Wheeled Mobile Robots

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

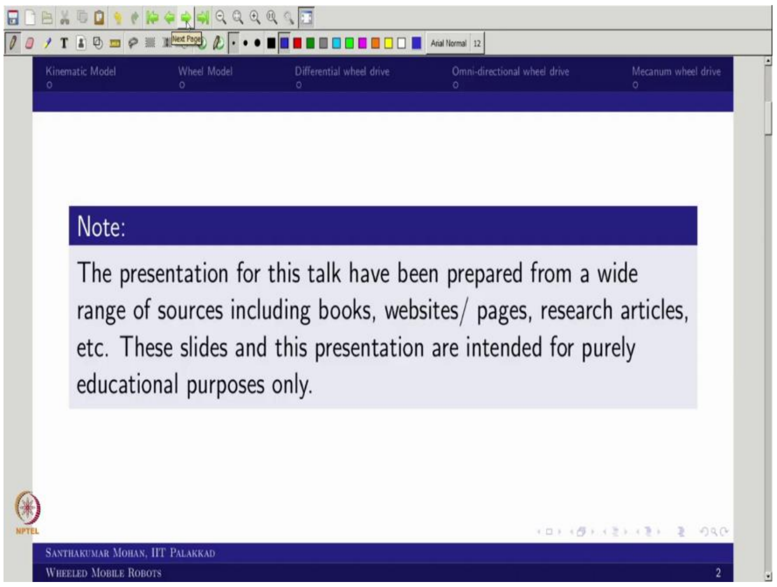
Lecture - 10
Kinematic Simulation of Wheeled Mobile Robots Part 1

Welcome back to Wheeled Mobile Robots. So, this is lecture 10, where we are going to talk about more you can say kinematic simulation aspect. So, last class we have seen last two classes, we have seen like design of maneuverability, based on that we have derived the generalized wheel model.

Based on the generalized wheel model, we have actually like got the final equation in the form of you can say angular velocity that you would be taking as a velocity input command. So, based on that what we are trying to do in this particular lecture; we would be doing a kinematic simulation on MATLAB.

Before that I will show you like what are the example we have seen, from that we will extract the equation and we will start incorporating in the MATLAB directly. So, let us actually like go the slide one by one.

(Refer Slide Time: 00:59)



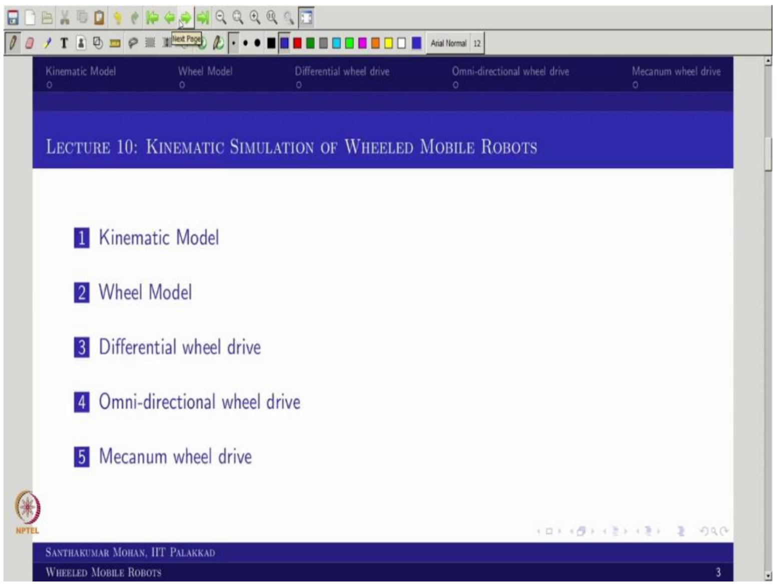
Note:

The presentation for this talk have been prepared from a wide range of sources including books, websites/ pages, research articles, etc. These slides and this presentation are intended for purely educational purposes only.

Santhakumar Mohan, IIT Palakkad
Wheeled Mobile Robots 2

So, we will actually like come back to the content of this particular slide or you can say lecture.

(Refer Slide Time: 01:01)


Kinematic Model Wheel Model Differential wheel drive Omni-directional wheel drive Mecanum wheel drive

LECTURE 10: KINEMATIC SIMULATION OF WHEELED MOBILE ROBOTS

1. Kinematic Model
2. Wheel Model
3. Differential wheel drive
4. Omni-directional wheel drive
5. Mecanum wheel drive

SANTHAKUMAR MOHAN, IIT PALAKKAD
WHEELED MOBILE ROBOTS 3

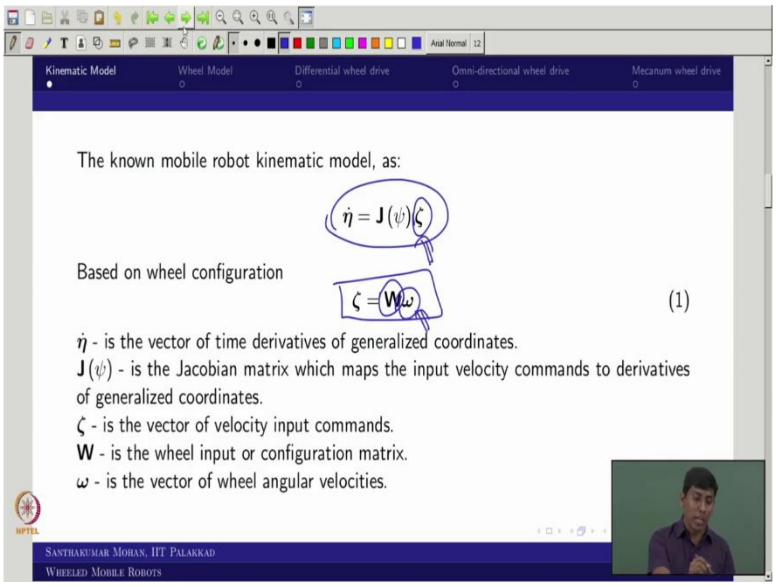
So, we would be recalling what is kinematic model we derived and based on the generalized wheel model, what we have obtained and we will take three example in this particular lecture. So, where we would be starting with a differential wheel drive as a prime most; because that is the simplest one and many of our you can say industrial wheeled mobile robot in the configuration of differential wheel.

And similarly if you see any shop floor or you can say warehouses; so now this Omni directional and Mecanum wheel drive mobile robots are coming in a very you can say bright way. So, in that sense we will try to cover these three and in the next lecture, this is the same thing we will extend as a part 2.

There we would see something like a specific configuration. For example, independent steerable wheel if we bring it; so how that would be coming into a picture or synchro drive or unicycle like that some specific example which is a special case that we will brought into a part 2.

But the part 1 would be a conventional one, which we have already derived in the generalize wheel model lecture. So, where if you refer lecture 8, then you can actually like find these all the you can say kinematic model ready.

(Refer Slide Time: 02:16)


Kinematic Model | Wheel Model | Differential wheel drive | Omni-directional wheel drive | Mecanum wheel drive

The known mobile robot kinematic model, as:

$$\dot{\eta} = \mathbf{J}(\psi)\zeta$$

Based on wheel configuration

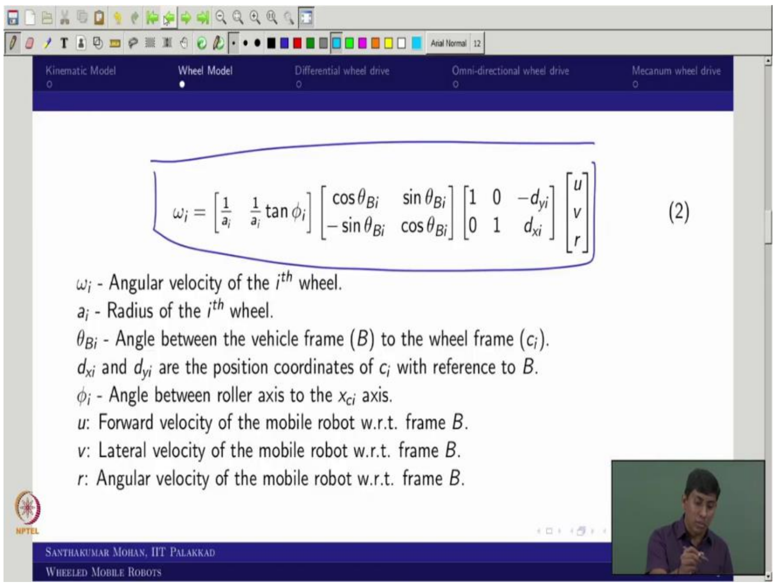
$$\zeta = \mathbf{W}\omega \qquad (1)$$

$\dot{\eta}$ - is the vector of time derivatives of generalized coordinates.
$\mathbf{J}(\psi)$ - is the Jacobian matrix which maps the input velocity commands to derivatives of generalized coordinates.
ζ - is the vector of velocity input commands.
$\mathbf{W}$ - is the wheel input or configuration matrix.
ω - is the vector of wheel angular velocities.

Santhakumar Mohan, IIT Palakkad
Wheeled Mobile Robots

So, in that sense what we are trying to recap. So, recapping is in the sense, we are actually like known what this, what you call kinematic model and based on this kinematic model, this is what you call the input; that input you can write in the form of you call, so W × ω. So, where this ω what you call actually like angular velocity; now this W what we derived based on so called the generalized wheel model.

(Refer Slide Time: 02:41)


Kinematic Model | Wheel Model | Differential wheel drive | Omni-directional wheel drive | Mecanum wheel drive

$$\omega_i = \begin{bmatrix} \frac{1}{a_i} & \frac{1}{a_i}\tan\phi_i \end{bmatrix} \begin{bmatrix} \cos\theta_{Bi} & \sin\theta_{Bi} \\ -\sin\theta_{Bi} & \cos\theta_{Bi} \end{bmatrix} \begin{bmatrix} 1 & 0 & -d_{yi} \\ 0 & 1 & d_{xi} \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix} \qquad (2)$$

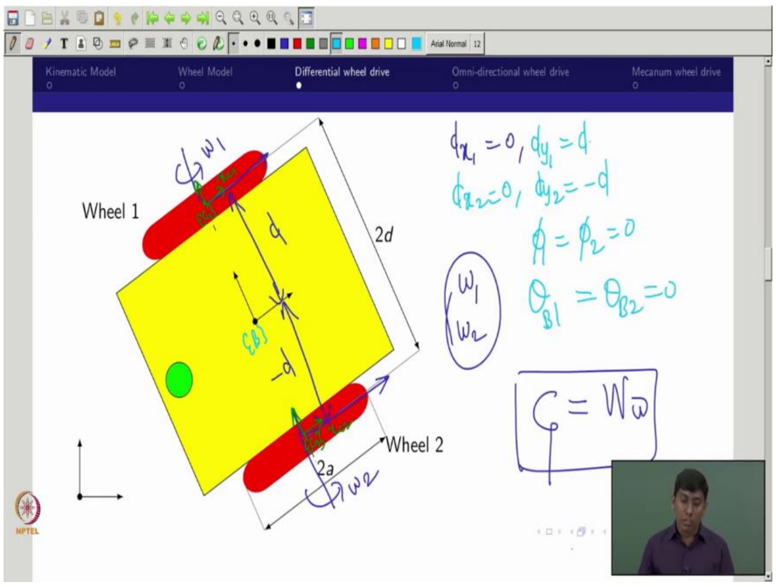
ω_i - Angular velocity of the i^{th} wheel.
a_i - Radius of the i^{th} wheel.
θ_{Bi} - Angle between the vehicle frame (B) to the wheel frame (c_i).
d_{xi} and d_{yi} are the position coordinates of c_i with reference to B.
ϕ_i - Angle between roller axis to the x_{ci} axis.
u: Forward velocity of the mobile robot w.r.t. frame B.
v: Lateral velocity of the mobile robot w.r.t. frame B.
r: Angular velocity of the mobile robot w.r.t. frame B.

Santhakumar Mohan, IIT Palakkad
Wheeled Mobile Robots

So, in that sense what we are actually trying to recall the generalized wheel model. So, if I actually like know there is a powered wheel, where the angular velocity is ω; so ith wheel angular velocity I can obtain based on this particular equation.

So, this is not the case in this particular lecture; this particular lecture what we have derived in the lecture 8. So, few of the examples we have seen; we will extract the equation and we will try to incorporate in a MATLAB simulation code and then we will show how this can be done.

(Refer Slide Time: 03:10)



So, for that we will take the first one. So, here actually like you can see that this is a wheel 1, where you would be having a you can say linear velocity here, another linear velocity here. So, where here you can see that this is ω_1 and here actually like ω_2.

So, these two we are actually like trying to derive based on the you call generalized wheel model, where we can actually like see ω_1 and ω_2 and we can actually like get the what you call the relation ξ. So, the ξ I can write in other way around; so, the ξ which we used to write in the, so this way; so the $\xi = W \times \omega$.

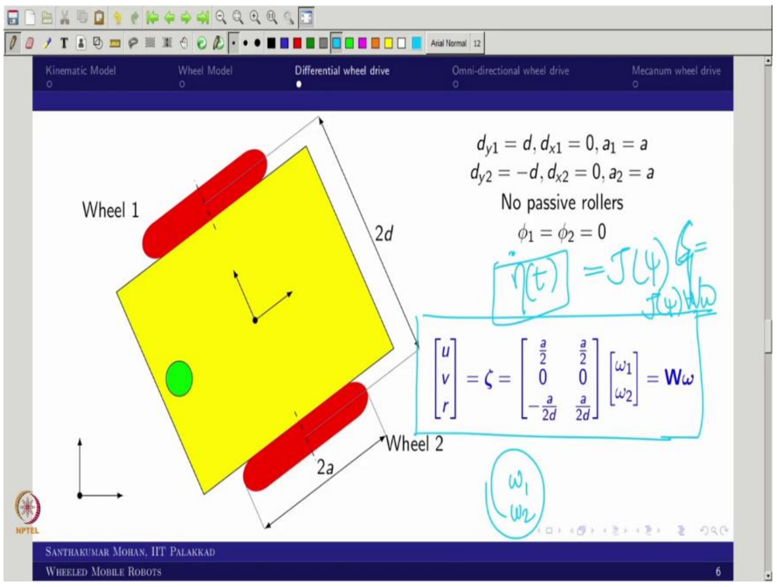
So, now we will actually like find this. So, based on the equation; so obviously you know like this is the distance which we call actually like d and this is also d, but this is in the opposite direction. So, I am putting - d. In that sense, you can see the d_{x1} would be 0;

because the wheel frame what we can actually like see here. So, I am just drawing, this is what the wheel frame, this is c_2 and this is what you call the c_1.

So, now this is x_{c1} and x_{c2} you can see. So, now, in that sense what one can see. So, the distance d_{x1} and d_{y1} easily you can get it. So, this is the first wheel. So, that is actually like this and d_{x2} and d_{y2} you can actually like get it again based on this. Since there is a no passive; you can say there is no passive roller, so you can say that ϕ_1 and ϕ_2 are 0.

In the sense what one can see based on this equation, you can substitute everything and then you can get it. Further you can see that the B frame and the c_1 and c_2 are actually like a parallel frames; in that sense the θ_1, which is I call θB_1 and θB_2 are 0.

(Refer Slide Time: 05:06)



$d_{y1} = d, d_{x1} = 0, a_1 = a$
$d_{y2} = -d, d_{x2} = 0, a_2 = a$
No passive rollers
$\phi_1 = \phi_2 = 0$

$\boxed{\dot{\eta}(t)} = J(\psi)\zeta = J(\psi)W\omega$

$$\boxed{\begin{bmatrix} u \\ v \\ r \end{bmatrix} = \zeta = \begin{bmatrix} \frac{a}{2} & \frac{a}{2} \\ 0 & 0 \\ -\frac{a}{2d} & \frac{a}{2d} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \mathbf{W}\omega}$$

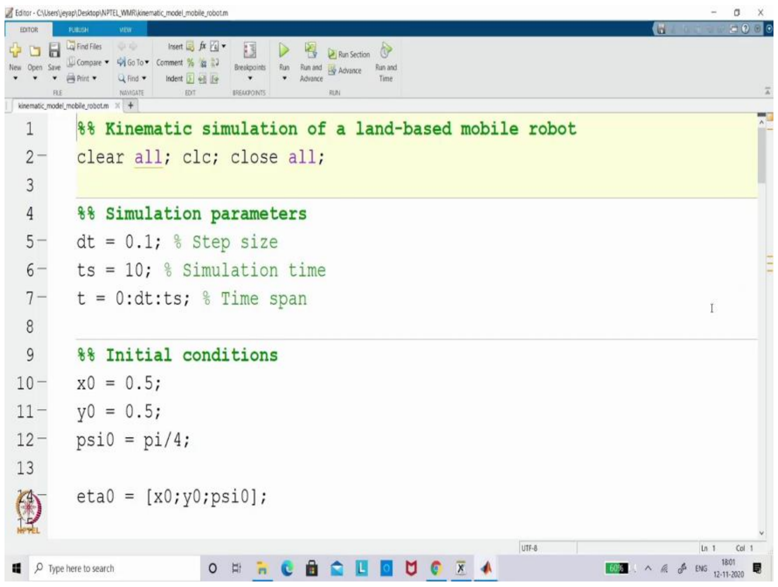
ω_1
ω_2

So, if that is the case, what would be the final equation which we obtained? So, these are we have actually like derived based on this. So, this is the final equation we obtained. So, this equation now we extracted. So, based on this, we will actually try to do a what you call kinematic simulation.

So, now we will actually like move to the MATLAB screen and we will try to incorporate this. So, now, if I vary this ω_1 and ω_2; so what will happen to your what you call η. So, that is actually like with respect to time; this is what we are trying to do as a kinematic simulation here. So, what we actually like know, this $\dot{\eta}$ we can write as $J(\Psi \times \xi)$.

But now, this sorry this ξ is actually like rewritten as, so W × ω. So, in the sense this we will actually like write it. So, now, we will substitute this ω vector, whereas earlier we have taken directly $\begin{bmatrix} u \\ v \\ r \end{bmatrix}$ we have given; but right now we will take the configuration which is given as to us, so which is ω_1 and ω_2. So, now, we will try to see that here in the MATLAB screen. So, let me go.

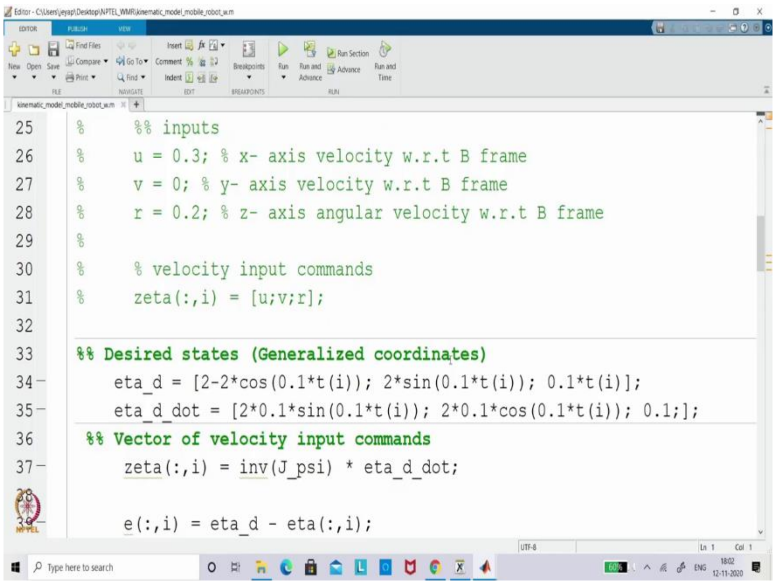
(Refer Slide Time: 06:13)



```
%% Kinematic simulation of a land-based mobile robot
clear all; clc; close all;

%% Simulation parameters
dt = 0.1; % Step size
ts = 10; % Simulation time
t = 0:dt:ts; % Time span

%% Initial conditions
x0 = 0.5;
y0 = 0.5;
psi0 = pi/4;

eta0 = [x0;y0;psi0];
```

So, you can see now. So, this is a MATLAB code which we have taken since last two classes; in the sense lecture 5 and 6, we have used this.

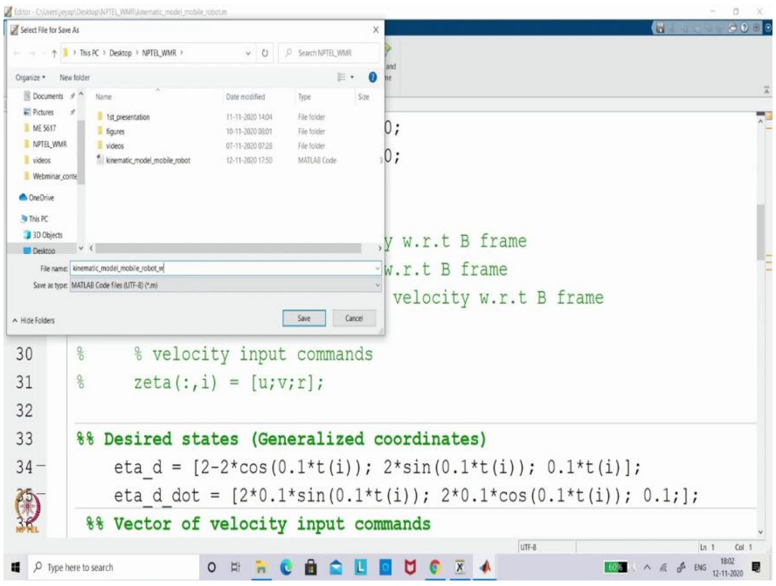
(Refer Slide Time: 06:24)



```
%       %% inputs
%       u = 0.3; % x- axis velocity w.r.t B frame
%       v = 0; % y- axis velocity w.r.t B frame
%       r = 0.2; % z- axis angular velocity w.r.t B frame
%
%       % velocity input commands
%       zeta(:,i) = [u;v;r];

%% 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;];
 %% Vector of velocity input commands
     zeta(:,i) = inv(J_psi) * eta_d_dot;

     e(:,i) = eta_d - eta(:,i);
```

So, now, what we are actually like trying to change. So, we are trying to change now, there is no desired. So, there is no actually like input. So, what we are trying to do?

(Refer Slide Time: 06:37)

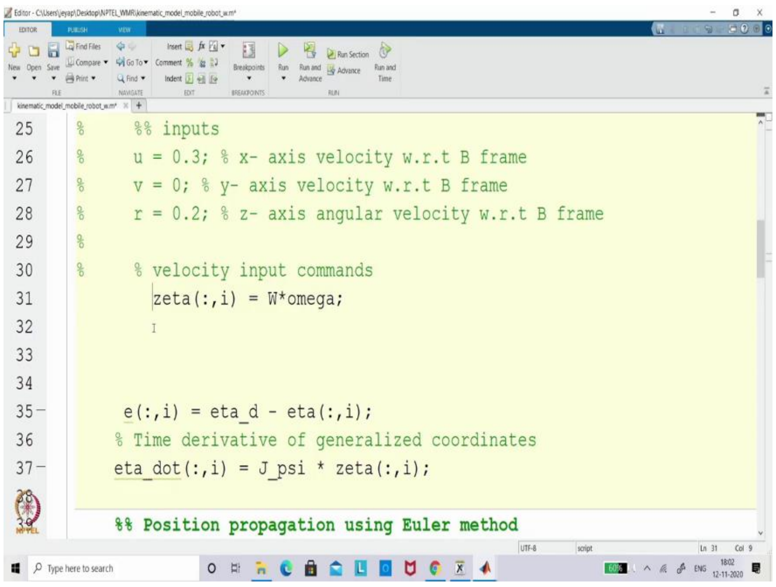


```
File name: kinematic_model_mobile_robot_w
Save as type: MATLAB Code files (UTF-8) (*.m)
```

```
%       % velocity input commands
%       zeta(:,i) = [u;v;r];

%% 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;];
 %% Vector of velocity input commands
```

So, I am actually like saving it this is in a different file, so that for my reference; I am just taking it this is v. So, I just put w. So, now what I am trying to do? I am actually taking out all those things, ok.

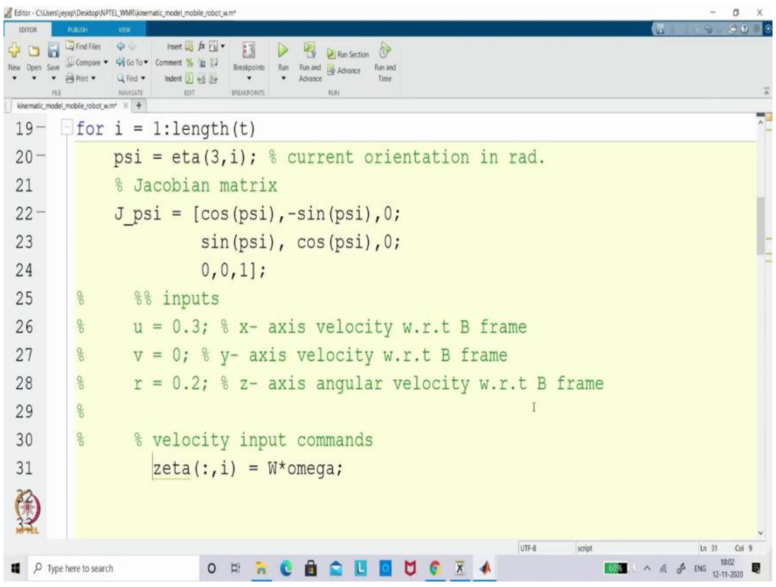
(Refer Slide Time: 06:48)



```
%     %% inputs
%     u = 0.3; % x- axis velocity w.r.t B frame
%     v = 0; % y- axis velocity w.r.t B frame
%     r = 0.2; % z- axis angular velocity w.r.t B frame
%
%     % velocity input commands
        zeta(:,i) = W*omega;



      e(:,i) = eta_d - eta(:,i);
    % Time derivative of generalized coordinates
    eta_dot(:,i) = J_psi * zeta(:,i);

    %% Position propagation using Euler method
```

So, now I am actually like saying this ξ is actually like written as W × ω, ok. So, I will write. So, ω vector, ok. So, this is what I am calling ξ, ok. So, in the sense, this is what my ξ.

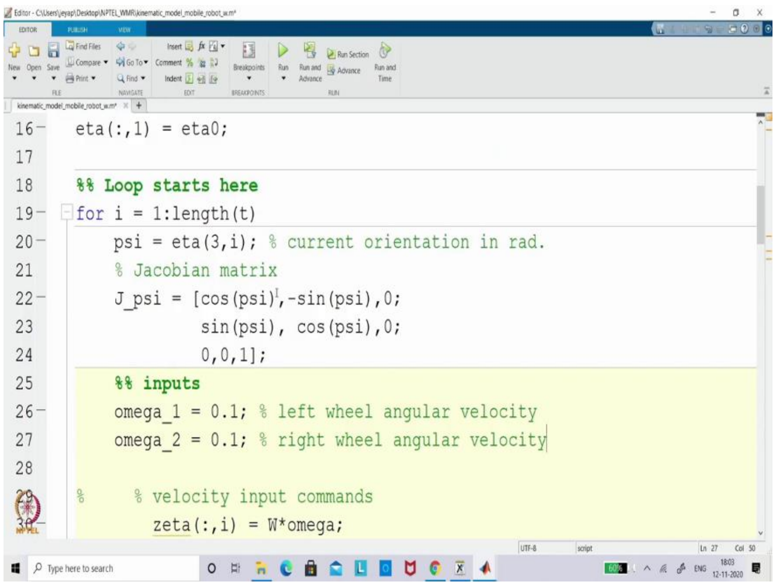
(Refer Slide Time: 07:07)



```
for i = 1:length(t)
    psi = eta(3,i); % current orientation in rad.
    % Jacobian matrix
    J_psi = [cos(psi),-sin(psi),0;
             sin(psi), cos(psi),0;
             0,0,1];
%     %% inputs
%     u = 0.3; % x- axis velocity w.r.t B frame
%     v = 0; % y- axis velocity w.r.t B frame
%     r = 0.2; % z- axis angular velocity w.r.t B frame
%
%     % velocity input commands
        zeta(:,i) = W*omega;
```

So, now, the ξ is obtained; but the ξ is obtained based on the wheel angular velocity, so that is what our input.

(Refer Slide Time: 07:22)



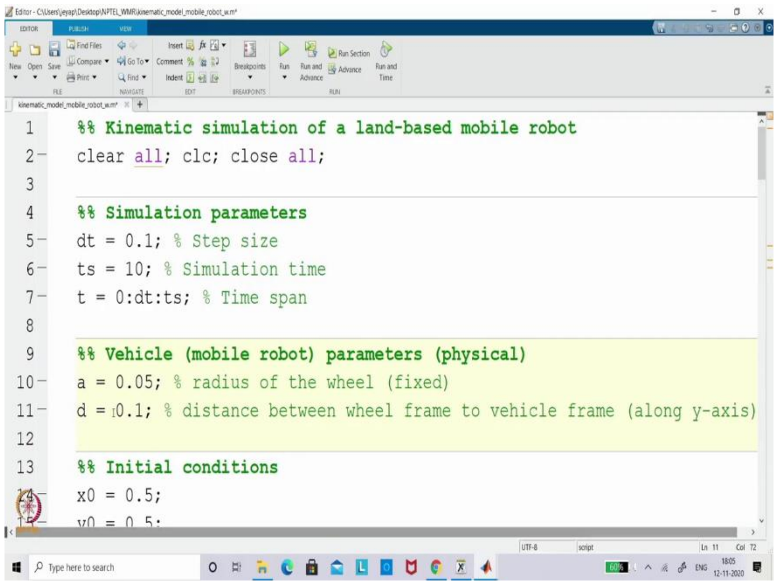
```
eta(:,1) = eta0;

%% Loop starts here
for i = 1:length(t)
    psi = eta(3,i); % current orientation in rad.
    % Jacobian matrix
    J_psi = [cos(psi),-sin(psi),0;
             sin(psi), cos(psi),0;
             0,0,1];
    %% inputs
    omega_1 = 0.1; % left wheel angular velocity
    omega_2 = 0.1; % right wheel angular velocity

%     % velocity input commands
      zeta(:,i) = W*omega;
```

So, in the sense I can write that as, so ω_1. So, which is actually like your first wheel, in the sense you can see in the top side; when you look at from the x y plane this is left wheel. So, I am taking that is 0.1 radian per second is the speed. So, this I call a left wheel angular velocity, ok.

So, then I am taking ω_2. So, which is actually like, for simplicity I am taking both are equal. So, we will see how that will work. So, right wheel angular velocity; you see so far what we have taken, we have taken a small patch, in the sense you have actually like seen $\begin{bmatrix} u \\ v \\ r \end{bmatrix}$ which is in the form of body fixed velocity that, directly you have given into $\dot{\eta}$ equation, where there is no geometry involved, right.

But now what happened? If you look at the equation what we have seen here; you can see that this is having a geometry, where a is coming into a picture and as well as d is coming into a picture, right. So, if you look at that, so then what you have to see; so you have to actually like use that. So, in the sense what I am trying to bring it.

(Refer Slide Time: 08:38)



```
%% Kinematic simulation of a land-based mobile robot
clear all; clc; close all;

%% Simulation parameters
dt = 0.1; % Step size
ts = 10; % Simulation time
t = 0:dt:ts; % Time span

%% Vehicle (mobile robot) parameters (physical)
a = 0.05; % radius of the wheel (fixed)
d = 0.1; % distance between wheel frame to vehicle frame (along y-axis)

%% Initial conditions
x0 = 0.5;
y0 = 0.5;
```

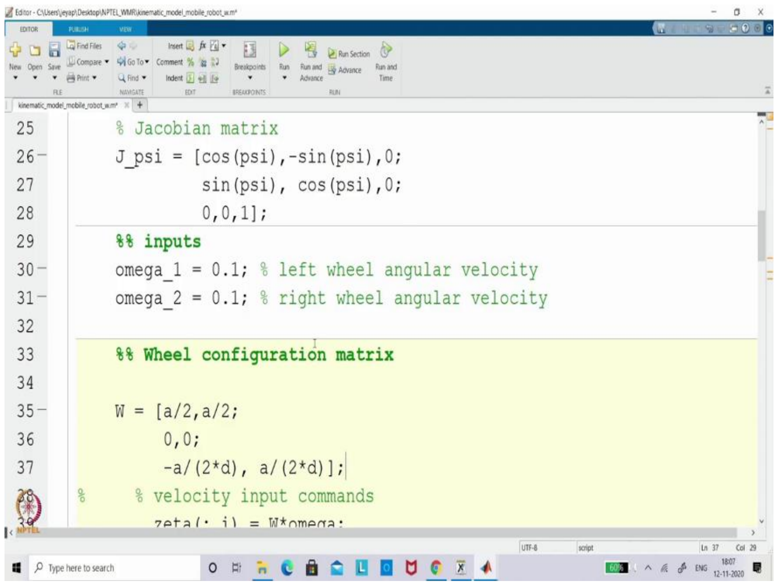
So, I am bringing it that as, so vehicle parameter. So, although it is a mobile robot, it is a vehicle that is what we can write. Some people may offend, so I am just putting at that is a mobile robot parameters. So, this is actually like a physical. So, these are the physical parameter; here the one is a, which is nothing but the radius of the wheel.

I assume that the diameter of the wheel is 10 centimeter; so the radius would be 0.05 meter, in the sense 5 centimeter. So, this is radius of the wheel, it is a fixed wheel, right. So, I am just giving a additional information here itself. So, then what you have, there is a l; not l here, because the length we are not using here and the you call W matrix, the d which is actually like distance between vehicle frame to the wheel.

So, now, we have taken two wheels, both are actually like symmetrically arranged with respect to your B frame; in the sense 2 d we have taken as a distance between two wheels. So, d I am extracting from there. I assume that this is something like 20 centimeter as the with vehicle; so obviously in that sense it is actually like 10 centimeter, it is look like very small, right.

So, still I will actually like use it; later on we will see how this will come. So, this is distance between you can say wheel frame to you can say vehicle frame, ok. So, that means actually like the distance between c to b and we are assuming that d_1 and d_2 are same; this is on the you can say y axis, I will just along y axis I will put it that is the right one, along y axis, right. So, now, what we have actually like seen; so we have seen that, a and d we have given. So, now, we will come back here. So, this is actually like depend on W.

(Refer Slide Time: 10:34)



```
% Jacobian matrix
J_psi = [cos(psi),-sin(psi),0;
          sin(psi), cos(psi),0;
          0,0,1];
%% inputs
omega_1 = 0.1; % left wheel angular velocity
omega_2 = 0.1; % right wheel angular velocity

%% Wheel configuration matrix

W = [a/2,a/2;
     0,0;
     -a/(2*d), a/(2*d)];
%    % velocity input commands
     zeta(:,i) = W*omega;
```

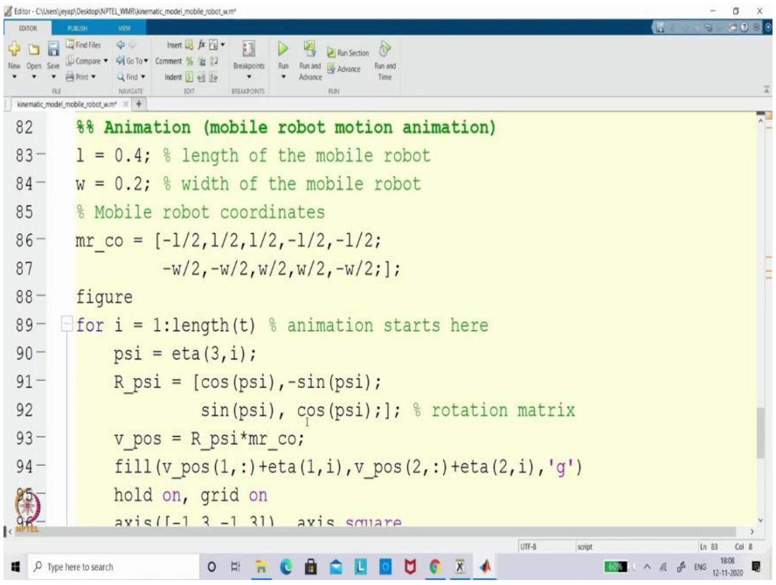
So, I am writing that W matrix. So, which is I am writing as wheel configuration matrix. So, wheel configuration matrix which I am calling W and you know like what is W. So, if you do not know; you can actually like recall this equation what we extracted. So, now, I am actually like going back to that.

So, now, you can see that this would be. So, a by 2 right comma a by 2 this is the first row; in the sense x axis velocity would be just mean of two angular velocity multiply with radius. Then what you have, actually like your y axis is 0 based on the fixed wheel configuration and then you have actually like minus or plus based on what way you are actually like taking it.

So, if you look it at this particular case; so where you have actually like taken the left wheel is your wheel 1. So, in the sense the left wheel alone is having a, you can say angular velocity; you assume that the second wheel is not having anything. Then what one can actually like see it. So, this would actually try to rotate in a clockwise.

So, in the sense what we have seen, the size actually like positive direction is counter clockwise; but this is rotating in the opposite side that is why you can see the r velocity component is coming as minus a by 2 d, ok. So, I just I am giving that you can say background. So, now, this is what you got it. So, now, you can see this is a by 2 d, right.

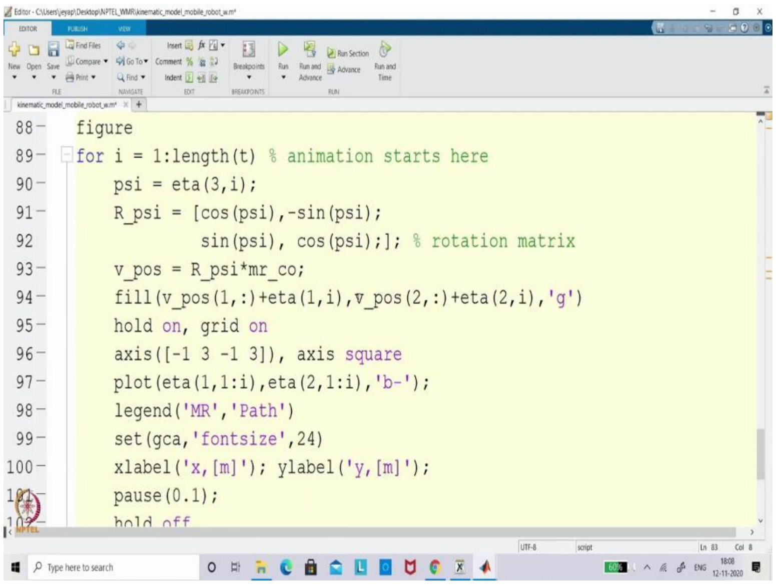
(Refer Slide Time: 12:17)



```
%% Animation (mobile robot motion animation)
l = 0.4; % length of the mobile robot
w = 0.2; % 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;];
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
```

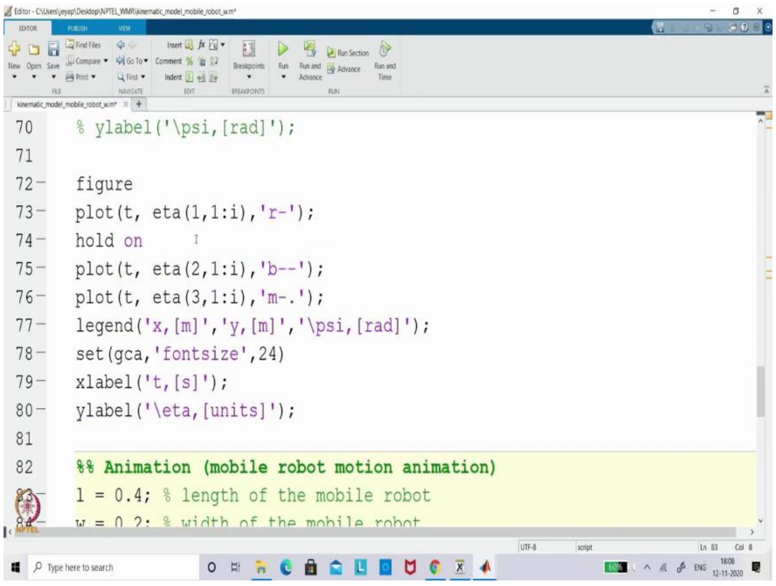
So, this is what your W matrix. So, right now we just incorporate what is W and ω and now we are trying to play this, ok. So, now, I will actually like take the same thing animation. So here, I have to change; because the width what I have taken is actually like 0.2 and the length I am actually like assuming it is 0.4, it is just you can say twice of the width. So, now, what else you need; here actually like there is no desired. So, I will actually like remove it that desired thing, ok.

(Refer Slide Time: 12:34)



```
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
```

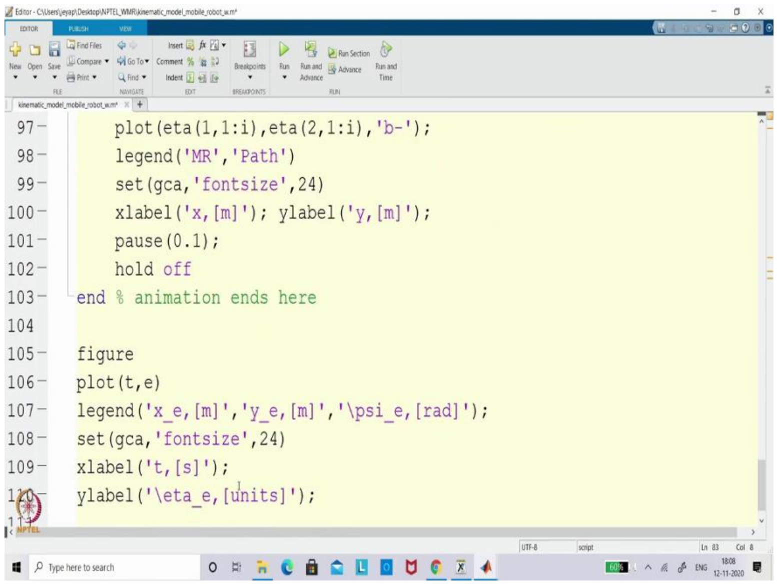
(Refer Slide Time: 12:38)



```
% ylabel('\psi,[rad]');

figure
plot(t, eta(1,1:i),'r-');
hold on
plot(t, eta(2,1:i),'b--');
plot(t, eta(3,1:i),'m-.');
legend('x,[m]','y,[m]','\psi,[rad]');
set(gca,'fontsize',24)
xlabel('t,[s]');
ylabel('\eta,[units]');

%% Animation (mobile robot motion animation)
l = 0.4; % length of the mobile robot
w = 0.2; % width of the mobile robot
```

So, now, if I run, this will give; you can see the x position, y position and angular value and then this animation code will actually like run it, right.

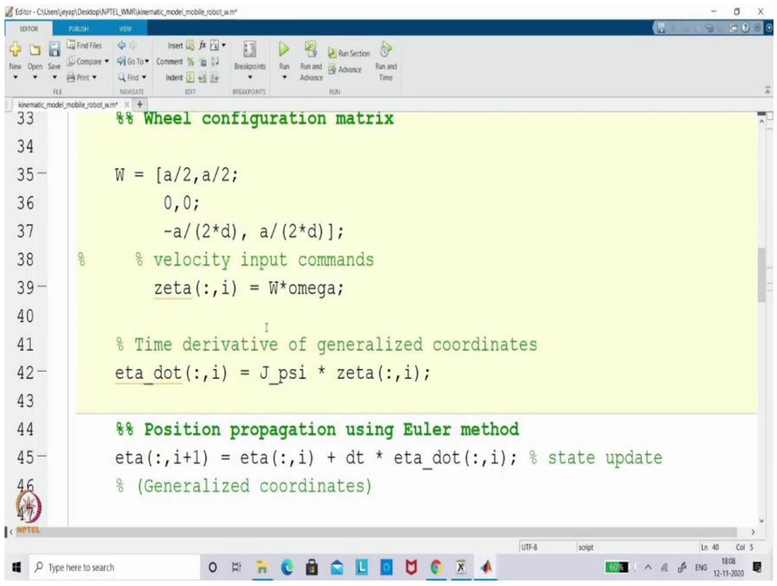
(Refer Slide Time: 12:47)



```
    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

figure
plot(t,e)
legend('x_e,[m]','y_e,[m]','\psi_e,[rad]');
set(gca,'fontsize',24)
xlabel('t,[s]');
ylabel('\eta_e,[units]');
```

So, now this error will not be there. So, this is the previous code where you have taken a inverse differential kinematics; in the sense this particular stuff also I will remove it.

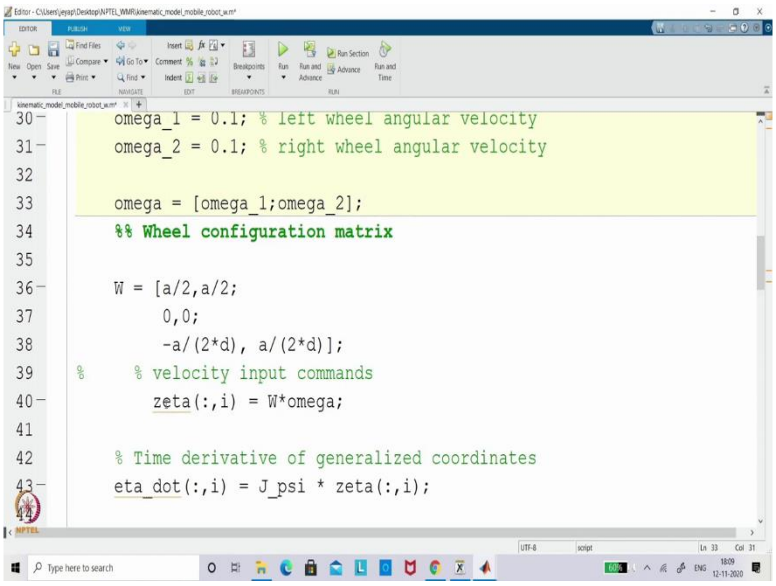
(Refer Slide Time: 12:56)



```
    %% Wheel configuration matrix

    W = [a/2,a/2;
         0,0;
         -a/(2*d), a/(2*d)];
%      % velocity input commands
       zeta(:,i) = W*omega;

    % Time derivative of generalized coordinates
    eta_dot(:,i) = J_psi * zeta(:,i);

    %% Position propagation using Euler method
    eta(:,i+1) = eta(:,i) + dt * eta_dot(:,i); % state update
    % (Generalized coordinates)
```

So, now, what we are trying to do? We are trying to give ω 1 and ω 2.

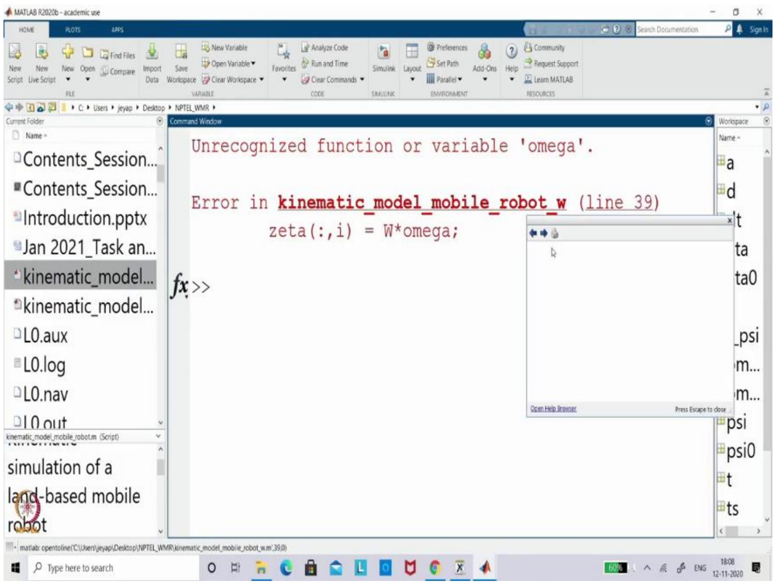
(Refer Slide Time: 13:05)



```
    omega_1 = 0.1; % left wheel angular velocity
    omega_2 = 0.1; % right wheel angular velocity

    omega = [omega_1;omega_2];
    %% Wheel configuration matrix

    W = [a/2,a/2;
         0,0;
         -a/(2*d), a/(2*d)];
%     % velocity input commands
        zeta(:,i) = W*omega;

    % Time derivative of generalized coordinates
    eta_dot(:,i) = J_psi * zeta(:,i);
```

And we are trying to see how that would be incorporated as a ξ, which is W × ω and we are trying to run, ok.

(Refer Slide Time: 13:23)

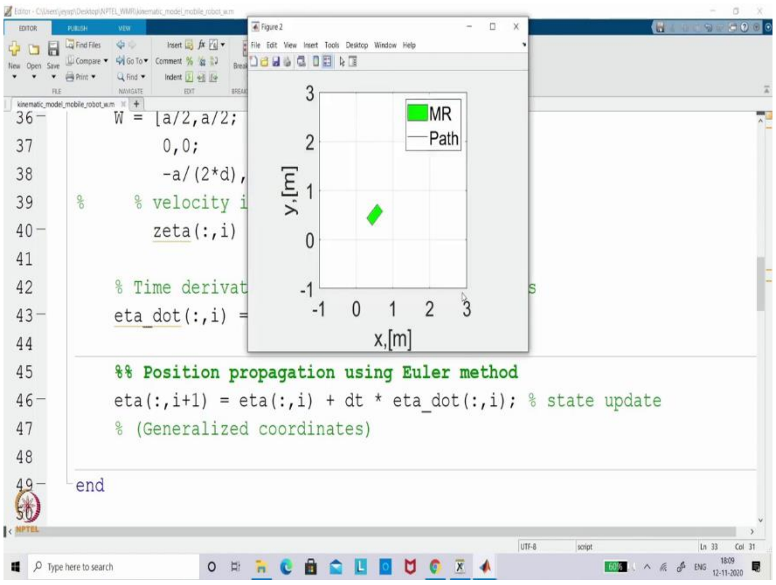


```
Unrecognized function or variable 'omega'.

Error in kinematic_model_mobile_robot_w (line 39)
        zeta(:,i) = W*omega;

fx >>
```

So, we will see whether this is having error free. So, far it shows no error; so obviously it is running. So, we can see; yeah, it is having a error, because we did not what mention what is ω, right. So, what is ω? ω is actually like vector of ω_1 and ω_2. So, I will write it that. So, ω is actually like vector of ω_1 and ω_2, right. So, now, these two are actually like incorporated in the sense you can actually like cross check. Now, ω is given and W is given, and you are recalculating ξ.

It is very very simple. So, you may feel it actually like it is just addition, but we will actually like try to do the inverse, then inverse kinematic simulation; then you will actually like see that, why this wheel configuration matrix is important, ok. So, right now we will actually like run it. So, I hope there is no error again; because we have taken the old code, right. So, that is why I was actually like bit of concern on the error side.

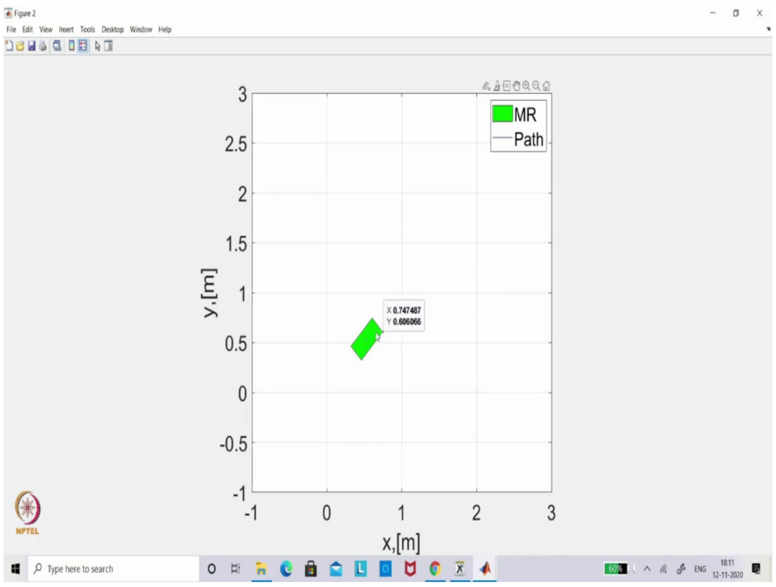
(Refer Slide Time: 14:22)



```
        W = [a/2,a/2;
             0,0;
             -a/(2*d),
%       % velocity i
            zeta(:,i)

        % Time derivat                                  s
        eta_dot(:,i) =

        %% Position propagation using Euler method
        eta(:,i+1) = eta(:,i) + dt * eta_dot(:,i); % state update
        % (Generalized coordinates)

    end
```

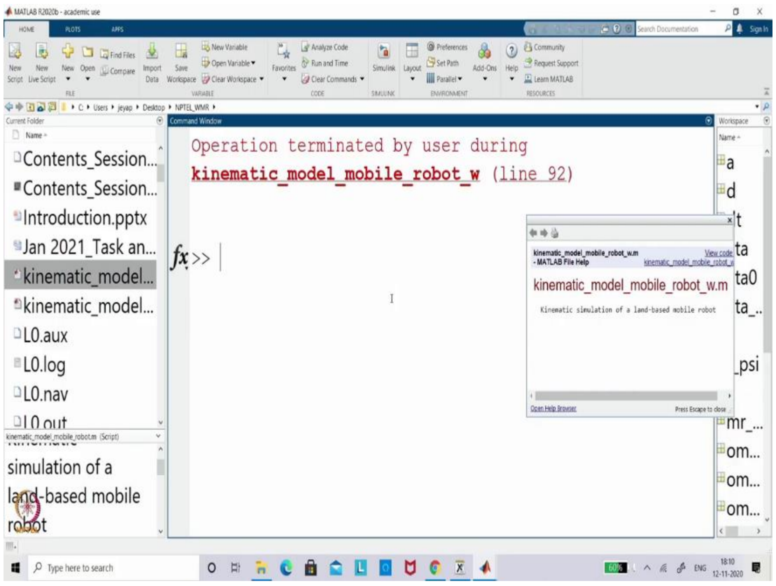
So, now, it actually like moved. So, now, what we have given; we have actually like given ω_1 and ω_2 = 0.1 radian/second and it is actually like trying to, you can see it tries supposed to go in a forward direction.

(Refer Slide Time: 14:32)



Already the vehicle is actually like inclined in what you call 45°. So, it is actually like trying to move in a forward direction; you can see like the vehicle is actually like very slowly moving, because we have given one second as the delay, ok. So, now, I will actually like go back, ok.

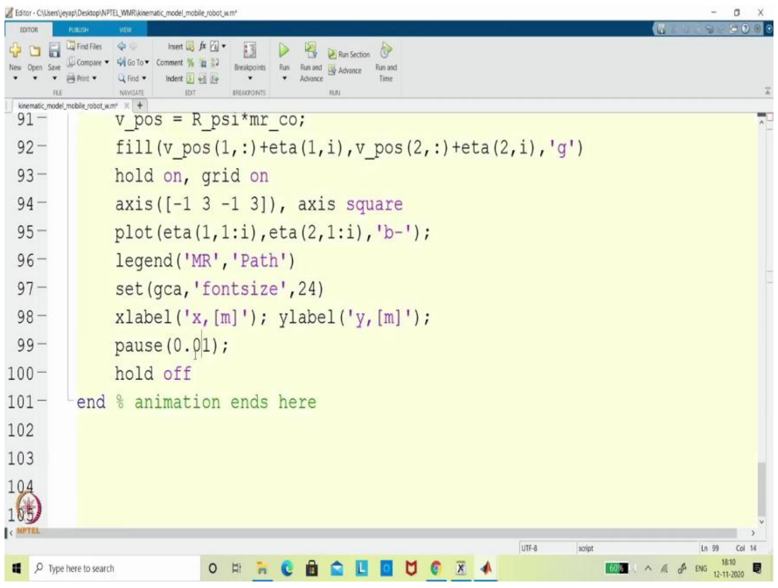
(Refer Slide Time: 14:54)



```
Operation terminated by user during
kinematic_model_mobile_robot_w (line 92)

fx >>
```

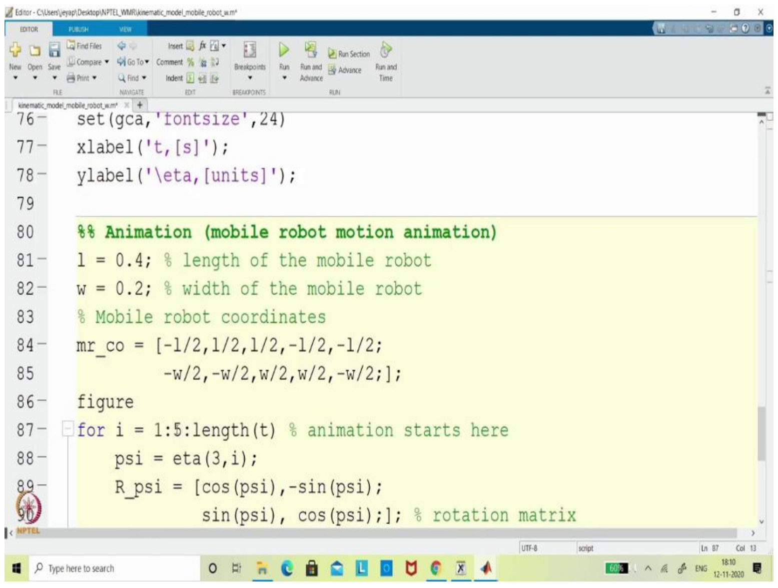
So, I will go back and change that delay, so that I can actually like make it.

(Refer Slide Time: 15:00)



```
    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(0.01);
    hold off
end % animation ends here
```

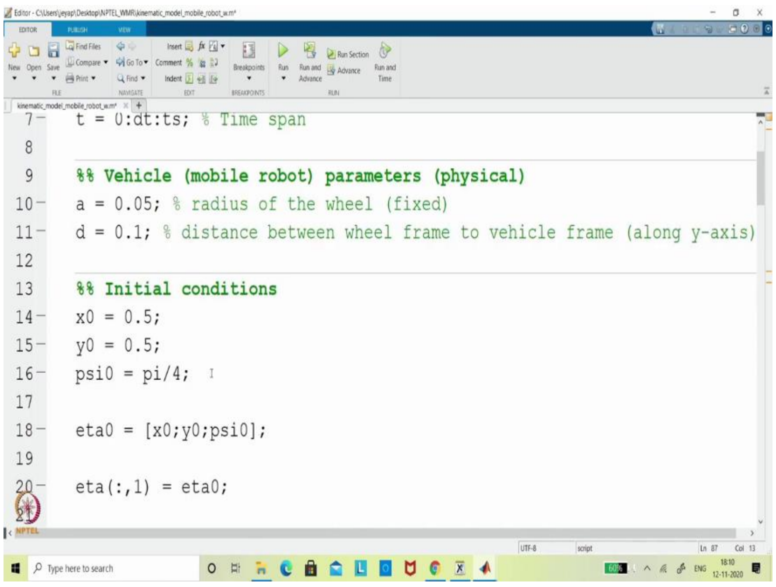
(Refer Slide Time: 15:07)



```
set(gca,'fontsize',24)
xlabel('t,[s]');
ylabel('\eta,[units]');

%% Animation (mobile robot motion animation)
l = 0.4; % length of the mobile robot
w = 0.2; % 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;];
figure
for i = 1:5:length(t) % animation starts here
    psi = eta(3,i);
    R_psi = [cos(psi),-sin(psi);
             sin(psi), cos(psi);]; % rotation matrix
```

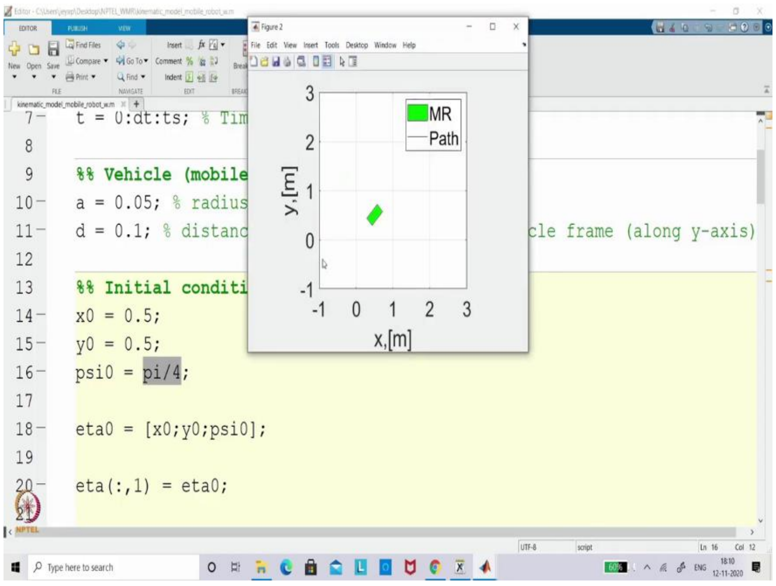
So, I will try to see this is five step size. So, I am just increasing the speed, ok. So, now, I am just seeing that this is 0.1 and this is also 0.1; in the sense both tangential velocity are equal, in the sense the vehicle will go forward.

(Refer Slide Time: 15:21)


```
t = 0:dt:ts; % Time span

%% Vehicle (mobile robot) parameters (physical)
a = 0.05; % radius of the wheel (fixed)
d = 0.1; % distance between wheel frame to vehicle frame (along y-axis)

%% Initial conditions
x0 = 0.5;
y0 = 0.5;
psi0 = pi/4;

eta0 = [x0;y0;psi0];

eta(:,1) = eta0;
```

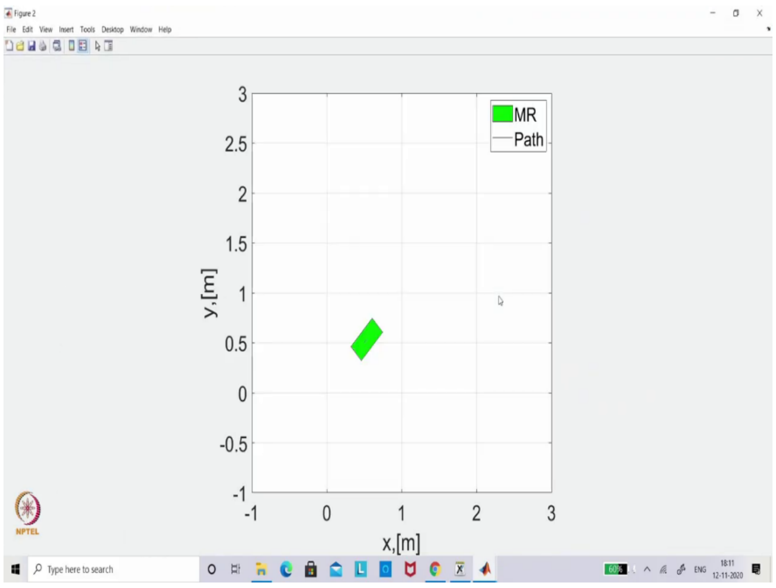
But if you look at the initial condition it is actually like π/4; in the sense it is looking in a 45 °line. So, it will move in a, you can say diagonal way.

(Refer Slide Time: 15:31)



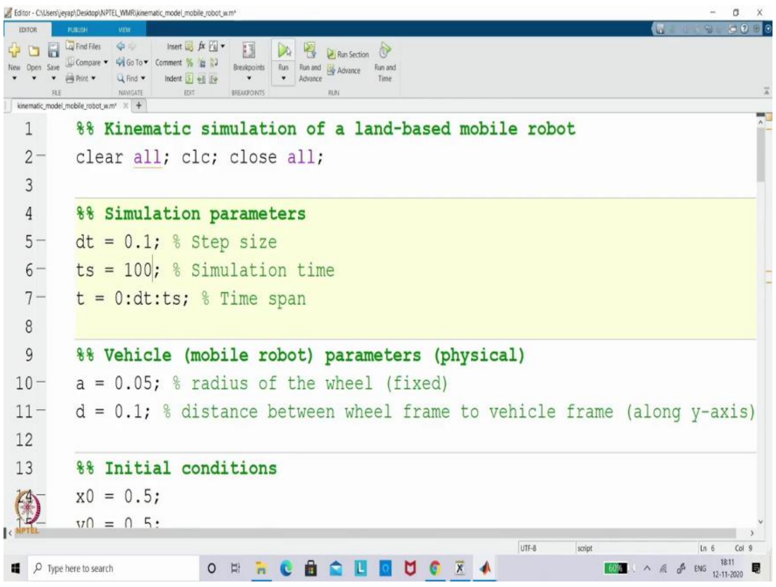
```
t = 0:dt:ts; % Tim
%% Vehicle (mobile
a = 0.05; % radius
d = 0.1; % distanc                 cle frame (along y-axis)
%% Initial conditi
x0 = 0.5;
y0 = 0.5;
psi0 = pi/4;

eta0 = [x0;y0;psi0];

eta(:,1) = eta0;
```

(Refer Slide Time: 15:33)



So, that is what you can actually like see it. So, now, so the vehicle is actually like; you can see this is the starting point, it is actually like moved diagonally, ok. So, now, you can actually like increase the what you call time.

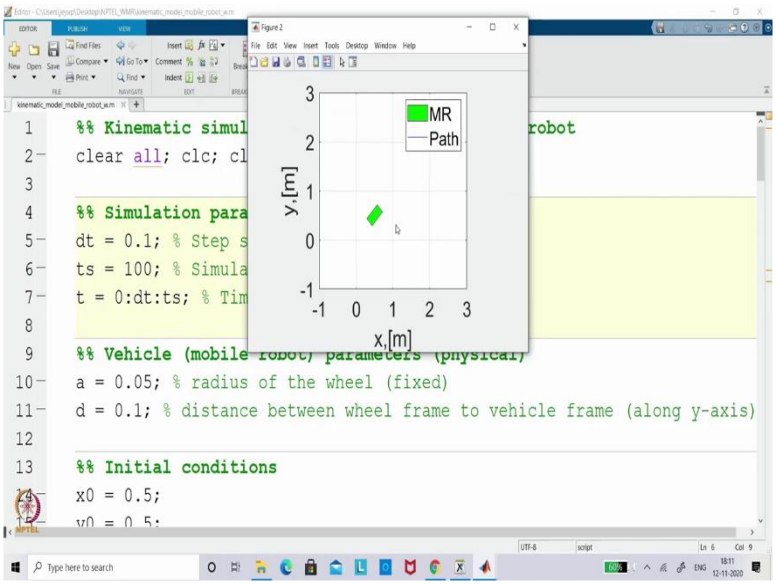
(Refer Slide Time: 15:46)



```
%% Kinematic simulation of a land-based mobile robot
clear all; clc; close all;

%% Simulation parameters
dt = 0.1; % Step size
ts = 100; % Simulation time
t = 0:dt:ts; % Time span

%% Vehicle (mobile robot) parameters (physical)
a = 0.05; % radius of the wheel (fixed)
d = 0.1; % distance between wheel frame to vehicle frame (along y-axis)

%% Initial conditions
x0 = 0.5;
y0 = 0.5;
```

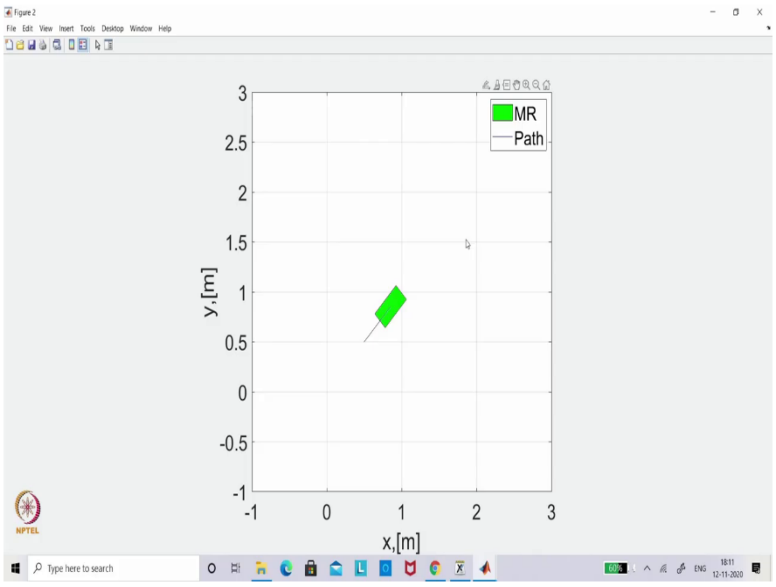
And you can actually like even find that. So, the how that is actually like happening; because it moved very small distance, right.

(Refer Slide Time: 15:50)



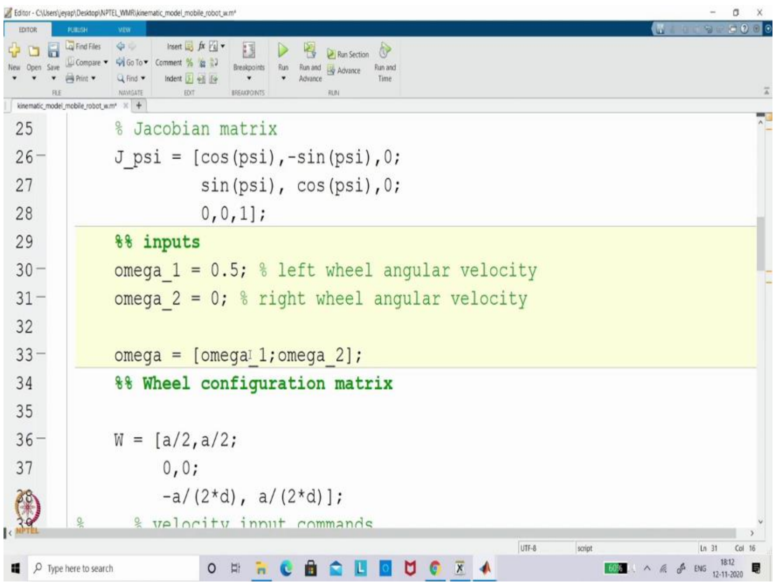
```
%% Kinematic simul                                    robot
clear all; clc; cl

%% Simulation para
dt = 0.1; % Step s
ts = 100; % Simula
t = 0:dt:ts; % Tim

%% Vehicle (mobile robot) parameters (physical)
a = 0.05; % radius of the wheel (fixed)
d = 0.1; % distance between wheel frame to vehicle frame (along y-axis)

%% Initial conditions
x0 = 0.5;
y0 = 0.5;
```

(Refer Slide Time: 15:52)



So, now, you can actually like see, you can wait; because my you can say screen resolution is actually like a very poor, it is showing although I make it a magnified this thing, it is showing as a small screen, ok. But you can see by even in the small screen, you can see that this green patch is actually like moving it. So, now, you can see it is a full size has come. So, it moved probably 0.5 to 0.52; now it is one by one, right. So, that is what it moved.

(Refer Slide Time: 16:28)

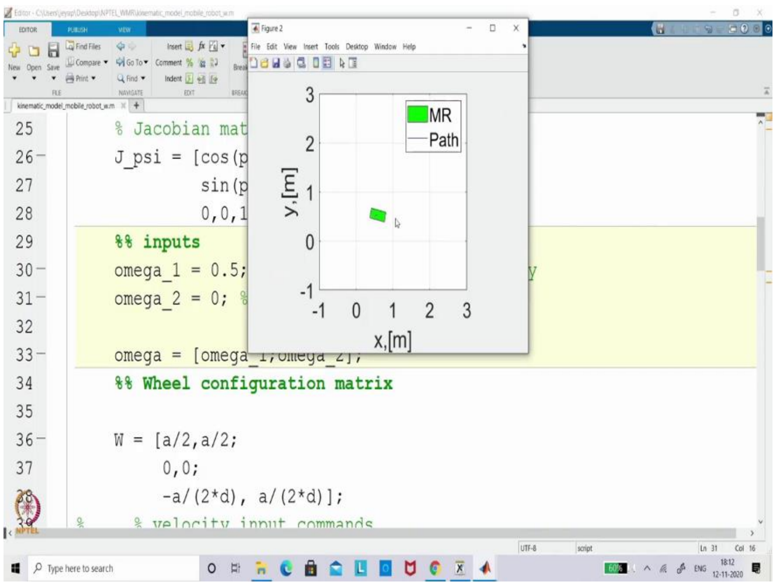


Editor - C:\Users\jeyap\Desktop\NPTEL_WMR\kinematic_model_mobile_robot_w.m*

```
% Jacobian matrix
J_psi = [cos(psi),-sin(psi),0;
         sin(psi), cos(psi),0;
         0,0,1];
%% inputs
omega_1 = 0.5; % left wheel angular velocity
omega_2 = 0; % right wheel angular velocity

omega = [omega_1;omega_2];
%% Wheel configuration matrix

W = [a/2,a/2;
     0,0;
     -a/(2*d), a/(2*d)];
%   % velocity input commands
```

So, now, in order to get more you can say beneficial, so I will just increase this and I am actually like giving a difference; in the sense the left wheel is actually like having 0.5 radian/second, whereas the right wheel is actually like giving 0.

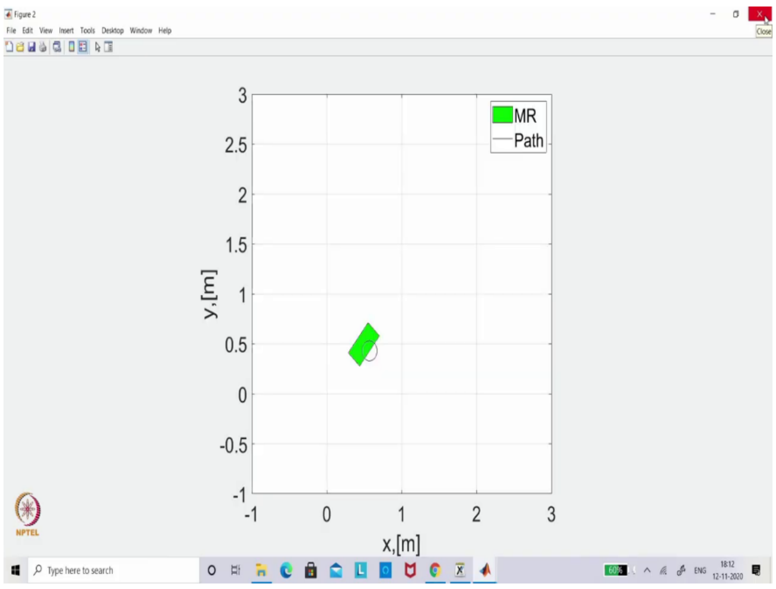
(Refer Slide Time: 16:41)



```
% Jacobian mat
J_psi = [cos(p
         sin(p
         0,0,1
%% inputs
omega_1 = 0.5;
omega_2 = 0; %

omega = [omega_1;omega_2];
%% Wheel configuration matrix

W = [a/2,a/2;
     0,0;
     -a/(2*d), a/(2*d)];
%   % velocity input commands
```

So, now if you are actually like running what one can actually like see it; it would actually try to rotate, you can see, right.

(Refer Slide Time: 16:44)



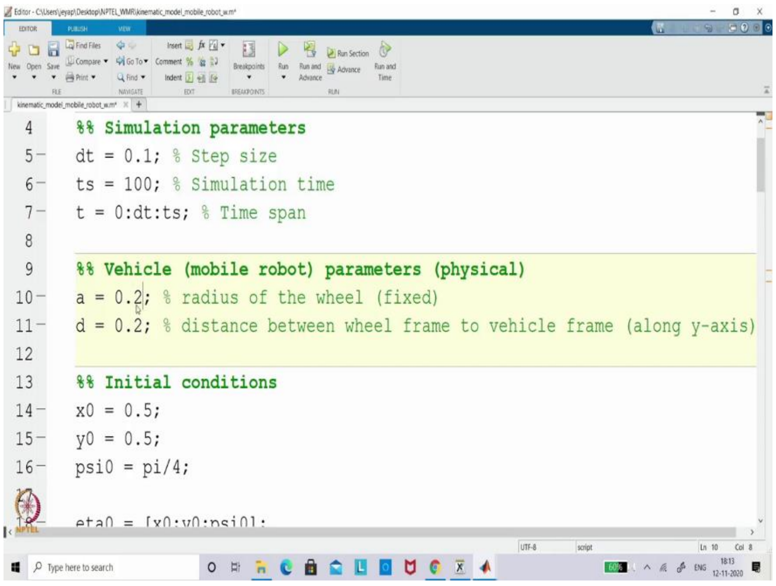
So, it is actually try to rotate and as well as try to move forward; but it is actually like having one side, so it is actually trying to make a loop and loop. So, it will actually like go in this. So, now, this is one of the turning radius you can find for this particular given thing.

So, now, you can actually like feel it why this is so small. So, now the play is coming. So, what that; I already told in the introduction class. So, what I said, how to select the wheel configuration, right. So, that we have known, but how to select the wheel size that is important, right.

Now, what we have given is the radius of the wheel is very small and the width of the vehicle also like very small 10; 10 centimeter is actually like, it is actually like small right, it is something like a toy. But you see the wheel size, wheel size is actually like 5 centimeter; in the sense you have a width, in the same size of the width you have your own wheel.

So, it is actually like not properly you can say configured, right. So, that is why this particular course is required. Now, you can see that, if you increase this you can say width as per the your proposition to the you call wheel size; then you will actually like feel it different.

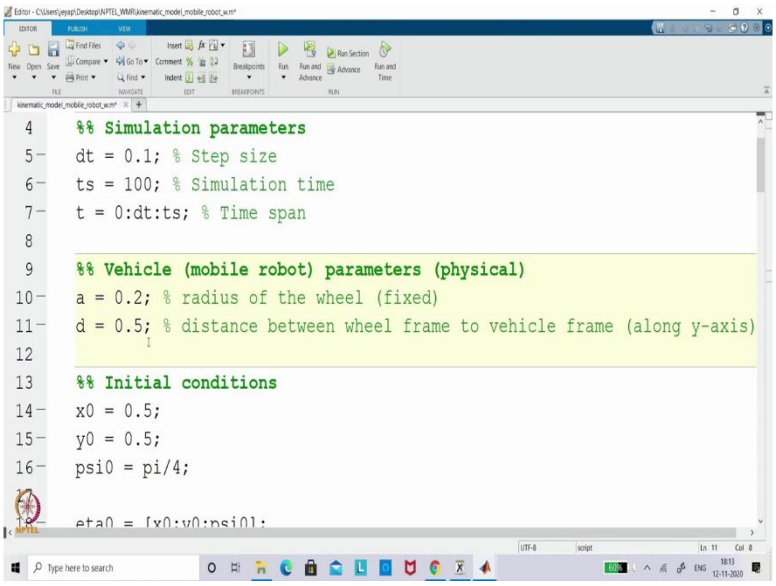
(Refer Slide Time: 17:54)



```
%% Simulation parameters
dt = 0.1; % Step size
ts = 100; % Simulation time
t = 0:dt:ts; % Time span

%% Vehicle (mobile robot) parameters (physical)
a = 0.2; % radius of the wheel (fixed)
d = 0.2; % distance between wheel frame to vehicle frame (along y-axis)

%% Initial conditions
x0 = 0.5;
y0 = 0.5;
psi0 = pi/4;

eta0 = [x0;y0;psi0];
```

Similarly, when you want faster speed in the sense go faster; so definitely you know like the a is the playing role, because the ω is actually like you would be giving 1 radian or 2 radian based on the motor. But what actually like go in to generate the traction or you call tangential velocity; that would be coming based on what you call a dependent.

(Refer Slide Time: 18:23)

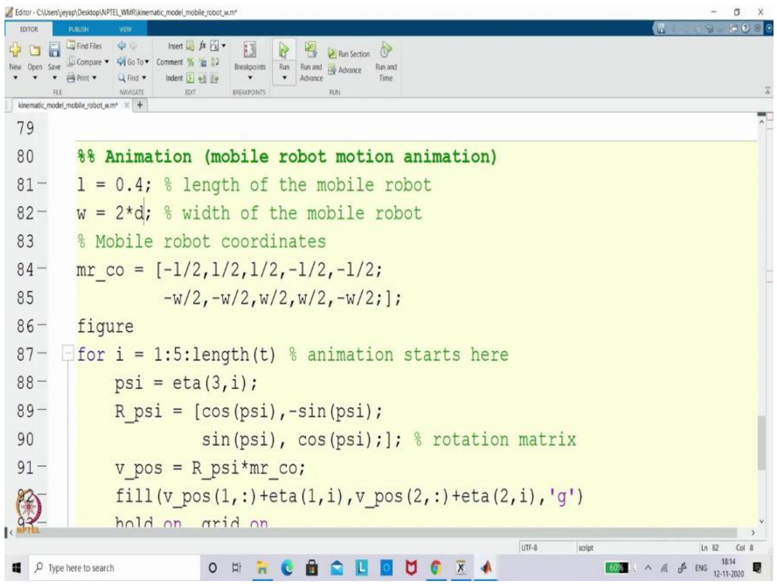


```
%% Simulation parameters
dt = 0.1; % Step size
ts = 100; % Simulation time
t = 0:dt:ts; % Time span

%% Vehicle (mobile robot) parameters (physical)
a = 0.2; % radius of the wheel (fixed)
d = 0.5; % distance between wheel frame to vehicle frame (along y-axis)

%% Initial conditions
x0 = 0.5;
y0 = 0.5;
psi0 = pi/4;

eta0 = [x0;y0;psi0];
```

So, now if I actually like increase this a just for understanding, you see it is actually like 20 centimeter and I am assuming that the depth is probably you can say 1 meter size, where actually like 20 like 40 centimeter as a dia wheel, which is very close to your trolley, ok.

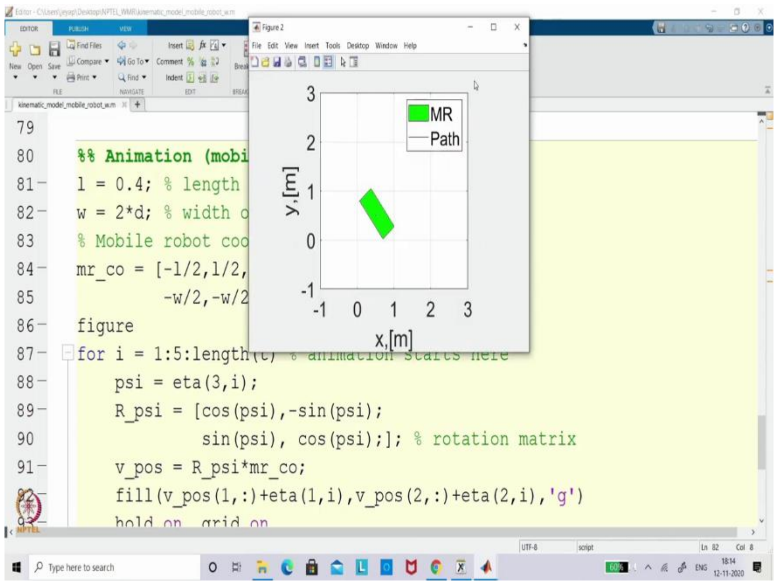
So, now, I am actually like putting it. So, this as you can say 2d, so that I no need to change every time. So, now what you will actually like see, it is actually like going little faster right and as well as you can see that visibility.

(Refer Slide Time: 18:40)



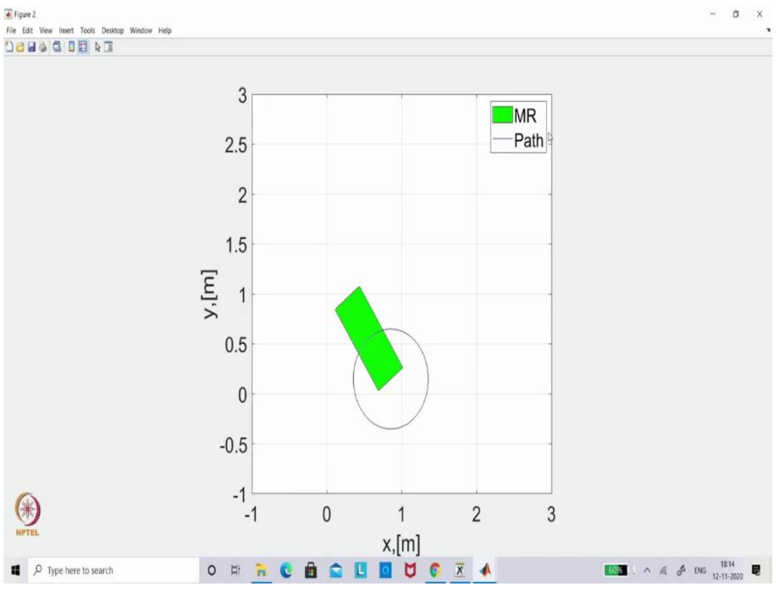
```
%% Animation (mobile robot motion animation)
l = 0.4; % length of the mobile robot
w = 2*d; % width of the mobile robot
% Mobile robot coordinates
mr_co = [-l/2,l/2,l/2,-l/2,-l/2;
         -w/2,-w/2,w/2,w/2,-w/2;];
figure
for i = 1:5:length(t) % animation starts here
    psi = eta(3,i);
    R_psi = [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
```

(Refer Slide Time: 18:46)



```
%% Animation (mobi
l = 0.4; % length
w = 2*d; % width o
% Mobile robot coo
mr_co = [-l/2,l/2,
         -w/2,-w/2
figure
for i = 1:5: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
```

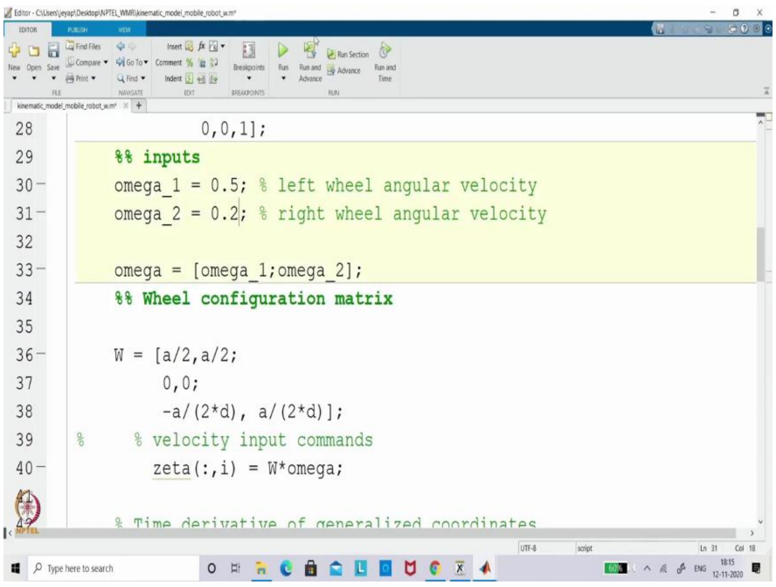
(Refer Slide Time: 18:47)



Because earlier what you have seen, the diameter of the wheel is actually like you can say 10 centimeter and your width of the vehicle also like you can say approximately 20 centimeter, which is actually like non comparable one. So, now, you can see like this is one side angular velocity only given; in the sense only one wheel is actually like rotating, the other wheel is actually like giving a resistant.

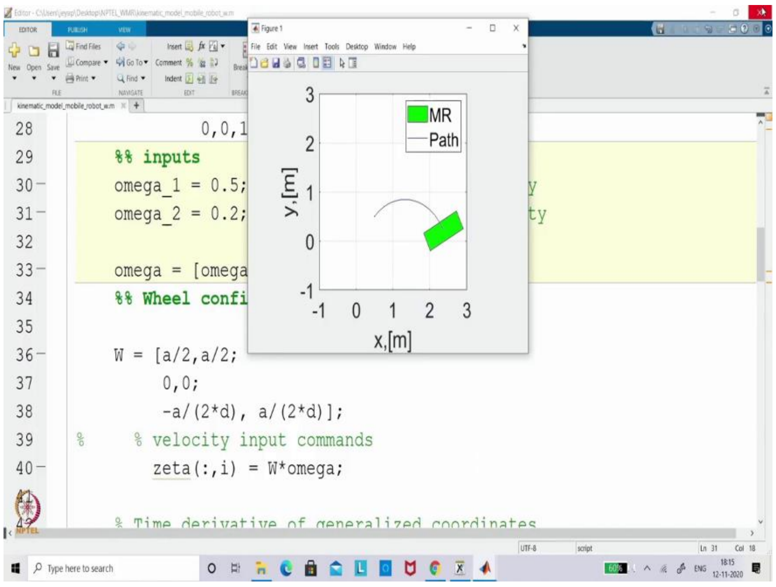
So, this is what the phenomena of fixed wheel; this is you can actually like feel it when you are dragging probably trolley suitcase, where one wheel gets stuck, in the sense it is actually like jammed and you are driving or you can say you are dragging. So, the other wheel start rotating; then you can see that the suitcase is start rotating right, the same scenario you can feel it.

(Refer Slide Time: 19:37)

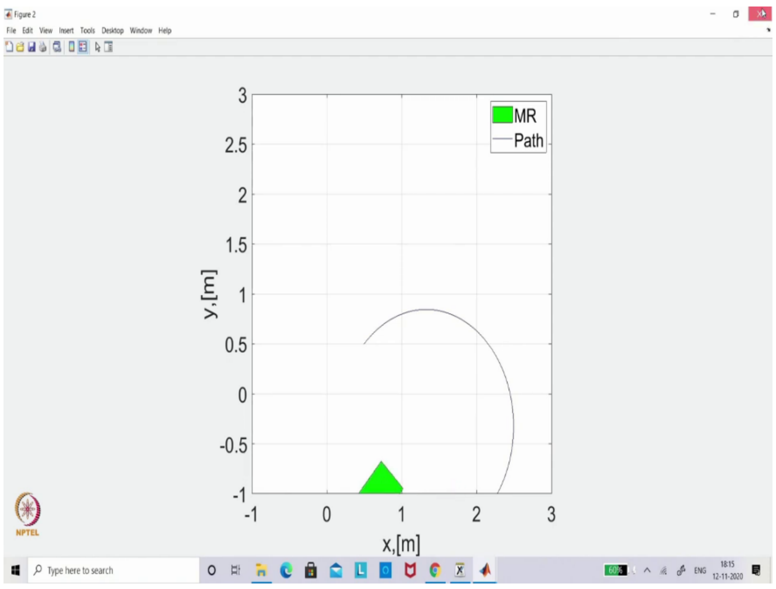

```
                  0,0,1];
    %% inputs
    omega_1 = 0.5; % left wheel angular velocity
    omega_2 = 0.2; % right wheel angular velocity

    omega = [omega_1;omega_2];
    %% Wheel configuration matrix

    W = [a/2,a/2;
         0,0;
         -a/(2*d), a/(2*d)];
%      % velocity input commands
       zeta(:,i) = W*omega;

    % Time derivative of generalized coordinates
```

So, now I am actually like giving a slightly a less comparison. So, now, what happened? There would be a resultant velocity; but still you can see that the left side wheel is dominating.

(Refer Slide Time: 19:43)

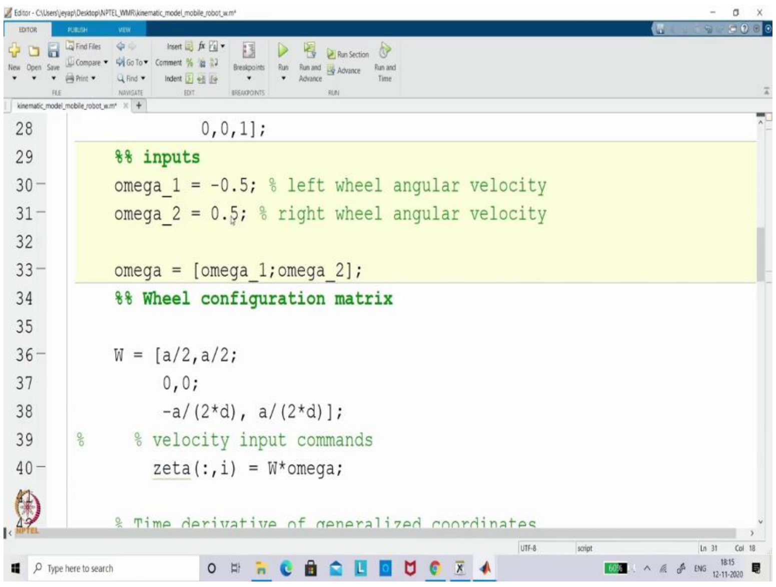


(Refer Slide Time: 19:46)



You can see earlier it was simply one small circle it make; but now it will try to make a spiral, ok. So, now, this is what the, what you call idea behind the differential wheel model. So, now, you got it little more clarity. So, now, we will actually like move it

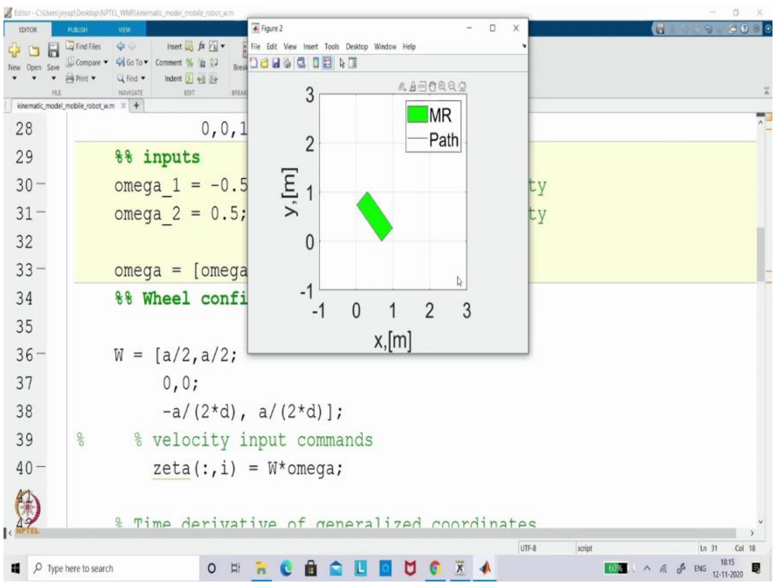
(Refer Slide Time: 20:06)



```
            0,0,1];
    %% inputs
    omega_1 = -0.5; % left wheel angular velocity
    omega_2 = 0.5; % right wheel angular velocity

    omega = [omega_1;omega_2];
    %% Wheel configuration matrix

    W = [a/2,a/2;
         0,0;
         -a/(2*d), a/(2*d)];
%     % velocity input commands
        zeta(:,i) = W*omega;

    % Time derivative of generalized coordinates
```

So, just I am giving equal and opposite angular velocity; in the sense I connect both the motor and I gave actually like same angular speed, but unknowingly actually like I put a polarity change, just imagine. So, then what one can expect? The vehicle will rotate its own point right; you can see that it is happening or not, right.

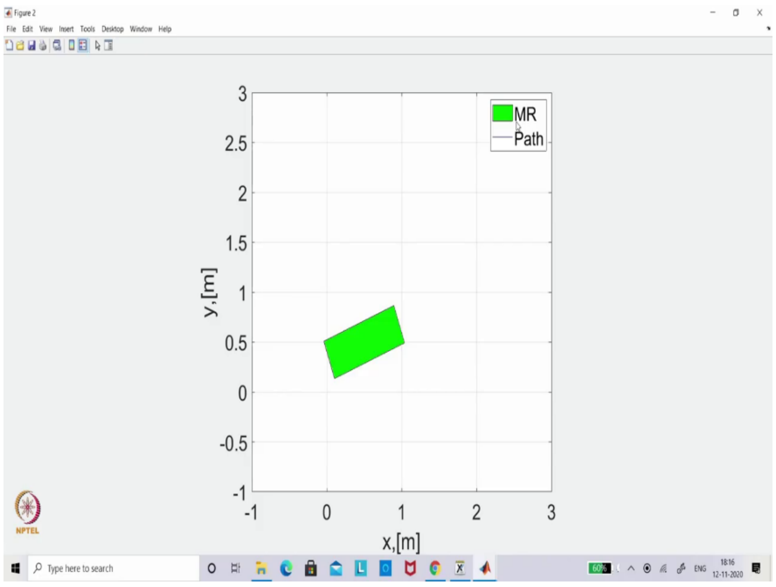
(Refer Slide Time: 20:25)



```
                    0,0,1
    %% inputs
    omega_1 = -0.5                               ty
    omega_2 = 0.5;                               ty

    omega = [omega
    %% Wheel confi

    W = [a/2,a/2;
         0,0;
         -a/(2*d), a/(2*d)];
%     % velocity input commands
      zeta(:,i) = W*omega;

    % Time derivative of generalized coordinates
```

It is actually like rotating about its own axis; why? Because both are actually like rotating, it is trying to make a couple that is what you can actually like feel it.

(Refer Slide Time: 20:36)



So, now what one can actually like see; this all the aspect of differential wheel you have seen, anyhow now you know how to write a code. So, now, you can actually like play further by changing ω change; like changing ω_1, ω_2 and you change a and d and even actually like you can try to change your initial condition and see how it is happening, so that you will get something like close to the original intention of this particular course will get it.

So, by doing this, keep on doing this what one can actually like get? So, we will get much more exposure. So, that is what supposed to be required. So, now, we have seen the, you can say differential wheel drive kinematic simulation by varying ω_1 and ω_2; that too like we have seen like if you take ω_1 and ω_2 opposite direction, you have seen that it is rotating about its point.

And if ω_1 is dominate, then that would take a left hand, in the sense it will rotate in a clockwise; where the ω_2 is dominant, then it would actually like rotate in the counterclockwise direction. If both are actually like equal, then the vehicle move in the forward. At any case the vehicle will not move in the lateral direction, until you take 90° turn and then move it.

So, in that sense what one can see, like this differential wheel drive kinematics simulation you have we have seen. So, the next part what we will come back. So, we will take two other example which we have we were actually like discuss in the beginning. So, the Omni wheel, you can say Omni directional wheel drive and as well as Mecanum wheel drive, we will see as a part 2 and part 3.

So, in that sense, so the next lecture what you can expect; something like more spice, because it is having actually like square matrix or the rectangular matrix with actually like more number of you can say columns. So, then you can actually like say that, lateral direction also possible; how that is happening that we will see in the part 2 and 3, ok. With that we will see, till then bye.