

Wheeled Mobile Robots
Prof. Santhakumar Mohan
Department of Mechanical Engineering
Indian Institute of Technology, Palakkad

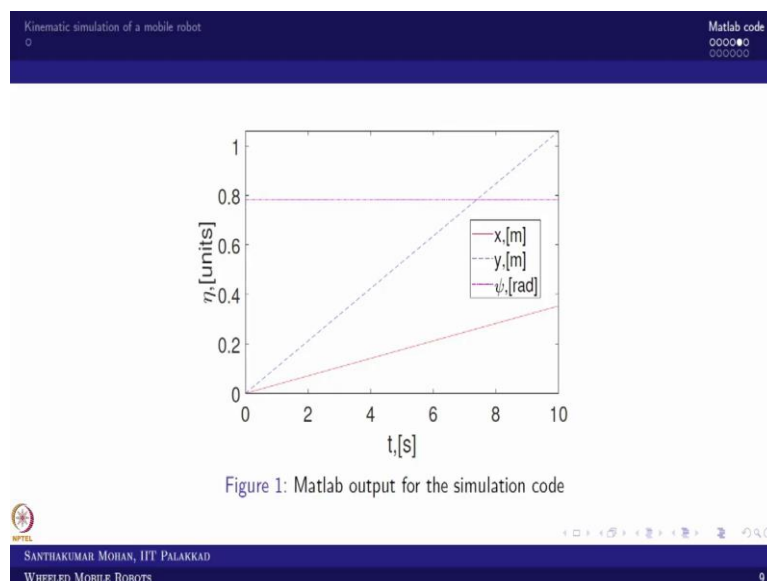
Lecture – 06
Kinematic Simulation and Motion Animation of A Mobile Robot (Land-based)

Welcome back, to the course on Wheeled Mobile Robot; this is lecture 6. You know lecture 5 we were talking about Kinematics Simulation of a Mobile Robot. In specific the land based mobile robot kinematics we have taken, and we have tried to simulate it with the help of Euler method. So, this particular lecture 6 we are going to talk about motion animation. So, why this is motion animation required with the help of the previous code.

And then how we will do the motion and animation that is what we are going to cover in this particular lecture. So, in that sense you can see that. So, I put two things so, the motion animation is the core for this particular lecture 6. So, for understanding this I am just giving you a very light way. So, we have used the Euler method in this form. So, based on that what we know the position propagation model is available.

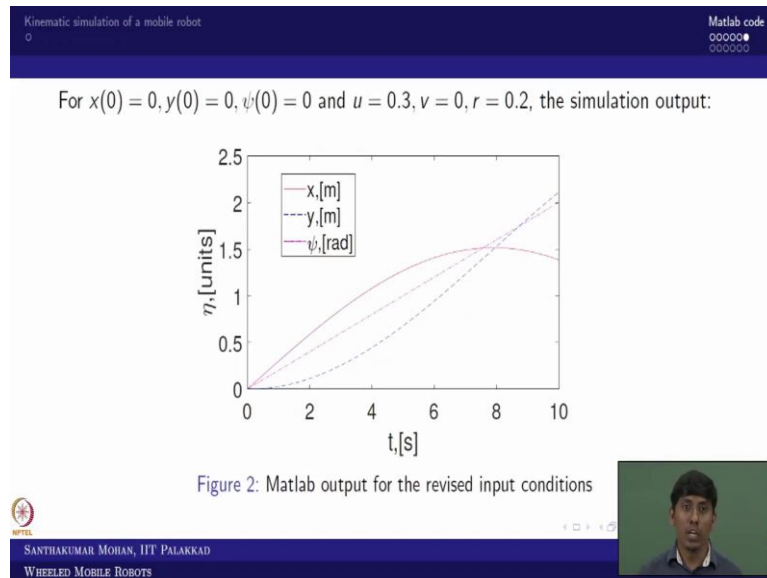
So, based on the position propagation model, we have used the already known differential kinematic model so we have actually like derive it. So, what we did? This is the code we did it in the last class. So, lecture 5 if you refer; so this simulation we have taken. So, now I am actually like in showing it here. So, this is the plot which we have done in the last class.

(Refer Slide Time: 01:19)



Where we have taken x , y and Ψ ; so, this particular plot is actually giving little more information, but what we wanted is slightly a different way. So, when I want to see the simulation result, if it gives in the exact vehicle orientation and vehicle motion that would be more beneficial.

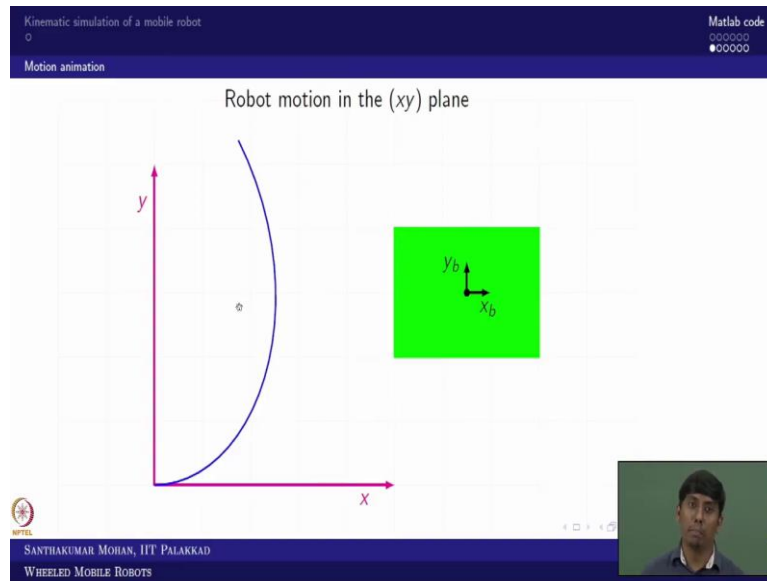
(Refer Slide Time: 01:40)



For example so, I am showing that the condition I have changed where the initial condition of that previous code in lecture 5 what I covered. So, I put the initial conditions all are 0, but I am giving the input instead of a constant input, I am actually giving something slightly varied, where u is 0.3 and there is a rotational angular velocity which is r as given as 0.2 radian/second.

So, now the simulation is giving this particular result, where x is actually like a slightly getting bend, but we do not know what the nature. And y also like getting curved its look like a parabola, and r is giving the ψ in the sense ψ would be gradually increasing. These all actually like getting, but you will see this code and this result, this result is not giving any fruitful information.

(Refer Slide Time: 02:30)

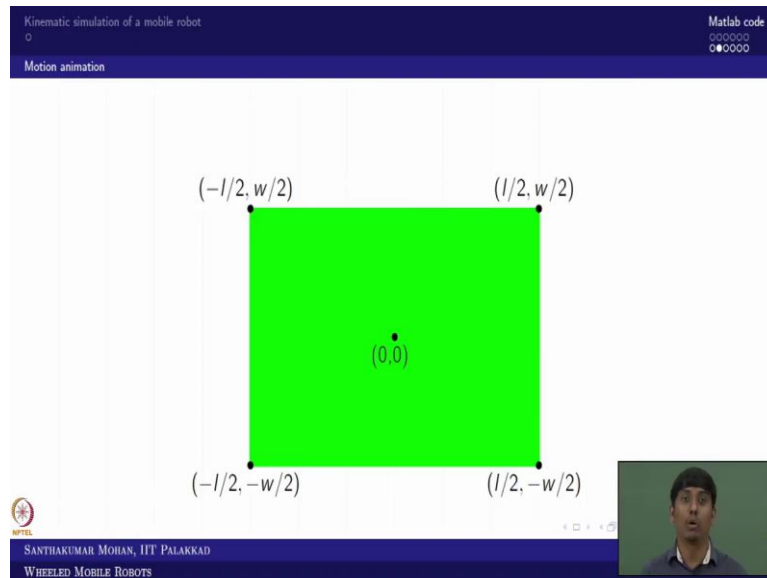


Now, then one can ask can I actually like plot x versus y ? Yes you can do, but still I am not getting the picture of what the ψ is happening. So, how the vehicle is actually like orient or whether the vehicle is actually like rotating about itself, these all not informatively given here right. So, then what one can see if I incorporate one of the block, which I call as a vehicle you know like vehicle we started with a small you call blue color block right.

The similar way I am trying to incorporate in animation as a green color block. So, now, this block can be actually like fit it into the x y plane. And I can show as a you can say image overlap I can show its as the animation right. So, this is what usually we are doing it. So now, you can record it as a video.

Now, how we can do this? So, far that what one can know so, the vehicle frame you know; which is we call b the point b that would be having a frame of x_b and y_b . Now, if I want to know this particular coordinate I already have it right. So, you see already this x y is there, but what I wanted I wanted this vehicle moving on this x y plane with the orientation.

(Refer Slide Time: 03:40)



So, in that sense what one can should know. So, one can should know that the coordinates of this point. So, I am taking as a general where the b point, which is actually like the body frame I assume as a origin. So, now, in that sense if the vehicle length I call l , and the vehicle width I call w , I can actually like bring the coordinates of these 4 corners in this form ok.

So, this is actually like very simple right. So, now, if I know this what additionally I can brought. So, now, if I actually like put x is actually like added in everywhere in the x coordinate what I will give the vehicle will actually like move in x , if I put y of the value which I obtained in the previous animation or you can just say simulation code, that will be added with all the y coordinate that will be taking the y addition right.

(Refer Slide Time: 04:30)

Kinematic simulation of a mobile robot
Motion animation

Matlab code

$$x'_1 = x_1 \cos \psi - y_1 \sin \psi$$

$$y'_1 = x_1 \sin \psi + y_1 \cos \psi$$

$$\begin{bmatrix} x'_1 \\ y'_1 \end{bmatrix} = \begin{bmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

SANTHAKUMAR MOHAN, IIT PALAKKAD
WHEELED MOBILE ROBOTS

But, what one supposed to know the vehicle is orient also right. So, in that the sense one suppose to understand, if you take a random point of x_1 and y_1 with the help of $(0, 0)$. So, now, you can see that this particular point I can draw a line.

So, now, this point is actually like you can say associated with the frame, which is x_b and y_b . Now, this frame itself is rotating in the sense the point x_1 and y_1 is part of this body that body is rotated. How that can be visualized? I can visualize with the small you can say picture here.

You can see that now, the frame b we rotated about z axis as a ψ angel. So, now, what would be the x_b ? x_b become x_b' and y_b become y_b' . Now, what happen to the point? The point x_1 and y_1 also part of this body right. So, now, that is also rotated. So, that is what happened? Now, if you look at this. So, what happened, this x_1 and y_1 with the respect to b frame is not changed. But, with respect to the original frame which we call initial frame it is getting changed right. So, how that can be realized?

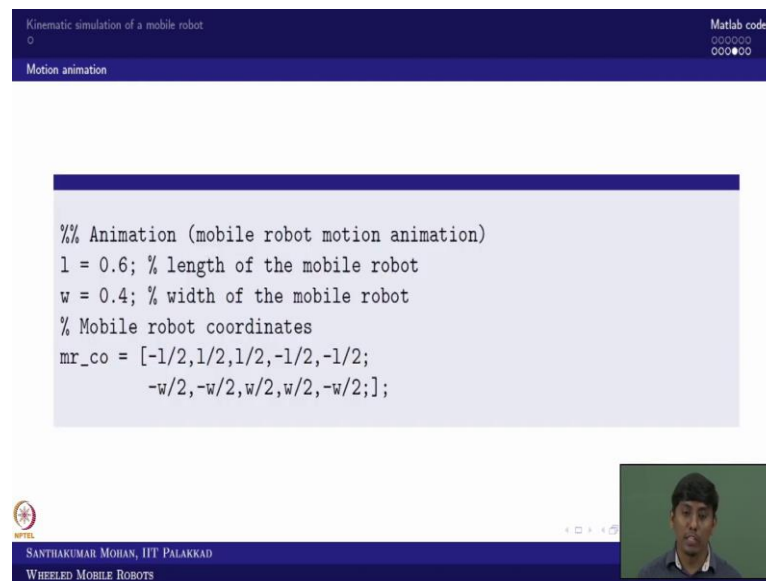
So, you can actually like projection you can use so, based on the projection what you can write. So, along with you call x_1 direction which is actually like x_b' direction which is I am assuming that that is parallel to x_b original. So, what I can right? So, that would be projected as $x_1 \cos(\psi) - y_1 \sin(\psi)$. So, how that can be done? You project this point. So, what it would give? It would actually like give two component right.

So, based on that component you can actually like get x'_1 and y'_1 . So, now, these are the coordinates which is actually like modified based on the ψ angle now, how I can actually like

rewrite the in a simplest form. So, one easiest way you see left hand side I can keep it x_1' and y_1' as a vector. So, the right hand side I can take one matrix and multiply with x_1 and y_1 .

So, in the sense what one can see it is a simple transformation matrix. So, this transformation matrix if I know, I can actually like easily portrait all the points and I can actually like start do the animation.

(Refer Slide Time: 06:43)



```
%% Animation (mobile robot motion animation)
l = 0.6; % length of the mobile robot
w = 0.4; % width of the mobile robot
% Mobile robot coordinates
mr_co = [-l/2,l/2,l/2,-l/2,-l/2;
        -w/2,-w/2,w/2,w/2,-w/2];
```

The screenshot shows a MATLAB code editor window. The title bar reads "Kinematic simulation of a mobile robot" and "Motion animation". The code defines the length (l = 0.6) and width (w = 0.4) of the mobile robot, and sets the mobile robot coordinates (mr_co) as a 2x5 matrix. The bottom of the window shows the name "SANTHAKUMAR MOHAN, IIT PALAKKAD" and "WHEELED MOBILE ROBOTS".

So, for that what we can actually like see, we are trying to do this incorporation in the MATLAB code. So, this is what we are actually trying to do, for animation purpose I am bringing the vehicle dimension. The vehicle length is actually like 0.6 meter and the width is actually 0.4 meter.

So, now you can see that the mobile robot coordinate, I can bring as mr_co which is nothing, but the mobile coordinates. So, which is I have written as what we did in the earlier slide, probably slide number 10 or something. So, you can refer it. So, now, this coordinate is incorporated what one can see. So, this is actually like where your x coordinate and y coordinate and the Ψ angle is actually getting change, the array would be length of t right.

(Refer Slide Time: 07:28)

```

Kinematic simulation of a mobile robot
Motion animation
Matlab code

figure
for i = 1:length(t) % animation starts here
    psi = eta(3,i);
    R_psi = [cos(psi),-sin(psi);
             sin(psi), cos(psi)]; % rotation matrix
    v_pos = R_psi*mr_co;
    fill(v_pos(1,:)+eta(1,i),v_pos(2,:)+eta(2,i),'g')
    hold on, grid on; axis([-1 3 -1 3]), axis square
    plot(eta(1,1:i),eta(2,1:i),'b-');
    legend('MR','Path'), set(gca,'fontsize',24)
    xlabel('x, [m]'); ylabel('y, [m]');
    pause(0.1); hold off
end % animation ends here

```

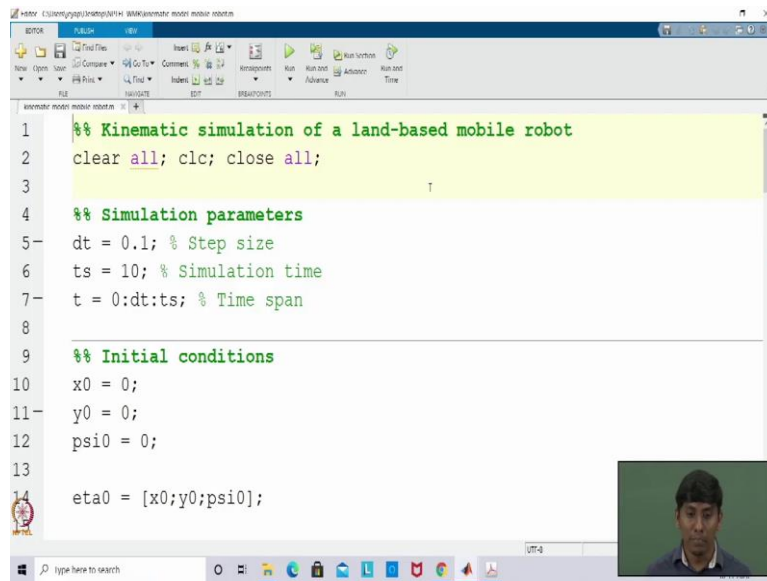
SANTHAKUMAR MOHAN, IIT PALAKKAD
WHEELED MOBILE ROBOTS

So, now that if I know I can actually like put in a for loop, where I can actually like run the loop from 1 to length of t, where I can say that this is animation start. If I know the psi the psi is nothing, but the third, you can say entity of the eta vector which is I am writing psi as eta of (3, 1) or in the sense it is i^{th} instant. So, now, I brought the you call the transformation matrix which I obtained in the you call slide number 13, where the rotation matrix I have incorporated that matrix I brought in.

So, now what I know already the mobile robot coordinate I know which is not rotated. But, now I brought that into rotation matrix. Now, what happen? The vehicle position is actually like rotated. Now, this vehicle position added with you can say x coordinate added with x value which is nothing, but eta of first component and, the y component which is v position of 2, which I added with second component, which is eta of second component which is (2, i).

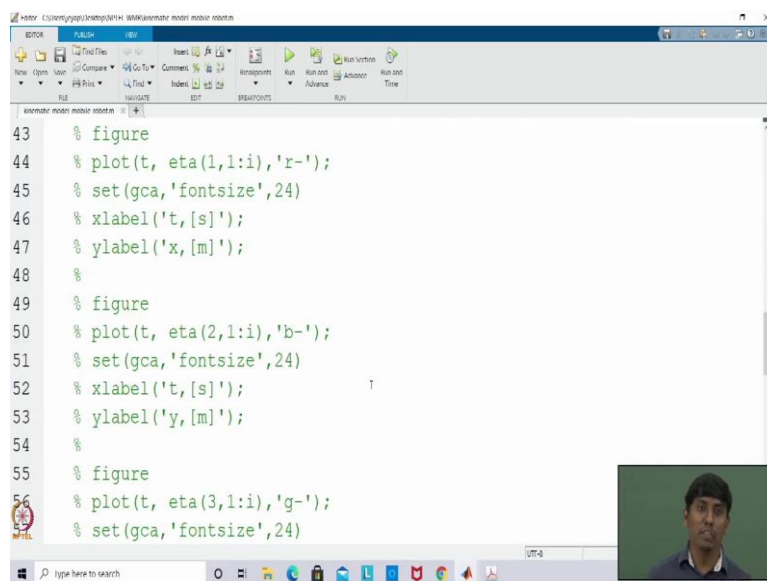
So, now what it will give? It will give a fill on the plot as a green patch. So, that green patch would be actually like try to show. Now, I want to show the profile or the path of the vehicle. So, I am plotting the eta (1, eta (2)) which is actually like I am plotting as up to i^{th} instant. So, now, you can see this in the MATLAB code. So, for that I am actually like moving into a MATLAB.

(Refer Slide Time: 08:52)



```
1 %% Kinematic simulation of a land-based mobile robot
2 clear all; clc; close all;
3
4 %% Simulation parameters
5 dt = 0.1; % Step size
6 ts = 10; % Simulation time
7 t = 0:dt:ts; % Time span
8
9 %% Initial conditions
10 x0 = 0;
11 y0 = 0;
12 psi0 = 0;
13
14 eta0 = [x0;y0;psi0];
```

(Refer Slide Time: 08:54)



```
43 % figure
44 % plot(t, eta(1,1:i),'r-');
45 % set(gca,'fontsize',24)
46 % xlabel('t,[s]');
47 % ylabel('x,[m]');
48 %
49 % figure
50 % plot(t, eta(2,1:i),'b-');
51 % set(gca,'fontsize',24)
52 % xlabel('t,[s]');
53 % ylabel('y,[m]');
54 %
55 % figure
56 % plot(t, eta(3,1:i),'g-');
57 % set(gca,'fontsize',24)
```

So, you can see that these are the things which we did in the lecture 5.

(Refer Slide Time: 08:57)

```
70
71 %% Animation (mobile robot motion animation)
72 l = 0.6; % length of the mobile robot
73 w = 0.4; % width of the mobile robot
74 % Mobile robot coordinates
75 mr_co = [-l/2,l/2,l/2,-l/2,-l/2;
76          -w/2,-w/2,w/2,w/2,-w/2];
77 figure
78 for i = 1:length(t) % animation starts here
79     psi = eta(3,i);
80     R_psi = [cos(psi),-sin(psi);
81             sin(psi), cos(psi)]; % rotation matrix
82     v_pos = R_psi*mr_co;
83     fill(v_pos(1,:)+eta(1,i),v_pos(2,:)+eta(2,i),'g')
84     hold on, grid on
```

So, now, what we added? So, these are the content we have added. So, where l w I have added as a length of the mobile robot and width of the mobile robot. And then I have added the mobile robot coordinate in this form. So, I said I want to actually like animate. So, in the sense I will be plotting sequence of image overlap one to another with passing the time right.

So, in that sense I am actually holding every image for few second or few millisecond, then I will be overlap one image to another image. So, for that what I am trying to do? First I am rotating the original you can say mobile robot coordinate with the help of rotation matrix.

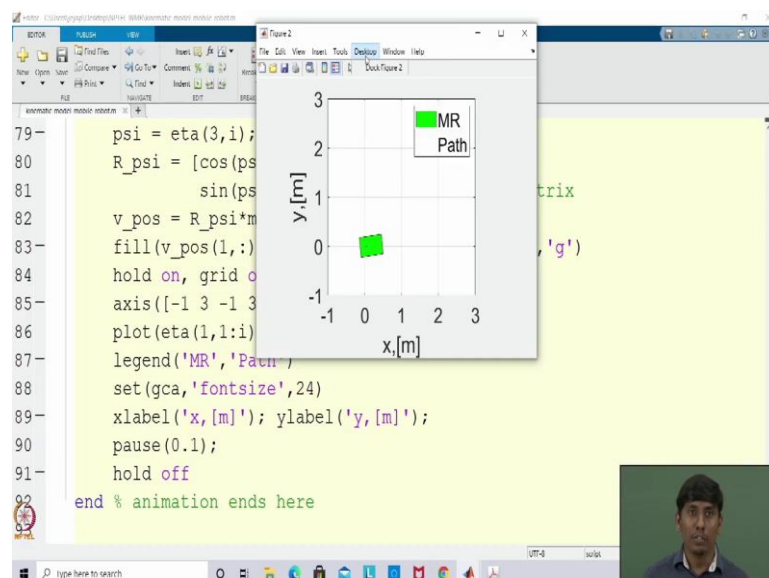
(Refer Slide Time: 09:37)

```
79     psi = eta(3,i);
80     R_psi = [cos(psi),-sin(psi);
81             sin(psi), cos(psi)]; % rotation matrix
82     v_pos = R_psi*mr_co;
83     fill(v_pos(1,:)+eta(1,i),v_pos(2,:)+eta(2,i),'g')
84     hold on, grid on
85     axis([-1 3 -1 3]), axis square
86     plot(eta(1,1:i),eta(2,1:i),'b-');
87     legend('MR','Path')
88     set(gca,'fontsize',24)
89     xlabel('x, [m]'); ylabel('y, [m]');
90     pause(0.1);
91     hold off
92 end % animation ends here
```

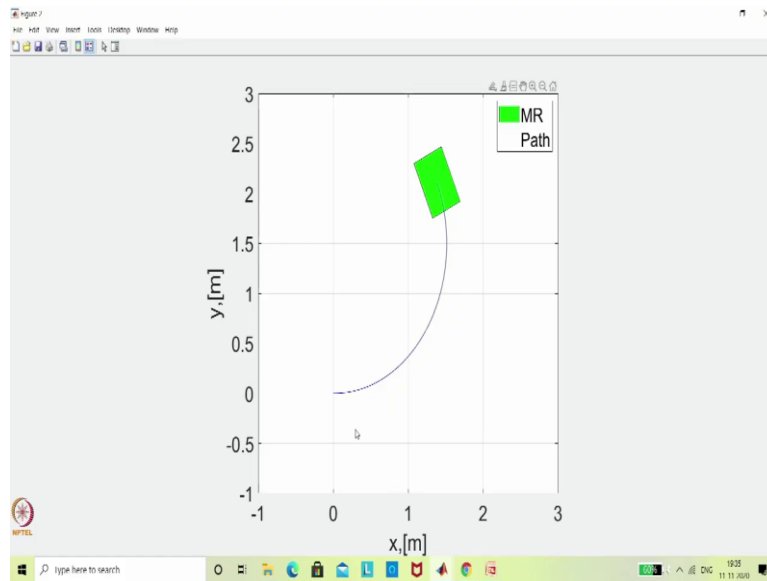
Then, I am actually like a filling as a patch. So, this is a simple MATLAB command where this is actually like x position and this is y position and this is the color. So, g means green. So, now what I am trying to do? I am actually like trying to show this. So, for that what I am trying to show you see I put a pass.

So, pass 0.1 means I am passing every segment as 100 millisecond, I am holding off why because every image need not to be super imposed with the previous image. So, in the sense hold on is required for initial, but hold off is required for making a animation as a moving. So, if I run this code you can easily realize what going to happen. So, you can see here. So, I hope you are able to see in few second, it takes time let see yeah.

(Refer Slide Time: 10:29)



(Refer Slide Time: 10:36)



So, now, you can actually like able to see yeah. So, you can see like this. So, the vehicle which is actually like look like a green patch. So, that is actually like moving right. So, this is what we call vehicle animation.

Now, you can see that that based on the orientation of the vehicle it is moving, on top of that the vehicle frame also looking you can see coordinate as b right that b frame is the point, where the blue line is actually like ending. So, in the sense you can see that where the vehicle coordinate all about and how the vehicle is oriented.

So, these all actually like you can see in a single you can say picture, which is actually like nothing, but as a animation here. So, now, you compare to the previous result although you plotted x y plane or you plotted individual coordinate, it is not giving this you can say fruitful information right. So, that is what once should know. So, how to present the result in a betterment?

So, that is why this vehicle animation is important, in the sense the motion animation is important. Now, once you know this so the next lecture what you call lecture 7, where you are bringing the you can say wheel kinematic model then you can actually put the wheel here, and you can see the overall animation.

So, how individual wheel would be contributing you can easily realize for example, you have a differential wheel drive. So, when you have a left drive and right drive how it is dominating,

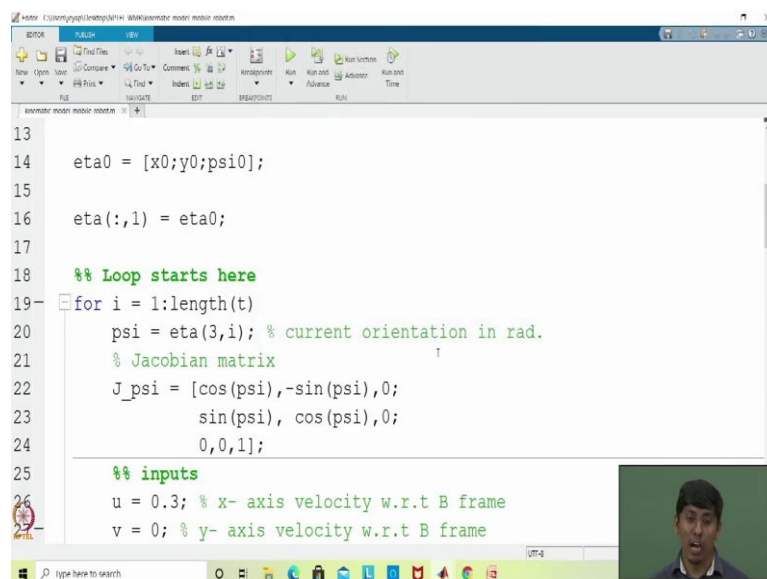
that you can see in the vehicle you can say motion animation rather than, just plotting time versus $x(t)$ $y(t)$ or time versus u and v that is not going to be helpful.

But, this motion animation would be very much helpful. I hope you got it how to do vehicle motion animation so, with that I am ending this particular lecture. So, for what we have seen is actually like forward kinematic model, along with the motion simulation. So, now, we will actually like see how this would be helpful if we do an inverse differential kinematic model taken place. In the sense the η desired and $\dot{\eta}$ desired is given.

So, how one can actually like simulate this particular system and how that would be different from the general controlling. Because, when I was defining what is forward and different you can say inverse differential kinematics, I said that forward differential kinematics equivalent to a simulating that we have done.

The inverse differential kinematic what we said as a controlling, but this is open loop control I already mentioned. So, now, we will actually like move to the MATLAB, you can see code what we have done so far.

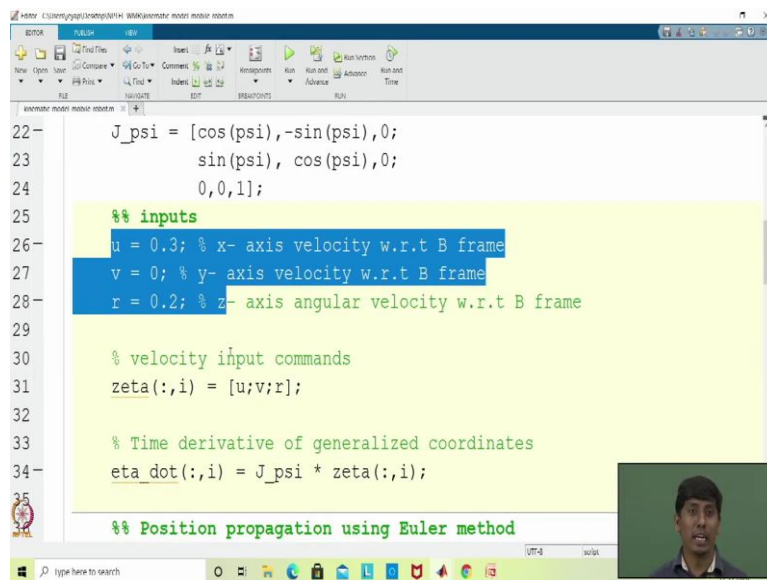
(Refer Slide Time: 12:58)



```
13
14 eta0 = [x0;y0;psi0];
15
16 eta(:,1) = eta0;
17
18 %% Loop starts here
19 for i = 1:length(t)
20     psi = eta(3,i); % current orientation in rad.
21     % Jacobian matrix
22     J_psi = [cos(psi), -sin(psi), 0;
23             sin(psi), cos(psi), 0;
24             0, 0, 1];
25
26     %% inputs
27     u = 0.3; % x- axis velocity w.r.t B frame
28     v = 0; % y- axis velocity w.r.t B frame
```

So, with that we will actually like change how that can be vary. For example, now I said that the η has given.

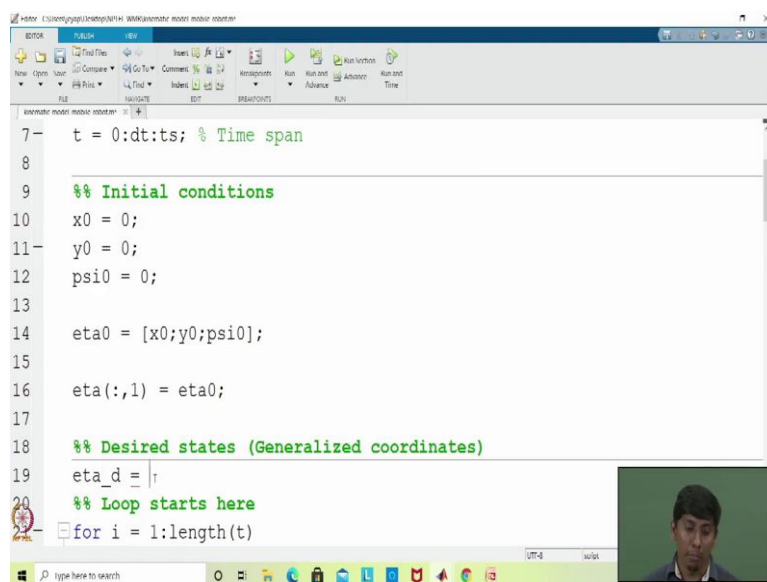
(Refer Slide Time: 13:04)



```
22-     J_psi = [cos(psi), -sin(psi), 0;
23-             sin(psi),  cos(psi), 0;
24-             0, 0, 1];
25-
26-     %% inputs
27-     u = 0.3; % x- axis velocity w.r.t B frame
28-     v = 0; % y- axis velocity w.r.t B frame
29-     r = 0.2; % z- axis angular velocity w.r.t B frame
30-
31-     % velocity input commands
32-     zeta(:,i) = [u;v;r];
33-
34-     % Time derivative of generalized coordinates
35-     eta_dot(:,i) = J_psi * zeta(:,i);
36-
37-     %% Position propagation using Euler method
```

But, what we have taken the inputs u v r is straightaway we have given as a user command. But, now this user command would go away what; that means, this input would be defined based on the you call η desired and $\dot{\eta}$ desired.

(Refer Slide Time: 13:24)



```
7-     t = 0:dt:ts; % Time span
8-
9-     %% Initial conditions
10-    x0 = 0;
11-    y0 = 0;
12-    psi0 = 0;
13-
14-    eta0 = [x0;y0;psi0];
15-
16-    eta(:,1) = eta0;
17-
18-    %% Desired states (Generalized coordinates)
19-    eta_d = |
20-    %% Loop starts here
21-    for i = 1:length(t)
```

So, for that what we supposed to know? I will actually like add a new thing, which is I called Desired parameter ok. So, that to like I would say Desired states. So, which I call generalize so, Generalized coordinates.

So, these are the things which we are going to give so; obviously, what you need to know the eta desired which I am putting as eta underscore d that would be a vector, but it can be a you call simple you call constant vector. But right now I am thinking of putting it as a you can say time varying. So, I am better I will put it this into a for loop itself, because I have already time inside.

(Refer Slide Time: 14:12)

```

25     0, 0, 1];
26     %% inputs
27     u = 0.3; % x- axis velocity w.r.t B frame
28     v = 0; % y- axis velocity w.r.t B frame
29     r = 0.2; % z- axis angular velocity w.r.t B frame
30     %%
31     % velocity input commands
32     zeta(:,i) = [u;v;r];
33
34     %% Desired states (Generalized coordinates)
35     eta_d = [2-2*cos(0.1*t(i)); 2*sin(0.1*t(i)); 0.1*t(i)];
36     eta_d_dot = [2*0.1*sin(0.1*t(i)); 2*0.1*cos(0.1*t(i)); 0.1];
37     zeta(:,i) = inv(J_psi) * eta_d_dot;
38     % Time derivative of generalized coordinates
39     eta_dot(:,i) = J_psi * zeta(:,i);

```

So, instead of this what I am actually like trying to put it so, that desired thing. So, this is desired state is coming here so, this would be as a time varying input. So, where I am assuming that this is actually like taking $2 \cos$ you can say $0.1 \times t_i$, where i^{th} instant of time we would be using.

And similarly the other one I am actually like probably I am taking it as $2 -$ so, that it would be easy for my further run ok. So, the other one is actually like I am saying $2 \sin$ of so, $0.1 \times t_i$. So, why I have taken $- 2 - 2$? Otherwise what happen my starting point which is actually like 0 in x axis, but in this case if I put t equal to 0 it suddenly start from 2.

So, in order to make it neutral so $2 - 2$ so, then you may ask $2 \cos$ of this $- 2$ also will work; yes that will work, but you know like the eta desire dot would be derivative of this, you know the cos derivative would be actually like minus sign. So, if I include the minus sign here that would be easy for further run that is what the whole idea. The vehicle would have a forward velocity rather than reverse velocity that is what the whole idea began bringing the you call $2 - 2$ ok.

So, now I assume that this would be simply 0.1 times of t ok. So, now, what you have derived? So, you gave eta desired which is nothing, but this is x desired this is y desired and this is psi desired. So, now, based on that what one can actually like find ex_desired_dot y_desired_dot all we can actually find. So, that I am putting as eta_d_dot I am writing as eta_d, that is desired underscore dot it says that it is time derivative. So, if I take the time derivative this would be $2 \times 0.1 \times \sin$ of so, 0.1 times of t right.

Desired states (Generalized coordinates)

```
eta_d = [2-2*cos(0.1*t(i)); 2*sin(0.1*t(i)); 0.1*t(i)];
eta_d_dot = [2*0.1*sin(0.1*t(i)); 2*0.1*cos(0.1*t(i)); 0.1];
zeta(:,i) = inv(J_psi) * eta_d_dot;
```

So, then it would be again 2 into 0.1 times of so cos so, 0.1 times of ti you see I did not change anything I am using the same code only thing I am adding instead of giving a user input directly, I am putting in a other way around ok. So, now, if I take time differentiation of this would be 0.1 right.

```
eta_dot(:,i) = J_psi * zeta(:,i);
```

So, these are the desired you can say state vectors and these are the desired state derivatives. Now, what we are actually like interested, we are interested to find what you call zeta. So, that is what you are actually like interested.

So, here actually like substituting into eta_dot maybe like redundant, but we are actually like doing it right perspective. So, what we are trying to see the zeta or you can say of i, I am calculating based on what we know like J_psi inverse right. So, the J_psi inverse I can write in a MATLAB you know like MATLAB command it is just inv; so inverse of this.

```
zeta(:,i) = inv(J_psi) * eta_d_dot;
```

```
eta(:,i+1) = eta(:,i) + dt * eta_dot(:,i);
```

So now, what you need to have? So, here actually eta desired dot. So, that here desired so I am actually like putting that here ok. So, now you can see that what we have done. So, we have taken desired state variable and that time derivative, I have taken now I am taking it this.

(Refer Slide Time: 17:40)

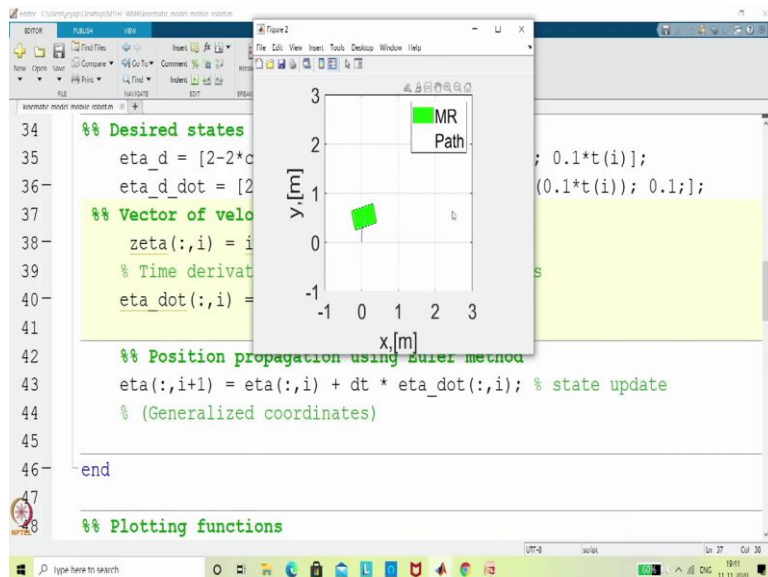
```

34 %% Desired states (Generalized coordinates)
35 eta_d = [2-2*cos(0.1*t(i)); 2*sin(0.1*t(i)); 0.1*t(i)];
36 eta_d_dot = [2*0.1*sin(0.1*t(i)); 2*0.1*cos(0.1*t(i)); 0.1];
37 %% Vector of velocity input commands
38 zeta(:,i) = inv(J_psi) * eta_d_dot;
39 % Time derivative of generalized coordinates
40 eta_dot(:,i) = J_psi * zeta(:,i);
41
42 %% Position propagation using Euler method
43 eta(:,i+1) = eta(:,i) + dt * eta_dot(:,i); % state update
44 % (Generalized coordinates)
45
46 end
47
48 %% Plotting functions

```

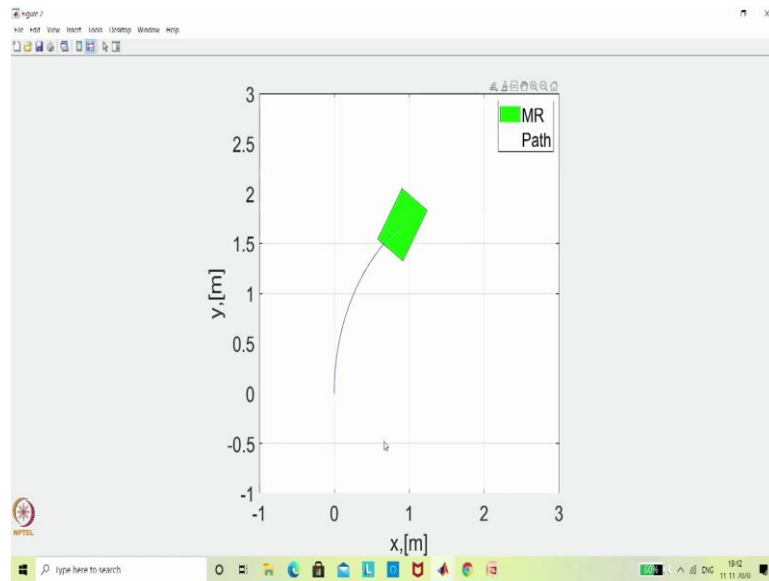
So, this is what you call? So, it is actually like vector of so, Vector of you can say velocity input commands. So, that you have actually like calculated right. So, now, if I run this what we have given? So, the profile which is look like a circle. So, that circle supposed to be followed, we will actually like cross check whether that is actually like done or not. So, we will just run whether there is an error free.

(Refer Slide Time: 18:09)



So, I will see that is actually like a error free. So, you can see like now it is it actually started from 0 point and it is actually trying to make a circle.

(Refer Slide Time: 18:18)

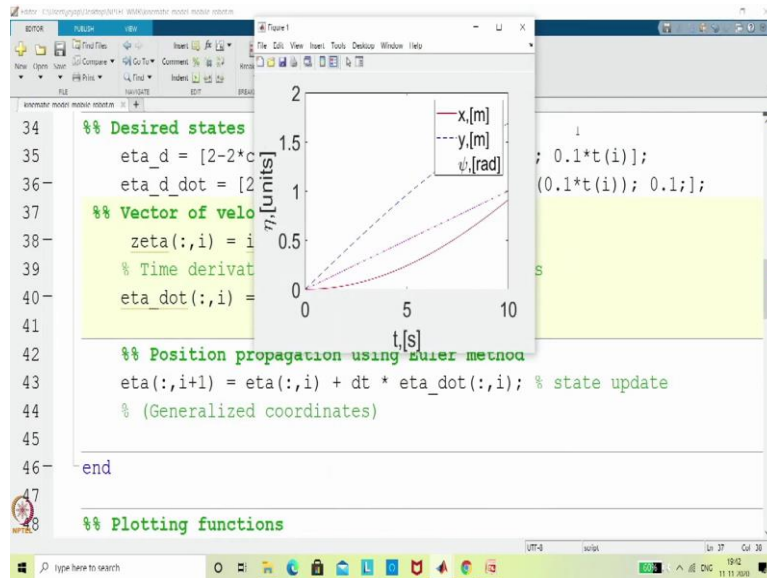


So, you can see here. So, now, you can actually like see that it is started from $(0, 0)$ that is what we have intend. So, you have desired and actual both are starting from 0 and it is actually like making a circle.

But one can actually like see that this is actually like not completing it. So, why it is not completing? Because, we have not given the side desired dot as per the what you call the circle. So, we have actually like given in a other way around, here actually like how the side desired supposed to be it would be actually like a you can say minus of this. Why it is so?

Because it is actually like rotating in the other direction; so, that is what we have to actually like see, in the sense if I actually change minus 0.1 times of t_i , then you can see that that will actually like align with this, but that is never less that is not important.

(Refer Slide Time: 19:08)



What important is actually like, what you have given as a user input that we are trying to follow. So, for that what we are actually like seen that this is actually like kind of control loop, but this is an open loop, still I am just checking.

(Refer Slide Time: 19:20)

```

34 %% Desired states (Generalized coordinates)
35 eta_d = [2-2*cos(0.1*t(i)); 2*sin(0.1*t(i)); 0.1*t(i)];
36 eta_d_dot = [2*0.1*sin(0.1*t(i)); 2*0.1*cos(0.1*t(i)); 0.1];
37 %% Vector of velocity input commands
38 zeta(:,i) = inv(J_psi) * eta_d_dot;
39
40 e(:,i) = eta_d - eta(:,i);
41 % Time derivative of generalized coordinates
42 eta_dot(:,i) = J_psi * zeta(:,i);
43
44 %% Position propagation using Euler method
45 eta(:,i+1) = eta(:,i) + dt * eta_dot(:,i); % state update
46 % (Generalized coordinates)
47
48 end

```

So, what would be the eta desired and what is your eta of that i^{th} instant ok. So, now I am actually like assuming that this is an error. So, this error which would be having a vector so, I am assuming that that would be having propagated. So, now, I can see whether the error exists or not ok.

(Refer Slide Time: 19:42)

```
43
44     %% Position propagation using Euler method
45     eta(:,i+1) = eta(:,i) + dt * eta_dot(:,i); % state update
46     % (Generalized coordinates)
47
48 end
49
50 %% Plotting functions
51 % figure
52 % plot(t, eta(1,1:i),'r-');
53 % set(gca,'fontsize',24)
54 % xlabel('t, [s]');
55 % ylabel('x, [m]');
56 % figure
```

(Refer Slide Time: 19:46)

```
95     legend('MR','Path')
96     set(gca,'fontsize',24)
97     xlabel('x, [m]'); ylabel('y, [m]');
98     pause(0.1);
99     hold off
100 end % animation ends here
101
102 figure
103 plot(t,e)
104 legend('x_e, [m]', 'y_e, [m]', '\psi_e, [rad]');
105 set(gca,'fontsize',24)
106 xlabel('t, [s]');
107 ylabel('\eta_e, [units]');
```

So now, if I run so I will actually like add a plot. So, at the end after the simulation or animation I will just add. So, there is a figure so, where plot t versus e. So, where there are 3 errors so, in the sense I can actually like take it legends and all.

(Refer Slide Time: 20:01)

```

72 plot(t, eta(2,1:i), 'b--');
73 plot(t, eta(3,1:i), 'm-.');
74 legend('x, [m]', 'y, [m]', '\psi, [rad]');
75 set(gca, 'fontsize', 24);
76 xlabel('t, [s]');
77 ylabel('\eta, [units]');
78
79 %% Animation (mobile robot motion animation)
80 l = 0.6; % length of the mobile robot
81 w = 0.4; % width of the mobile robot
82 % Mobile robot coordinates
83 mr_co = [-l/2, l/2, l/2, -l/2, -l/2;
84         -w/2, -w/2, w/2, w/2, -w/2];
85 figure
86 for i = 1:length(t) % animation starts here

```

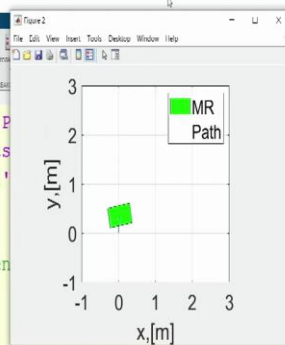
So, I will just use it these things it is easy for me. So, I am just copying it, but you can type it yourself. So, now, this is x error and this is actually like y error in meter and this is psi error in radians, and this is actually like I call eta error ok.

(Refer Slide Time: 20:23)

```

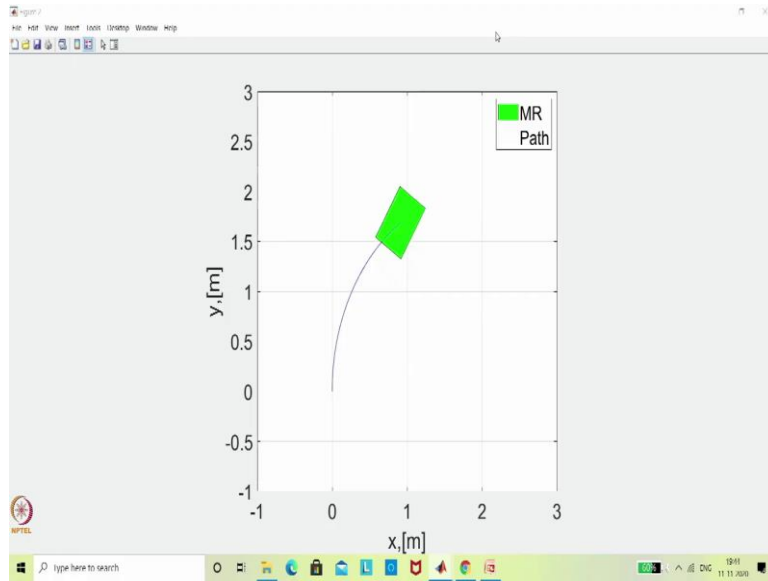
95 legend('MR', 'P
96 set(gca, 'font
97 xlabel('x, [m]
98 pause(0.1);
99 hold off
100 end % animation en
101
102 figure
103 plot(t,e)
104 legend('x_e, [m]', 'y_e, [m]', '\psi_e, [rad]');
105 set(gca, 'fontsize', 24);
106 xlabel('t, [s]');
107 ylabel('\eta_e, [units]');
108
109

```



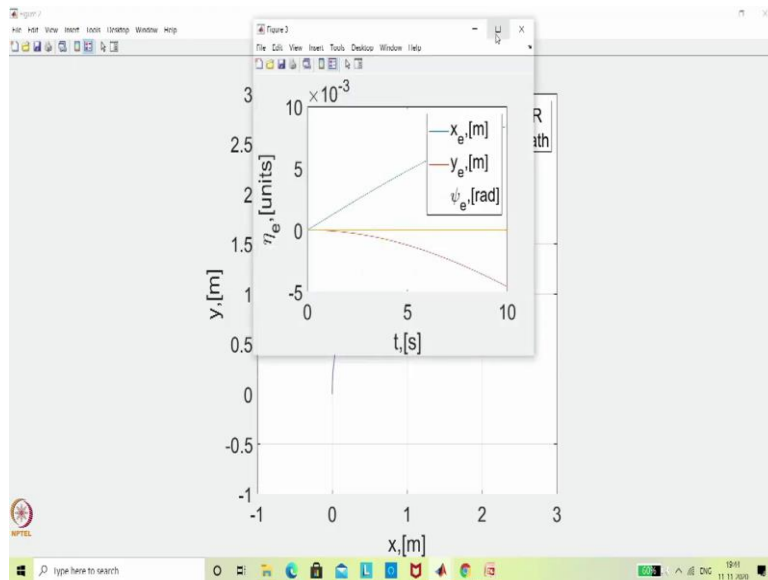
So now, if I actually like run this what I will get finally, after the simulation or you can say animation, it would be giving the error value.

(Refer Slide Time: 20:29)

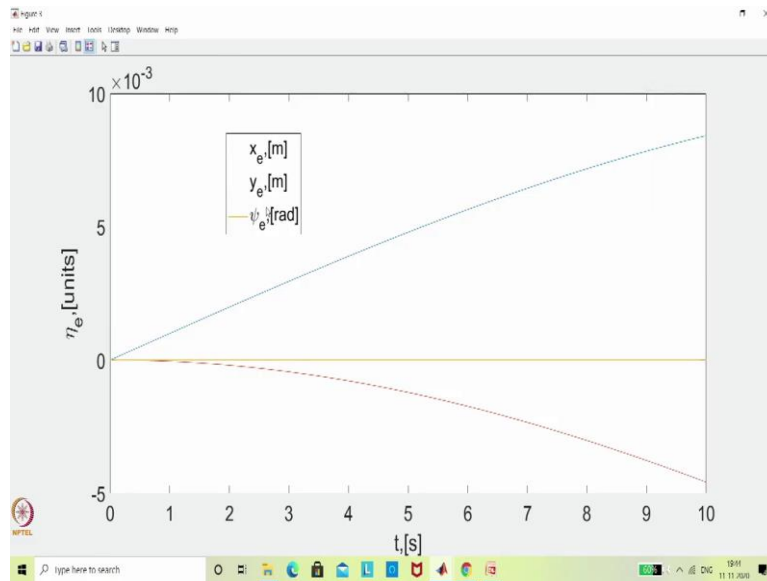


So, we can actually like see that. So, now, I am just rerunning, but we can actually like you can see play this particular code in n number of ways and you can do it, but what we have done here is actually like last code the same code. But without adding this what we have done is a forward differential kinematics. But, here we have done the inverse differential kinematics.

(Refer Slide Time: 20:43)



(Refer Slide Time: 20:46)



So, now you can see that there is a very small error, if I actually like see the magnitude it is actually like in the millimeter level, where the psi e also like you can see that it is a 0. Whereas, you can see x position and y position error is actually like very small, it is actually like open loop.

Now, one situation I am actually like adding it just for your benefit, you see that I am actually like giving a situation, where the initial condition is actually like different from what you have given a desired value.

(Refer Slide Time: 21:17)

```
1 %% Kinematic simulation of a land-based mobile robot
2 clear all; clc; close all;
3
4 %% Simulation parameters
5 dt = 0.1; % Step size
6 ts = 10; % Simulation time
7 t = 0:dt:ts; % Time span
8
9 %% Initial conditions
10 x0 = 0.5;
11 y0 = 0.5;
12 psi0 = 0;
13
14 eta0 = [x0;y0;psi0];
```

So, for example, earlier what we have seen that the desired, when you put t equal to 0 it start from 0 and 0 right, x desired and y desired is 0. But, now the initial condition I changed is 0.5 meter and 0.5 meter. So, now, you can see that the code is completely different right.

(Refer Slide Time: 21:41)

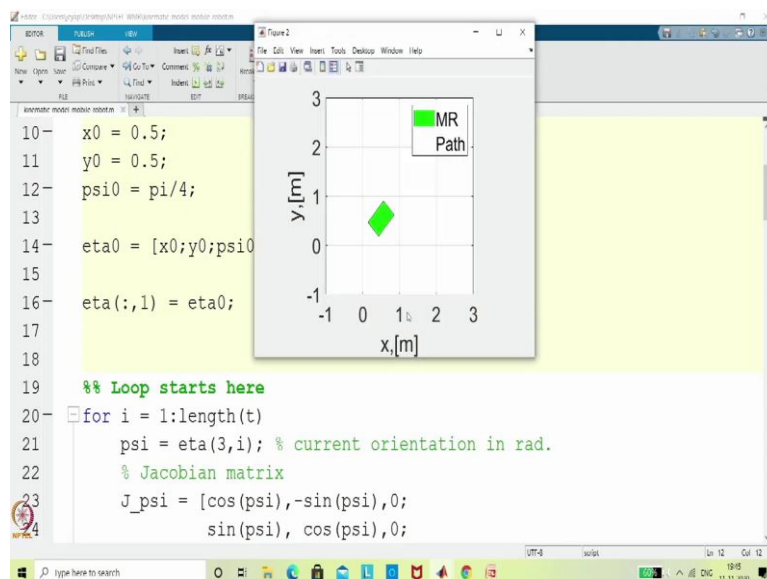
```

7- t = 0:dt:ts; % Time span
8
9 %% Initial conditions
10 x0 = 0.5;
11 y0 = 0.5;
12 psi0 = pi/4;
13
14 eta0 = [x0;y0;psi0];
15
16 eta(:,1) = eta0;
17
18
19 %% Loop starts here
20 for i = 1:length(t)
21     psi = eta(3,i); % current orientation in rad.

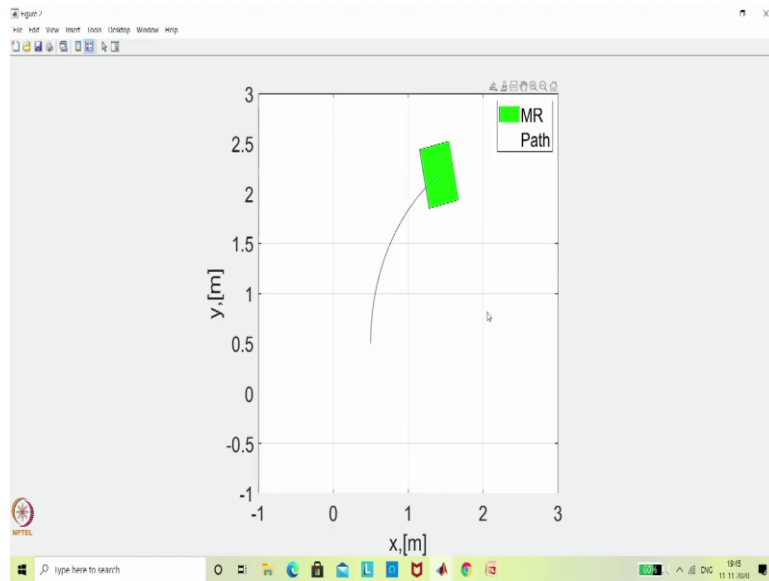
```

So, what that means so, you have error. Further I assuming that this is actually like rotated pi by 4 so, I am just showing it. So, instead of vehicle is actually like 0 position it is actually like inclined in a 45 degree. Now, you can see that the circle would be with respect to the 45 degree and with respect to you can say the starting point that is why I said it is a open loop.

(Refer Slide Time: 22:01)

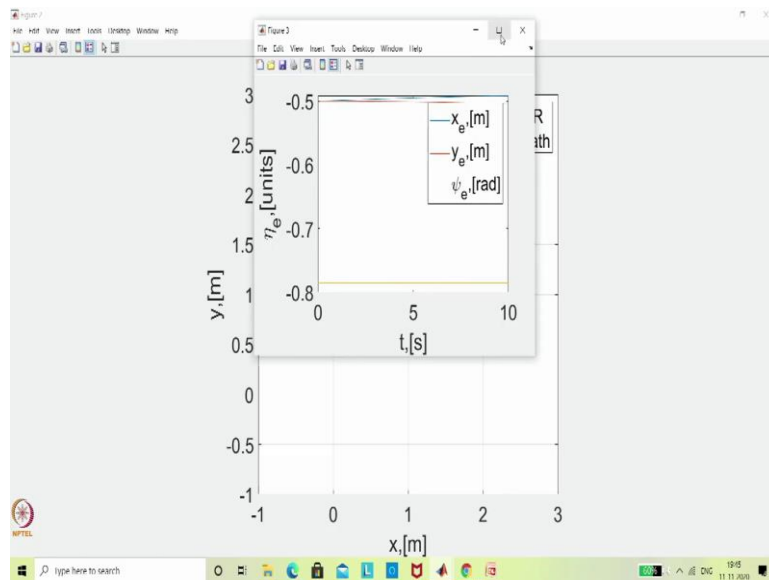


(Refer Slide Time: 22:03)



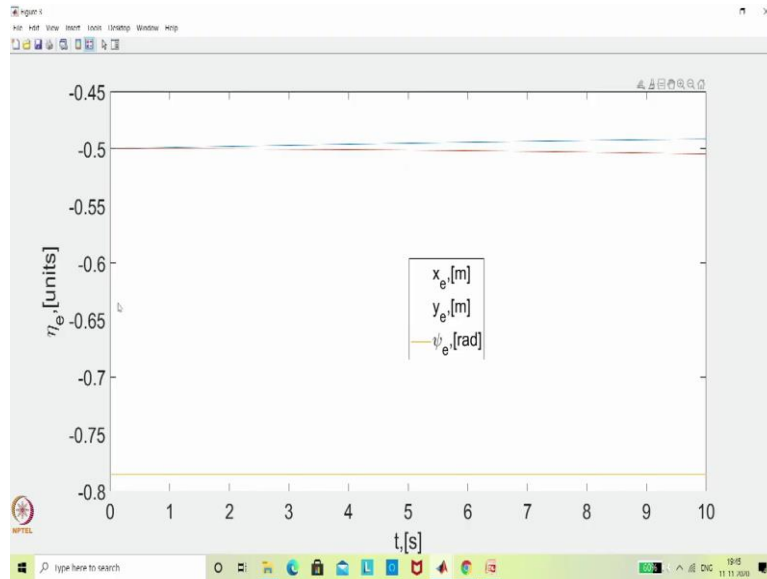
So, that is what the whole idea of actually like talking about this particular you can say, inverse differential kinematics. The inverse differential kinematics is actually like open loop control, not closed loop control that is what the intention of bringing this inverse differential kinematics into this particular kinematic simulation code.

(Refer Slide Time: 22:22)



Now, you can see right earlier it was different, but now it is actually like totally different.

(Refer Slide Time: 22:25)



So, and the error also like you have given actually like starting from supposed to be from 0, but it is starting from 0.5 right. And similarly this is actually like starting from minus 45 degree as the error. So, now, you got it a clarity, so you have seen like lecture 5 how to do the kinematics simulation, but in lecture 6 what we have seen?

This is the lecture 6 what we have seen is actually like the motion animation, along with the forward and inverse differential kinematics, in the sense what one can see. If you bring the vehicle configuration along with wheel, you can definitely add this particular code and then you can make it. So, in the sense the next lecture what we are trying to see. So, how to bring that you can say wheel configuration into the vehicle?

So, for that we are trying to bring the generalized wheel kinematic model so, that is what we are going to see in lecture 7. And then we will move forward and end of that particular week, we will be doing the motion, animation along with wheel configuration. Even there we will do a kinematic control also then; you can see that more close to the real time that is what we are interested.

So, with that the lecture 6 you can say the kinematics simulation along with motion animation is done and we will meet on lecture 7 and see you then, bye.