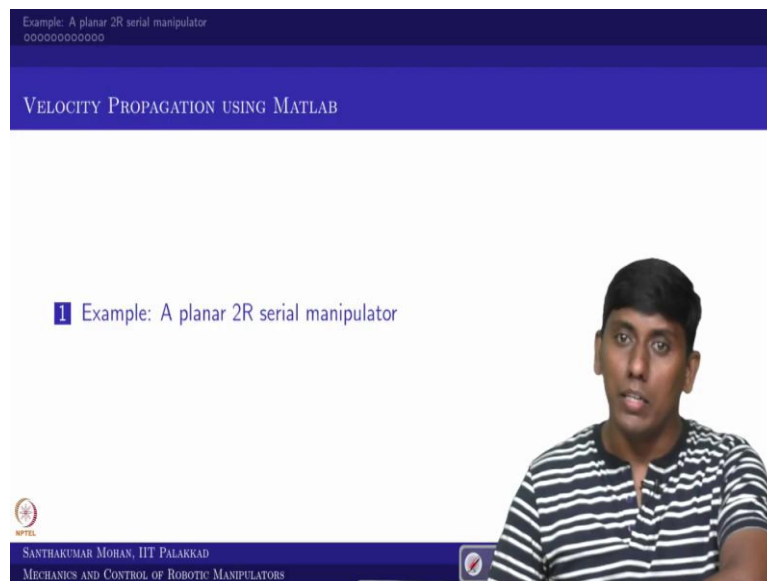


**Mechanic and Control of Robotic Manipulators**  
**Professor. Santhakumar Mohan**  
**Department of Mechanical Engineering**  
**Indian Institute of Technology, Palakkad**  
**Lecture No. 21**  
**Velocity Propagation model using MATLAB**

Welcome back to Mechanics and Control of Robotic Manipulator. The last class we have seen how to you can say generate Jacobian matrix and as well as how to obtain velocity for different joints starting from 0 to the end effector we call base to E, which we simply call velocity propagation model.

So, this is what we have seen and in the end of the lecture itself I said we will be seeing the same thing can be used in MATLAB as an efficient tool with a simple symbolic math code and we can see that in this particular lecture.

(Refer Slide Time: 0:50)



So, in that sense what we are trying to do is we will take the same example, which we have seen in the last lecture, the 2 R serial planar manipulator will take and then we will derive it that in MATLAB and see whether the same equation which we obtained in the analytical method we are getting into MATLAB or not.

(Refer Slide Time: 1:10)

Example: A planar 2R serial manipulator

$${}^0_1\mathbf{R} = \begin{bmatrix} C_1 & -S_1 & 0 \\ S_1 & C_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, {}^1_2\mathbf{R} = \begin{bmatrix} C_2 & -S_2 & 0 \\ S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, {}^2_3\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$${}^0_1\mathbf{P} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, {}^1_2\mathbf{P} = \begin{bmatrix} L_1 \\ 0 \\ 0 \end{bmatrix}, {}^2_3\mathbf{P} = \begin{bmatrix} L_2 \\ 0 \\ 0 \end{bmatrix} \quad (1)$$

SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

So, for that first we will recall. So, the 2 R serial manipulator is having 2 rotary joints. So, these are the; you can say rotation matrices and the position vectors.

(Refer Slide Time: 1:20)

Example: A planar 2R serial manipulator

Base frame angular and linear velocity vectors,

$${}^0_0\boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, {}^0_0\mathbf{v} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \quad (2)$$


SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS


So, based on this if you take base velocities are 0, both you call angular velocity and as well as you call linear velocity, both are 0.

(Refer Slide Time: 1:31)

Example: A planar 2R serial manipulator  
○○●○○○○○○○○

Angular velocity propagation:

$${}^{i+1}\omega = {}^i R^{i+1} \omega + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix} \quad (3)$$
$${}^1\omega = {}^0 R^1 \omega + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \quad (4)$$
$${}^2\omega = {}^1 R^2 \omega + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix} \quad (5)$$



 SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS


Then you can propagate the model based on the angular velocity propagation system.

(Refer Slide Time: 1:37)

Example: A planar 2R serial manipulator  
○○●○○○○○○○○

Angular velocity propagation:

$${}^3\omega = {}^2 R^3 \omega = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix} \quad (6)$$


 SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

So, in that you can start omega 1 omega 2.


(Refer Slide Time: 1:40)

Example: A planar 2R serial manipulator  
○○○○●○○○○○

Linear velocity propagation:

$${}_{i+1}^{i+1}\mathbf{v} = {}_i^{i+1}\mathbf{R} ({}_i^i\mathbf{v} + {}_i^i\boldsymbol{\omega} \times {}_i^{i+1}\mathbf{P}) \quad (7)$$

$${}^1_1\mathbf{v} = {}^1_0\mathbf{R} ({}_0^0\mathbf{v} + {}_0^0\boldsymbol{\omega} \times {}_0^1\mathbf{P}) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (8)$$

$${}^2_2\mathbf{v} = {}^2_1\mathbf{R} ({}_1^1\mathbf{v} + {}_1^1\boldsymbol{\omega} \times {}_1^2\mathbf{P}) = \begin{bmatrix} L_1 S_2 \dot{\theta}_1 \\ L_1 C_2 \dot{\theta}_1 \\ 0 \end{bmatrix} \quad (9)$$



NPTEL  
SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

And then omega three. So, similar way we can start with a linear velocity propagation model and where we start from the linear base velocity is 0 and then we can propagate v 1, v 2 and v3.

(Refer Slide Time: 1:51)

Example: A planar 2R serial manipulator  
○○○○●○○○○○

Linear velocity propagation:

$${}^3_3\mathbf{v} = {}^3_2\mathbf{R} ({}_2^2\mathbf{v} + {}_2^2\boldsymbol{\omega} \times {}_2^3\mathbf{P}) = \begin{bmatrix} L_1 S_2 \dot{\theta}_1 \\ L_1 C_2 \dot{\theta}_1 + L_2 (\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix} \quad (10)$$


NPTEL  
SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

So, once we obtained we can see, we can get to the what you call end effector velocity with respect to you can say base frame, in the sense of velocity of 3 with respect to what you call zeroth frame.


(Refer Slide Time: 2:05)

Example: A planar 2R serial manipulator  
○○○○○○●○○○○

End-effector linear velocities with respect to base frame

$${}^0_3\mathbf{v} = {}^0_3\mathbf{R} {}^3_3\mathbf{v}$$

$${}^0_3\mathbf{v} = \begin{bmatrix} C_{12} & -S_{12} & 0 \\ S_{12} & C_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L_1 S_2 \dot{\theta}_1 \\ L_1 C_2 \dot{\theta}_1 + L_2 (\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix} \quad (11)$$

$${}^0_3\mathbf{v} = \begin{bmatrix} -(L_1 S_1 + L_2 S_{12}) \dot{\theta}_1 - L_2 S_{12} \dot{\theta}_2 \\ (L_1 C_1 + L_2 C_{12}) \dot{\theta}_1 + L_2 C_{12} \dot{\theta}_2 \\ 0 \end{bmatrix}$$


SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

We can see it here. So, in that sense of what I can find, so this will give Jacobian matrix.


(Refer Slide Time: 2:12)

Example: A planar 2R serial manipulator  
○○○○○○●○○○○

End-effector linear velocities with respect to base frame

$${}^0_3\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} -(L_1 S_1 + L_2 S_{12}) & -L_2 S_{12} \\ (L_1 C_1 + L_2 C_{12}) & +L_2 C_{12} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \quad (12)$$

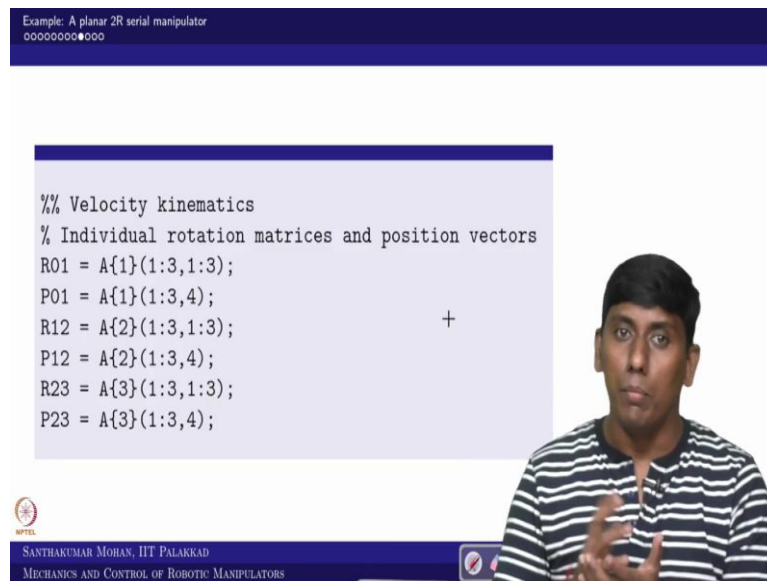
If we consider, it is only a 2 DoF system (further it is a planar system),

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -(L_1 S_1 + L_2 S_{12}) & -L_2 S_{12} \\ (L_1 C_1 + L_2 C_{12}) & +L_2 C_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \quad (13)$$


SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

That is also like we have seen in the last class. So, now the same thing, can we do it in MATLAB in the same format yes, we can do it. So, for that if you are used the live script even it would give in a symbolic manner, but we are not doing it live script, we would be using a simple math script. So, we can see MATLAB script code.

(Refer Slide Time: 2:36)



Example: A planar 2R serial manipulator  
oooooooooooo

```
%% Velocity kinematics
% Individual rotation matrices and position vectors
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);
```

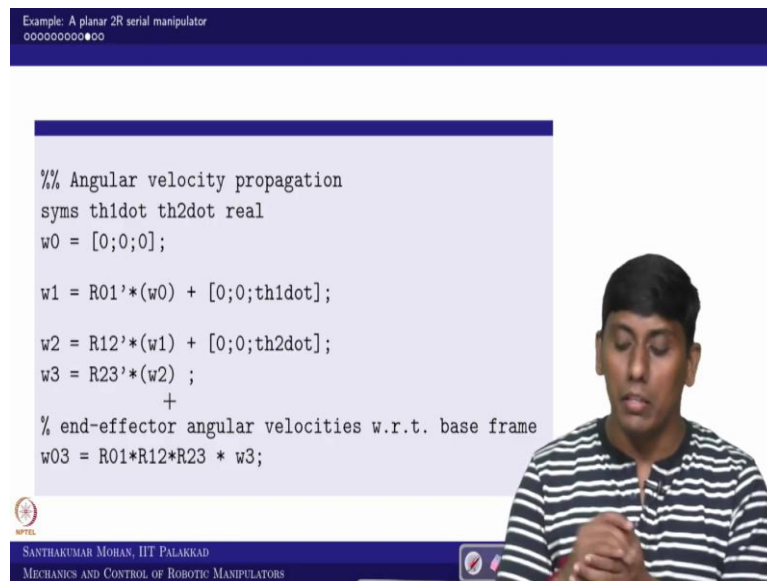
SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

So, for that I am showing the code here. So, these are the velocity, you can say kinematics which we are trying to do it. So, for that we are taking the rotational matrices and position vectors, which we have obtained from the previous you can say simulation code, we can obtain where A would be the cell that would be having 3 cells.

So, each cell would be having a rotation and position, rotation matrix and position vectors. So, from there you can take, for example if you take a 1 which is the first cell, the first 3 cross 3 would be corresponding to the rotational matrix and the next you can say 3 of the fourth column would be equivalent into position vector, this is we all know.

So, the same way we can do it for the second cell and the third cell. So, now what we have done, so previous code we have taken and from that, we can find the rotation matrices and position vectors, once we obtain this.

(Refer Slide Time: 3:33)



Example: A planar 2R serial manipulator  
ooooooooo●oo

```
%% Angular velocity propagation
syms th1dot th2dot real
w0 = [0;0;0];

w1 = R01'*(w0) + [0;0;th1dot];

w2 = R12'*(w1) + [0;0;th2dot];
w3 = R23'*(w2) ;
+
% end-effector angular velocities w.r.t. base frame
w03 = R01*R12*R23 * w3;
```

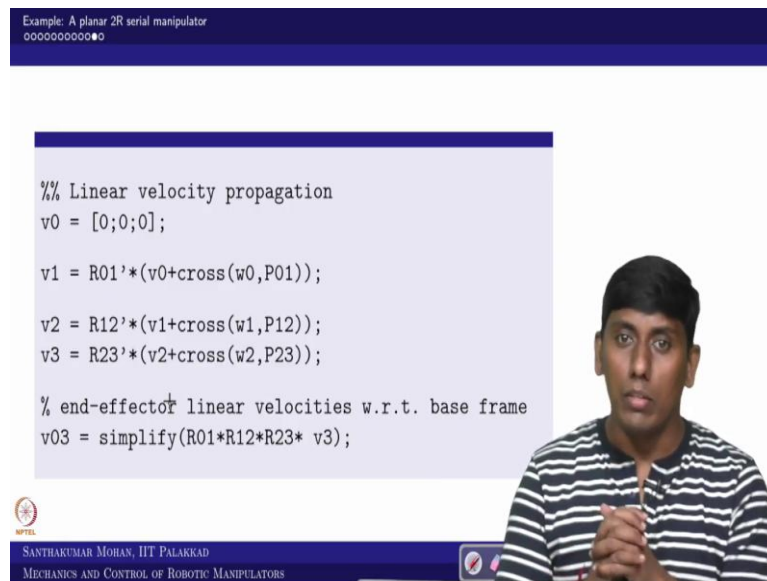
SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

So, what we can do we can do the angular velocity propagation, where we can see there are 2 additional variable would becoming virtually these are theta 1 dot and theta 2 dot, so here we cannot write theta 1 theta 2 dot I have returned th1 dot th2 dot, so both are I represented as real and further what we have taken, we are taken that the base angular velocity would be starting from 0 because it is fixed. So, in the sense of omega 0 which I have written as w 0 equal to 0 0 0 in the sense, you can say zeros of 3 comma 1.

So, now, once you obtain this what we can do, we can substitute the velocity propagation model where omega i plus 1 to i plus 1 which we can write as R i plus 1 to i, omega i to i plus, so the variable is addition in this case, so we have already know the rotation matrix that transpose would be coming here because it is inverse and omega 0 is known and this is the active joint. So, theta 1 dot is added which is third quadrant, not quadrant third element, in the sense z axis. So, this will give omega 1 which I have written as w 1.

So, similarly we can calculate w 2 and we 3 once you obtain w 1, you can calculate w 3 and you can calculate w you can say 2 and 3 through this sequence. Once you obtain what one can do. So, this is equivalent to omega 3 with respect to third frame. So, in that sense, you can calculate what you can say. So, the omega 3 with respect to 0. So, these are the things we are trying to see it in MATLAB. So, let us go one step further.

(Refer Slide Time: 5:17)



Example: A planar 2R serial manipulator  
oooooooooooo●

```
%% Linear velocity propagation
v0 = [0;0;0];

v1 = R01*(v0+cross(w0,P01));

v2 = R12*(v1+cross(w1,P12));
v3 = R23*(v2+cross(w2,P23));

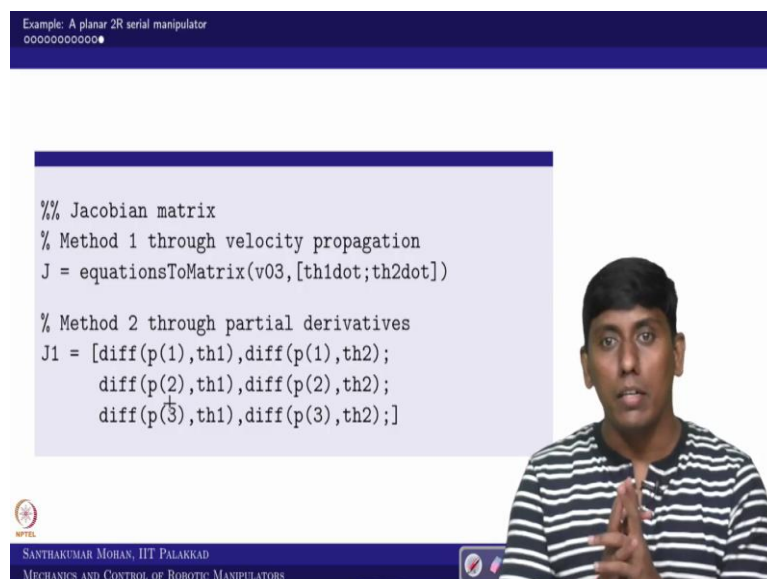
% end-effector linear velocities w.r.t. base frame
v03 = simplify(R01*R12*R23* v3);
```

SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS

So, what we can do we can try to understand the linear velocity propagation. So, for that again the base we are starting the base it is fixed, so the linear velocity would be 0 0 0, if it is put it on the mobile base then this would come with the mobile base velocities. So, once this is all done, so we can go the velocity propagation model where the slip velocity and the tangential velocity would be coming.

So, once you calculate this, so  $v_1$  then  $v_2$  and  $v_3$  we can calculate, once you calculate  $v_3$  you can get the  $v_{03}$  in a sense, so velocity of third frame with respect to zeroth frame we can calculate. So, here the simplify command is just to simplify the overall equations. So, now we will see this further.

(Refer Slide Time: 6:04)



Example: A planar 2R serial manipulator  
oooooooooooo●

```
%% Jacobian matrix
% Method 1 through velocity propagation
J = equationsToMatrix(v03,[th1dot;th2dot])

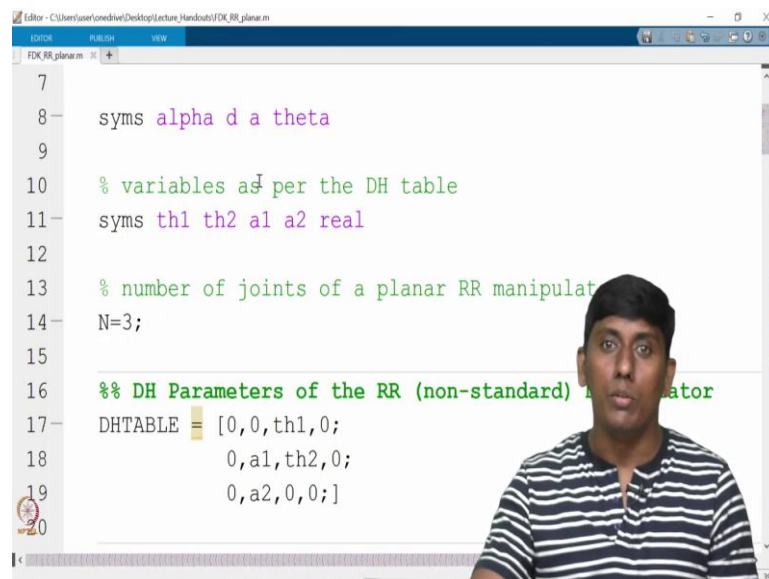
% Method 2 through partial derivatives
J1 = [diff(p(1),th1),diff(p(1),th2);
      diff(p(2),th1),diff(p(2),th2);
      diff(p(3),th1),diff(p(3),th2);]
```

SANTHAKUMAR MOHAN, IIT PALAKKAD  
MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS



So, once you obtain this what one can do, we wanted Jacobian matrix which would be you can say coefficient of your you can say theta 1 dot and theta 2 dot, so this coefficient we can use a simple MATLAB command called equations to matrix or once you know the position vector from the forward kinematic model, you can do the partial differentiation with respect to Q vector. So, that is what we have done here. So, now we have all seen, so how the MATLAB code look like in a slide format, we will go to the original MATLAB code.

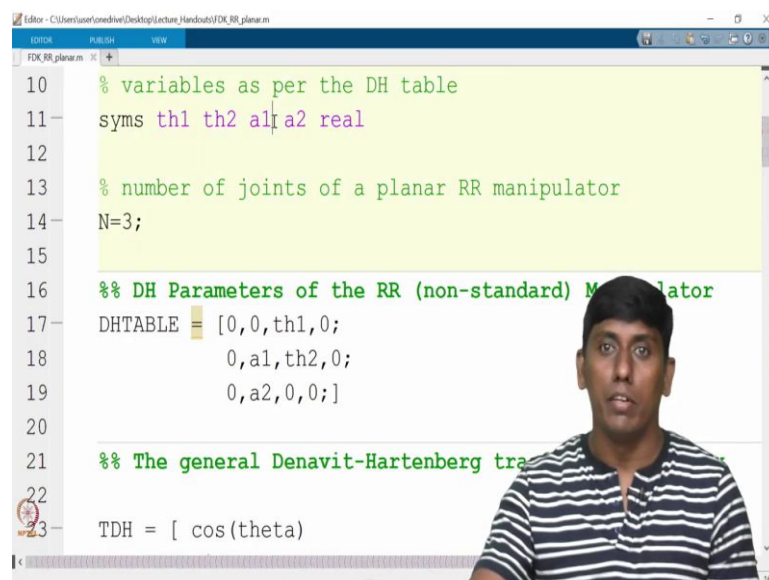
(Refer Slide Time: 6:39)



```

7
8 syms alpha d a theta
9
10 % variables as per the DH table
11 syms th1 th2 a1 a2 real
12
13 % number of joints of a planar RR manipulator
14 N=3;
15
16 %% DH Parameters of the RR (non-standard) Manipulator
17 DHTABLE = [0,0,th1,0;
18            0,a1,th2,0;
19            0,a2,0,0;]
20

```



```

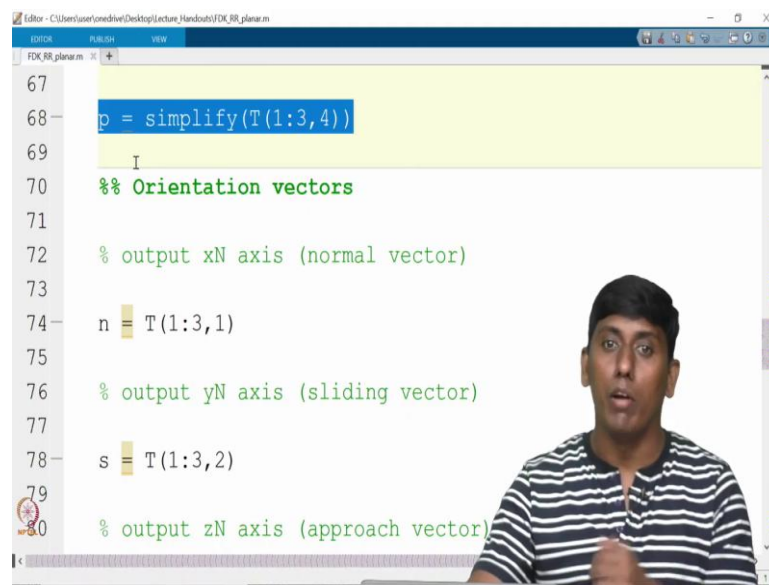
10 % variables as per the DH table
11 syms th1 th2 a1 a2 real
12
13 % number of joints of a planar RR manipulator
14 N=3;
15
16 %% DH Parameters of the RR (non-standard) Manipulator
17 DHTABLE = [0,0,th1,0;
18            0,a1,th2,0;
19            0,a2,0,0;]
20
21 %% The general Denavit-Hartenberg tra
22
23 TDH = [ cos(theta)

```

So, this is the direct kinematic you can say code which we have done in the previous simulation class. So, this is the; you can say the generalized variable and a variable based on the DH table we have actually given and here the number of joints are including you can say end effector it is 3. But excluding the zeroth frame.

Then the DH parameter as per the table which you are derived that we can substitute and the arm matrix we can actually get it here we are going to use the non standard one So, these all we have seen. So, based on that what one can see, you can actually create A would consist of 3 cells. So, each cell would be having a transformation matrix. So, these all we have seen in the previous simulation class. Right now, what we are taking.

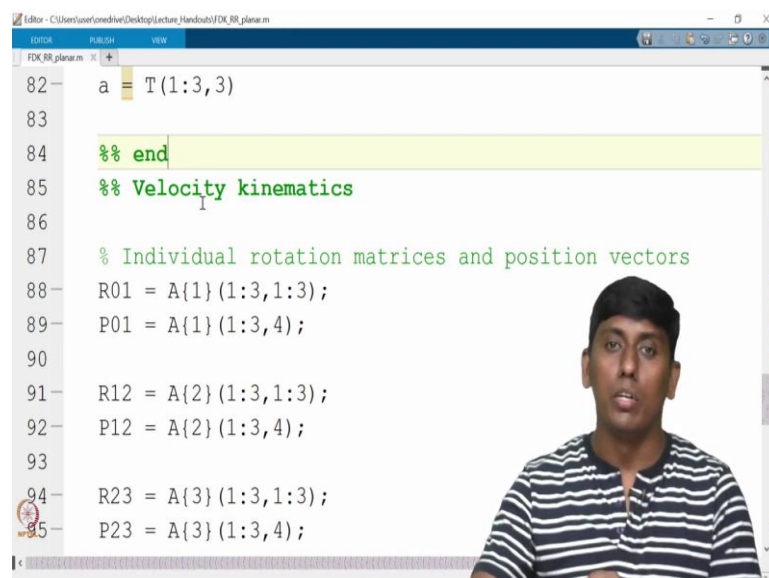
(Refer Slide Time: 7:26)



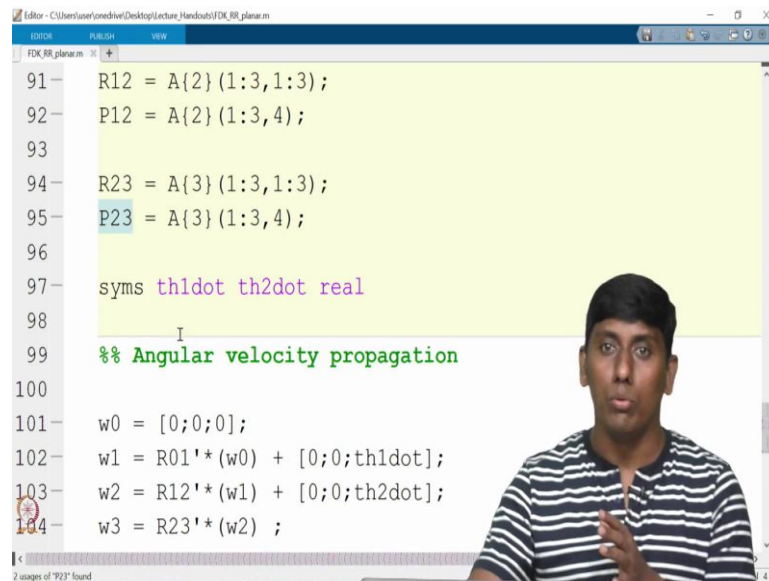
```
67
68 p = simplify(T(1:3,4))
69
70 %% Orientation vectors
71
72 % output xN axis (normal vector)
73
74 n = T(1:3,1)
75
76 % output yN axis (sliding vector)
77
78 s = T(1:3,2)
79
80 % output zN axis (approach vector)
```

So, we have taken the P for the Jacobian, but right now we are trying to see the; you can say velocity propagation model.

(Refer Slide Time: 7:33)



```
82 a = T(1:3,3)
83
84 %% end
85 %% Velocity kinematics
86
87 % Individual rotation matrices and position vectors
88 R01 = A{1}(1:3,1:3);
89 P01 = A{1}(1:3,4);
90
91 R12 = A{2}(1:3,1:3);
92 P12 = A{2}(1:3,4);
93
94 R23 = A{3}(1:3,1:3);
95 P23 = A{3}(1:3,4);
```



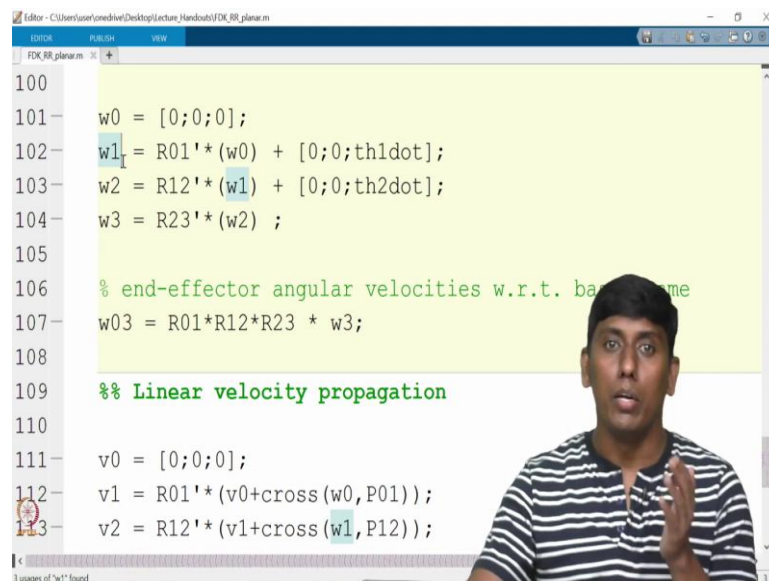
```
91- R12 = A{2} (1:3, 1:3);
92- P12 = A{2} (1:3, 4);
93
94- R23 = A{3} (1:3, 1:3);
95- P23 = A{3} (1:3, 4);
96
97- syms th1dot th2dot real
98
99- %% Angular velocity propagation
100
101- w0 = [0;0;0];
102- w1 = R01'*(w0) + [0;0;th1dot];
103- w2 = R12'*(w1) + [0;0;th2dot];
104- w3 = R23'*(w2);
```

So, for that we are actually coming to the velocity kinematics. So, till last simulation class we have ended at this end. So, we are adding the code whatever I have shown in the slide here from starting from here. So, this is the velocity kinematics. So, for starting the velocity kinematics you need to know the individual rotation matrices and position vectors.

So, this you can get it from the you can say cell which you have created in the previous direct kinematic model. So, these all we have obtained. So, in this case rotation matrix of 1 with respect to 0 2 rotation matrix of 3 with respect to 2. Similarly, the position vector 0 1 with respect to 0 2 position vector of 3 with respect to 2. So, these are all we obtained.

So, then we are going to the velocity propagation for that we need to create a variable. So, here there are two active variables which is theta 1 dot and theta 2 dot which we have written as theta like th1 dot th2 dot which we consider as real.

(Refer Slide Time: 8:31)

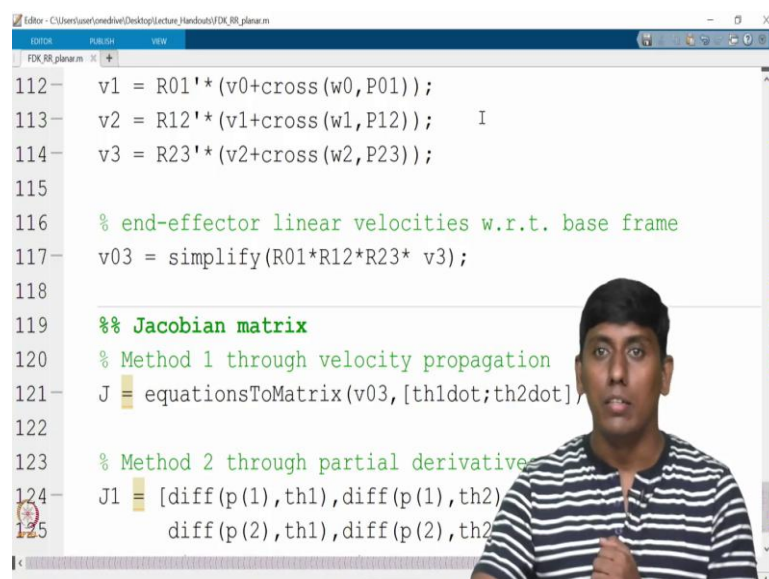


```
100
101 w0 = [0;0;0];
102 w1 = R01'*(w0) + [0;0;thldot];
103 w2 = R12'*(w1) + [0;0;th2dot];
104 w3 = R23'*(w2) ;
105
106 % end-effector angular velocities w.r.t. base frame
107 w03 = R01*R12*R23 * w3;
108
109 %% Linear velocity propagation
110
111 v0 = [0;0;0];
112 v1 = R01'*(v0+cross(w0,P01));
113 v2 = R12'*(v1+cross(w1,P12));
```

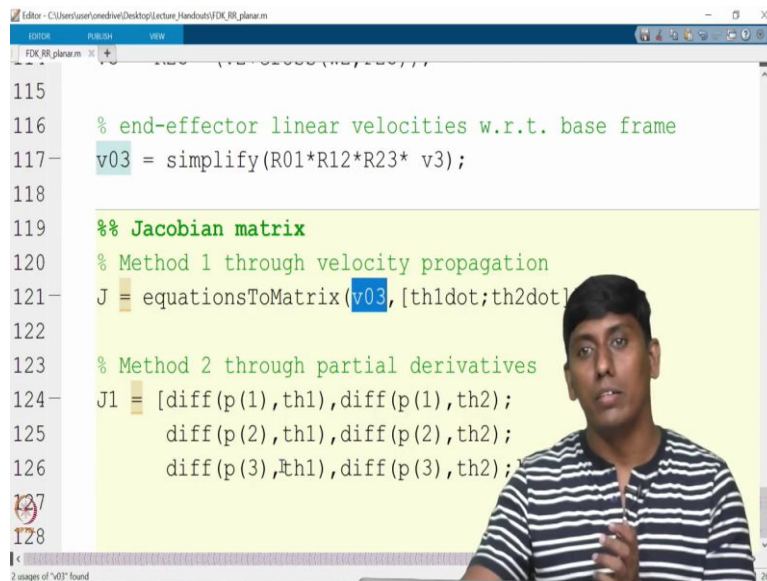
So, now this is the; you can say code which is taken from the velocity propagation. So, once you have the angular velocity is zeroth frame is known, then you can calculate the further proceeding frames 1, 2, 3 and all. So, this is the code, and we can find the end effector angular velocity.

So, then we can go to the linear velocity propagation because the linear velocity propagation required both or you can say previous frame linear velocity and angular velocity and position vector. Position vector by the way we got it from the direct kinematic model, but the angular velocity needs to be calculated that is why we have calculated already. So, now we can do the linear velocity propagation.

(Refer Slide Time: 9:14)



```
112 v1 = R01'*(v0+cross(w0,P01));
113 v2 = R12'*(v1+cross(w1,P12));
114 v3 = R23'*(v2+cross(w2,P23));
115
116 % end-effector linear velocities w.r.t. base frame
117 v03 = simplify(R01*R12*R23* v3);
118
119 %% Jacobian matrix
120 % Method 1 through velocity propagation
121 J = equationsToMatrix(v03, [thldot;th2dot]);
122
123 % Method 2 through partial derivatives
124 J1 = [diff(p(1),th1),diff(p(1),th2),
125       diff(p(2),th1),diff(p(2),th2)];
```



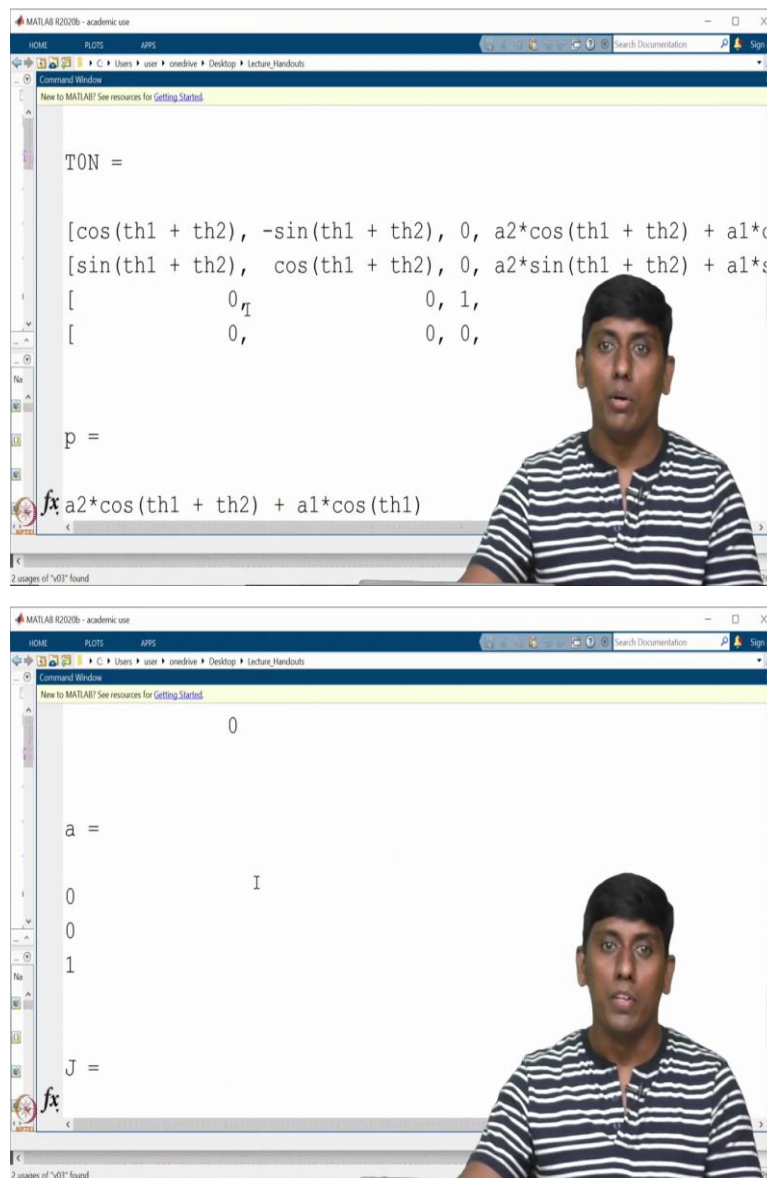
```
115
116 % end-effector linear velocities w.r.t. base frame
117 v03 = simplify(R01*R12*R23* v3);
118
119 %% Jacobian matrix
120 % Method 1 through velocity propagation
121 J = equationsToMatrix(v03, [th1dot;th2dot]);
122
123 % Method 2 through partial derivatives
124 J1 = [diff(p(1),th1),diff(p(1),th2);
125       diff(p(2),th1),diff(p(2),th2);
126       diff(p(3),th1),diff(p(3),th2)];
127
128
```

So, these are the propagated model. Since in this particular example, there is no linear joint, so everything is rotary, so there is no addition of  $D^{-1} \dot{p}_i$  or you can say  $D^{-1} \dot{p}_i + 1$  that is we have not added. So, finally you can calculate the 3. So, with respect to base also can calculate. So, now this is what the end effector linear velocity with respect to base. In the sense, the end effector velocity, which is here as a third frame that velocity with zeroth frame you can calculate, which is as equal to  $\dot{p}_x \dot{p}_y \dot{p}_z$ .

So, since it is in a planar the  $\dot{p}_z$  will be 0. So, we can take from there I already said, so the command called equations to matrix we can do it. So, where  $v03$  is the final end effector velocity, so I want to get it that partial derivative. So, I can take it a coefficient, this is one way or the other way, you can do it method through partial derivative.

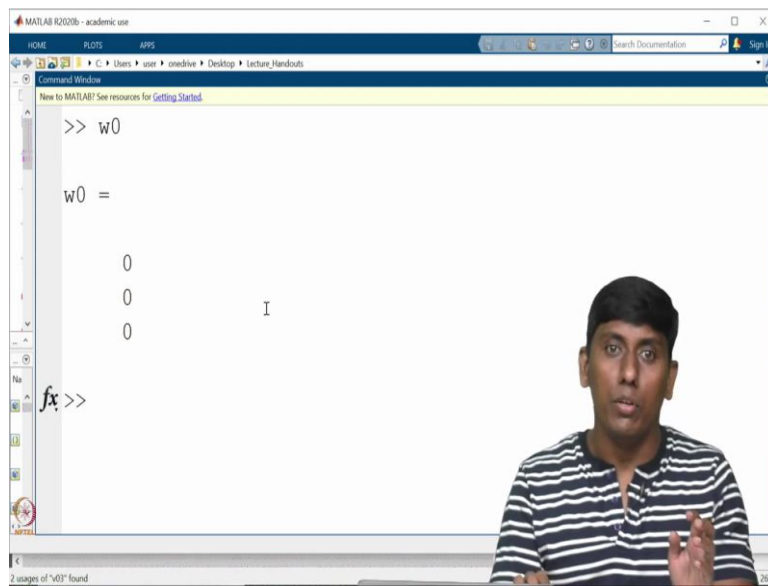
So, we will go one by one. So, as I already said, I have minimized this editor window. So, you click the editor window, and you can run this code. So, this code is actually like, you can say error free.

(Refer Slide Time: 10:28)



So, now if you ran this code, you can see that the result would be obtained here. So, we can start from this. So, these are all we have seen in your kinematic model. So, up to what you call approach, we will start from what you call omega 0, omega 0, we have, I will put it.

(Refer Slide Time: 10:45)



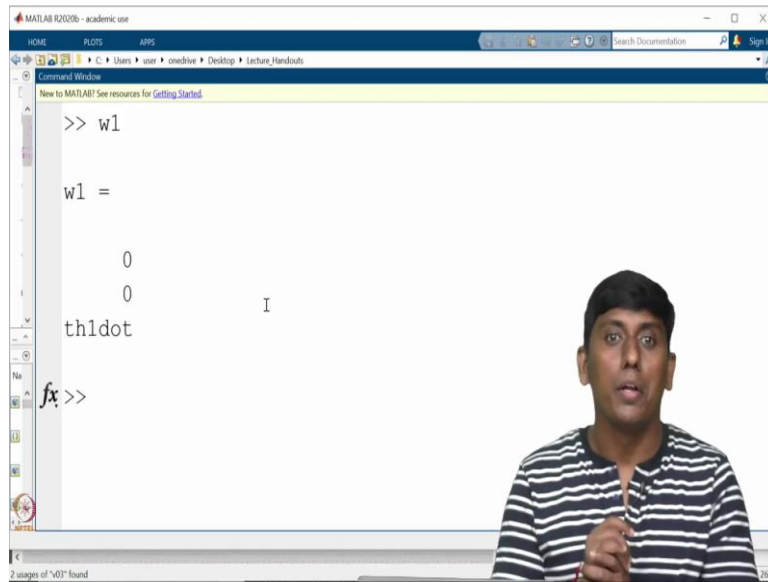
MATLAB R2020b - academic use

```
>> w0  
  
w0 =  
  
    0  
    0  
    0
```

*fx* >>

2 usages of "w0" found

The image shows a MATLAB Command Window with the following content: The user enters the command `w0`. The output is a column vector of three zeros. A handwritten *fx* is next to the prompt `>>`. The instructor is visible in the bottom right corner.



MATLAB R2020b - academic use

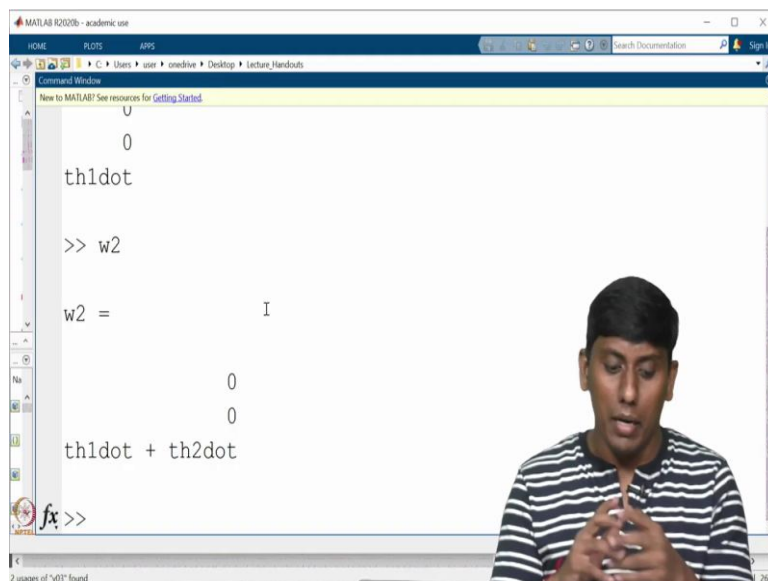
```
>> w1  
  
w1 =  
  
    0  
    0
```

th1dot

*fx* >>

2 usages of "w0" found

The image shows a MATLAB Command Window with the following content: The user enters the command `w1`. The output is a column vector of two zeros. The text `th1dot` is visible above the prompt. A handwritten *fx* is next to the prompt `>>`. The instructor is visible in the bottom right corner.



MATLAB R2020b - academic use

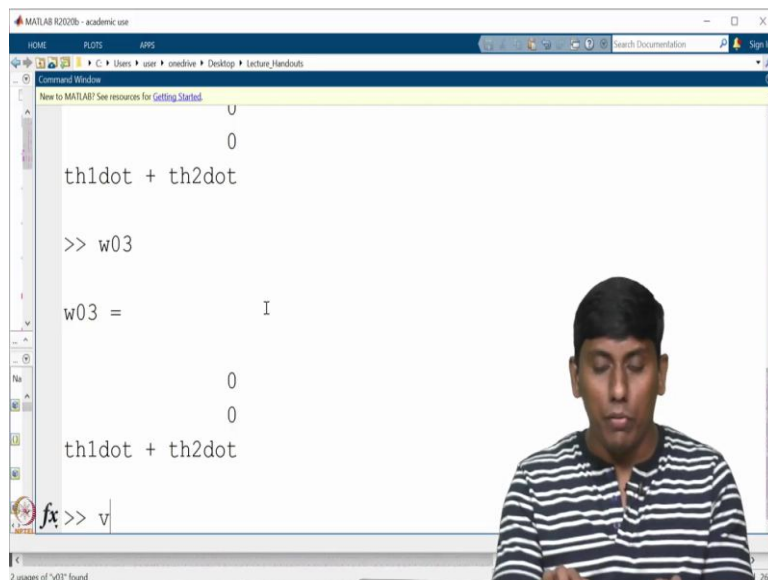
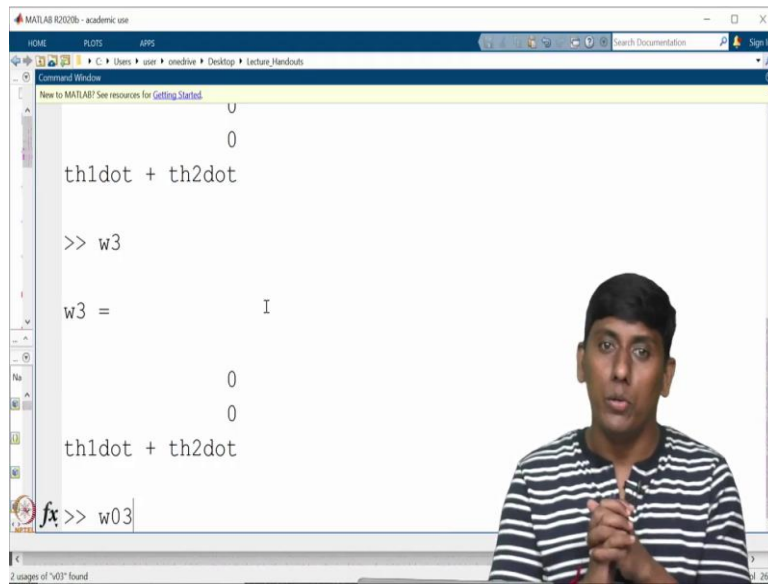
```
th1dot  
  
>> w2  
  
w2 =  
  
    0  
    0
```

th1dot + th2dot

*fx* >>

2 usages of "w0" found

The image shows a MATLAB Command Window with the following content: The text `th1dot` is visible above the prompt. The user enters the command `w2`. The output is a column vector of two zeros. The text `th1dot + th2dot` is visible below the output. A handwritten *fx* is next to the prompt `>>`. The instructor is visible in the bottom right corner.



So, omega 0 we have considered as 0 0. So, we will see what is omega 1 omega 1 which we got it from the analytical model list, so 0 0 theta 1 dot whether we are getting it or not, yes, we are getting it. So, similarly, omega 2 would be theta 1 dot plus theta 2 in z axis. So, that is also we obtained.

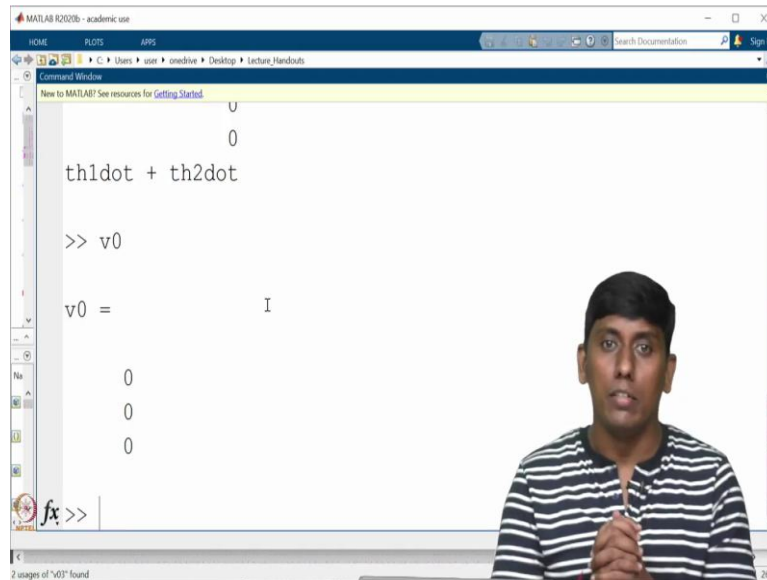
The same as you can see omega 3, because it is the same link that is also, we obtain. So, now we can see, what would be the end effector. So, end effector also straightforward, because it is actually like, simple but you have a rotation matrix, if the theta 1 and theta 2 are nonzero, then that also will play, but in this case, I do not think that would be having a play, because your rotation matrix would be, so cos theta 1 plus theta 2. So, like that it comes.

So, this all you can say thetas would be in the x and y, but your omega 3 is having x and y is 0 0. So, that is what you can see. So, in the sense, so the third frame, you can say z axis and

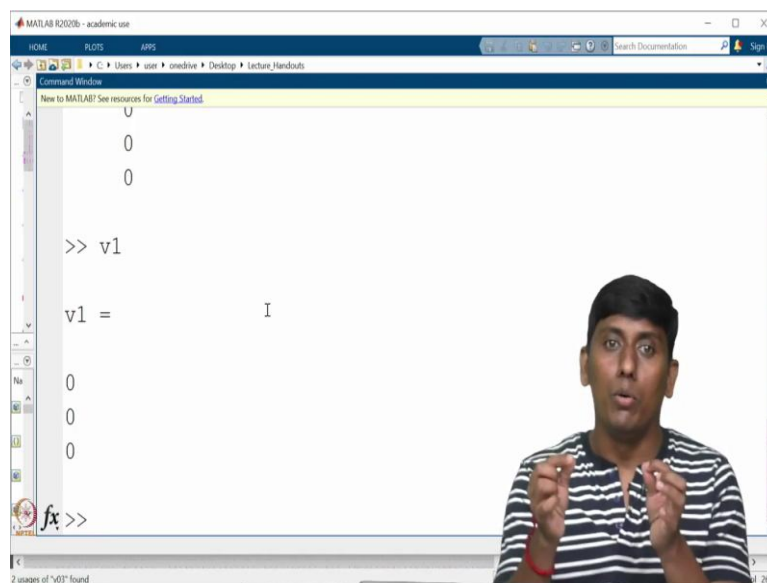


zeroth frame z axis all are parallel. So, in the sense whatever you have the third frame velocity that would be directly transformed to you can say with respect to zeroth frame. So, this is correct, you got it. So, this is we are verified in the analytical, analytical also giving the same.

(Refer Slide Time: 12:09)



A screenshot of the MATLAB R2020b Command Window. The window title is "MATLAB R2020b - academic use". The Command Window shows the following text: `U`, `0`, `th1dot + th2dot`, `>> v0`, and `v0 =` followed by a 3x1 column vector of zeros. A cursor is positioned at the end of the vector. A small inset video of a man in a striped shirt is visible in the bottom right corner of the MATLAB window.



A screenshot of the MATLAB R2020b Command Window. The window title is "MATLAB R2020b - academic use". The Command Window shows the following text: `U`, `0`, `0`, `>> v1`, and `v1 =` followed by a 3x1 column vector of zeros. A cursor is positioned at the end of the vector. A small inset video of a man in a striped shirt is visible in the bottom right corner of the MATLAB window.

```

MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\OneDrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
v
0
0
>> v2
v2 =          I
a1*th1dot*sin(th2)
a1*th1dot*cos(th2)
0
fx >>
2 usages of 'v0' found

```

So, we start from  $v_1$ , so  $v_0$  we assumed as 0. So,  $v_1$  would be what we expect because 0 and 1 on the same point, so there would be 0 linear velocity that is what we have seeing in the analytical also. So, here also the same, but when you come to the second you can see, so there would be  $L_1$  distance, and the tangential velocity would be coming. So, that would come here.

So, if you put  $\theta_2$  is 0, so what you can see, if you put  $\theta_2$  to 0, so it is velocity would be only in y axis that is very clear,  $a_1 \dot{\theta}_1$  would be the magnitude that would be direct parallel to Y axis. So, that is visible here also and the x axis 0 velocity that is also clear. So, now, you put  $\theta_2$  to is 90 degrees. So, you can see in that case.

So,  $a_1 \dot{\theta}_1$  would be having perpendicular which is approach side. So, that is also. So,  $a_1 \dot{\theta}_1$  would be there. So, in that sense this direction, but if you take it from zeroth frame that would be a negative direction, that you can a cross check it.

(Refer Slide Time: 13:23)

```

MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
a1*th1dot*cos(th2)
a1*th1dot*sin(th2)
0
>> v3
v3 =
I
a1*th1dot*sin(th2)
a2*(th1dot + th2dot) + a1*th1dot*cos(th2)
fx >>

```

```

MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
>> v03
v03 =
- a1*th1dot*sin(th1) - a2*th1dot*sin(th1 + th2) - a2*th2dot*sin(th1 + th2)
a1*th1dot*cos(th1) + a2*th1dot*cos(th1 + th2) + a2*th2dot*cos(th1 + th2)
fx >>

```

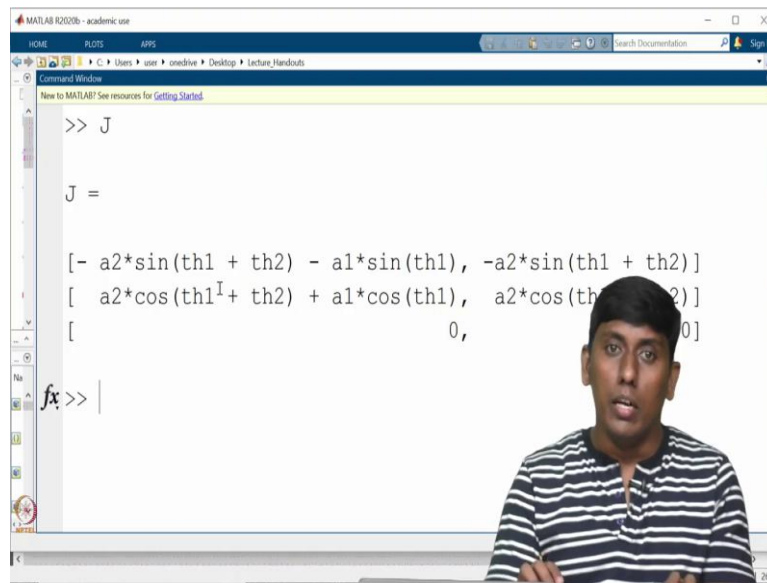
```

MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
- a1*th1dot*sin(th1) - a2*th1dot*sin(th1 + th2) - a2*th2dot*sin(th1 + th2)
a1*th1dot*cos(th1) + a2*th1dot*cos(th1 + th2) + a2*th2dot*cos(th1 + th2)
0
fx

```

So, let us go to the v3. So, v3 would be added. So, I will show it clear. So, these are the v3 components. So, you can see now, the v3 the x axis 1 velocity added but y axis you can see something is addition. So, this is all with respect to something but what we wanted. So, v03. So, so, v03 is what you wanted, so this is having a lengthy equation. So, but this lengthy equation you can reduce it to a simple one.

(Refer Slide Time: 14:08)



The image shows a MATLAB Command Window with the following text:

```
>> J

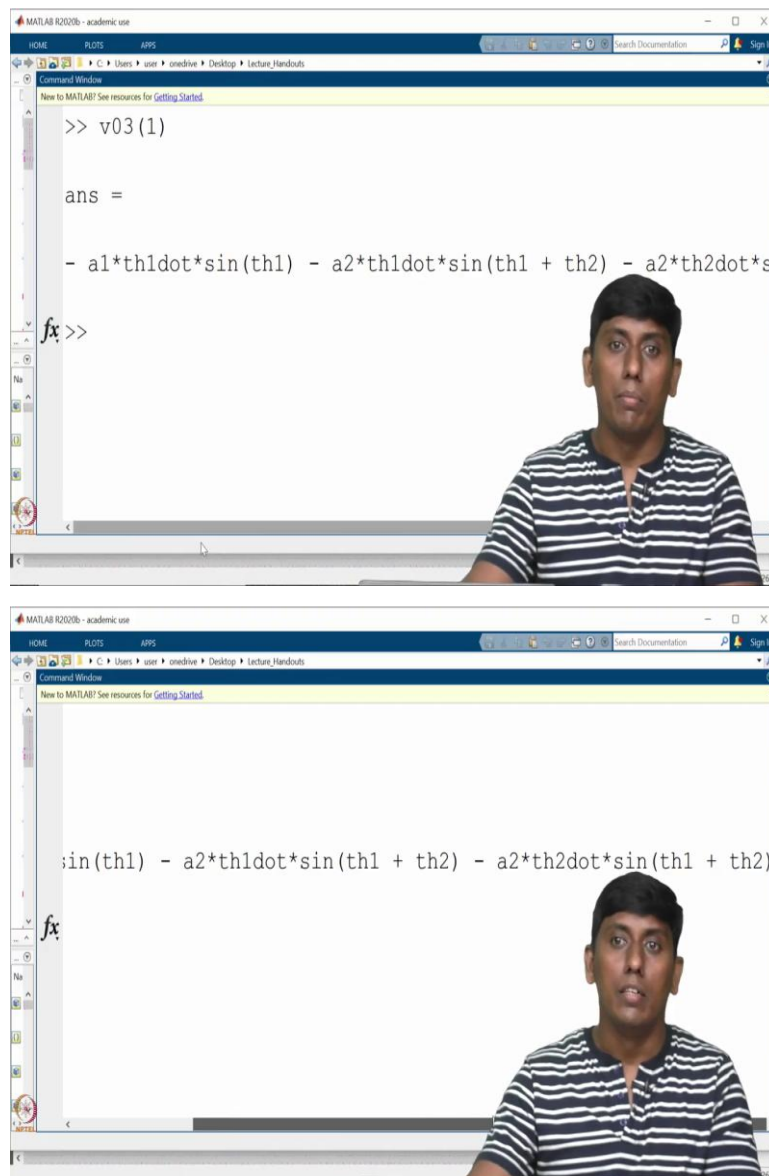
J =

[- a2*sin(th1 + th2) - a1*sin(th1), -a2*sin(th1 + th2)]
[ a2*cos(th1 + th2) + a1*cos(th1), a2*cos(th1 + th2)]
[ 0, 0]
```

A small inset image of a man in a striped shirt is overlaid on the bottom right of the MATLAB window.

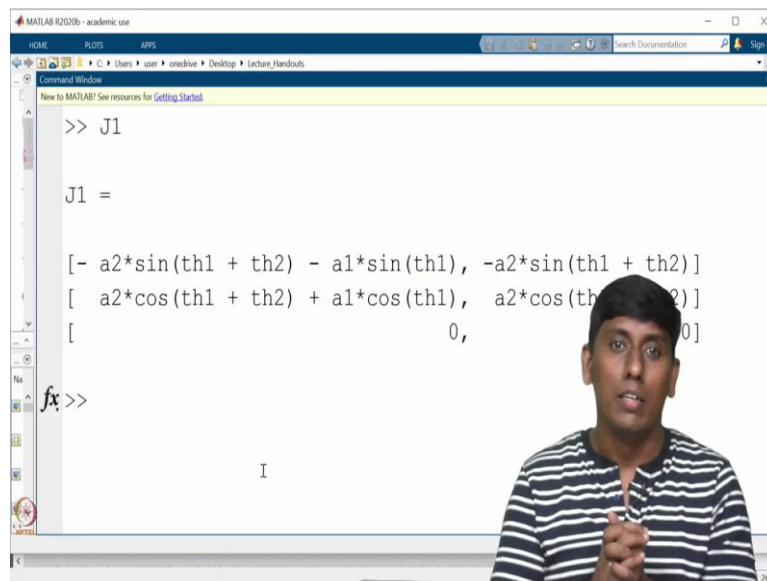
So, for that what we are trying to do the Jacobian. So, you can see the Jacobian. So, Jacobian you can get it. So, you can see, this is a2 sine theta 1 plus theta 2 minus a1 sine theta 1, this is what we also obtained in the earlier one and here also you can see this is a2 sine theta 1 plus theta 2. So, the minus sign is very clear. So, similar way you can see the x and y. So, in that sense, I want to show like this is the first component.

(Refer Slide Time: 14:39)



So, I will just make it this is the first component, you can see first component is having, so theta 1 dot is having you can say two places, theta 2 dot is having two places, theta 2 place is having once place. So, that is what we can see it as Jacobian.

(Refer Slide Time: 15:02)



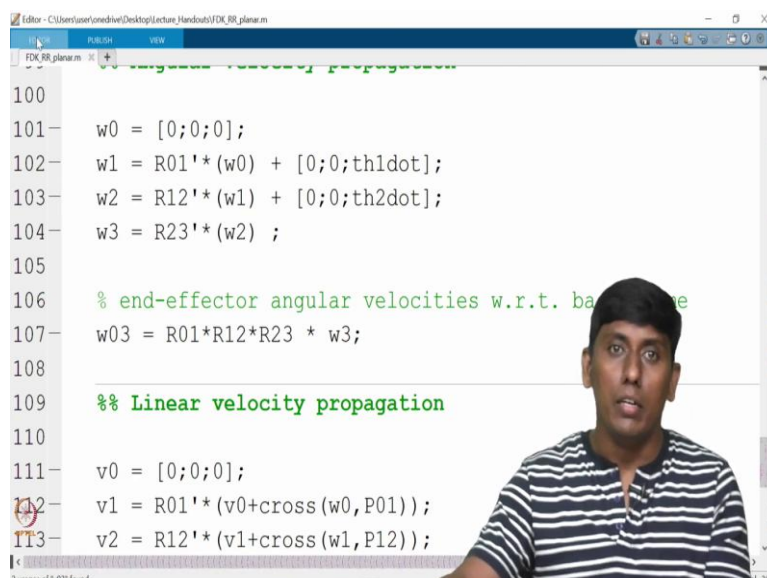
```
>> J1

J1 =

[-a2*sin(th1 + th2) - a1*sin(th1), -a2*sin(th1 + th2)]
[ a2*cos(th1 + th2) + a1*cos(th1), a2*cos(th1 + th2)]
[ 0, 0]
```

The same thing you can expect in J 1 with the partial derivative also. So, that is what we have obtained. So, you can see this is also the same case what we see. So, in the sense of what one can clearly see. So, whatever you have done in analytical the same thing, you can put it as a code in MATLAB and do it even this can be even further simplified as a simple generic code. But that is a little complex to understand. So, that is why I have written this code as a lengthy one.

(Refer Slide Time: 15:34)



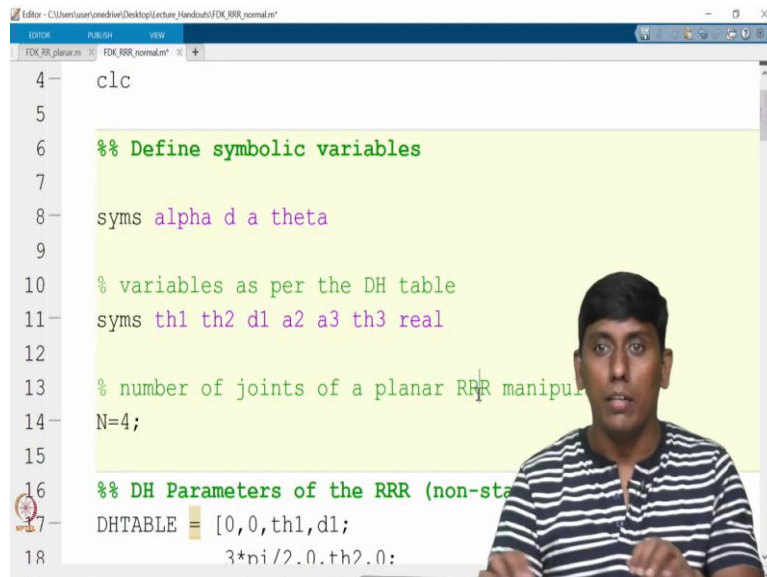
```
100
101- w0 = [0;0;0];
102- w1 = R01'*(w0) + [0;0;thldot];
103- w2 = R12'*(w1) + [0;0;th2dot];
104- w3 = R23'*(w2) ;
105
106 % end-effector angular velocities w.r.t. base
107- w03 = R01*R12*R23 * w3;
108
109 %% Linear velocity propagation
110
111- v0 = [0;0;0];
112- v1 = R01'*(v0+cross(w0,P01));
113- v2 = R12'*(v1+cross(w1,P12));
```

So, instead of this will omega 1 omega 2 omega 3, I can put a for loop, so where I can for loop will start from i equal to you can say 1 to n and inside I would say that whether it is a rotary joint or prismatic joint I can put a condition. So, based on that I can iterate and then I

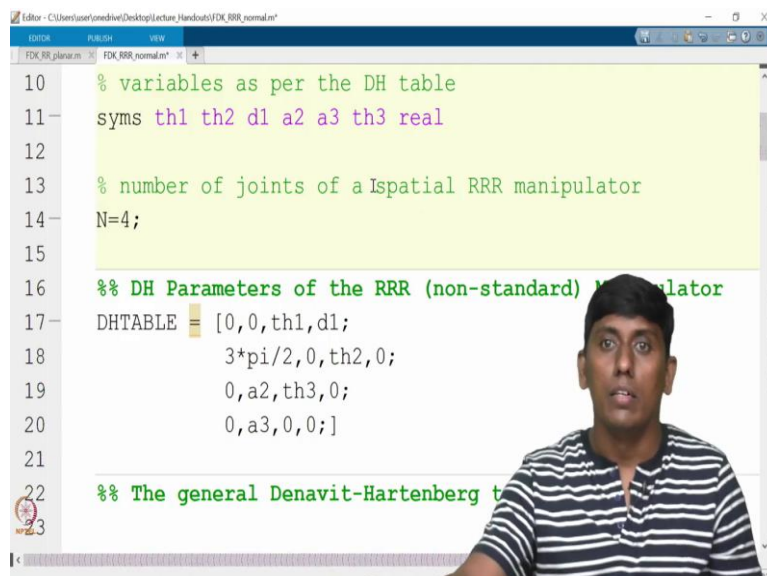
can do it, but that would give a confusing end to you. So, I have returned in a propagated way.

Some people will say that say it is a recursive one, why you have done in a sequence just to give a clarity, the same thing can be realized in you can see other one, other one in the sense we can go another example.

(Refer Slide Time: 16:18)



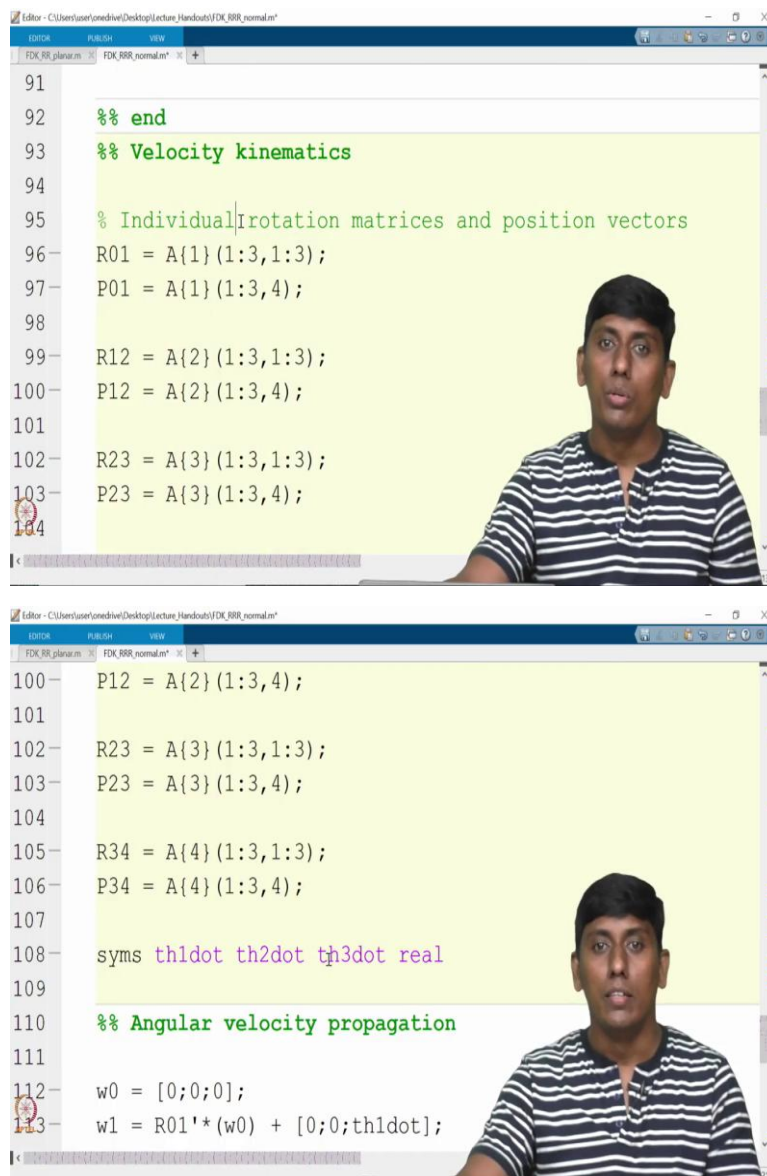
```
4- clc
5-
6- %% Define symbolic variables
7-
8- syms alpha d a theta
9-
10- % variables as per the DH table
11- syms th1 th2 d1 a2 a3 th3 real
12-
13- % number of joints of a planar RRR manipulator
14- N=4;
15-
16- %% DH Parameters of the RRR (non-standard) Manipulator
17- DHTABLE = [0,0,th1,d1;
18-            3*pi/2,0,th2,0;
```



```
10- % variables as per the DH table
11- syms th1 th2 d1 a2 a3 th3 real
12-
13- % number of joints of a spatial RRR manipulator
14- N=4;
15-
16- %% DH Parameters of the RRR (non-standard) Manipulator
17- DHTABLE = [0,0,th1,d1;
18-            3*pi/2,0,th2,0;
19-            0,a2,th3,0;
20-            0,a3,0,0;]
21-
22- %% The general Denavit-Hartenberg table
```

You can see that is also like seen in the forward kinematic model or direct kinematic model where the three R serial robot, we have seen, this is R R R. So, this serial robot we have seen it is spatial. So, this robot we have seen this is a DH table and these all we have seen in your previous simulation you can say lecture.

(Refer Slide Time: 16:47)



The image shows two screenshots of a MATLAB script editor. The top screenshot displays lines 91 to 104 of the script, which define rotation matrices and position vectors for the first three frames. The bottom screenshot displays lines 100 to 113, continuing the script with the fourth frame and defining the angular velocity propagation.

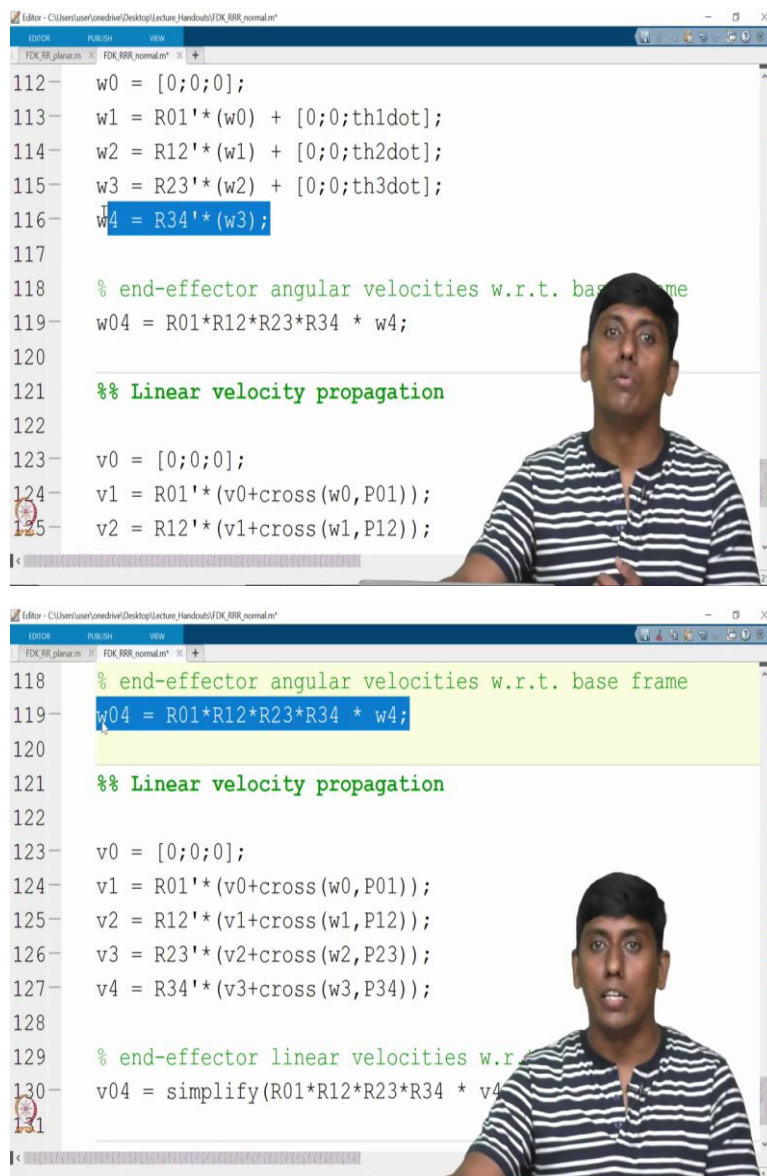
```
91
92 %% end
93 %% Velocity kinematics
94
95 % Individual rotation matrices and position vectors
96 R01 = A{1} (1:3,1:3);
97 P01 = A{1} (1:3,4);
98
99 R12 = A{2} (1:3,1:3);
100 P12 = A{2} (1:3,4);
101
102 R23 = A{3} (1:3,1:3);
103 P23 = A{3} (1:3,4);
104

100 P12 = A{2} (1:3,4);
101
102 R23 = A{3} (1:3,1:3);
103 P23 = A{3} (1:3,4);
104
105 R34 = A{4} (1:3,1:3);
106 P34 = A{4} (1:3,4);
107
108 syms th1dot th2dot th3dot real
109
110 %% Angular velocity propagation
111
112 w0 = [0;0;0];
113 w1 = R01'*(w0) + [0;0;th1dot];
```

So, now, we will start only from here you can see this is the same thing what only one thing is added the rotation matrix of the fourth frame and the position vector added. So, similarly, one additional active variable which is theta 3 dot that is added and here you can see that we have actually extended up to omega 4.



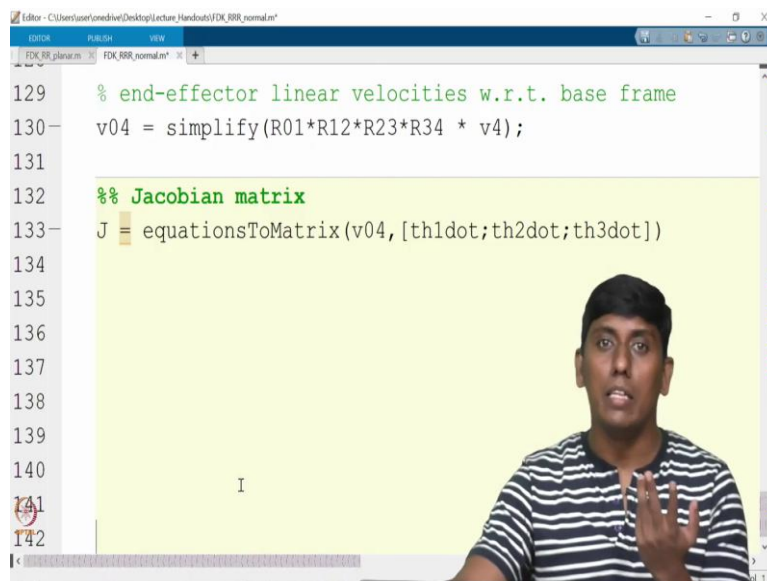
(Refer Slide Time: 17:06)



```
112- w0 = [0;0;0];
113- w1 = R01'*(w0) + [0;0;th1dot];
114- w2 = R12'*(w1) + [0;0;th2dot];
115- w3 = R23'*(w2) + [0;0;th3dot];
116- w4 = R34'*(w3);
117
118- % end-effector angular velocities w.r.t. base frame
119- w04 = R01*R12*R23*R34 * w4;
120
121- %% Linear velocity propagation
122
123- v0 = [0;0;0];
124- v1 = R01'*(v0+cross(w0,P01));
125- v2 = R12'*(v1+cross(w1,P12));
126- v3 = R23'*(v2+cross(w2,P23));
127- v4 = R34'*(v3+cross(w3,P34));
128
129- % end-effector linear velocities w.r.t. base frame
130- v04 = simplify(R01*R12*R23*R34 * v4);
131
```

So, since we have extended. So, this end effector velocity is  $\omega_4$  with respect to 0 that is what with respect to base frame. Similarly, the linear velocity is extended up to this. So, this also we can extend. Since we know this method also giving the same result. So, I did not like what you call did the second method.

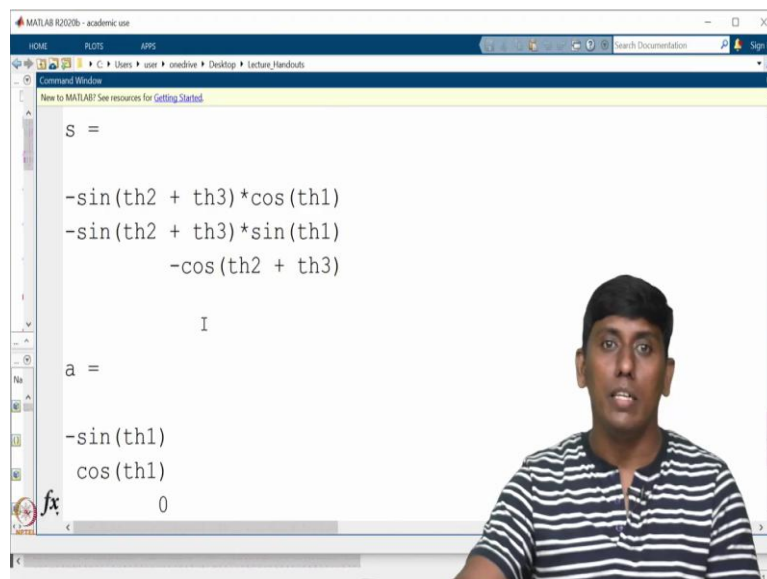
(Refer Slide Time: 17:32)



```
129 % end-effector linear velocities w.r.t. base frame
130 v04 = simplify(R01*R12*R23*R34 * v4);
131
132 %% Jacobian matrix
133 J = equationsToMatrix(v04, [th1dot;th2dot;th3dot])
134
135
136
137
138
139
140
141
142
```

The J I calculated in the earlier code that is I did not do it. So, now if I ran this obviously this will give you the right result you can also like verify, I will be sure I will be sharing these codes in your lecture material. So, you can find it.

(Refer Slide Time: 17:50)



```
Command Window
New to MATLAB? See resources for Getting Started

s =

-sin(th2 + th3)*cos(th1)
-sin(th2 + th3)*sin(th1)
-cos(th2 + th3)

I

a =

-sin(th1)
cos(th1)
0
```

```

MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
cos(th1)*(a3*cos(th2 + th3) + a2*cos(th2))
sin(th1)*(a3*cos(th2 + th3) + a2*cos(th2))
d1 - a3*sin(th2 + th3) - a2*sin(th2)

n =
      I
cos(th2 + th3)*cos(th1)
cos(th2 + th3)*sin(th1)
-sin(th2 + th3)

```

```

MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
-sin(th1)
cos(th1)
0

J =
      I
[-sin(th1)*(a3*cos(th2 + th3) + a2*cos(th2))
cos(th1)*(a3*cos(th2 + th3) + a2*cos(th2)),
]
fx >>

```

So, now this was run, and you can see these are you can say you are a normal sliding and approach vector and this is what the Jacobian.

(Refer Slide Time: 17:58)

```
MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
[ cos(th1)*(a3*cos(th2 + th3) + a2*cos(th2)), a2*cos(th2 + th1)
[
0,
>> w1
w1 =
I
0
0
th1dot
fx >>
```

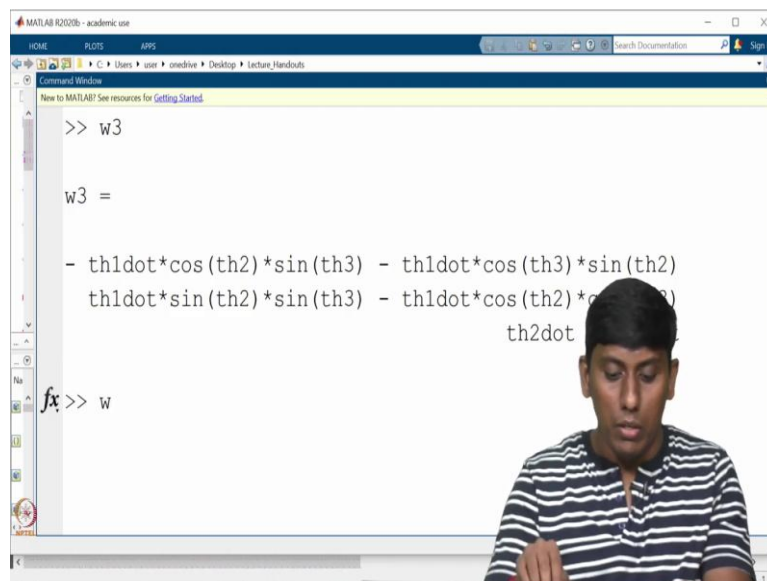
```
MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
0
th1dot
>> w2
w2 =
I
-th1dot*sin(th2)
-th1dot*cos(th2)
th2dot
fx >>
```

```
MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
-th1dot*cos(th2)
th2dot
>> w3
w3 =
- th1dot*cos(th2)*sin(th3) - th1dot*cos(th3)
th1dot*sin(th2)*sin(th3) - th1dot*cos(th2)*sin(th3)
th2dot
fx >> cl
```

And you can see this is what omega 1 and this is the omega 2. So, here it is the first joint is vertical rotary and the second joint is planar in the y z plane, or you can say x z plane whatever plane you consider.

So, in that sense you can see that there is the angular velocity, this is theta 1 dot here, but when it comes to this axis, so it would be having a transformation that is what you can see it from this result. So, similarly omega 3 it would be parallel. So, you can see that just added.

(Refer Slide Time: 18:33)

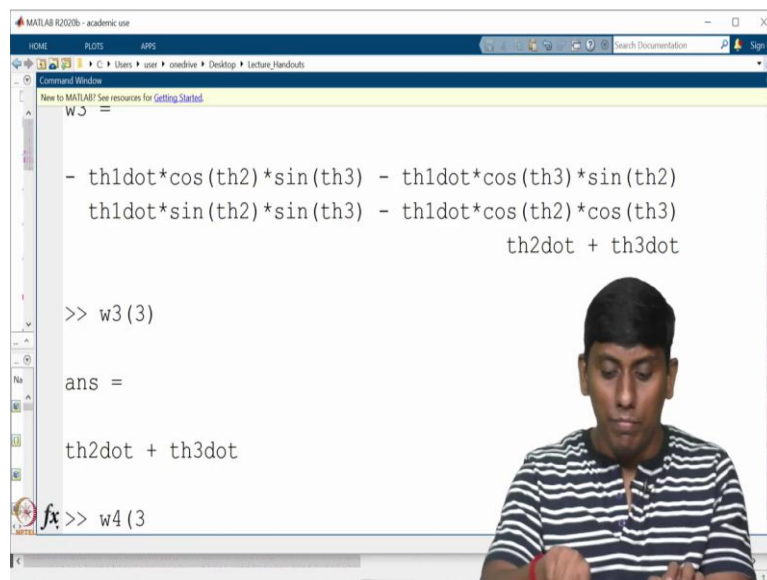


```

MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
>> w3

w3 =

- th1dot*cos(th2)*sin(th3) - th1dot*cos(th3)*sin(th2)
  th1dot*sin(th2)*sin(th3) - th1dot*cos(th2)*cos(th3)
  th2dot
fx >> w
  
```



```

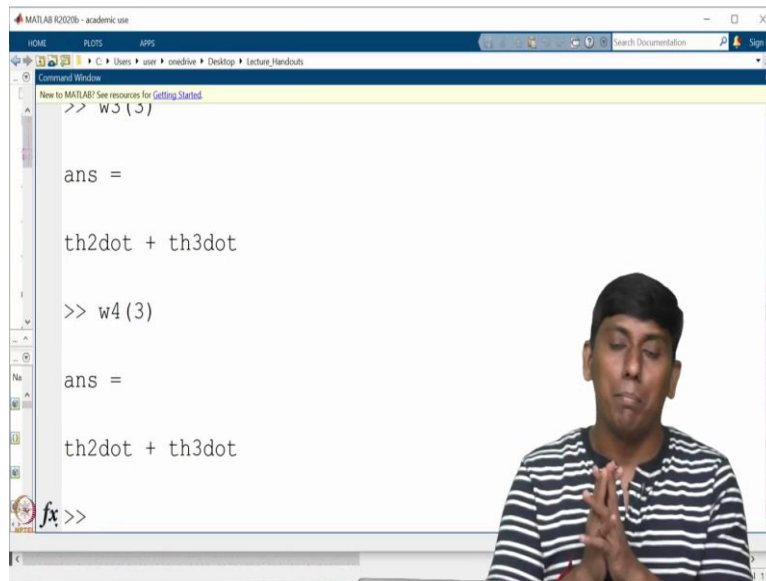
MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
w3 =

- th1dot*cos(th2)*sin(th3) - th1dot*cos(th3)*sin(th2)
  th1dot*sin(th2)*sin(th3) - th1dot*cos(th2)*cos(th3)
  th2dot + th3dot

>> w3(3)

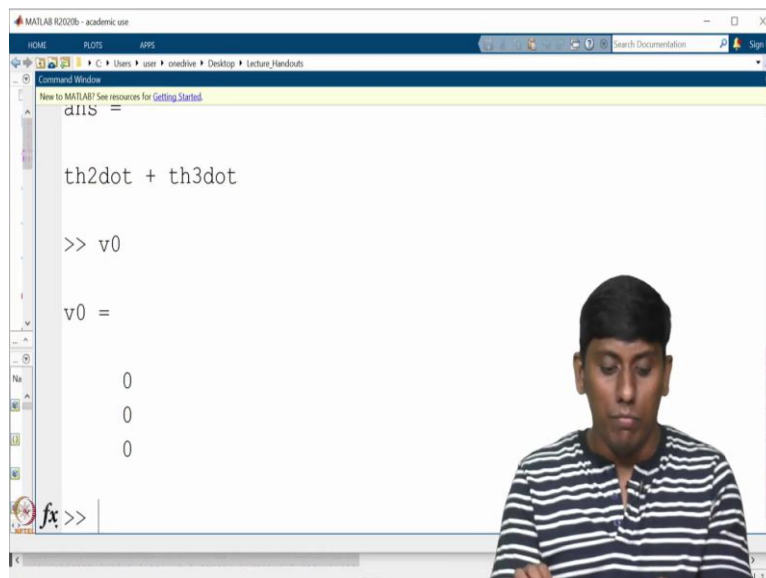
ans =

th2dot + th3dot
fx >> w4(3)
  
```



And the; what you call the coordinate would be added. So, that is it. If you look at it the third component of omega 3 you can see it is theta 2 dot plus theta 3 dot why it is, so because z axis are parallel for 2 and 3. So, the same sense if you go for omega 4 you can say 3, so that is also same because the third frame and fourth frame are the same link and the fourth frame is not having any active joint and we have assumed that both axis are parallel.

(Refer Slide Time: 19:07)



MATLAB R2020b - academic use

HOME PLOTS APPS

C:\Users\user\onedrive\Desktop\Lecture\_Handouts

Command Window

New to MATLAB? See resources for Getting Started


```
U
0
0

>> v1

v1 =

0
0
0

fx >>
```



MATLAB R2020b - academic use

HOME PLOTS APPS

C:\Users\user\onedrive\Desktop\Lecture\_Handouts

Command Window

New to MATLAB? See resources for Getting Started


```
U
0
0

>> v2

v2 =

0
0
0

fx >>
```



MATLAB R2020b - academic use

HOME PLOTS APPS

C:\Users\user\onedrive\Desktop\Lecture\_Handouts

Command Window

New to MATLAB? See resources for Getting Started


```
U
0
0

>> v3

v3 =

a2*th2dot*sin(th3)
a2*th2dot*cos(th3)
a2*th1dot*cos(th2)

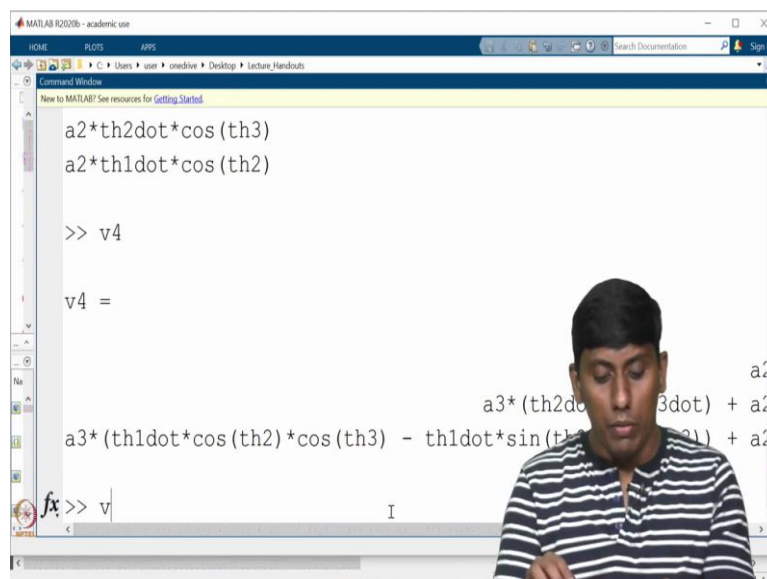
fx >>
```



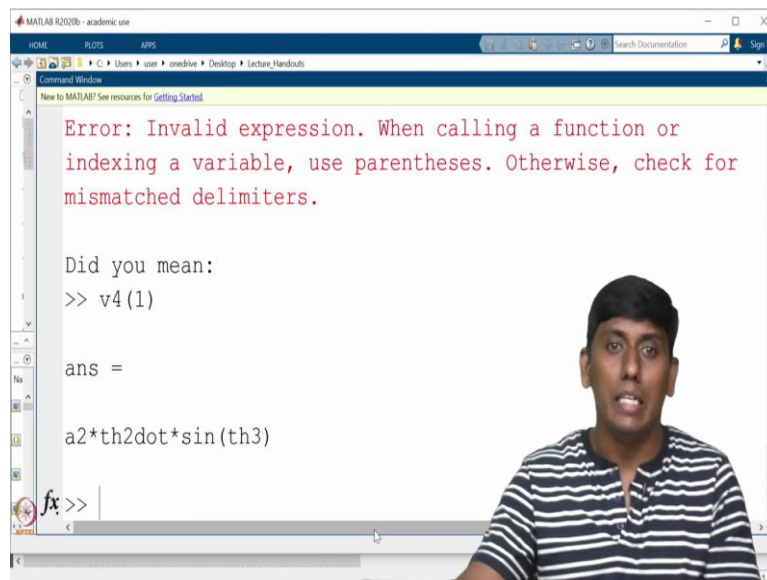
In the same sense we can start we assume 0, we because both are same point and v2 you can see 0, because in this case if you recall R R R, three R manipulator. So, the first axis comes here, second axis goes inside the you can say picture or inside the screen, in the sense 0 1 2 3 all having in the same line, especially 1 and 2 are having the same point 0 and 1 are D1 distance but the D1 distance is constant. So, there is no linear velocity happening at you can 0 1 and 2 that is why you can see up to v2 it is having 0.

When you come to the third. So, v3 you can see like that tangential velocity is coming, here only tangential component will come because we have considered all are rotary joint. So probably one additional example I may put it somewhere in the middle. So, you can see if the prismatic joint comes out it will come. Anyway, in Dynamics we will be seeing that in a prismatic. So, we can try.

(Refer Slide Time: 20:14)





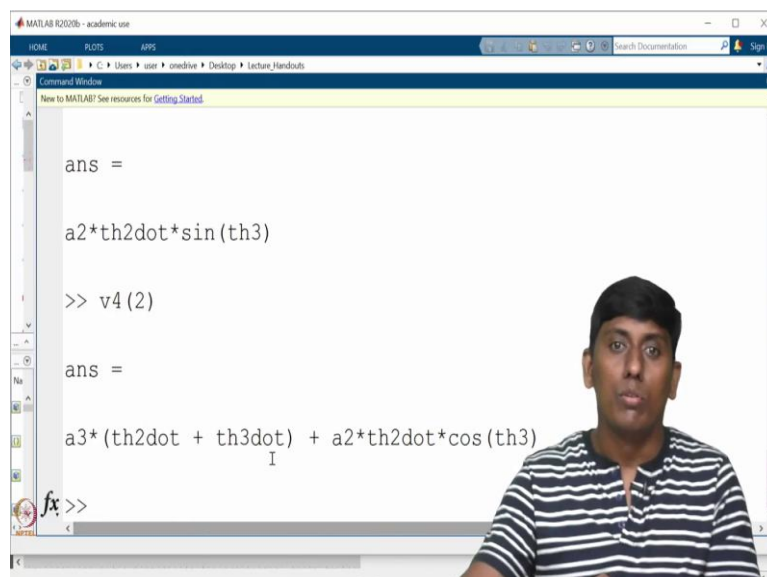


```
MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started
Error: Invalid expression. When calling a function or indexing a variable, use parentheses. Otherwise, check for mismatched delimiters.

Did you mean:
>> v4(1)

ans =

a2*th2dot*sin(th3)
fx >>
```



```
MATLAB R2020b - academic use
HOME PLOTS APPS
C:\Users\user\onedrive\Desktop\Lecture_Handouts
Command Window
New to MATLAB? See resources for Getting Started

ans =

a2*th2dot*sin(th3)

>> v4(2)

ans =

a3*(th2dot + th3dot) + a2*th2dot*cos(th3)
fx >>
```

So, similarly v4 you can see, So, v4 would be lengthy because it is having a connection, So, v4 of 1, v4 of 1 you can see this is, so v4 of 2 you can see it that way. So, like that you can see it would be expanded.

(Refer Slide Time: 20:34)

```

a3*(th2dot + th3dot) + a2*th2dot*cos(th3)

>> J

J =

[-sin(th1)*(a3*cos(th2 + th3) + a2*cos(th2)), th2 + th3]
[ cos(th1)*(a3*cos(th2 + th3) + a2*cos(th2)), th2 + th3]
[
fx>>

```

```

:th2 + th3)*cos(th1)*sin(th3) - sin(th2 + th3)*(a3 +
:th2 + th3)*sin(th1)*sin(th3) - sin(th2 + th3)*sin(th1)*(a3 +
- a3*cos(th3) -
fx>>

```

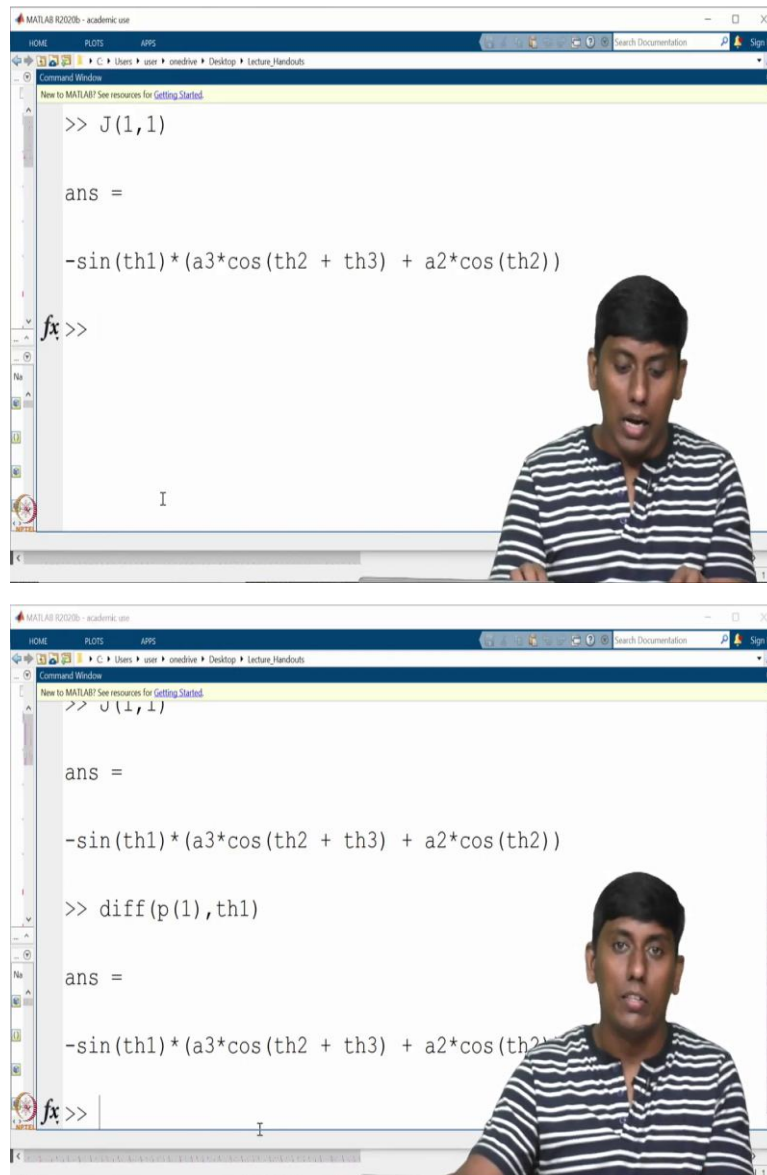
```

3)*cos(th1)*(a3 + a2*cos(th3)), -a3*sin(th2 + th3)*sin(th1)]
3)*sin(th1)*(a3 + a2*cos(th3)), -a3*sin(th2 + th3)*cos(th1)]
3*cos(th2 + th3) - a2*cos(th2), -a2*cos(th2 + th3)]
fx>>

```

And now, you can find the J, so J would be small extension of the other one, this can be obtained with the same thing.

(Refer Slide Time: 20:51)



So, for example, J of 1 comma 1 I am just showing it. So, I will just make it this. So, J of 1 comma 1 and differentiation of, so P of 1, comma so theta 1 would be the same. So, what I did not simplify, so that is why it is coming, slightly different, but you can see like, these two are same. So, that is what we can see it. So, in that sense, I am ending this particular lecture.

So, where have seen like how to do the velocity propagation model of a serial manipulator with the help of MATLAB. So, this code is very generic, you can just you can say put it as long as you have a direct kinematic model which you have written in MATLAB, it is simple

extension, if not, you should have a rotation matrices and position vectors in your hand and then you can it make the propagation model.

So, either way you can do it, but I already did the direct kinematic model in MATLAB I just extended, otherwise you have to write the rotation matrix then there. So, that is the only thing and please make sure, so when you are calculating  $\omega_{i+1}$  with respect to  $i+1$  frame, so the rotation matrix is that you can say inverse are transpose of the normal matrix in the sense.

So, when you calculate  $\omega_1$ , so the rotation matrix is  $R_{01}^T$ , or you have to write  $R_{10}$ . That is, we do not have, so we are taking a transpose of  $R_{01}$ . So, that make it clear. So, with that, we are ending this particular lecture. The next lecture would be talking about the statics, and you can say singularity of the serial manipulator. So, we can see until then, thank you. Bye.