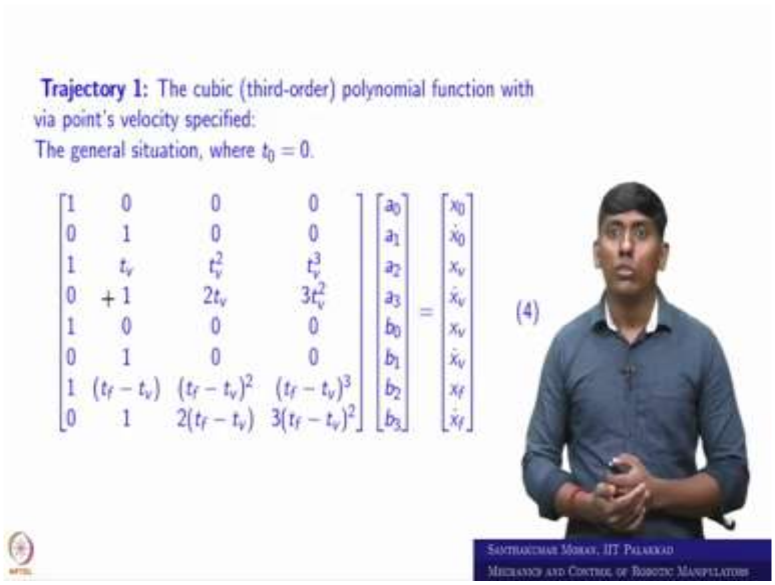
So, these are the outcome, so, these outcomes like can be written as the M of q and n of q comma q dot and this is a way, we can write the final dynamic model. So, once you derive the dynamic model what we wanted, we wanted the control aspect. So, before that, I just want to show what is Fx what is Fy and what is Mz.

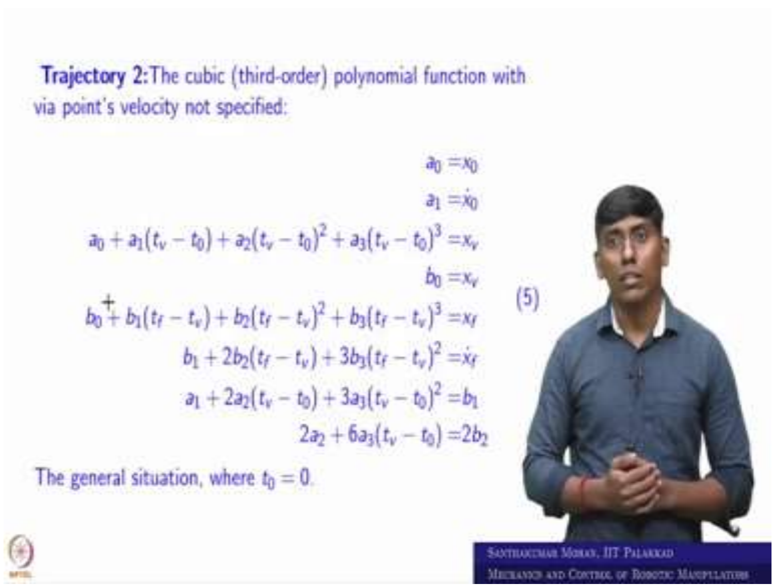
So, these are the values which you can like see it which we have derived in the MATLAB even you can do it in by hand, but MATLAB is easy that is what we have seen in the regular course. So, based on that, we can go for the control aspect. Even before that you want to make a dynamic model then this is a dynamic model relation where q double dot double integrate you will get q, but now we are interested in the control aspect. So, in this sense, if I see the tau based on the desired and the model parameter, this is the case.

So, now we can try even two different controls just for demonstration you can see the computed torque control we can use which is very popular in our course. Similarly, the PD control is other popular. So, here there is no gravity coming that is why it is a simple PD control because it is a land based mobile robot. So, based on that what we wanted, we wanted to write a controller code.

So, we can like see for that there are several segments, I just want to show one simple segment where we are like having a via point. So, if it is a via point what you can see there is two segments will come. So, individual segment we can do it in that case, even in the motion planning or trajectory generation we have seen there are two sub cases, one is with the via point velocity specified then it is a two independent segments.

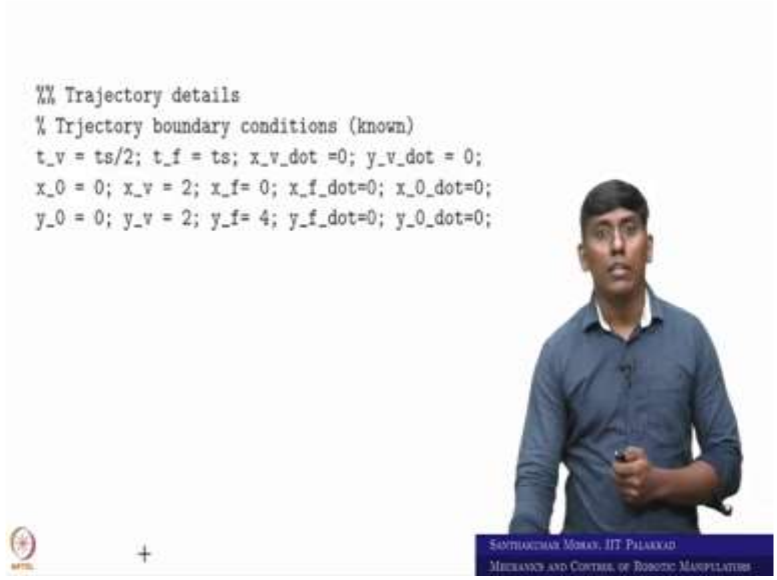
(Refer Slide Time: 19:57)

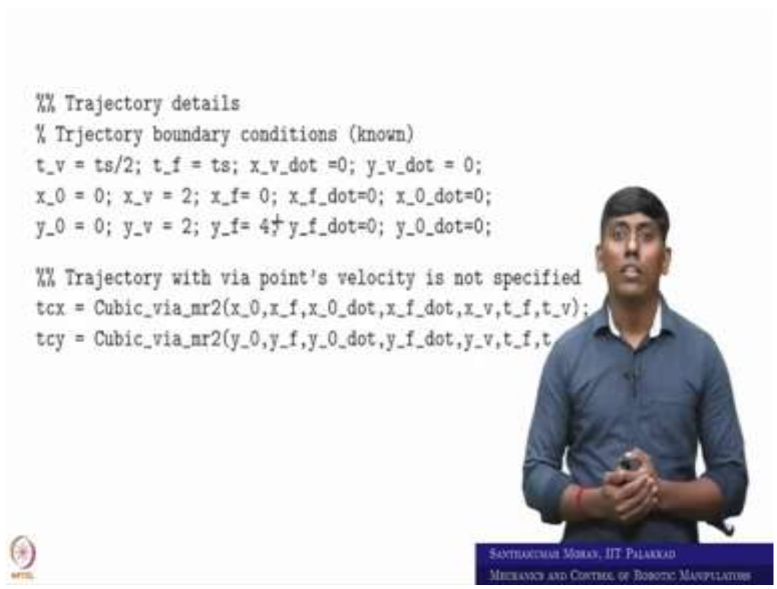




```
function tc = Cubic_via_mr2(x_0,x_f,x_0_dot,x_f_dot,xv,tf,tv)
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0 , 0 , 0 , 0 , 0;
    0 , 1 , 0 , 0 , 0 , 0 , 0 , 0;
    1 , t_v , t_v^2 , t_v^3 , 0 , 0 , 0 , 0 ;
    0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 ;
    0 , 0 , 0 , 0 , 1 , (t_f-t_v) , (t_f-t_v)^2 , (t_f-t_v)
    0 , 0 , 0 , 0 , 0 , 1 , 2*(t_f-t_v) , 3*(t_f-t_v)^2
    0 , 1 , 2*t_v , 3*t_v^2 , 0 , -1 , 0 , 0 ;
    0 , 0 , 2 , 6*t_v , 0 , 0 , -2 , 0];
%% known inputs
b = [x_0; x_0_dot ; x_v ; x_v ; x_f ; x_f_dot ;0 ;0
%% Trajectory coeeficients
tc = inv(A)*b;
end
```


```
%% Dynamic model of a generalized mobile robot
% (land-based)in inertial frame
clear all; clc; close all;
%% Simulation parameters (Euler's method)
dt = 0.1; % step size
ts = 60; % total simulation time
t = 0:dt:ts; % span 0,0.1,0.2,...,9.9,10.
%% Physical parameters
m = 30;
Icz = 0.1;
xbc = 0; ybc = 0;
%% Initial conditions
q0 = [0;0;0]; % initial conditions
qdot0 = [0;0;0];
q(:,1) = q0;
q_dot(:,1) = qdot0;
```


%% Trajectory details
% Trjectory boundary conditions (known)
t_v = ts/2; t_f = ts; x_v_dot =0; y_v_dot = 0;
x_0 = 0; x_v = 2; x_f= 0; x_f_dot=0; x_0_dot=0;
y_0 = 0; y_v = 2; y_f= 4; y_f_dot=0; y_0_dot=0;
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


%% Trajectory details
% Trjectory boundary conditions (known)
t_v = ts/2; t_f = ts; x_v_dot =0; y_v_dot = 0;
x_0 = 0; x_v = 2; x_f= 0; x_f_dot=0; x_0_dot=0;
y_0 = 0; y_v = 2; y_f= 4; y_f_dot=0; y_0_dot=0;
%% Trajectory with via point's velocity is not specified
tcx = Cubic_via_mr2(x_0,x_f,x_0_dot,x_f_dot,x_v,t_f,t_v);
tcy = Cubic_via_mr2(y_0,y_f,y_0_dot,y_f_dot,y_v,t_f,t
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

```
%% Trajectory details
% Trjectory boundary conditions (known)
t_v = ts/2; t_f = ts; x_v_dot =0; y_v_dot = 0;
x_0 = 0; x_v = 2; x_f= 0; x_f_dot=0; x_0_dot=0;
y_0 = 0; y_v = 2; y_f= 4; y_f_dot=0; y_0_dot=0;

%% Trajectory with via point's velocity is not specified
tcx = Cubic_via_mr2(x_0,x_f,x_0_dot,x_f_dot,x_v,t_f,t_v);
tcy = Cubic_via_mr2(y_0,y_f,y_0_dot,y_f_dot,y_v,t_f,t_v);

%% Trajectory with via point's velocity is specified
tcx = Cubic_via_mr1(x_0,x_f,x_0_dot,x_f_dot,x_v,x_v_do
tcy = Cubic_via_mr1(y_0,y_f,y_0_dot,y_f_dot,y_v,y_v_d
```


```
%% Trajectory details
% Trjectory boundary conditions (known)
t_v = ts/2; t_f = ts; x_v_dot =0; y_v_dot = 0;
x_0 = 0; x_v = 2; x_f= 0; x_f_dot=0; x_0_dot=0;
y_0 = 0; y_v = 2; y_f= 4; y_f_dot=0; y_0_dot=0;

%% Trajectory with via point's velocity is not specified
tcx = Cubic_via_mr2(x_0,x_f,x_0_dot,x_f_dot,x_v,t_f,t_v);
tcy = Cubic_via_mr2(y_0,y_f,y_0_dot,y_f_dot,y_v,t_f,t_v);

%% Trajectory with via point's velocity is specified
tcx = Cubic_via_mr1(x_0,x_f,x_0_dot,x_f_dot,x_v,x_v_dot,t_f,t_v);
tcy = Cubic_via_mr1(y_0,y_f,y_0_dot,y_f_dot,y_v,y_v_dot,t_f,t_v);

%% Circular trajectory details
rx = 3; ry = 3; wx = 0.2; wy = 0.1;
```

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, that is what we can like see where the via are point velocity specified? So, this is the equation which we have derived that can be portrayed in MATLAB. So, this is the way we can portrait. Similarly, the trajectory two which is like you can write as a third order polynomial, but you can see via point velocity is not specified. So, then you can see this will you eight cross eight and eight unknowns and then you can do it the same way we can like write it in a MATLAB code.

So, these all like we have done then what we can do even some cases we want to do the; you can say simulation we can like make it this. So, for that you can see the same Euler method we have used the vehicle mass is 30 kilograms and all other cases we have given, and you can see these are the trajectory generation for that we have considered the you can see trajectory boundary conditions.

And then this is the first you can say via point case where there is no velocity specified then this is like with velocity specified even further you want to go with a circular profile can we like bring it the circular trajectory details. So, once these all obtained, we can like do the numerical integration. So, you can see like for that so, we are like taking two segments. So, t of i which goes up to t 0 to t v that is the first segment and then t v to t f that is the second segment.

(Refer Slide Time: 21:19)

```
%% Desired values based on a circular trajectory
    x(i) = rx*sin(wx*t(i));
    xdot(i) = rx*wx*cos(wx*t(i));
    xddot(i) = -rx*wx^2*sin(wx*t(i));
    y(i) = ry-ry*cos(wy*t(i));
    ydot(i) = ry*wy*sin(wy*t(i));
    yddot(i) = ry*wy^2*cos(wy*t(i));
```



```
psi_desired = wrapTo2Pi(atan2(ydot(i),xdot(i)));
q_desired(:,i) = [x(i);
                  y(i);
                  psi_desired];
if xdot(i)==0 && ydot(i)~=0
  psi_dot_desired =0;
else
  psi_dot_desired = (yddot(i)/xdot(i)...
  -(ydot(i)*xddot(i))/xdot(i)^2)/(ydot(i)^2/xdot(i)^2
end
q_desired_dot(:,i) = [xdot(i);
                      ydot(i);
                      psi_dot_desired];
q_desired_double_dot(:,i) = [xddot(i);
                             yddot(i);
                             0];
```

```
%% system dynamic terms
 q(:,1) = q_desired(:,1); q_dot(:,1) = q_desired_dot(:,1);
 psi = q(3,i); psidot = q_dot(3,i);
 J_q = [cos(psi),-sin(psi),0;
        sin(psi),+cos(psi),0;
        0,0,1;];  % Jacobain matrix
 n_v_mu = [-m*psidot^2*(xbc*cos(psi) - ybc*sin(psi));
           -m*psidot^2*(ybc*cos(psi) + xbc*sin(psi));
            0;]; %% Other effects
 %% Inertia matrix
 D_mu = [m,0, -m*(ybc*cos(psi) + xbc*sin(psi));
         0,m,  m*(xbc*cos(psi) - ybc*sin(psi));
         -m*(ybc*cos(psi) + xbc*sin(psi)),...
          m*(xbc*cos(psi) - ybc*sin(psi)),...
          Icz + m*(xbc^2 + ybc^2)];
```

Santhakumar Mohan, IIT Palakkad

Mechanics and Control of Robotic Manipulators

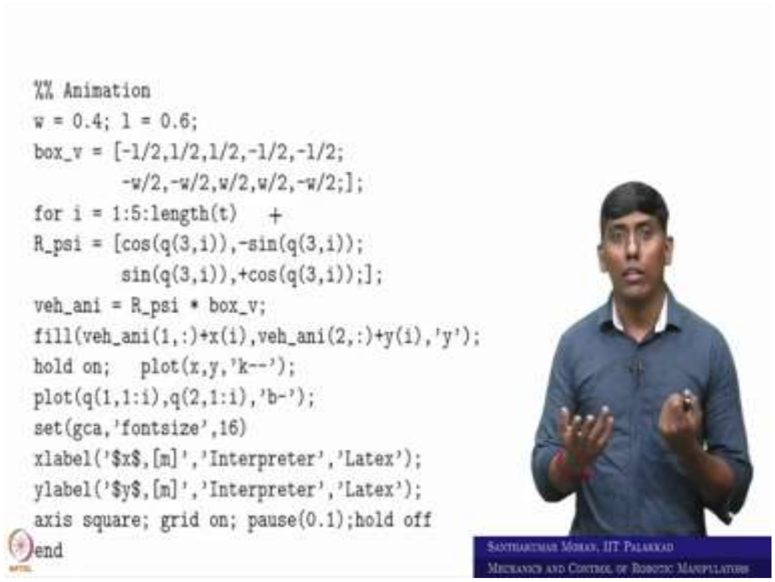
```
%% Errors
    q_tilda(:,i) = q_desired(:,i) - q(:,i);
    q_tilda_dot(:,i) = q_desired_dot(:,i) - q_dot(:,i);

%% Input vector based on Computed-torque control
tau_mu(:,i) = D_mu * (q_desired_double_dot(:,i) ...
    +4*q_tilda_dot(:,i)+4*q_tilda(:,i))+n_v_mu;
```

Santhakumar Mohan, IIT Palakkad

Mechanics and Control of Robotic Manipulators

```
%% Errors
    q_tilda(:,i) = q_desired(:,i) - q(:,i);
    q_tilda_dot(:,i) = q_desired_dot(:,i) - q_dot(:,i);

%% Input vector based on Computed-torque control
tau_mu(:,i) = D_mu * (q_desired_double_dot(:,i) ...
    +4*q_tilda_dot(:,i)+4*q_tilda(:,i))+n_v_mu;

%% Input vector based on PD control
    tau_mu(:,i) = diag([120,120,0.4])*q_tilda_dot(:,i)
                  +diag([120,120,0.4])*q_tilda(:,i);
```


```
    %% Accelerations
    q_double_dot(:,i) = inv(D_mu)*(tau_mu(:,i)-n_v_mu);
    % Velocities (time update 1)
    q_dot(:,i+1) = q_dot(:,i) ...
        + dt*(q_double_dot(:,i));
    % Positions (time update 2)
    q(:,i+1) = q(:,i) + dt*(q_dot(:,i))...
        +1/2*dt^2*(q_double_dot(:,i));
end % numerical integration ends here
```

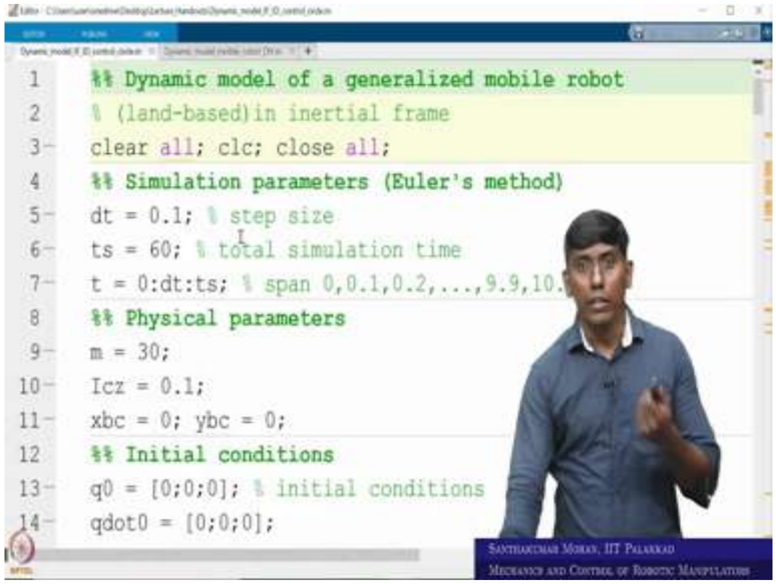


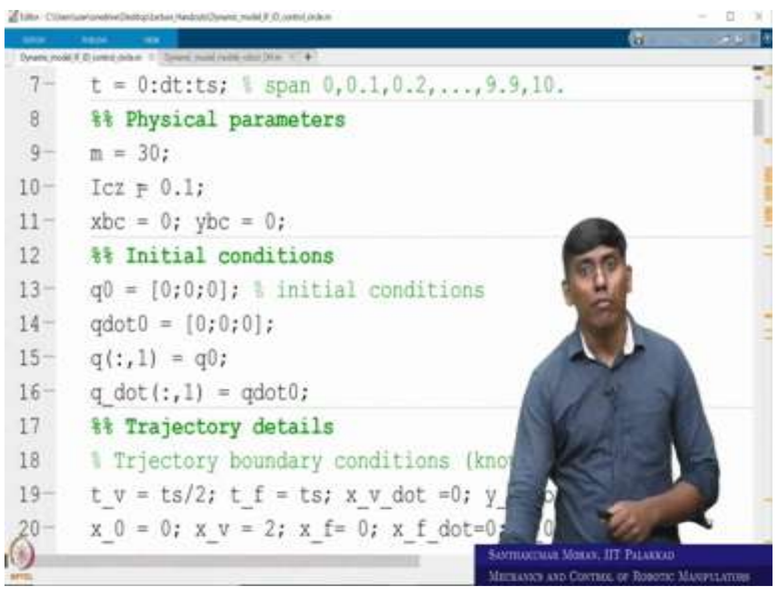
So, this is what we have done then we come back to the desired values based on the circular trajectory if I want to do it, so, this is a simple circular, where rx is the radius if rx and ry are different than it will give an ellipse. And similarly, omega x and omega y are different then that would be giving different profiles we can see in MATLAB.

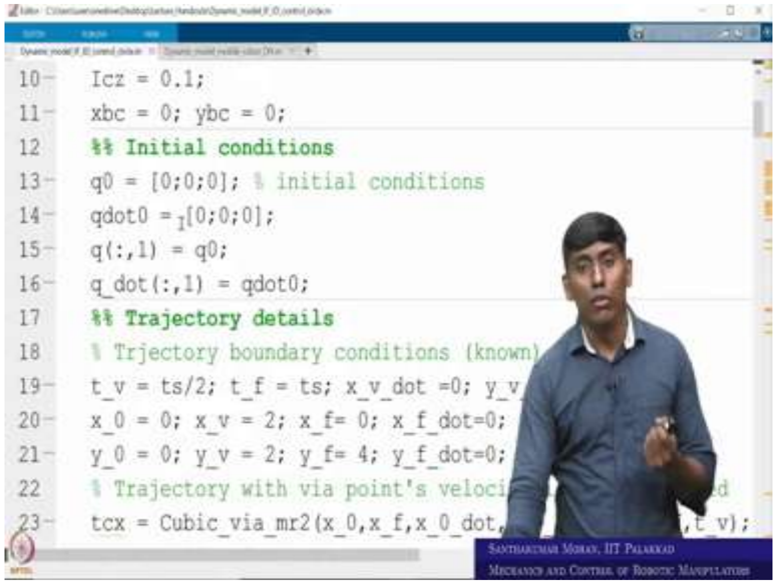
So, now we can like see this is we assume the circular trajectory. So, now, based on that we can like see that we can go for the circular trajectory cases also then we can come to the system dynamics. System dynamics required Jacobian matrix just for further end. So, then you can see like so, n v underscore mu and D mu we have got it. Here like it not supposed to be called D mu and n mu because here mu and q are same.

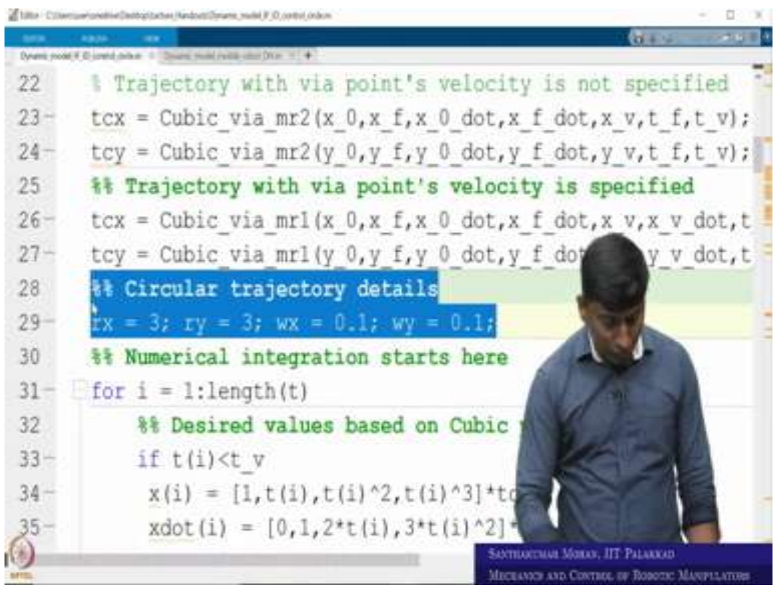
So, that way we can like get it then we calculate the error just for even you can say position and the velocity error then we can like go for the tau. So, then you can see like it is a simple computed torque control or we can go for the velocity control you can see the same thing. So, based on that we can like go for the system update, then the acceleration calculated then time velocity time update for the velocity and the position update which is the second velocity update then we can like plot as animated view where the box is moving because we have considered the rectangular box as the vehicle. So, that is what the case.

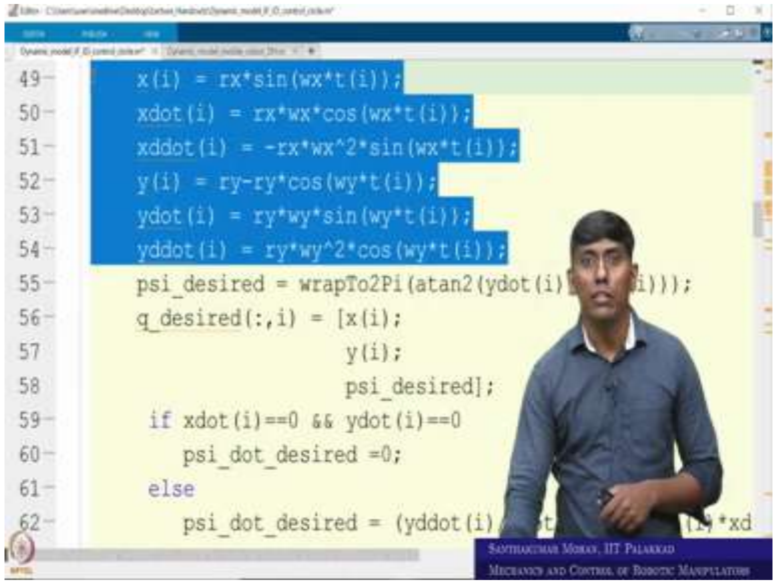
(Refer Slide Time: 22:49)

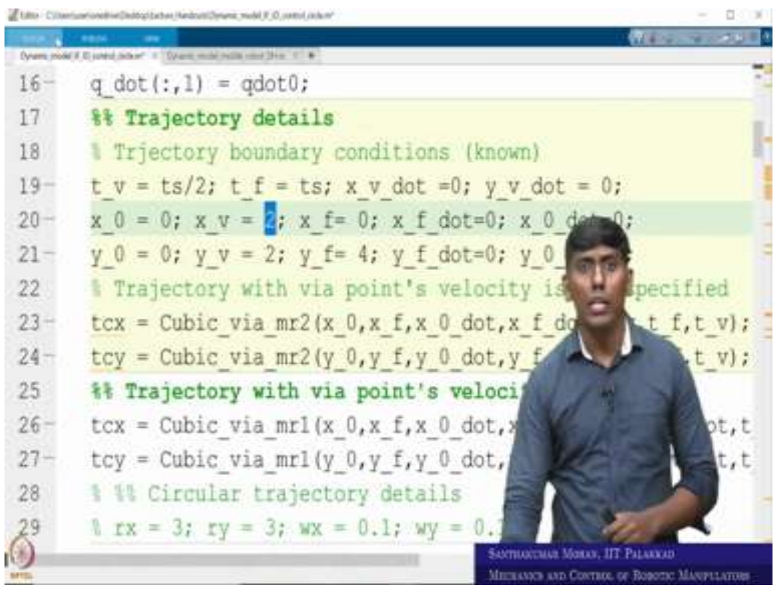





Icz = 0.1;
xbc = 0; ybc = 0;
%% Initial conditions
q0 = [0;0;0]; % initial conditions
qdot0 = [0;0;0];
q(:,1) = q0;
q_dot(:,1) = qdot0;
%% Trajectory details
% Trjectory boundary conditions (known)
t_v = ts/2; t_f = ts; x_v_dot =0; y_v
x_0 = 0; x_v = 2; x_f= 0; x_f_dot=0;
y_0 = 0; y_v = 2; y_f= 4; y_f_dot=0;
% Trajectory with via point's veloci
tcx = Cubic_via_mr2(x_0,x_f,x_0_dot,
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


% Trajectory with via point's velocity is not specified
tcx = Cubic_via_mr2(x_0,x_f,x_0_dot,x_f_dot,x_v,t_f,t_v);
tcy = Cubic_via_mr2(y_0,y_f,y_0_dot,y_f_dot,y_v,t_f,t_v);
%% Trajectory with via point's velocity is specified
tcx = Cubic_via_mr1(x_0,x_f,x_0_dot,x_f_dot,x_v,x_v_dot,t
tcy = Cubic_via_mr1(y_0,y_f,y_0_dot,y_f_dot
%% Circular trajectory details
rx = 3; ry = 3; wx = 0.1; wy = 0.1;
%% Numerical integration starts here
for i = 1:length(t)
%% Desired values based on Cubic
if t(i)<t_v
x(i) = [1,t(i),t(i)^2,t(i)^3]*t
xdot(i) = [0,1,2*t(i),3*t(i)^2]*
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators



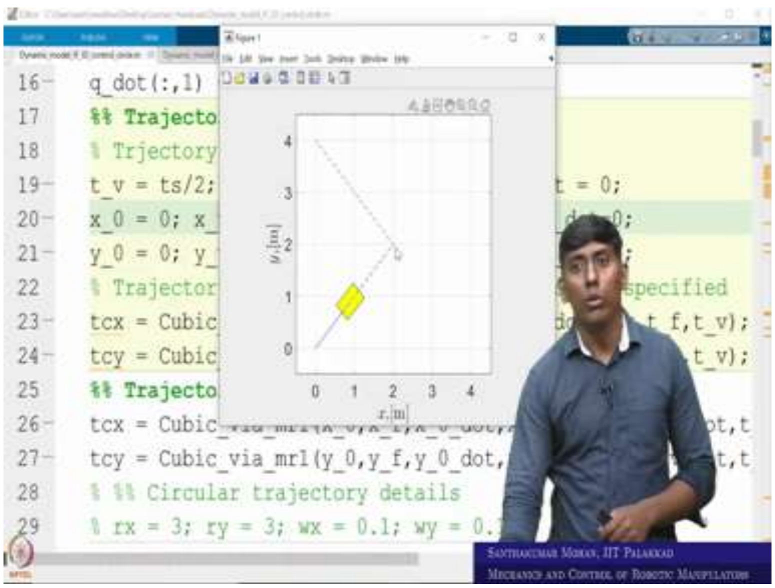


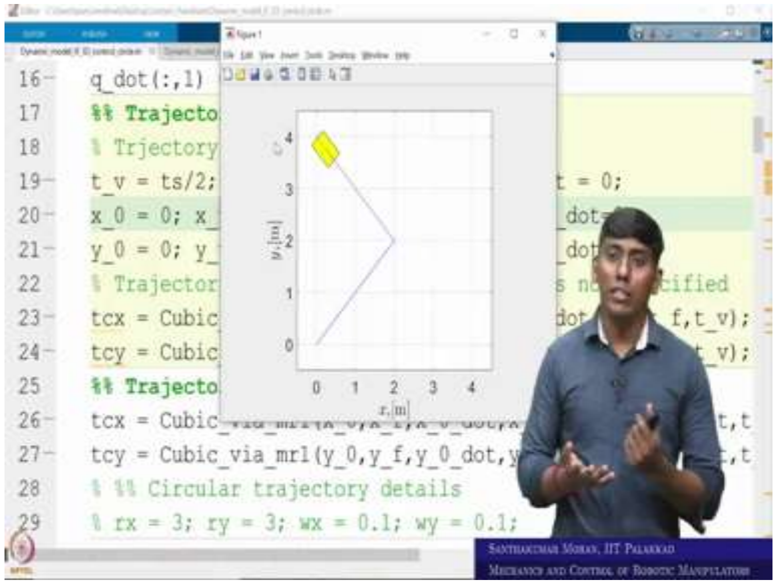
So, now we can like go to the MATLAB window where I already open it here for your benefit. So, you can see like this is the dynamic model code. In fact, here I already made the controller code. So, you can see like this is the mass value this is the inertia value need not to be 0.1 even you can substitute different value, but I have taken a rough and I assume that the centroid and the body center are like same that is why xbc and ybc are 0.

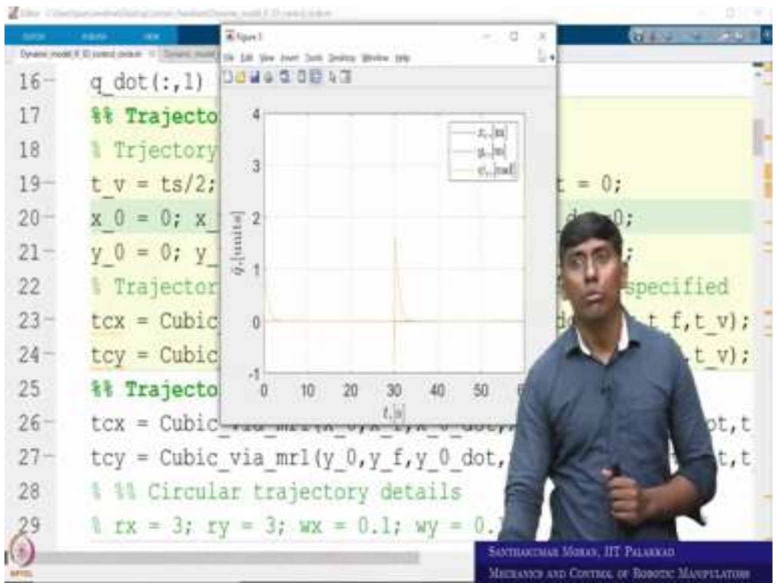
And then initial condition I assume that the vehicle starting from 0 with the 0 orientation, then we can like see that the trajectory coefficients are given. And we can like generate so right now I am like trying to see the first case which I am like trying to see that the; you can say trajectory with via points velocity is specified. So, in that case, so I am like making this circular profile, or actual like subdue, so I am just commenting this.

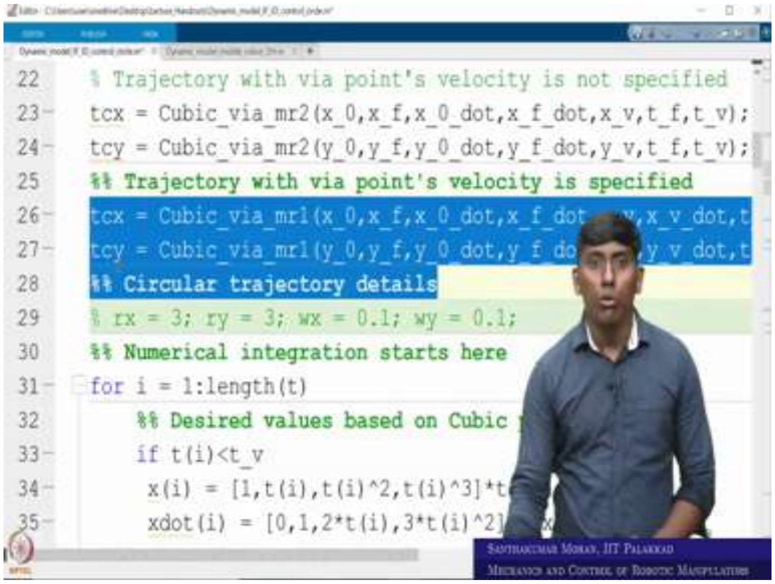
So, then we can like see that this is like trying to follow what you call this via point. So, you can like see what via point I have given here it is really tricky. So, you can see like x displacement at via is like 2 and y also 2, but the second point which is the final point x axis there is no displacement and y is like 4, in the sense it is like v shape in the sense you can see it is like greater than or equal or greater symbol. So, that is what we are trying to see. I will just try to run it first.

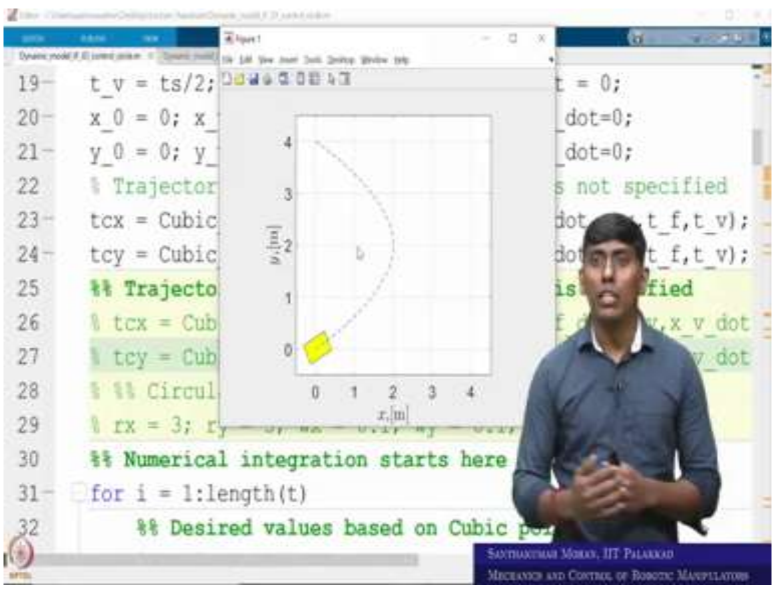
(Refer Slide Time: 24:24)

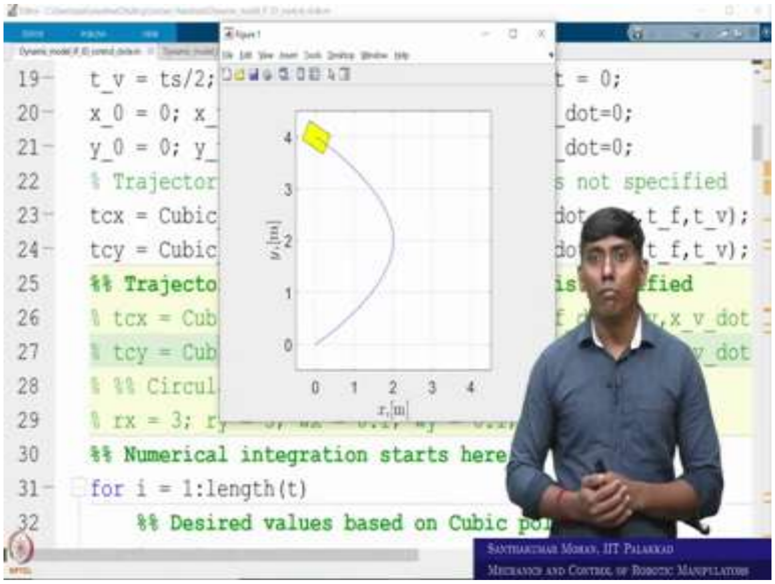


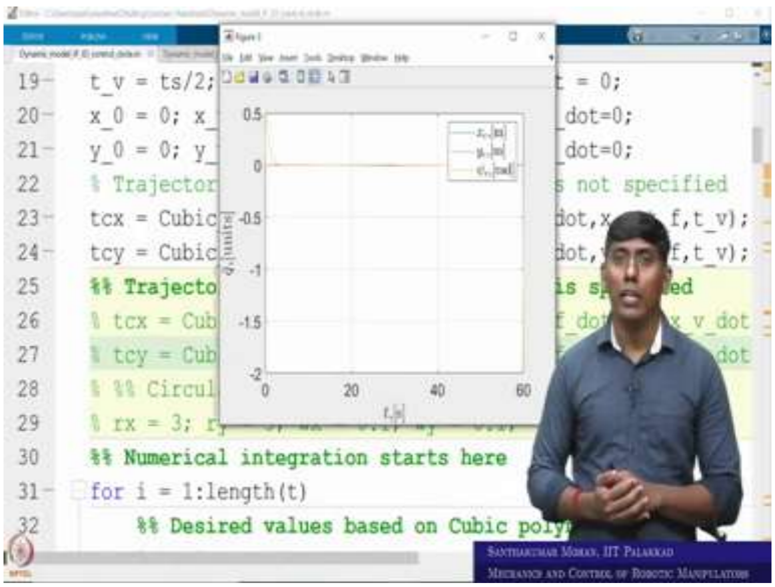

16 q_dot(:,1)
17 %% Trajecto
18 % Trjectory
19 t_v = ts/2;
20 x_0 = 0; x_
21 y_0 = 0; y_
22 % Trajector
23 tcx = Cubic
24 tcy = Cubic
25 %% Trajecto
26 tcx = Cubic
27 tcy = Cubic_via_mrl(y_0,y_f,y_0_dot,y
28 % %% Circular trajectory details
29 % rx = 3; ry = 3; wx = 0.1; wy = 0.1;
Figure 1
$y, [m]$
$x, [m]$
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

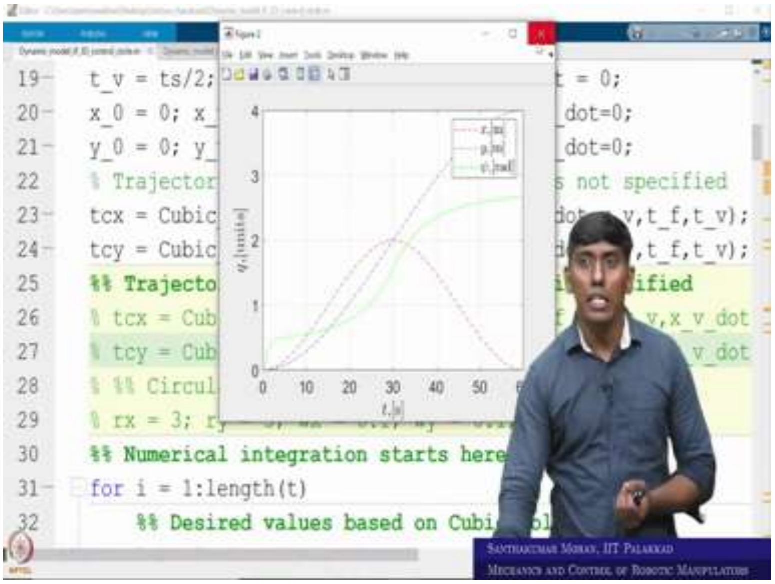

16 q_dot(:,1)
17 %% Trajecto
18 % Trjectory
19 t_v = ts/2;
20 x_0 = 0; x_
21 y_0 = 0; y_
22 % Trajector
23 tcx = Cubic
24 tcy = Cubic
25 %% Trajecto
26 tcx = Cubic
27 tcy = Cubic_via_mrl(y_0,y_f,y_0_dot,
28 % %% Circular trajectory details
29 % rx = 3; ry = 3; wx = 0.1; wy = 0.
Figure 1
$t, [s]$
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


% Trajectory with via point's velocity is not specified
tcx = Cubic_via_mr2(x_0,x_f,x_0_dot,x_f_dot,x_v,t_f,t_v);
tcy = Cubic_via_mr2(y_0,y_f,y_0_dot,y_f_dot,y_v,t_f,t_v);
%% Trajectory with via point's velocity is specified
%% Circular trajectory details
% rx = 3; ry = 3; wx = 0.1; wy = 0.1;
%% Numerical integration starts here
for i = 1:length(t)
%% Desired values based on Cubic
if t(i)<t_v
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


t_v = ts/2;
x_0 = 0;
y_0 = 0;
% Trajector
tcx = Cubic
tcy = Cubic
%% Trajecto
%% Numerical integration starts here
for i = 1:length(t)
%% Desired values based on Cubic
y, [m]
x, [m]
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


t_v = ts/2;
x_0 = 0; x_
y_0 = 0; y_
% Trajector
tcx = Cubic
tcy = Cubic
%% Trajecto
% tcx = Cub
% tcy = Cub
% %% Circul
% rx = 3; r
%% Numerical integration starts here
for i = 1:length(t)
%% Desired values based on Cubic po
Figure 1
y, [m]
x, [m]
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


t_v = ts/2;
x_0 = 0; x_
y_0 = 0; y_
% Trajector
tcx = Cubic
tcy = Cubic
%% Trajecto
% tcx = Cub
% tcy = Cub
% %% Circul
% rx = 3; r
%% Numerical integration starts here
for i = 1:length(t)
%% Desired values based on Cubic poly
Figure 1
t, [s]
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators



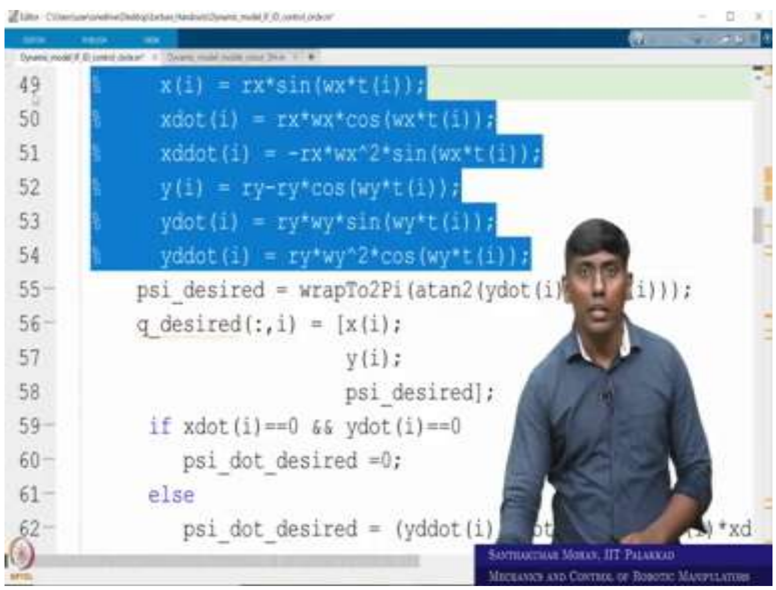
First of all, it is running or not we can see you can see this is with two segments in the sense the endpoint velocity 0 and you can see it goes and it is like goes backward. So, you can like see it is like making it. So, there is a switchover happening. Because it is ending, but you want to make it this in a mobile robot it is not really preferred. You want to like smooth so for that what we are like trying to do.

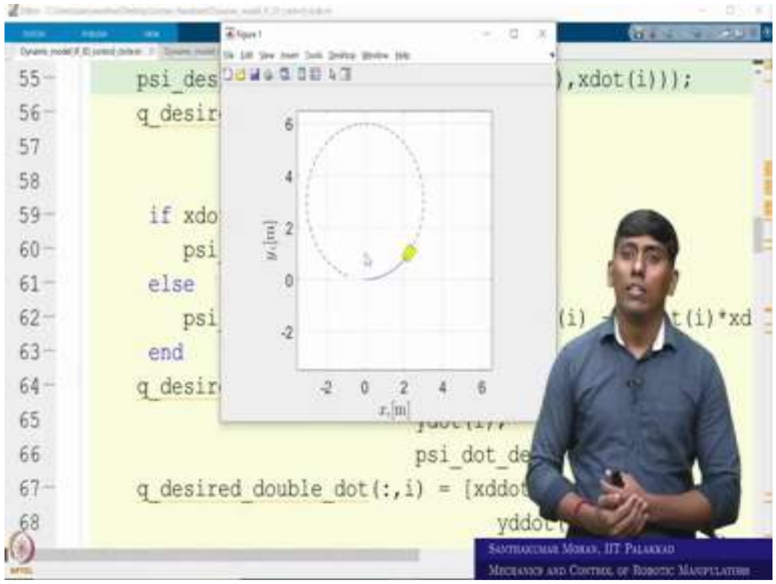
So, we are like trying to make a smooth profile. So, that is what I said via our point velocity is not specified in the sense, the initial what you call so, initial you can see velocity of the second segment is the final velocity of the first segment. Similarly, inertial acceleration of the second segment, we assumed the final acceleration of the first segment.

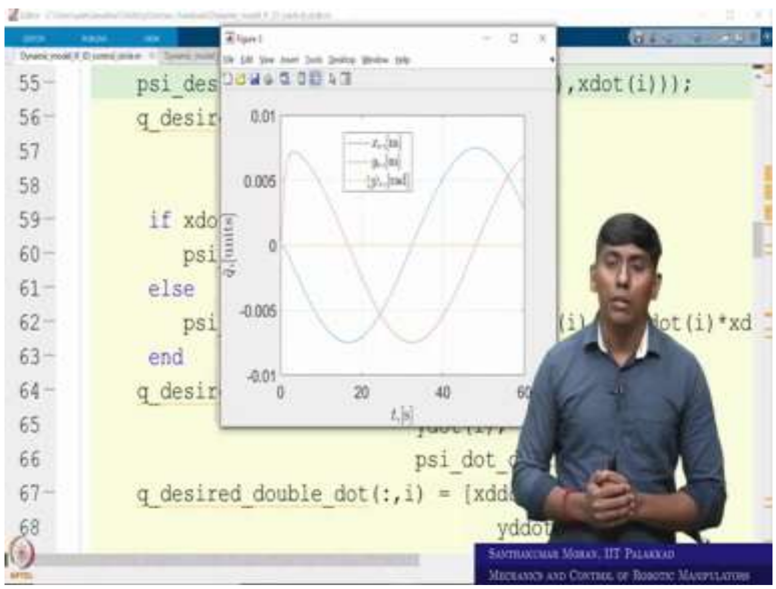
So, second segment initial and final would be matched with the first segment that is what the idea so, in this sense the second profile we take. So, now in that case so, if I ran you can see like the same via point would be maintained, but you can see that the profile itself smooth what is the via point it is 2 comma 2 which is like fulfilling but what addition the vehicle is like making a smooth profile.

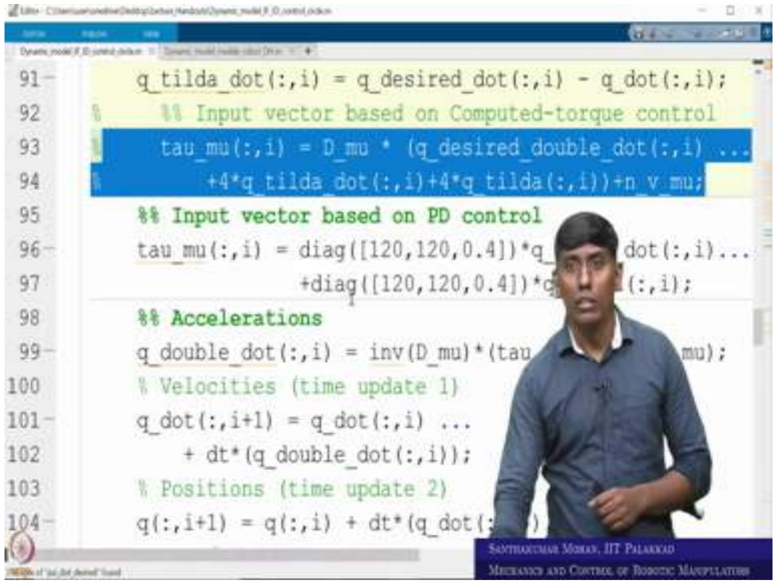
So, this is the beneficial of the even the trajectory generation which we have generated because in the manipulator it is not very clear, but the vehicle profile you can see it is very very important. So, now, you can see in the error case it is like one pulse only. So, whereas, the other case you can see like there are two pulses have come. So, this is what the idea behind.

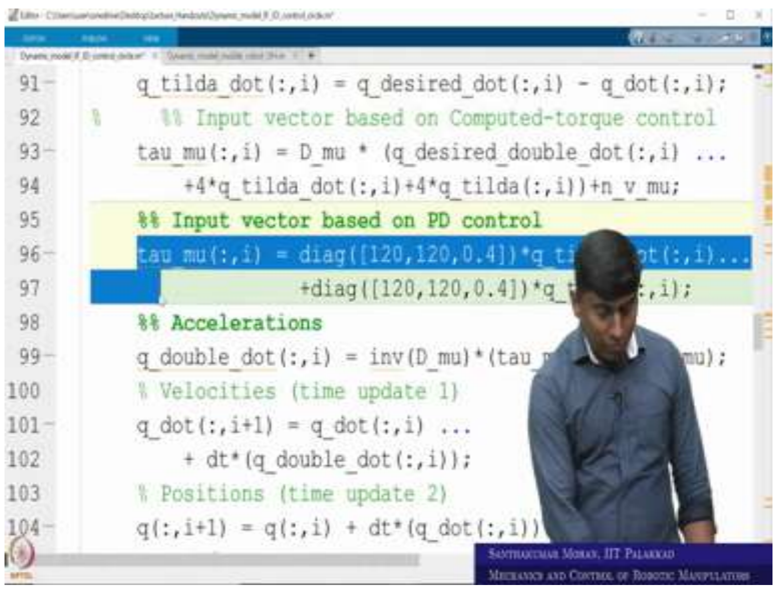
(Refer Slide Time: 26:03)

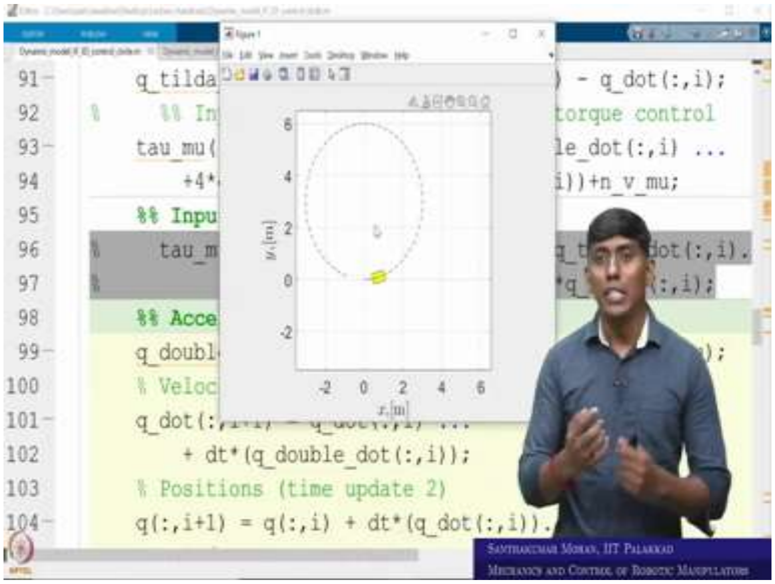


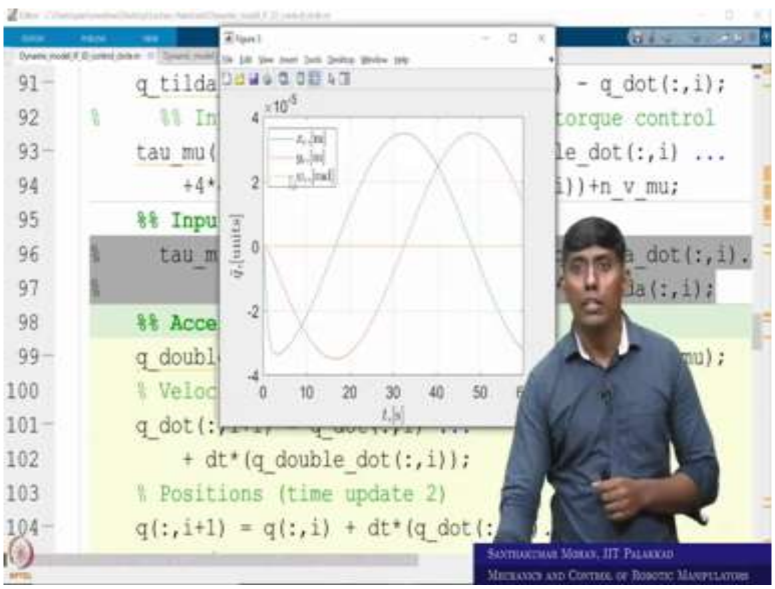

psi_des
),xdot(i)));
q_desir
if xdo
psi
else
psi
(i)
t(i)*xd
end
q_desir
psi_dot_de
q_desired_double_dot(:,i) = [xddot
yddot
x, [m]
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


psi_des
),xdot(i)));
q_desir
if xdo
psi
else
psi
(i)
ot(i)*xd
end
q_desir
psi_dot_
q_desired_double_dot(:,i) = [xdd
yddot
t, [s]
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


q_tilda_dot(:,i) = q_desired_dot(:,i) - q_dot(:,i);
% %% Input vector based on Computed-torque control
% tau_mu(:,i) = D_mu * (q_desired_double_dot(:,i) ...
% +4*q_tilda_dot(:,i)+4*q_tilda(:,i))+n_v_mu;
%% Input vector based on PD control
tau_mu(:,i) = diag([120,120,0.4])*q_
dot(:,i)...
+diag([120,120,0.4])*q
(:,i);
%% Accelerations
q_double_dot(:,i) = inv(D_mu)*(tau
mu);
% Velocities (time update 1)
q_dot(:,i+1) = q_dot(:,i) ...
+ dt*(q_double_dot(:,i));
% Positions (time update 2)
q(:,i+1) = q(:,i) + dt*(q_dot(:
SANTHAKUMAR MOHAN, IIT PALAKKAD
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS


q_tilda_dot(:,i) = q_desired_dot(:,i) - q_dot(:,i);
% %% Input vector based on Computed-torque control
tau_mu(:,i) = D_mu * (q_desired_double_dot(:,i) ...
+4*q_tilda_dot(:,i)+4*q_tilda(:,i))+n_v_mu;
%% Input vector based on PD control
tau_mu(:,i) = diag([120,120,0.4])*q_ti
ot(:,i)...
+diag([120,120,0.4])*q_
:,i);
%% Accelerations
q_double_dot(:,i) = inv(D_mu)*(tau_
mu);
% Velocities (time update 1)
q_dot(:,i+1) = q_dot(:,i) ...
+ dt*(q_double_dot(:,i));
% Positions (time update 2)
q(:,i+1) = q(:,i) + dt*(q_dot(:,i))
SANTHAKUMAR MOHAN, IIT PALAKKAD
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS





So, now, even you want to give us a circular profile. So, I assume that there is a circular profile we want to indent. So, I assume that the circular radius are like equals. So, I just see that is like happening here. So, the circular radius is equal and the frequency also equal. So, in the sense it is supposed to follow a circular profile we can see whether it is like happening or not.

Now, you can see it is like line of sight also we have used so, in the site it is like following on the line. So, this all like happened. So, these all the whole idea behind the Denavit Hartenberg approach. So, now you can see even the mobile robot can be used in the case of what you call in the sense of you can see Denavit Hartenberg approach we can derive the equation and even we have done the motion control. So, here we have done the, you can see motion control with the computed torque control even for the mobile robot even simple PD control is sufficient.

So, just to demonstrate you can see already the PD control is executed now, I am trying to show the computed torque control performance. So, you can see them the performance is like little more improved. So, that is what I just wanted to show it. So, now you can see like this is a computed torque control performance. So, I hope you have enjoyed the whole course is like given a whole idea and this particular lecture is like recap of whatever we have seen from the beginning to the end that is what the whole idea.

So, now even if you look at the error, so error earlier it was like 0.01 or something now it is like 10 power minus 5 that is the beauty of what you call model-based control. If your model is very accurate, then you can go the classical control like model base. So that is the whole idea behind here. I hope you have actual like got the whole essence what we intended this particular lecture. So, with that, I am like ending here. So, see you again. Thank you. Bye, take care.