Mechanics and Control of Robotic Manipulators

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

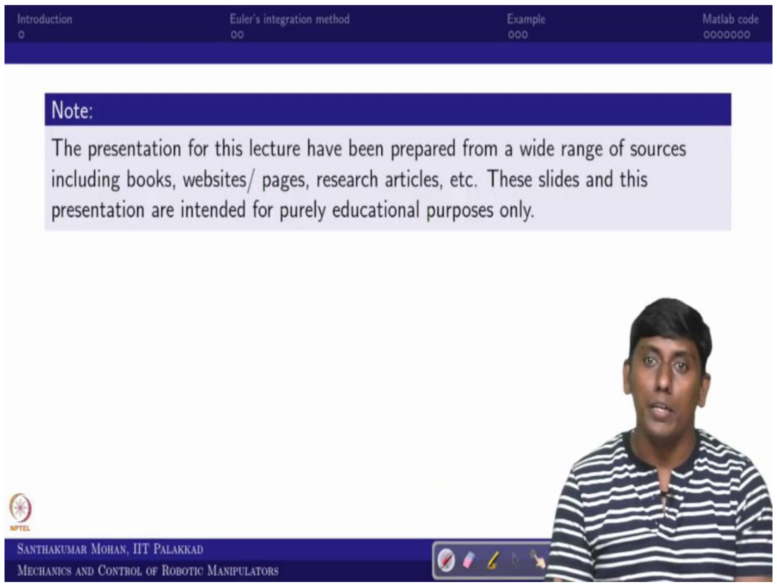
Lecture No. 28

Dynamic Model Derivation Using Newton-Euler Method in MATLAB

Hi, welcome back to mechanics and control of robotic manipulator. Last few classes what we have seen is how to derive a dynamic model of a robotic manipulator in specific serial manipulator. We have seen two methods. So, these even MATLAB, or you can say using MATLAB how to derive the equation of motion also, we have seen.

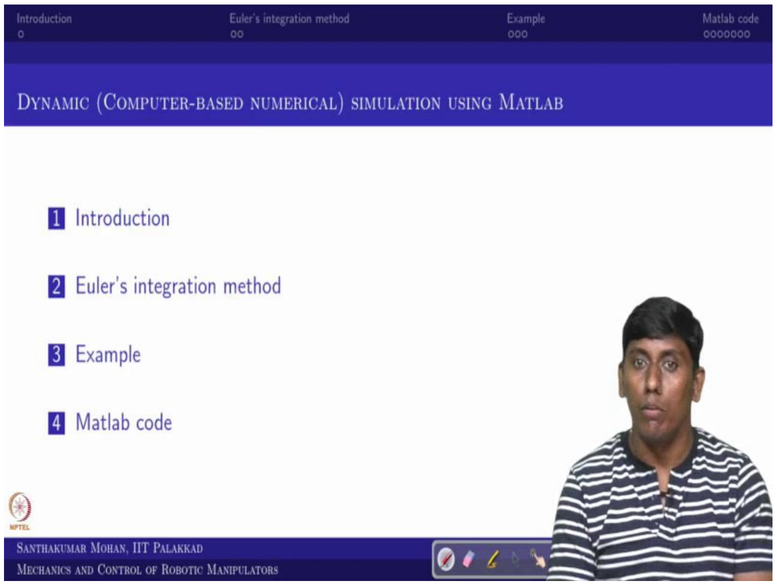
This particular class or this particular lecture, we are going to talk about how to do a dynamic simulation. So, whatever we derive that equation that equation how we can use for a dynamic simulation model.

(Refer Slide Time: 0:43)


Introduction
Euler's integration method
Example
Matlab code
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

Note:

The presentation for this lecture 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.



Introduction
Euler's integration method
Example
Matlab code

Dynamic (Computer-based numerical) simulation using Matlab

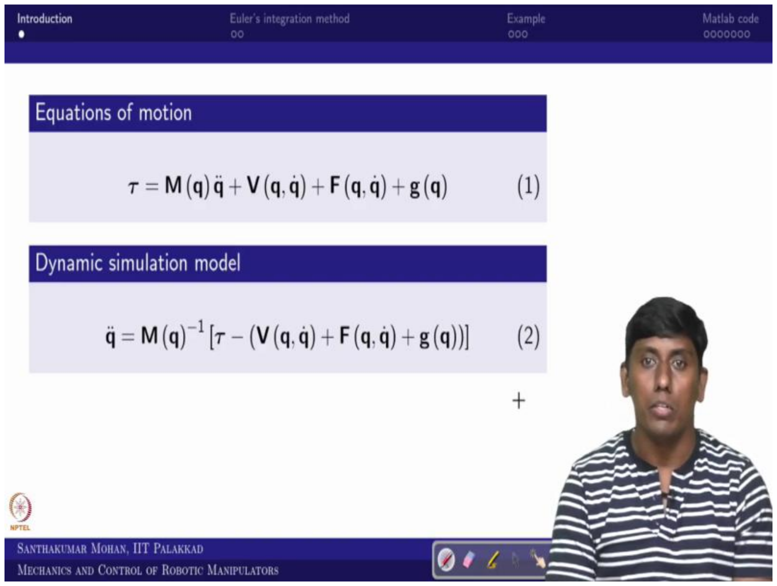
1. Introduction
2. Euler's integration method
3. Example
4. Matlab code

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, in the sense we are trying to see basically like you can say the equation of motion we are taking and trying to do the forward dynamics in this particular lecture. So, for that we need to have some numerical integration method. So, here we are going to talk about Euler integration method that would be very simple. It is a first order integration method that we can use if you are necessary or you feel like it is necessary, even we can use second order or fourth order you can say numerical integration methods.

But here we are going to use the simple first order which is what we call Euler integration method. Then we will take one simple example and we will see how that example can be ported in MATLAB and based on that how we can do the dynamic simulation along with what you call motion animation.

(Refer Slide Time: 1:33)


Introduction | Euler's integration method | Example | Matlab code

Equations of motion

$$\tau = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) \quad (1)$$

Dynamic simulation model

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}[\tau - (\mathbf{V}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}))] \quad (2)$$

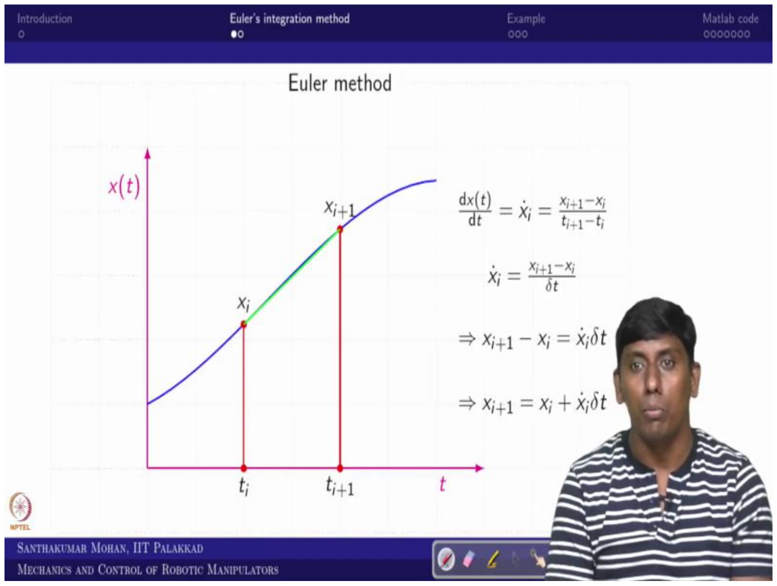
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, this is all what we are trying to cover in that sense the generalized you can say equation of motion of robotic manipulator can be written in this form. So, this equation we can modify for a forward dynamics, although we are doing in a forward dynamics in the sense what we know the tau is known. And we are trying to find out what is the motion. In that sense, we will start with the initial stage initially the; you call system variable called q q dot would be known.

Then tau is known, then how we can get q double dot this is what we are trying to do. So, for that we are calling this is a dynamic simulation model. We are writing everything in acceleration form. In the sense so, q double dot can be written in this simple equation. So, now, you can see here the initial condition q q dot and the input tau are known and the frictional quantity gravity vector and all other vectors can be found based on that.

If we even write this in you can say you call even if we draw in a block diagram base, it is very clear. But now, we will do it in MATLAB. In the next lecture, we will see in a block diagram approach. So, in the sense that the MATLAB case we need to write everything in a code form.

(Refer Slide Time: 2:50)


Introduction
Euler's integration method
Example
Matlab code
$x(t)$
x_{i+1}
x_i
t_i
t_{i+1}
t
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

Euler method

$$\frac{dx(t)}{dt} = \dot{x}_i = \frac{x_{i+1}-x_i}{t_{i+1}-t_i}$$

$$\dot{x}_i = \frac{x_{i+1}-x_i}{\delta t}$$

$$\Rightarrow x_{i+1} - x_i = \dot{x}_i \delta t$$

$$\Rightarrow x_{i+1} = x_i + \dot{x}_i \delta t$$

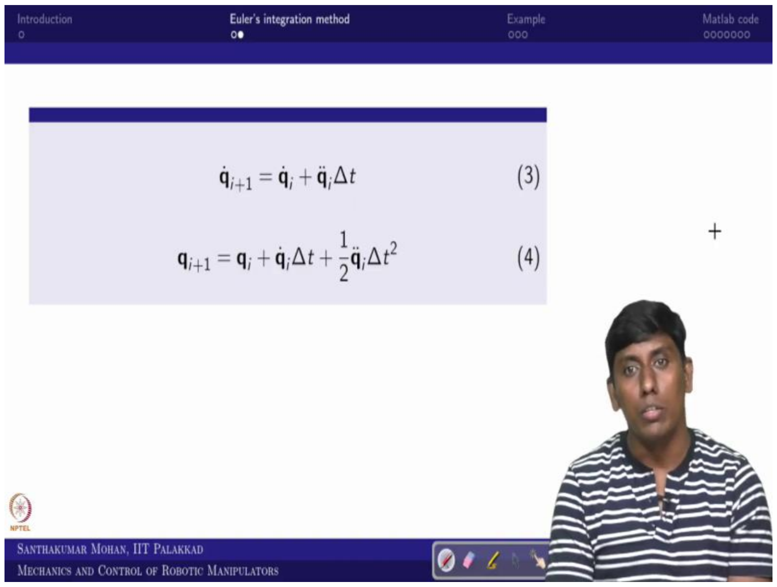
So, for doing the numerical integration, we will try to use the Euler method. So, how we can get the Euler method by you can say demonstrating in this particular plot. So, you can see like here that x of t and the time we have achieved here in a plot. Where time is the independent variable, and the x of t is the dependent variable. So, right now the x of t is given in this particular polynomial.

So, this particular curve. So, now what I am trying to take I am trying to take as a random ti point. So, in a sense first instant I am taking so, what is the corresponding value I call x of i. So, I am taking another step which is we call t i plus 1. So, that would be correspond to you can say x of or x i plus 1. Now, although the curve is somewhat curved in the sense, it is not straight line. But I am trying to make a straight line.

You can see in a green line, so, where xi and xi plus one it is connecting. So, now if we see the straight line. What would be the slope? The slope would be x dot of i that x dot of i how I can write x of i plus 1 or xi plus 1 minus xi whole divided by ti plus 1 minus ti, So, this is the way we can write. So, now based on the slope, one can easily find if I know x of i and I know x dot of i and if I know what is the time step I am going to consider then I can find x i plus 1.

So, in the sense this equation what I have written I can rewrite, so, x dot of i multiply with this we call time step. And then I can rewrite this equation in the sense I consider this as a delta t. So, then you can see x dot of i into delta t plus x of i is equal to xi plus 1. So, this is what we are going to call as a numerical integration that to like Euler’s method. So, in the sense this equation is going to use hereafter in the dynamic simulation for our robotic manipulator.

(Refer Slide Time: 5:04)


Introduction
Euler's integration method
Example
Matlab code
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

$$\dot{\mathbf{q}}_{i+1} = \dot{\mathbf{q}}_i + \ddot{\mathbf{q}}_i \Delta t \tag{3}$$

$$\mathbf{q}_{i+1} = \mathbf{q}_i + \dot{\mathbf{q}}_i \Delta t + \frac{1}{2}\ddot{\mathbf{q}}_i \Delta t^2 \tag{4}$$

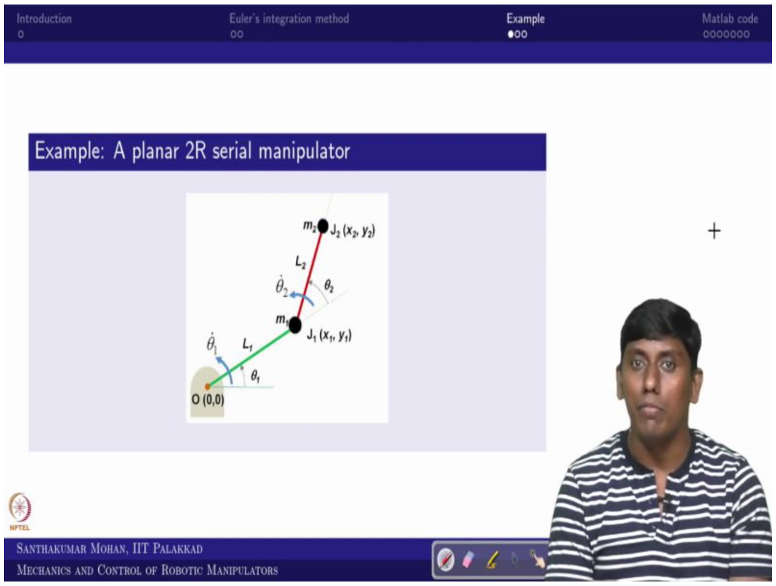
So, now if that is the case how I can write in a generalized form in robotic manipulator. Here there are two things will come. Because the robotic system is the equation of motion written in second order equation. In the sense so, q dot can be obtained by taking q double dot as the input. Similarly, like q we can get it from q dot.

So, in that sense this is straightforward from the Euler integration. But this we can rewrite in that way. So, we know equation of motion for a simple particle where we call you can say v; v equal to u plus at and s is equal to s naught plus ut plus half at squared. The same way here this is equal into what you call the velocity relation. So, in the sense q dot i plus 1 is the next one. So, that can be obtained from the qi q dot i plus q double dot of i into delta t.

So, right now, we want to go with the position, then you can see like it depends on you can say initial velocity initial position and initial acceleration at top. So, in that sense what we can rewrite this equation. So, this is further we are differentiating. So, that is what we are trying to do. So, now this is integrating it. So, if we integrate what we get this is going to be q i plus 1 and this is going to be q i and this will get something addition. Why?

Because this is going to give two set of equation. So, that is what we have written. So, in the sense you can see the q dot easily we can obtain. So, once we obtain the q dot we can go to q. So, that is a way we can a proceed.

(Refer Slide Time: 6:55)



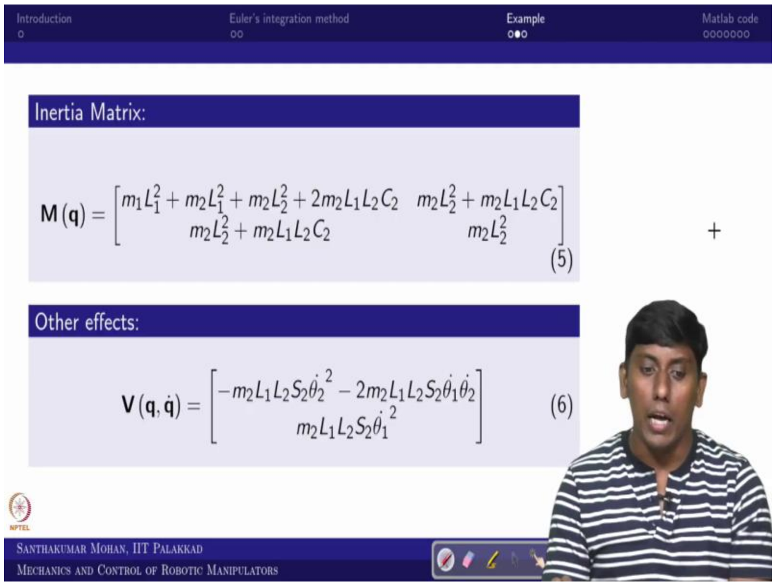
Introduction | Euler's integration method | Example | Matlab code

Example: A planar 2R serial manipulator

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, if that is the case, how we can take it to a MATLAB you can see simulation for that we will take one of the simplest examples which all studied so, far in all the; you can say constraint for example, you talk about velocity or you talk about equation of motion. We have taken planar 2R serial manipulator. The same example we will take.

(Refer Slide Time: 7:17)



Introduction | Euler's integration method | Example | Matlab code

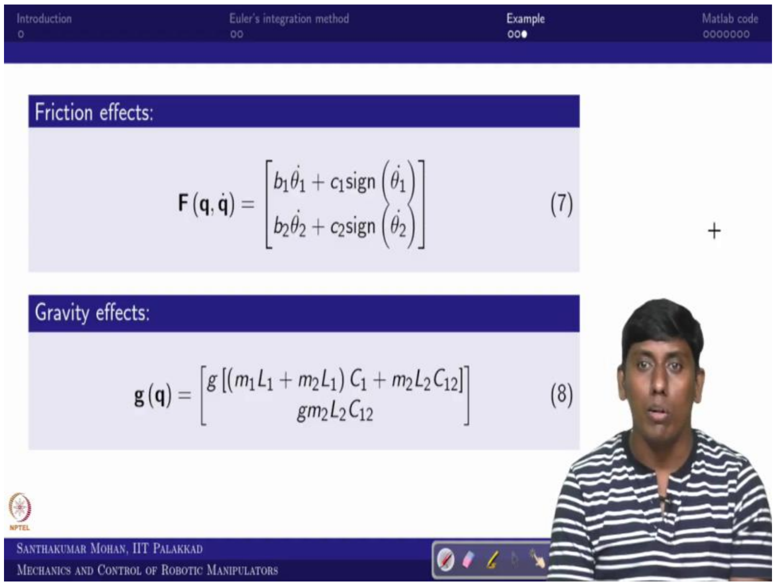
Inertia Matrix:

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} m_1L_1^2 + m_2L_1^2 + m_2L_2^2 + 2m_2L_1L_2C_2 & m_2L_2^2 + m_2L_1L_2C_2 \\ m_2L_2^2 + m_2L_1L_2C_2 & m_2L_2^2 \end{bmatrix} \quad (5)$$

Other effects:

$$\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -m_2L_1L_2S_2\dot{\theta}_2^2 - 2m_2L_1L_2S_2\dot{\theta}_1\dot{\theta}_2 \\ m_2L_1L_2S_2\dot{\theta}_1^2 \end{bmatrix} \quad (6)$$

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


Introduction
Euler's integration method
Example
Matlab code

Friction effects:

$$\mathbf{F}(\mathbf{q},\dot{\mathbf{q}}) = \begin{bmatrix} b_1\dot{\theta}_1 + c_1\text{sign}\left(\dot{\theta}_1\right) \\ b_2\dot{\theta}_2 + c_2\text{sign}\left(\dot{\theta}_2\right) \end{bmatrix} \quad (7)$$

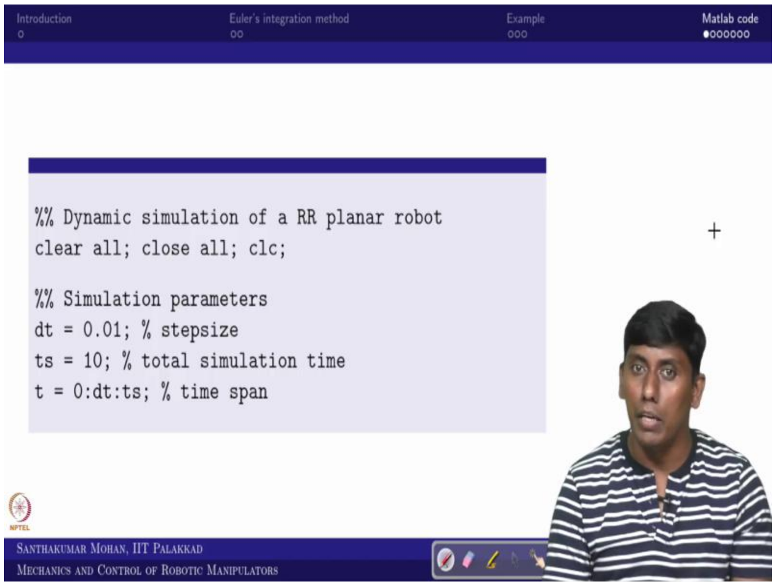
Gravity effects:

$$\mathbf{g}(\mathbf{q}) = \begin{bmatrix} g\left[(m_1L_1 + m_2L_1)\,C_1 + m_2L_2C_{12}\right] \\ gm_2L_2C_{12} \end{bmatrix} \quad (8)$$

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, what we have, so, M of q is known so, then other effort v q comma q dot is known. And g of you can say f of q comma q dot also, known and the g of q is not the sense all you can say four subcomponents are known. So, now if I know tau, how can I get this with the initial condition? Because this is ordinary differential equation for solving ordinary differential equation, you need one of the boundary condition, here we call initial condition. If you know initial condition and input, then you can solve this particular equation.

(Refer Slide Time: 7:55)


Introduction
Euler's integration method
Example
Matlab code

```
%% Dynamic simulation of a RR planar robot
clear all; close all; clc;

%% Simulation parameters
dt = 0.01; % stepsize
ts = 10; % total simulation time
t = 0:dt:ts; % time span
```

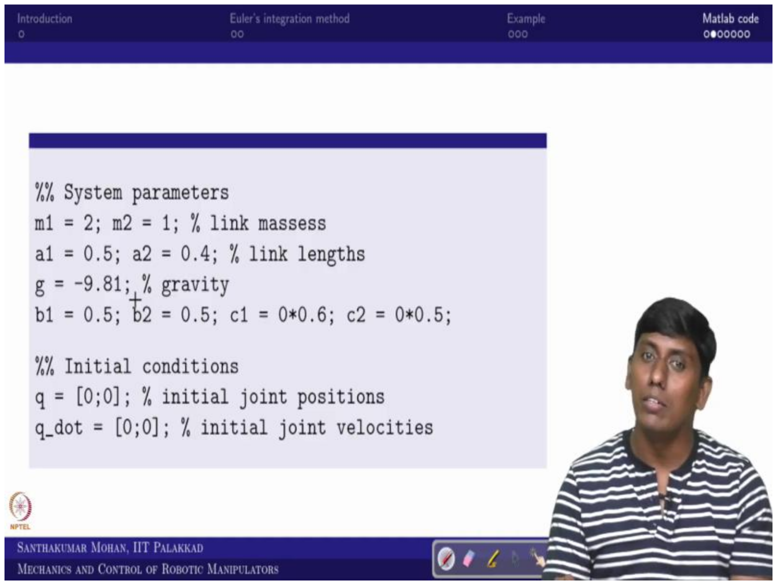
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, that is what we are trying to do for that we are taking simulation code. So, we are again taking it everything we are making it clear and sometimes we are making a plot. So, that is why we are closing all the figure window and clearing the command history. So, that we can understand what was the error command or what is the output. Then what we are going since it is having a several steps, first, we need to define the integration parameter.

So, here we are doing Euler integration. Euler integration is a simple first order system. So, the dt is what equal to delta t which is nothing but the step size. Here I have taken as a 10 millisecond it did not to be, but we have taken 10 millisecond. Further, we need to see what simulation range you want to run this particular MATLAB code. So, that is what I call total simulation time which is I have written as ts.

Then, this is going to actual like a propagate from 0 to ts with the integral of delta t. In the sense I am defining the time span which is starting t starting from 0 to ts with the interval of you can see our step size of dt here. So, this is the first case we have defined the simulation parameter. Once we know the simulation parameter then we can go to the system.

(Refer Slide Time: 9:13)


Introduction
Euler's integration method
Example
Matlab code
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

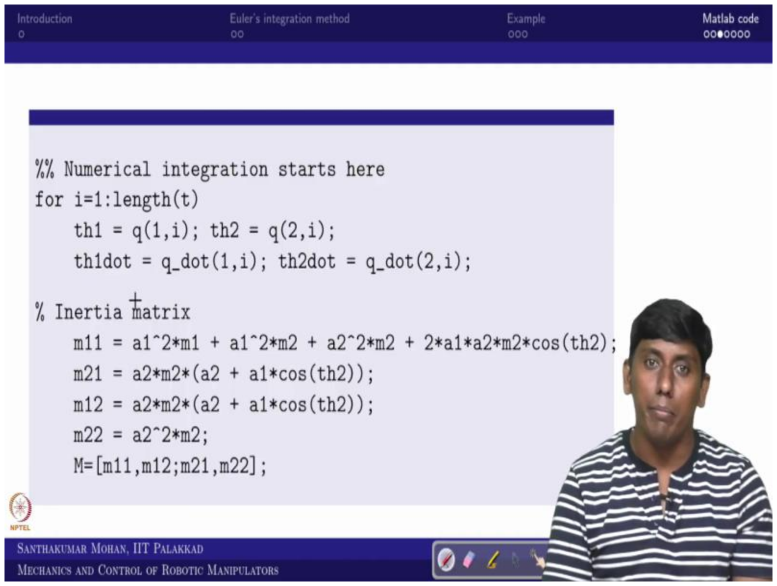
```
%% System parameters
m1 = 2; m2 = 1; % link massess
a1 = 0.5; a2 = 0.4; % link lengths
g = -9.81; % gravity
b1 = 0.5; b2 = 0.5; c1 = 0*0.6; c2 = 0*0.5;

%% Initial conditions
q = [0;0]; % initial joint positions
q_dot = [0;0]; % initial joint velocities
```

Then we are talking about the system parameter. Since it is a 2R serial manipulator that to in a plane. So, then we are assuming that it is a vertical planar. So, the g we defined as 9.81 then you are link length a1 and a2 defined. And your masses of link 1 and link 2 we define. Further we are taking in the non-rigid body effect. So, in that sense the frictional coefficients where the viscous friction and the coulomb friction we are considering.

So, the c1 and c2 I assume to be 0. It can be considered any Coulomb friction factors. But, usually we assume that it is a viscous friction. So, that is why b1 and b2 are nonzero value. So, once you obtain the system parameter what we need to do? Because we are trying to do the system integration, or you can say the state integration or in the other way around we can call numerical integration.

For that one important step is your initial condition should be known. So, we are defining the initial condition. So, here the system states are two, one is joint position, the other one is joint velocity. So, the joint velocity I am calling q dot q underscore dot. And the joint position I am calling q. So, q is equal to theta 1 and theta 2. So, q dot is equal to so, theta 1 dot and theta 2 dot. So, right now we assume both are 0's there is theta 1 and theta 2 are 0 and theta 1 dot and theta 2 dot also, 0. So, once we all done what is what else we need to do.

(Refer Slide Time: 10:45)


Introduction
Euler's integration method
Example
Matlab code
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

```
%% Numerical integration starts here
for i=1:length(t)
    th1 = q(1,i); th2 = q(2,i);
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);

% Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2*a1*a2*m2*cos(th2);
    m21 = a2*m2*(a2 + a1*cos(th2));
    m12 = a2*m2*(a2 + a1*cos(th2));
    m22 = a2^2*m2;
    M=[m11,m12;m21,m22];
```

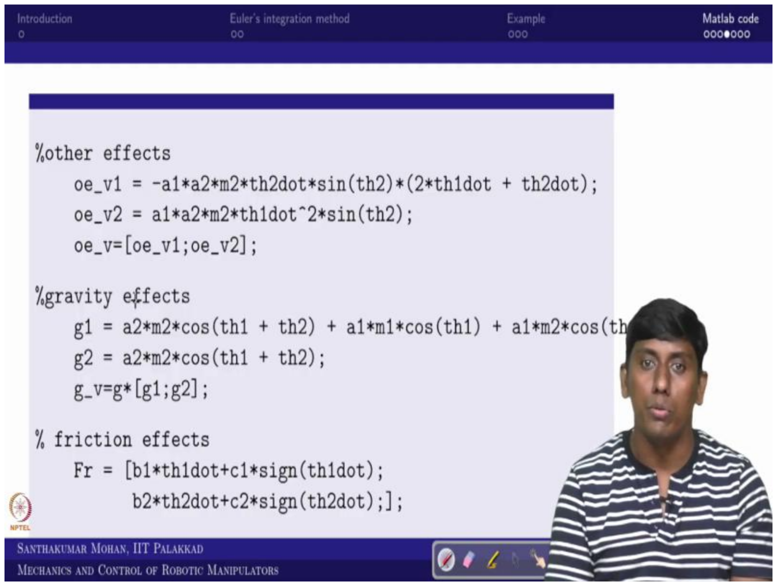
We can start the numerical integration since it is iteration because it is starting from time step t equal to 0 to ts as you can see stage by stage as a propagation. So, better we have to use one of the loops. So, simplest loop which we can use in MATLAB is for. So, I am using a for loop where it starts as the iteration or loop count as i which starts from 1 to you can say length of t. So, here length of t is so, 0 to ts how much it comes in this case it is 1001.

So, then I am calling like theta 1 and theta 2 is your system state. So, here the system states the joint position we have defined as q vector. So, the q vector first term equal to theta 1 and the second term equal to theta 2 but this q vector is going to propagate. So, that is why the ith instant equal to theta 1 and theta 2. The similar sense q underscore dot would give theta 1 dot and theta 2 dot. So, that is equal to so, theta 1 as q underscore dot of 1 comma i.

And 2 comma i is theta 2 dot. So, once you define what you can do you can start making what you call your individual components. So, we can start with the inertia matrix. So, you can see the inertia matters we can write as you can see four components. Because it is a 2 cross 2. So, all component I can write here. So, which start from m11 to m22 so, there are four terms. So, already a1, a2 we define and theta 1 theta 2 also, be defined which is going to be propagate.

And m1 and m2 are constant. So, in the sense we can find the inertia matrix. Once we know the inertia matrix, what we can do? We can do the other vector.

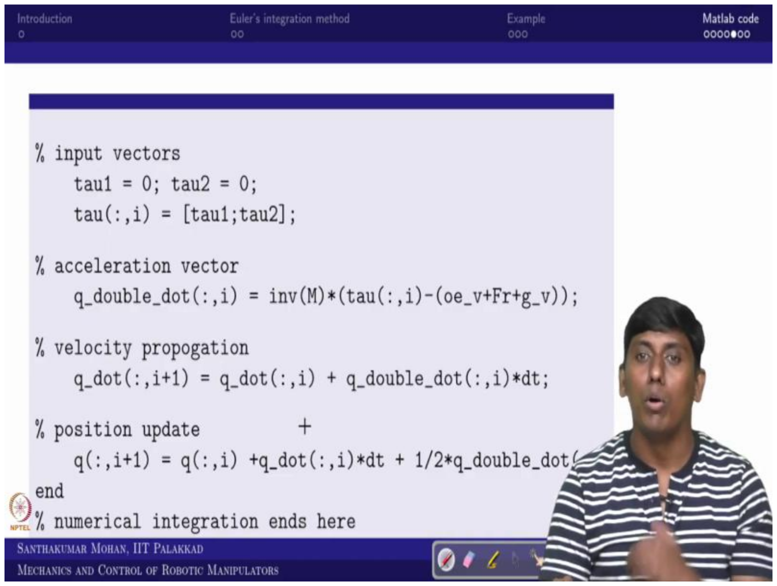
(Refer Slide Time: 12:32)


Introduction
Euler's integration method
Example
Matlab code
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

```
%other effects
    oe_v1 = -a1*a2*m2*th2dot*sin(th2)*(2*th1dot + th2dot);
    oe_v2 = a1*a2*m2*th1dot^2*sin(th2);
    oe_v=[oe_v1;oe_v2];

%gravity effects
    g1 = a2*m2*cos(th1 + th2) + a1*m1*cos(th1) + a1*m2*cos(th
    g2 = a2*m2*cos(th1 + th2);
    g_v=g*[g1;g2];

% friction effects
    Fr = [b1*th1dot+c1*sign(th1dot);
          b2*th2dot+c2*sign(th2dot);];
```

Which we call other effects here. So, which is equal to v of q comma q dot. So, this is also, like it is nothing but centripetal and you call Coriolis effect that can be given it as this. So, then we can go to the gravity effect. Finally, you can come with the frictional effect. So, although c1 and c2 are 0. But I have make it as a generalized one. So, in that sense c1 and c2 can be nonzero also. So, then you can see what you have to do now you have to give input.

(Refer Slide Time: 13:01)


Introduction
Euler's integration method
Example
Matlab code
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

```
% input vectors
    tau1 = 0; tau2 = 0;
    tau(:,i) = [tau1;tau2];

% acceleration vector
    q_double_dot(:,i) = inv(M)*(tau(:,i)-(oe_v+Fr+g_v));

% velocity propogation
    q_dot(:,i+1) = q_dot(:,i) + q_double_dot(:,i)*dt;

% position update
    q(:,i+1) = q(:,i) +q_dot(:,i)*dt + 1/2*q_double_dot(
end
% numerical integration ends here
```

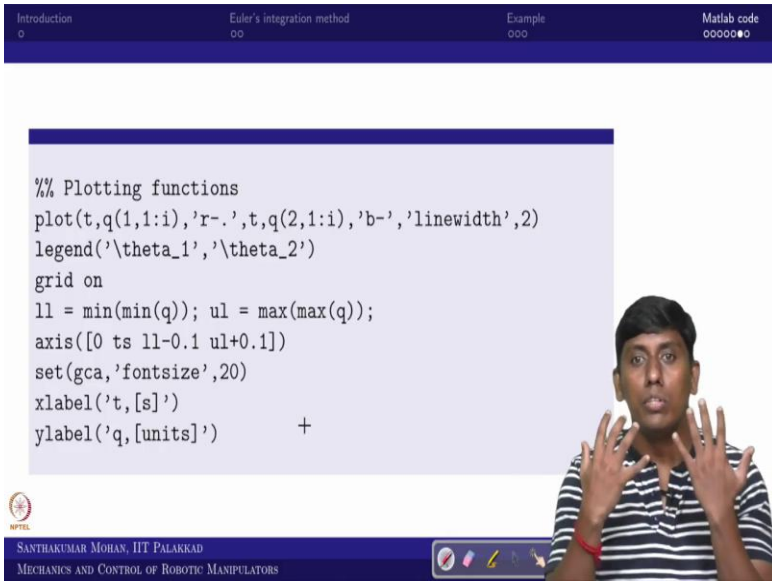
So, the input vector here I call tau 1 and tau 2 are 0. So, then you can ask what you want to expect here even you assume that theta 1 and theta 2 are 0 it is this extended position. Because our frame arrangement where theta 1 is parallel to x axis and theta 2 equal to 0 means that is again parallel to x1 in the sense it is just extended position. So, now there is a gravity's there even if I put tau 1 and tau 2 0.

And I introduced the friction what happened? This will come down and swing and it gets settled that I want to simulate. Later on, I can use some tau 1 and tau 2 as the input and see. So, now what we need to do? We have to find the q double dot term which is what we call acceleration vector. That is inverse of m into tau minus v of q comma q dot plus a frictional component plus gravitational component these are all we have used whatever the equation we have written in the previous slide.

So, then once you know q double dot what we can do we can propagate the velocity. So, in the sense q underscore dot of i plus 1 if we can do. So, that we can do it with a simple Euler integration then we can go to the position propagation. So, with that this would be the end. So, here this is q dot so, where you can say comma i that is what the case. So, whatever it is coming that is what, but it is multiplied with the delta t squared.

So, that we can see in the MATLAB code, then the numerical integration is the end. So, what exactly we expect we want to see what is the outcome once you run this program.

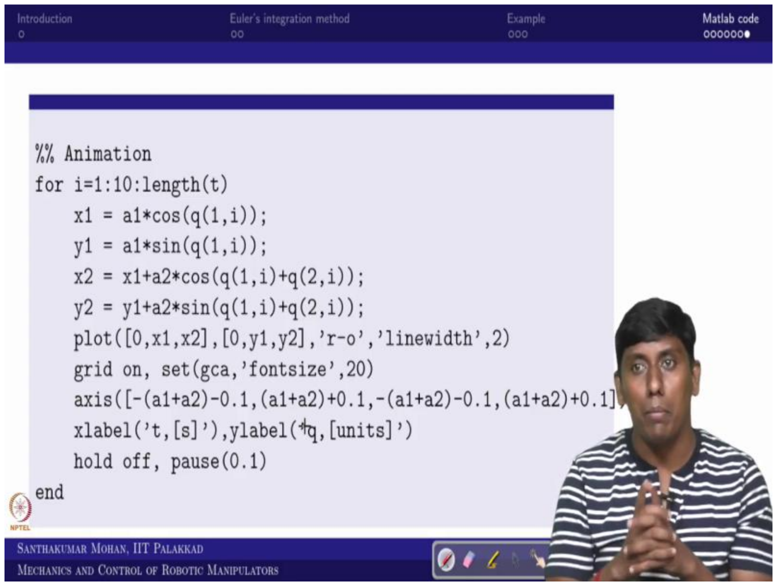
(Refer Slide Time: 14:47)


Introduction
Euler's integration method
Example
Matlab code
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

```
%% Plotting functions
plot(t,q(1,1:i),'r-.',t,q(2,1:i),'b-','linewidth',2)
legend('\theta_1','\theta_2')
grid on
ll = min(min(q)); ul = max(max(q));
axis([0 ts ll-0.1 ul+0.1])
set(gca,'fontsize',20)
xlabel('t,[s]')
ylabel('q,[units]')
```

So, I can plot the positional information along with the time in the sense that time trend or time history of the you can say joint position I can see. So, which is converge to some position that I can see. But in order to see more intuitive way so, then better we can do the animation.

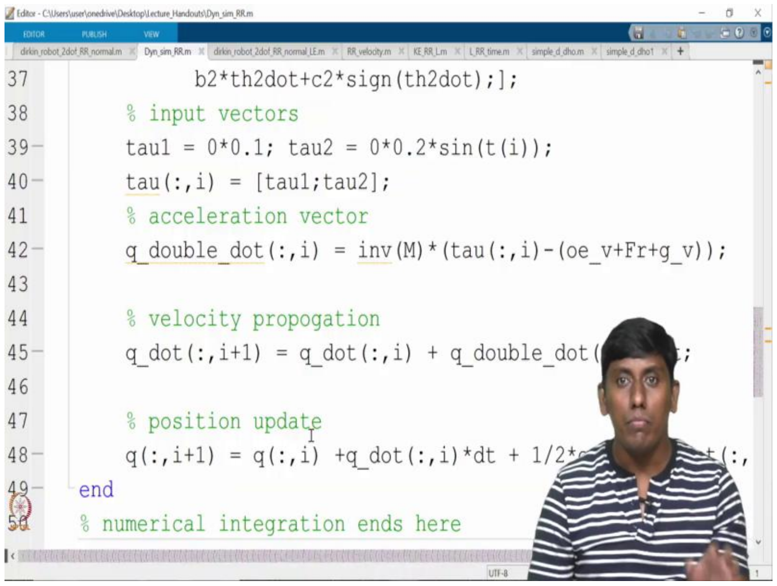
(Refer Slide Time: 15:07)


Introduction
Euler's integration method
Example
Matlab code
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

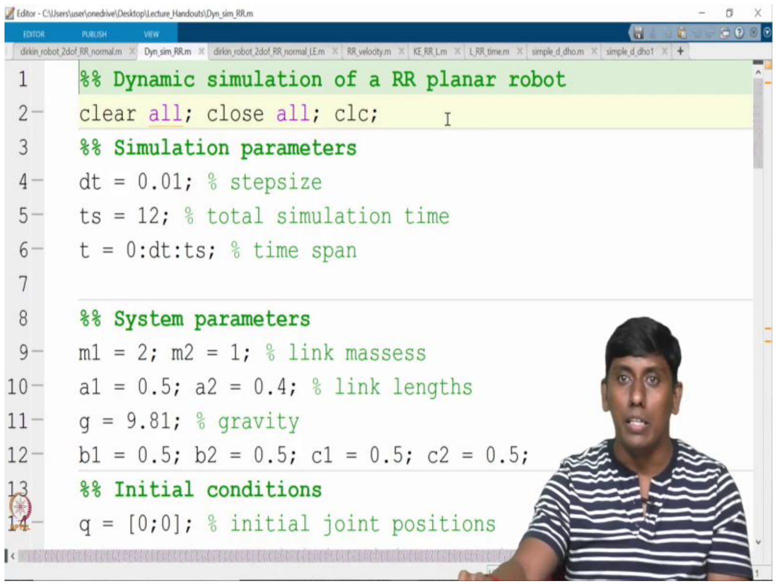
```
%% Animation
for i=1:10:length(t)
    x1 = a1*cos(q(1,i));
    y1 = a1*sin(q(1,i));
    x2 = x1+a2*cos(q(1,i)+q(2,i));
    y2 = y1+a2*sin(q(1,i)+q(2,i));
    plot([0,x1,x2],[0,y1,y2],'r-o','linewidth',2)
    grid on, set(gca,'fontsize',20)
    axis([-(a1+a2)-0.1,(a1+a2)+0.1,-(a1+a2)-0.1,(a1+a2)+0.1]
    xlabel('t,[s]'),ylabel('q,[units]')
    hold off, pause(0.1)
end
```

The animation part we can do. Here I assume that this is origin then 1 and 2 you can say the; you can say body case. So, this is the way we will explain I will explain this into the MATLAB code. Then that would be very clear to you.

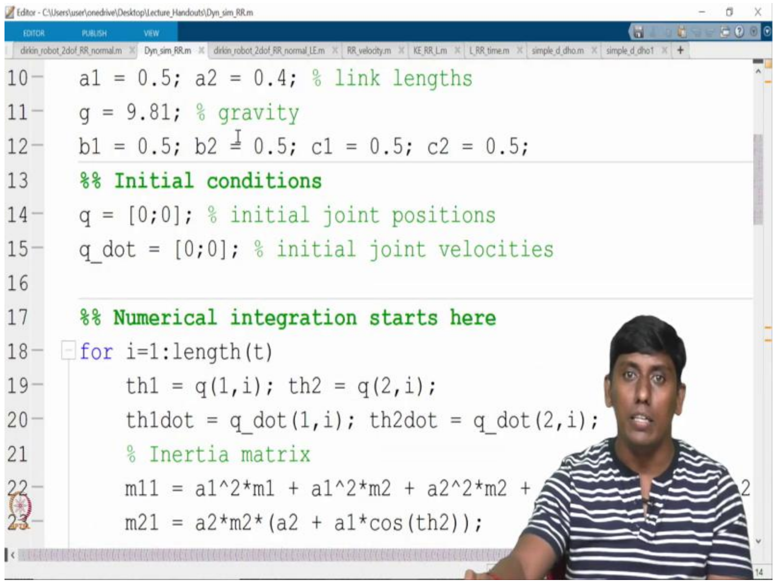
(Refer Slide Time: 15:23)



```
            b2*th2dot+c2*sign(th2dot);];
        % input vectors
        tau1 = 0*0.1; tau2 = 0*0.2*sin(t(i));
        tau(:,i) = [tau1;tau2];
        % acceleration vector
        q_double_dot(:,i) = inv(M)*(tau(:,i)-(oe_v+Fr+g_v));

        % velocity propogation
        q_dot(:,i+1) = q_dot(:,i) + q_double_dot(      ;

        % position update
        q(:,i+1) = q(:,i) +q_dot(:,i)*dt + 1/2*        t(:,
    end
    % numerical integration ends here
```



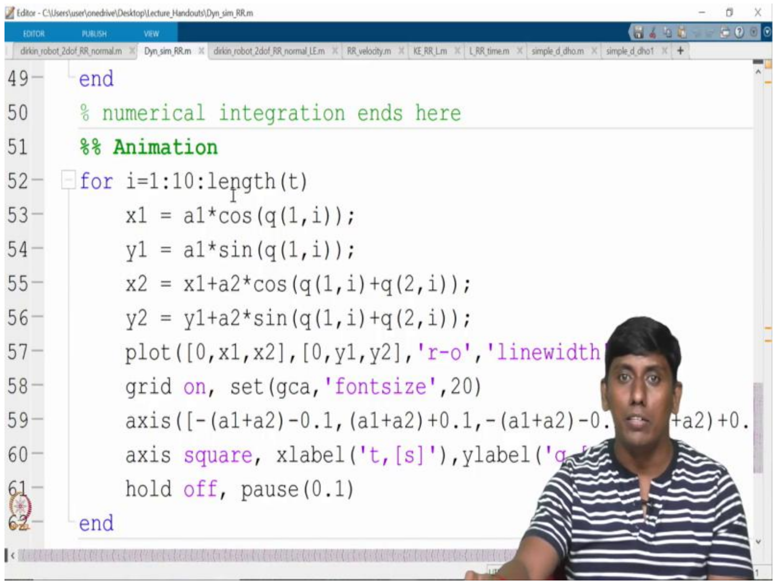
```
%% Dynamic simulation of a RR planar robot
clear all; close all; clc;
%% Simulation parameters
dt = 0.01; % stepsize
ts = 12; % total simulation time
t = 0:dt:ts; % time span

%% System parameters
m1 = 2; m2 = 1; % link massess
a1 = 0.5; a2 = 0.4; % link lengths
g = 9.81; % gravity
b1 = 0.5; b2 = 0.5; c1 = 0.5; c2 = 0.5;
%% Initial conditions
q = [0;0]; % initial joint positions
```



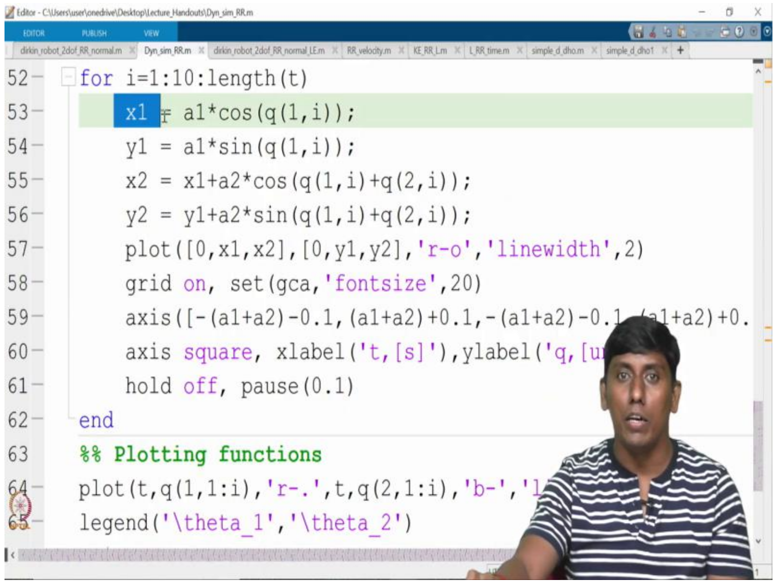
```
a1 = 0.5; a2 = 0.4; % link lengths
g = 9.81; % gravity
b1 = 0.5; b2 = 0.5; c1 = 0.5; c2 = 0.5;
%% Initial conditions
q = [0;0]; % initial joint positions
q_dot = [0;0]; % initial joint velocities

%% Numerical integration starts here
for i=1:length(t)
    th1 = q(1,i); th2 = q(2,i);
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +                    2
    m21 = a2*m2*(a2 + a1*cos(th2));
```

So, this is the MATLAB code, which we were seen in the slide. So, you can see that the dynamic simulation of RR planar robot we have done. So, this is what I mean. And these all we have given. So, right now, we assume that all frictional coefficients are 0. And here I have assumed the gravity is vertically down. That is why I modified as 9.81 instead of minus 9.81. Because when I derived, I have derived opposite direction. So, that is the case I have modified this. Why I did? I will show you once the simulation done.

(Refer Slide Time: 16:00)



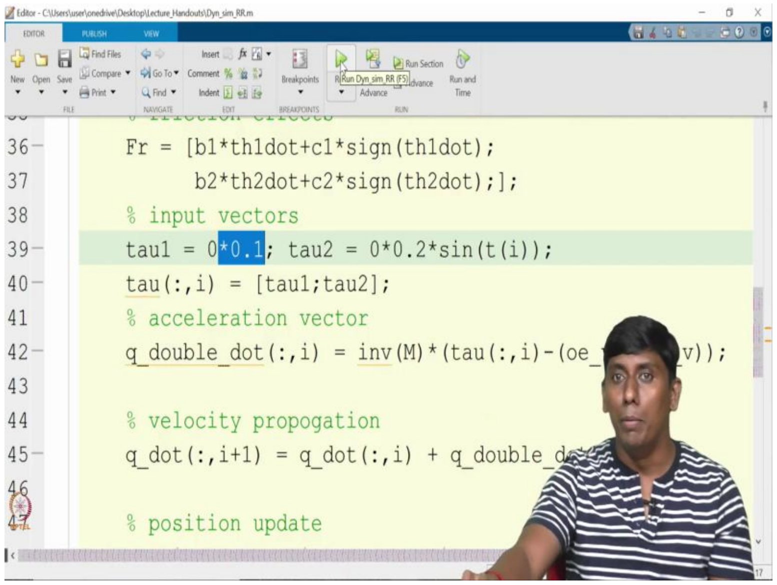
```
end
% numerical integration ends here
%% Animation
for i=1:10:length(t)
    x1 = a1*cos(q(1,i));
    y1 = a1*sin(q(1,i));
    x2 = x1+a2*cos(q(1,i)+q(2,i));
    y2 = y1+a2*sin(q(1,i)+q(2,i));
    plot([0,x1,x2],[0,y1,y2],'r-o','linewidth
    grid on, set(gca,'fontsize',20)
    axis([-(a1+a2)-0.1,(a1+a2)+0.1,-(a1+a2)-0.   +a2)+0.
    axis square, xlabel('t,[s]'),ylabel('q
    hold off, pause(0.1)
end
```

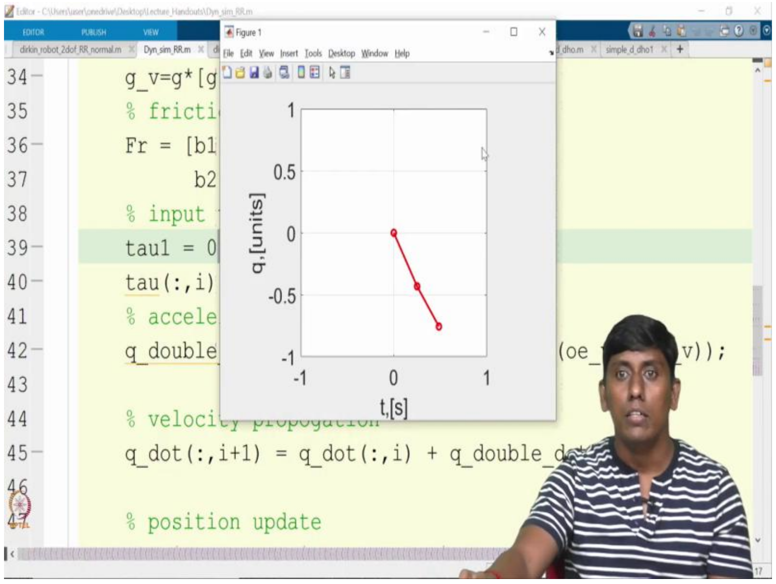

```
for i=1:10:length(t)
    x1 = a1*cos(q(1,i));
    y1 = a1*sin(q(1,i));
    x2 = x1+a2*cos(q(1,i)+q(2,i));
    y2 = y1+a2*sin(q(1,i)+q(2,i));
    plot([0,x1,x2],[0,y1,y2],'r-o','linewidth',2)
    grid on, set(gca,'fontsize',20)
    axis([-(a1+a2)-0.1,(a1+a2)+0.1,-(a1+a2)-0.1,(a1+a2)+0.
    axis square, xlabel('t,[s]'),ylabel('q,[u
    hold off, pause(0.1)
end
%% Plotting functions
plot(t,q(1,1:i),'r-.',t,q(2,1:i),'b-','l
legend('\theta_1','\theta_2')
```

So, you can see these all. So, this is the simulation animation part. So, you can see like here I am calculating the first you can see link end which I call x 1 and x 2 you can recall in the slide also, the j of x 1 comma x 2 sorry x 1 comma y 1. And j of x 2 comma y 2 that is what we have written it here. So, now if I start from origin then 1 and 2. So, if I make as a line diagram with a marker that looked like a manipulator. So, better I will run this simulation I will explain later on.

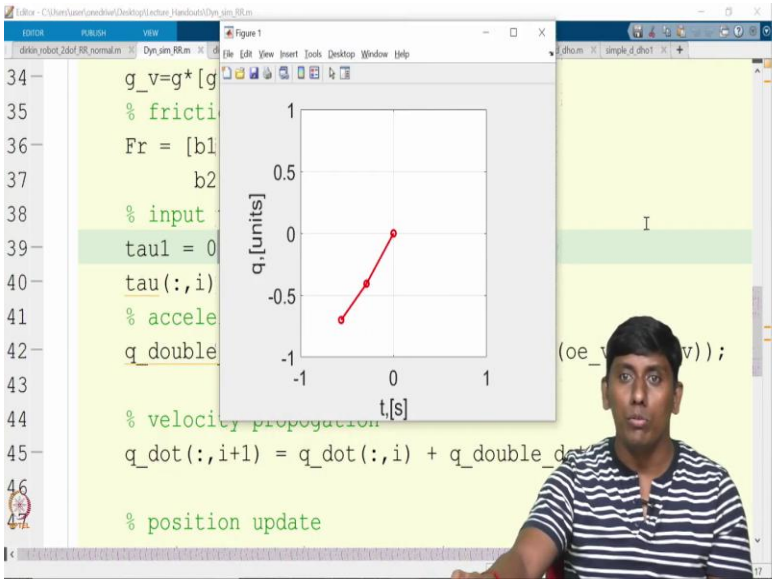
So, now, what we have given the 0 initial condition. So, you can see that this is 0 initial condition in the sense. So, q vector and q dot vector are 0's.

(Refer Slide Time: 16:48)

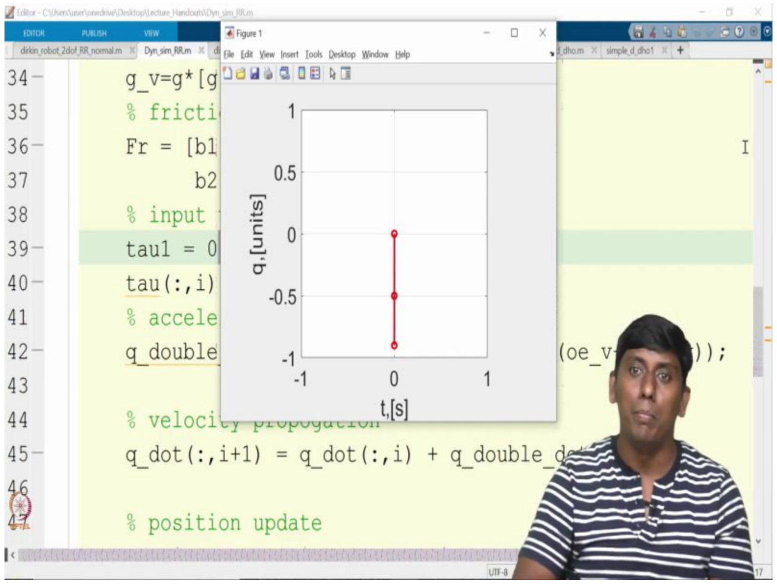

```
    Fr = [b1*th1dot+c1*sign(th1dot);
          b2*th2dot+c2*sign(th2dot);];
    % input vectors
    tau1 = 0*0.1; tau2 = 0*0.2*sin(t(i));
    tau(:,i) = [tau1;tau2];
    % acceleration vector
    q_double_dot(:,i) = inv(M)*(tau(:,i)-(oe_         v));

    % velocity propogation
    q_dot(:,i+1) = q_dot(:,i) + q_double_d

    % position update
```


Editor - C:\Users\user\onedrive\Desktop\Lecture_Handouts\Dyn_sim_RR.m
EDITOR
PUBLISH
VIEW
dirkin_robot_2dof_RR_normal.m
Dyn_sim_RR.m
simple_d_dho1
Figure 1
File Edit View Insert Tools Desktop Window Help
q,[units]
t,[s]

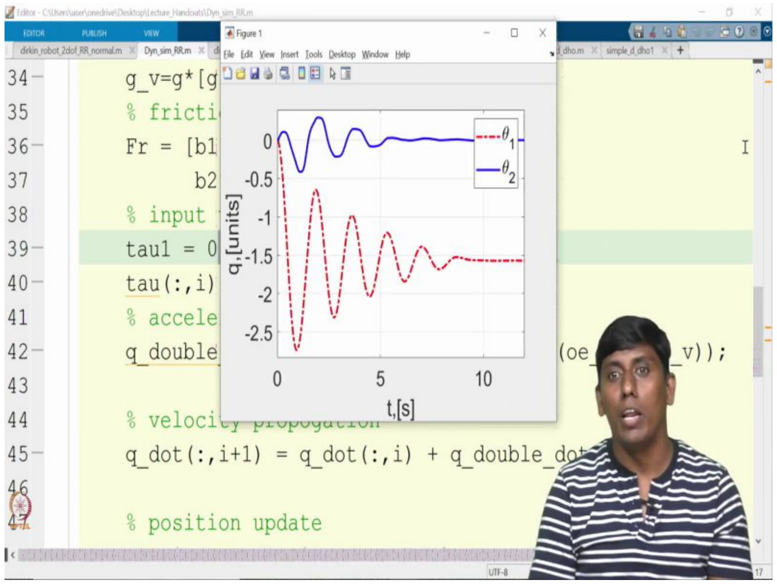
```
g_v=g*[g
% fricti
Fr = [b1
      b2
% input
tau1 = 0
tau(:,i)
% accele
q_double                              (oe_      v));

% velocity propogation
q_dot(:,i+1) = q_dot(:,i) + q_double_d

% position update
```


Editor - C:\Users\user\onedrive\Desktop\Lecture_Handouts\Dyn_sim_RR.m
EDITOR
PUBLISH
VIEW
dirkin_robot_2dof_RR_normal.m
Dyn_sim_RR.m
simple_d_dho1
Figure 1
File Edit View Insert Tools Desktop Window Help
q,[units]
t,[s]

```
g_v=g*[g
% fricti
Fr = [b1
      b2
% input
tau1 = 0
tau(:,i)
% accele
q_double                              (oe_      v));

% velocity propogation
q_dot(:,i+1) = q_dot(:,i) + q_double_d

% position update
```



```
    g_v=g*[g
    % fricti
    Fr = [b1
          b2
    % input
    tau1 = 0
    tau(:,i)
    % accele
    q_double                                    (oe_v      ));

    % velocity propogation
    q_dot(:,i+1) = q_dot(:,i) + q_double_d

    % position update
```

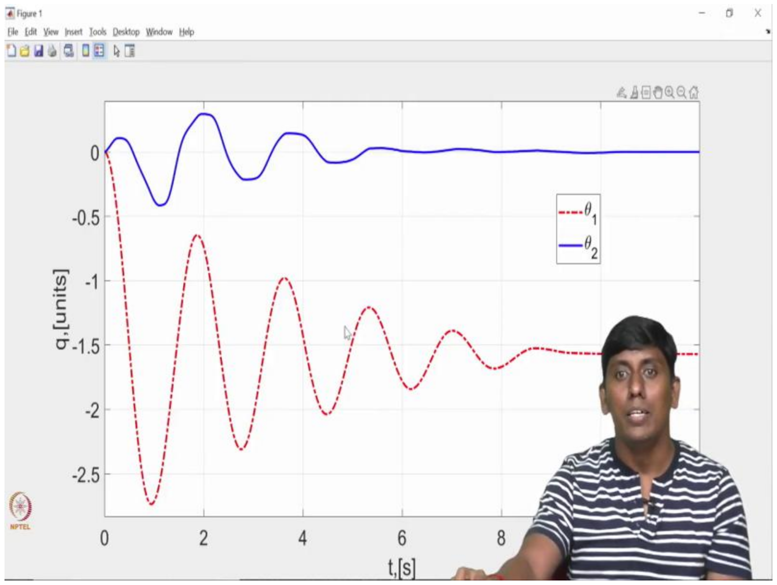


```
    g_v=g*[g
    % fricti
    Fr = [b1
          b2
    % input
    tau1 = 0
    tau(:,i)
    % accele
    q_double                                    (oe_      _v));

    % velocity propogation
    q_dot(:,i+1) = q_dot(:,i) + q_double_do

    % position update
```

And further the tau 1 and tau 2 also, 0 although I have put it some value but the equal is 0. So, now if I run this so, you can see like so, due to the gravity is the manipulator will start oscillating as a double pendulum. We can see that okay. So, now you can see. So, it is very clear now. So, it is a red you can say line connected this is what you call as you can say to a serial manipulator. So, this is the origin this is x1 y1 and this is x2 y2.

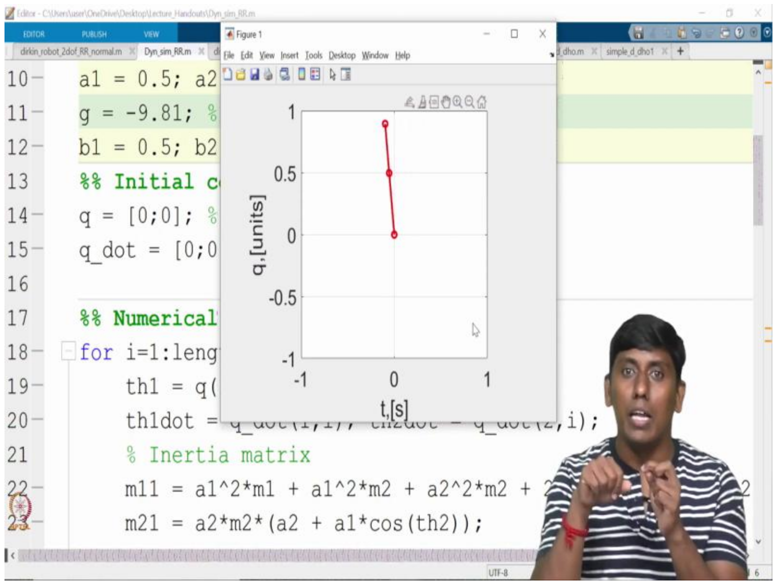
And now due to the gravity it is a swinging like as a double pendulum and it is ended.

(Refer Slide Time: 17:28)

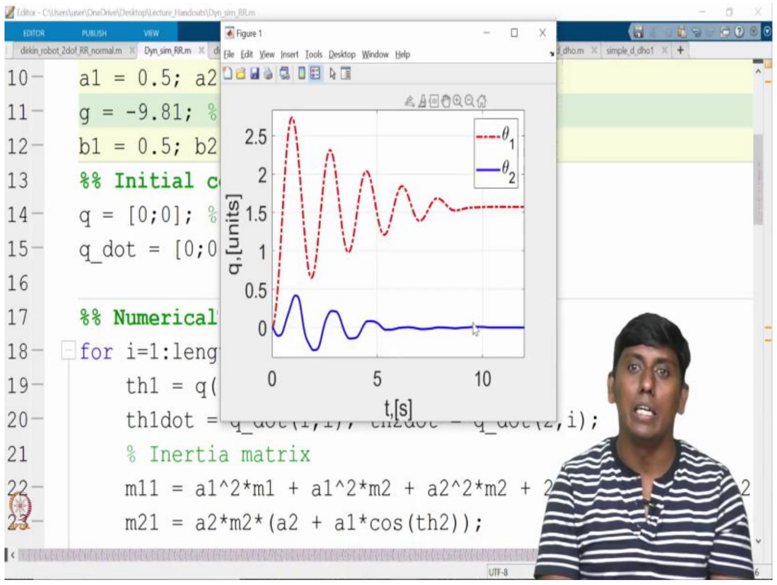


So, now if you see the plotting option also, you can see that initially it is start from 0. But due to the gravity it is start oscillating. And it ends with what you call 0 for theta 2 because theta 1 and theta 2 are collinear that is why theta 2 is 0. And theta 1 is minus 1.57 something which is equal to pi by 2 which is nothing but in degrees it is 90 degrees. So, that is what we have seen.

(Refer Slide Time: 17:54)


Editor - C:\Users\user\OneDrive\Desktop\Lecture_Handouts\Dyn_sim_RR.m
EDITOR
PUBLISH
VIEW
dirkin_robot_2dof_RR_normal.m
Dyn_sim_RR.m
dho.m
simple_d_dho1
Figure 1
File Edit View Insert Tools Desktop Window Help
q,[units]
t,[s]
1
0.5
0
-0.5
-1
-1
0
1
UTF-8

```
    a1 = 0.5; a2
    g = -9.81; %
    b1 = 0.5; b2
    %% Initial c
    q = [0;0]; %
    q_dot = [0;0

    %% Numerical
    for i=1:leng
        th1 = q(
        th1dot =                                    ,i);
        % Inertia matrix
        m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +
        m21 = a2*m2*(a2 + a1*cos(th2));
```

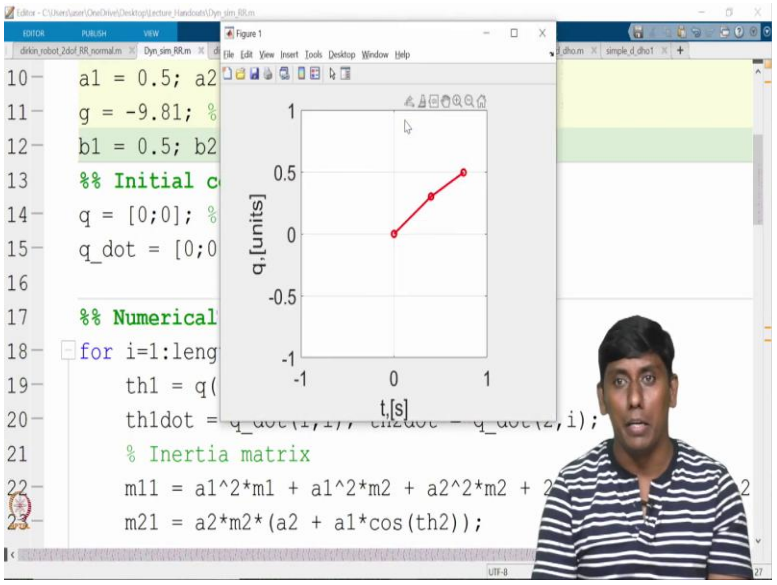


```
a1 = 0.5; a2
g = -9.81; %
b1 = 0.5; b2
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2
    m21 = a2*m2*(a2 + a1*cos(th2));
```

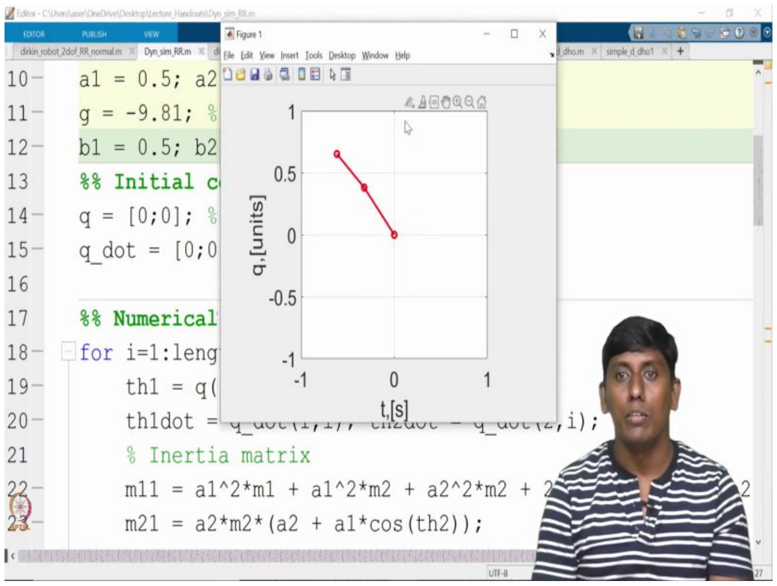
So, now there is one question will come. So, if I take this is minus 9.81 it will show that is the opposite direction. Why? Because we assume that the gravity is vertically down, and you choose the direction is minus. So, then it is swinging swing up and hold it. It is not going to happen because the gravity is eventually pulled down.

So, I just to show you here you can see like this is going up this is not going to happen in real. So, that is why we have changed the direction because this is what we have taken as a syntax what direction we have taken accordingly it is coming up. So, in the gravity vector we have assume it is vertically down. So, that is why it is opposite direction. I hope now you are clear why g is considered that way.

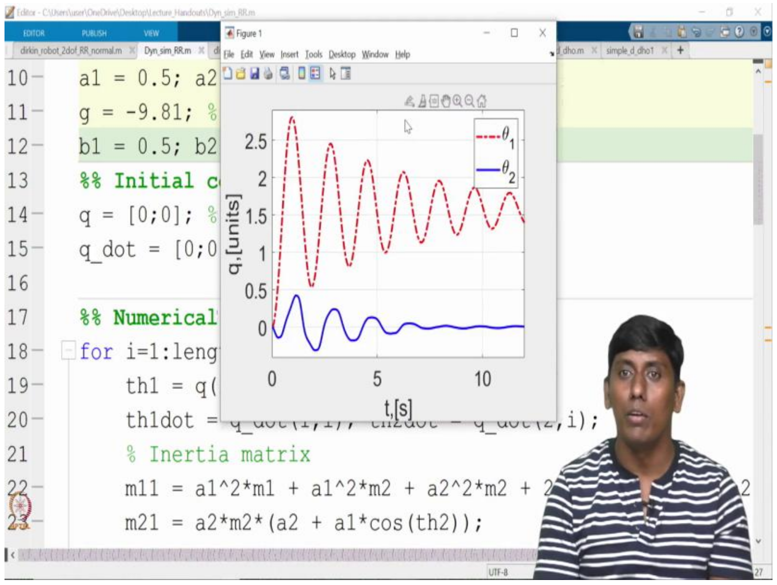
(Refer Slide Time: 18:45)



```
a1 = 0.5; a2
g = -9.81; %
b1 = 0.5; b2
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2                    2
    m21 = a2*m2*(a2 + a1*cos(th2));
```



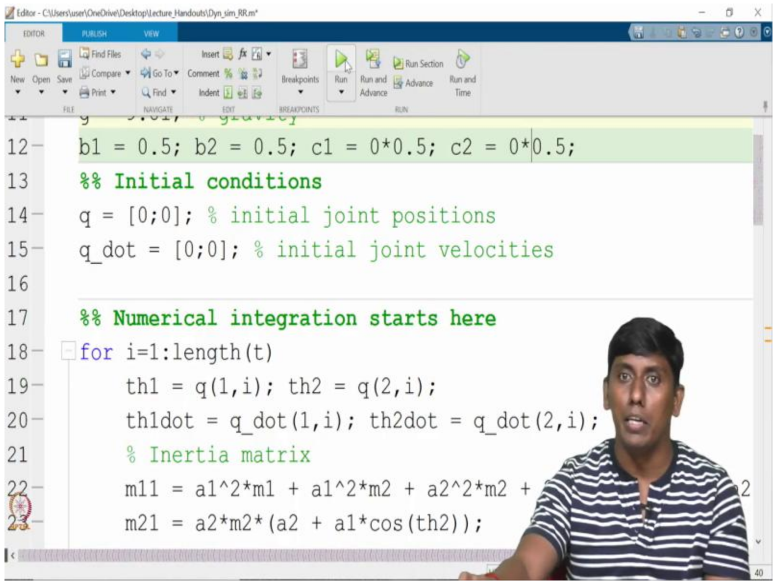
```
a1 = 0.5; a2
g = -9.81; %
b1 = 0.5; b2
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2                    2
    m21 = a2*m2*(a2 + a1*cos(th2));
```



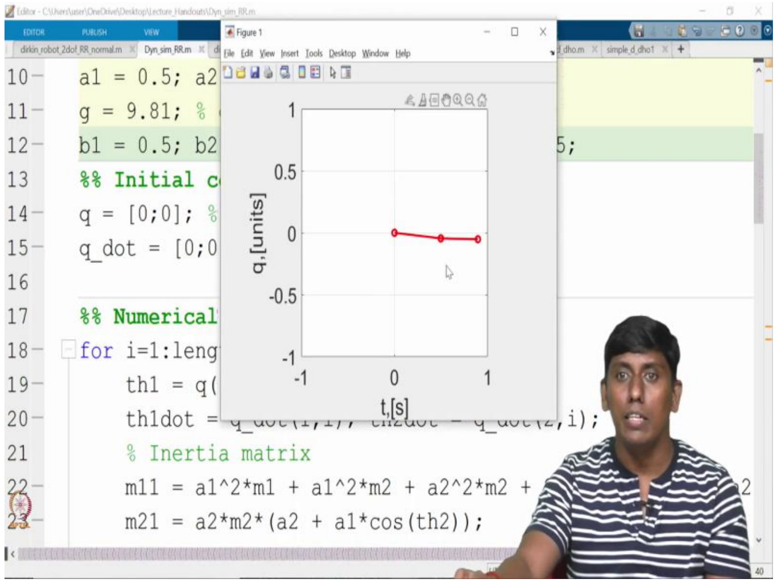
```
a1 = 0.5; a2
g = -9.81; %
b1 = 0.5; b2
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2
    m21 = a2*m2*(a2 + a1*cos(th2));
```

So, now in order to understand more and more clear you can change the frictional component for example, there is no Coulomb friction of the first joint. So, you can see like that what you call oscillation is completely different. So, you can see it right it is something like it is not getting settle. Because the viscous friction only there it is taking a long time to settle that you can feel it. But if the Coulomb friction is there it is one kind of static it is making further because it is a constant force acting opposite direction.

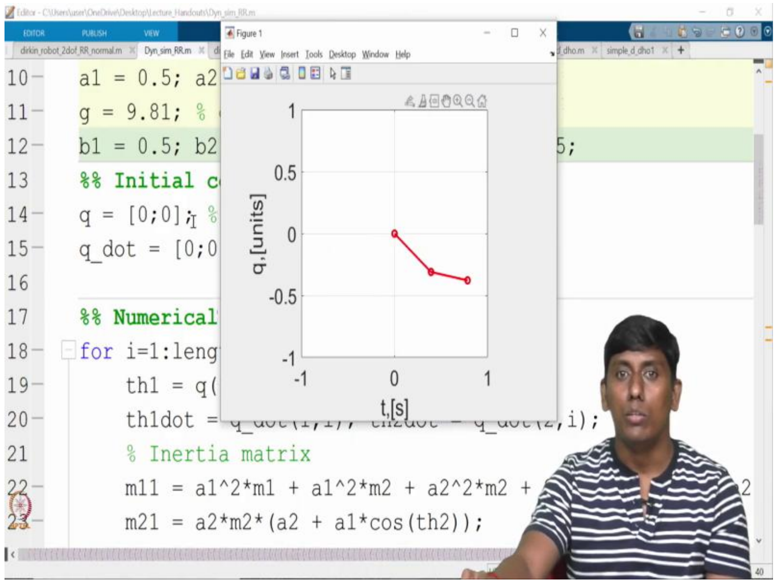
(Refer Slide Time: 19:16)



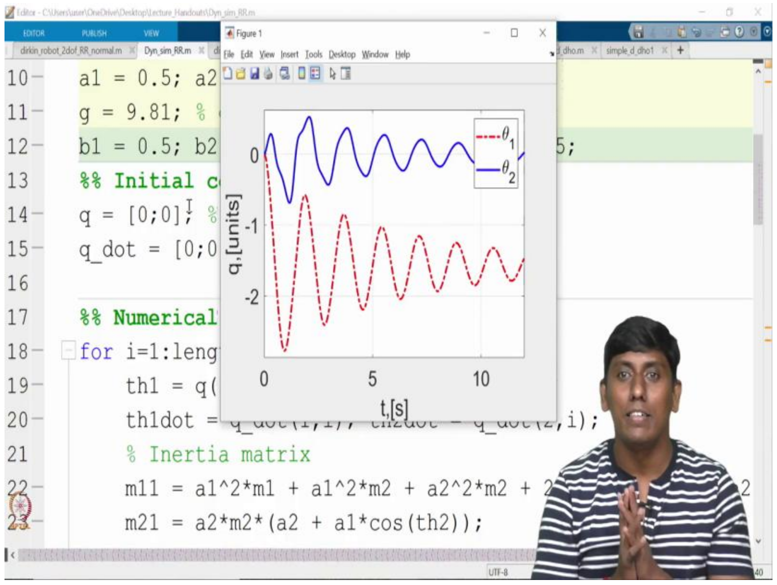
```
b1 = 0.5; b2 = 0.5; c1 = 0*0.5; c2 = 0*0.5;
%% Initial conditions
q = [0;0]; % initial joint positions
q_dot = [0;0]; % initial joint velocities

%% Numerical integration starts here
for i=1:length(t)
    th1 = q(1,i); th2 = q(2,i);
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +
    m21 = a2*m2*(a2 + a1*cos(th2));
```


Editor - C:\Users\user\OneDrive\Desktop\Lecture_Handouts\Dyn_sim_RR.m
EDITOR
PUBLISH
VIEW
dirkin_robot_2dof_RR_normal.m
Dyn_sim_RR.m
simple_d_dho1
Figure 1
File Edit View Insert Tools Desktop Window Help
q,[units]
t,[s]
1
0.5
0
-0.5
-1
-1
0
1

```
a1 = 0.5; a2
g = 9.81; %
b1 = 0.5; b2                                5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                              ,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +       2
    m21 = a2*m2*(a2 + a1*cos(th2));
```

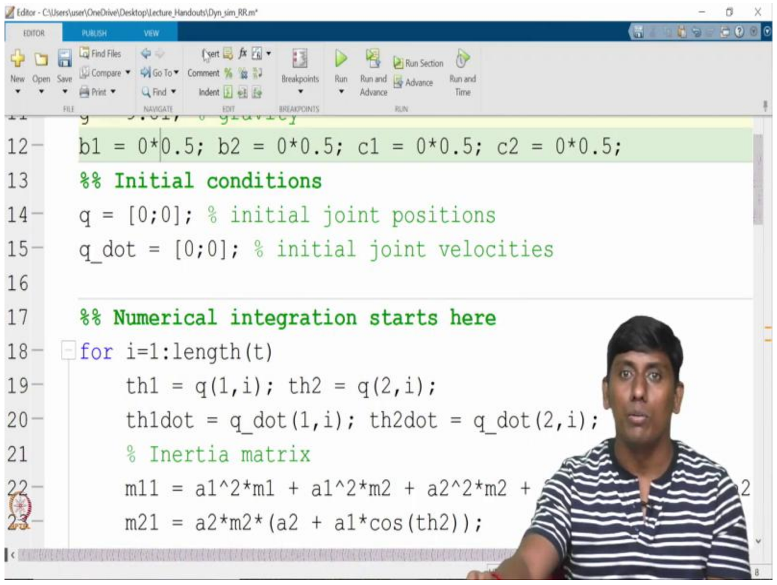

Editor - C:\Users\user\OneDrive\Desktop\Lecture_Handouts\Dyn_sim_RR.m
EDITOR
PUBLISH
VIEW
dirkin_robot_2dof_RR_normal.m
Dyn_sim_RR.m
simple_d_dho1
Figure 1
File Edit View Insert Tools Desktop Window Help
q,[units]
t,[s]
1
0.5
0
-0.5
-1
-1
0
1

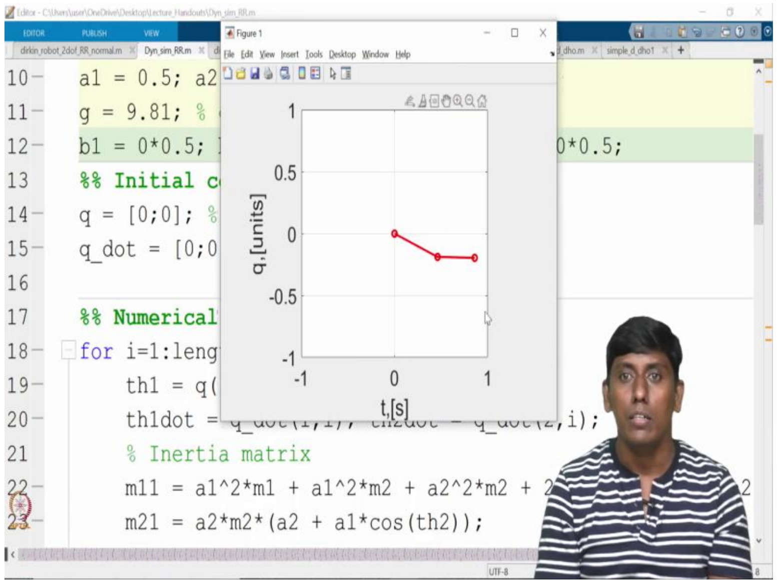
```
a1 = 0.5; a2
g = 9.81; %
b1 = 0.5; b2                                5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                              ,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +       2
    m21 = a2*m2*(a2 + a1*cos(th2));
```


```
a1 = 0.5; a2
g = 9.81; %
b1 = 0.5; b2                                5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                   q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2                   2
    m21 = a2*m2*(a2 + a1*cos(th2));
```

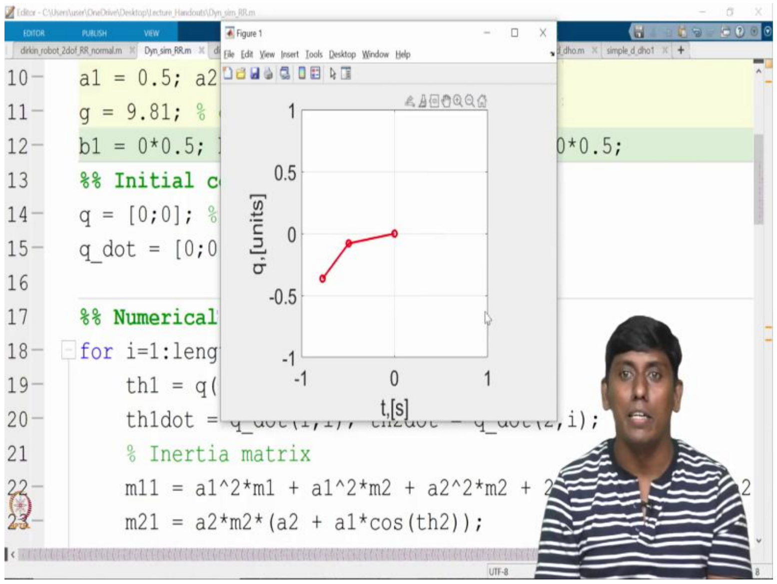
So, that is what you can see it. So, now I assume that. So, there is you can say both Coulomb frictions are 0. So, I am checking it with gravity is pulling down. So, now you can see that it is further it is taking time even the 20 second or whatever I given that duration it is not sufficient to get settle you can see. So, now if I eliminate the friction, you will feel a completely different phenomena. What different phenomena? It will get some kind of startup energy.

(Refer Slide Time: 19:57)

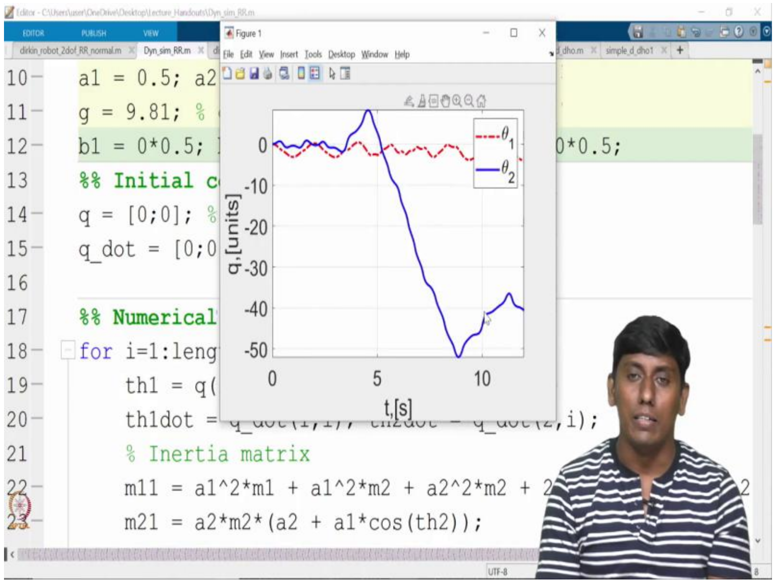

```
b1 = 0*0.5; b2 = 0*0.5; c1 = 0*0.5; c2 = 0*0.5;
%% Initial conditions
q = [0;0]; % initial joint positions
q_dot = [0;0]; % initial joint velocities

%% Numerical integration starts here
for i=1:length(t)
    th1 = q(1,i); th2 = q(2,i);
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +                 2
    m21 = a2*m2*(a2 + a1*cos(th2));
```



```
a1 = 0.5; a2
g = 9.81; %
b1 = 0*0.5;                                    0*0.5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                               q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2            2
    m21 = a2*m2*(a2 + a1*cos(th2));
```



```
a1 = 0.5; a2
g = 9.81; %
b1 = 0*0.5;                                    0*0.5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                               q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2            2
    m21 = a2*m2*(a2 + a1*cos(th2));
```



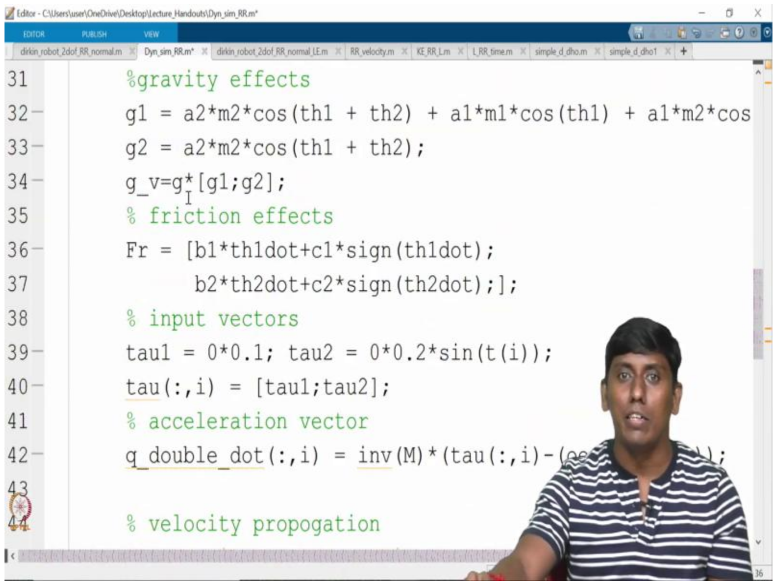
```
a1 = 0.5; a2
g = 9.81; %
b1 = 0*0.5;                                  0*0.5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                          q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2
    m21 = a2*m2*(a2 + a1*cos(th2));
```

You can see it is supposed to be marginally stable. Whether that marginally stable system in the sense it would be a continuous oscillation with the equal interval that is happening or not you can see it. So, you can see you. Now, you would have seen it is getting oscillation, it is completely not acceptable in real time. So, if you leave it the system will not propagate. So, now one can see why this is happening it is very simple. Because we have done with a simple numerical integration.

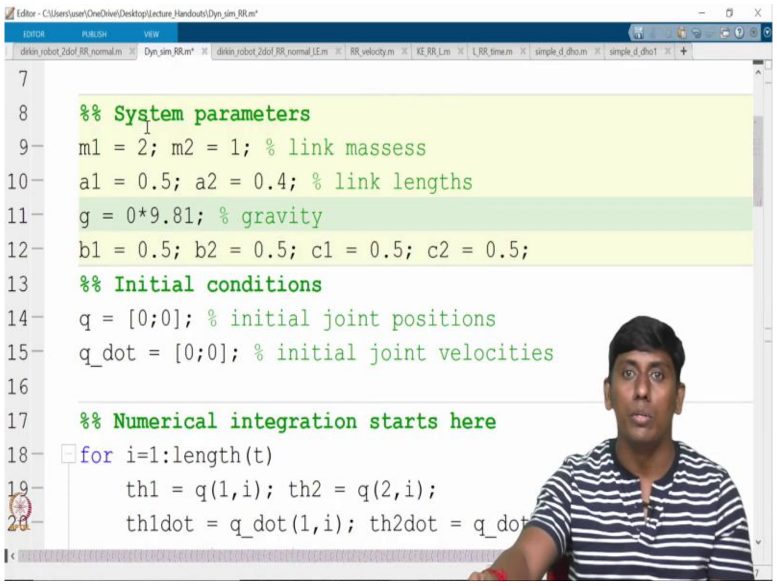
So, this particular phenomenon is not able to observe with a simple Euler integration. Further there is no friction. So, because of that what happened it is the second bar which is making a swing up or you can say it is producing kind of energy. So, theoretically it is not justified because so, as per the energy conservation principle it cannot be done this way. So, that is why we are introduced this friction.

And that way we can get it more. You can say realistic result. So, now, even the Coulomb friction you want to add or not it is up to you.

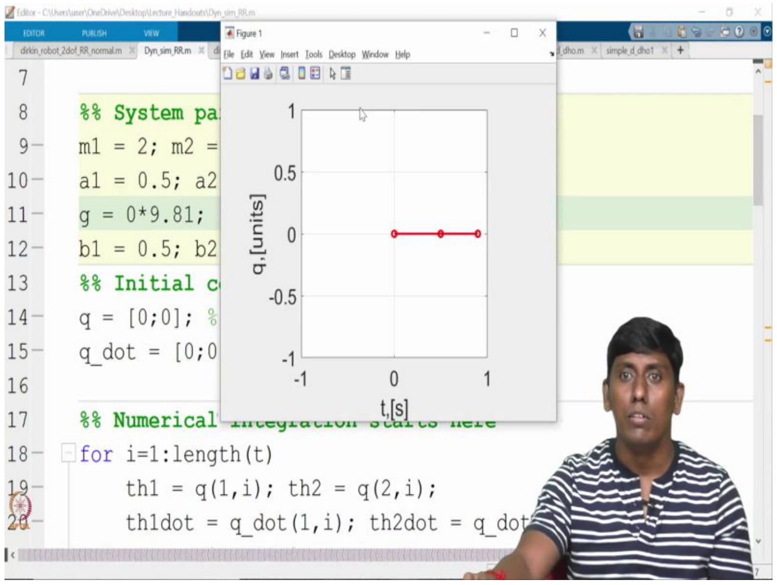
(Refer Slide Time: 21:16)



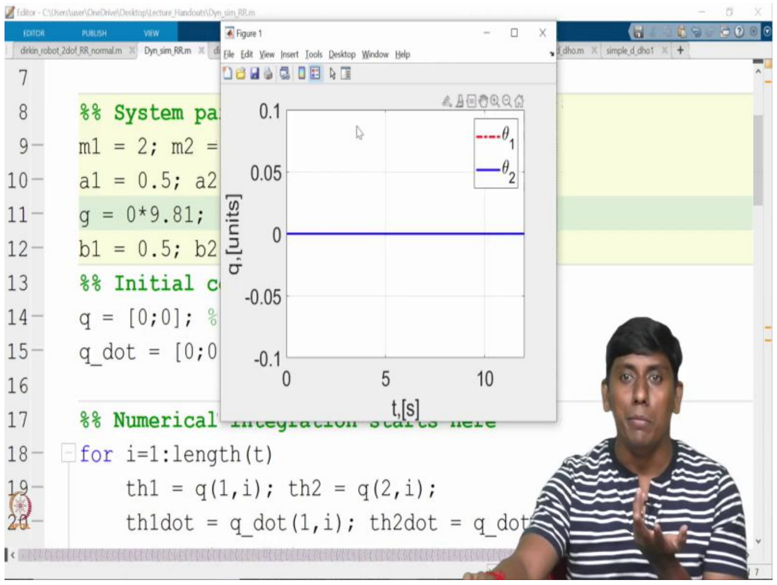
```
    %gravity effects
    g1 = a2*m2*cos(th1 + th2) + a1*m1*cos(th1) + a1*m2*cos
    g2 = a2*m2*cos(th1 + th2);
    g_v=g*[g1;g2];
    % friction effects
    Fr = [b1*th1dot+c1*sign(th1dot);
          b2*th2dot+c2*sign(th2dot);];
    % input vectors
    tau1 = 0*0.1; tau2 = 0*0.2*sin(t(i));
    tau(:,i) = [tau1;tau2];
    % acceleration vector
    q_double_dot(:,i) = inv(M)*(tau(:,i)-(
    
    % velocity propogation
```



```

%% System parameters
m1 = 2; m2 = 1; % link massess
a1 = 0.5; a2 = 0.4; % link lengths
g = 0*9.81; % gravity
b1 = 0.5; b2 = 0.5; c1 = 0.5; c2 = 0.5;
%% Initial conditions
q = [0;0]; % initial joint positions
q_dot = [0;0]; % initial joint velocities

%% Numerical integration starts here
for i=1:length(t)
    th1 = q(1,i); th2 = q(2,i);
    th1dot = q_dot(1,i); th2dot = q_dot
```



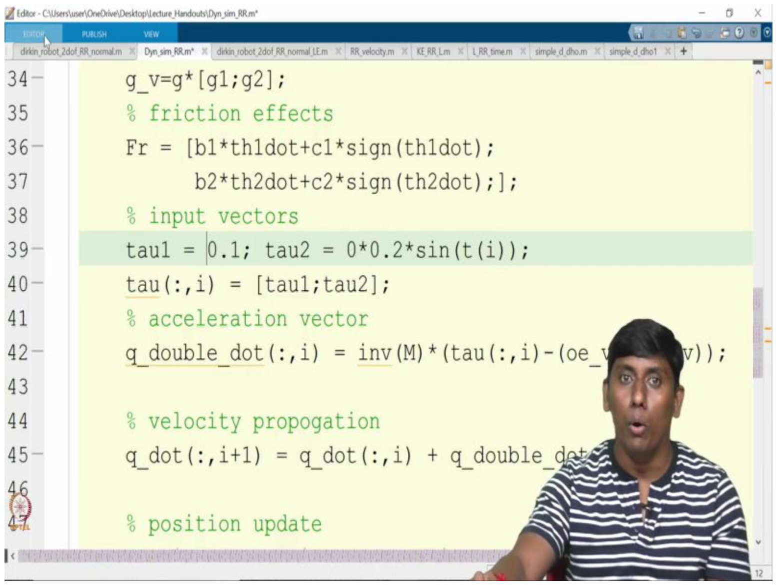
```
%% System pa
m1 = 2; m2 =
a1 = 0.5; a2
g = 0*9.81;
b1 = 0.5; b2
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical integration starts here
for i=1:length(t)
    th1 = q(1,i); th2 = q(2,i);
    th1dot = q_dot(1,i); th2dot = q_dot
```



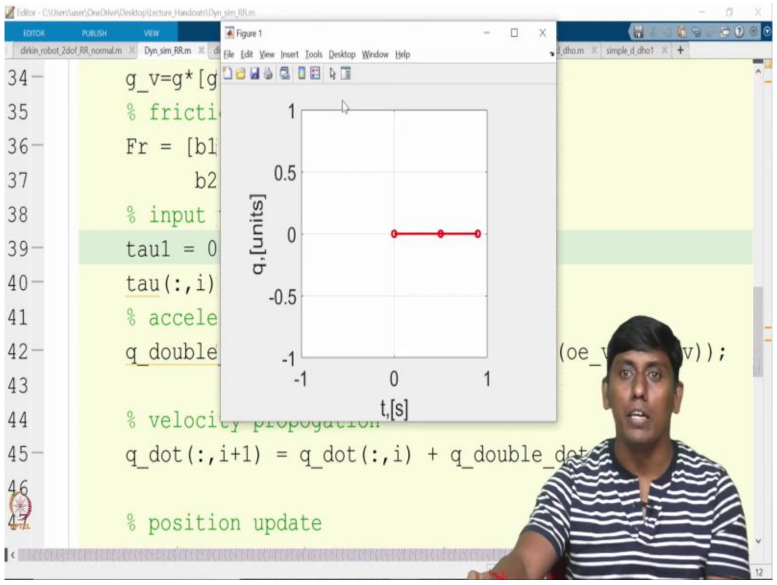
```
%% System pa
m1 = 2; m2 =
a1 = 0.5; a2
g = 0*9.81;
b1 = 0.5; b2
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical integration starts here
for i=1:length(t)
    th1 = q(1,i); th2 = q(2,i);
    th1dot = q_dot(1,i); th2dot = q_dot
```

But now, if I take the gravity vector 0 in the sense, I said that there is no gravity at all in the sense I put g equal to 0. Then what one can expect it would be remaining the same place. That is supposed to be. Yes that is matching. Because you have tau 1 and tau 2 are 0 and you do not have any input, or any external impact, or any external effort. So, obviously the system will remind as same. So, now, if you provide input what do you call tau 1 and tau 2.

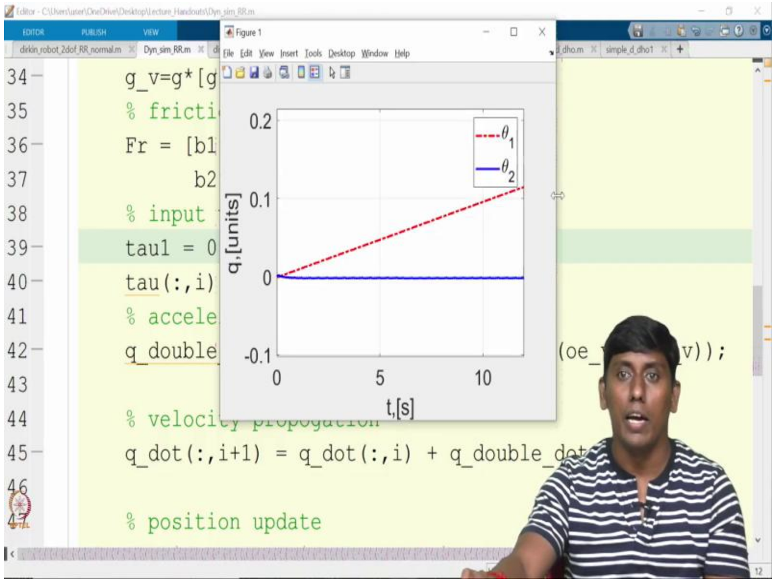
(Refer Slide Time: 21:49)



```
g_v=g*[g1;g2];
% friction effects
Fr = [b1*th1dot+c1*sign(th1dot);
      b2*th2dot+c2*sign(th2dot);];
% input vectors
tau1 = 0.1; tau2 = 0*0.2*sin(t(i));
tau(:,i) = [tau1;tau2];
% acceleration vector
q_double_dot(:,i) = inv(M)*(tau(:,i)-(oe_v      v));

% velocity propogation
q_dot(:,i+1) = q_dot(:,i) + q_double_dot

% position update
```



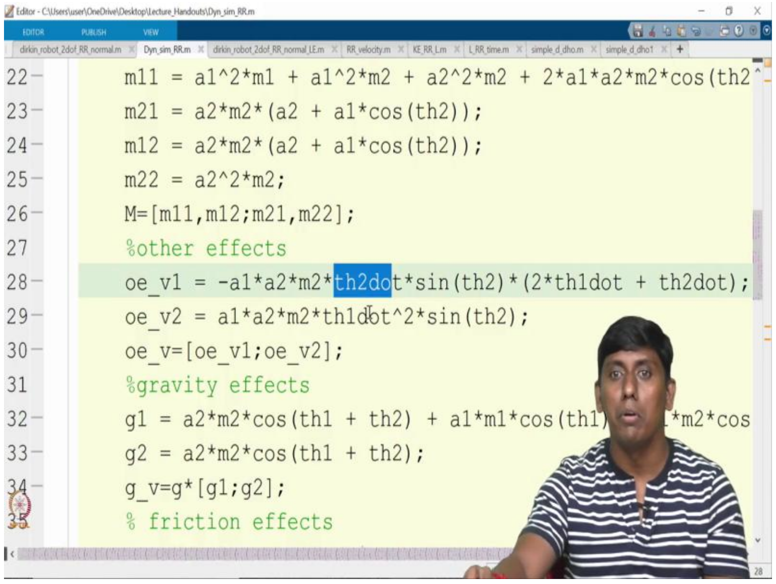
```
g_v=g*[g
% fricti
Fr = [b1
      b2
% input
tau1 = 0
tau(:,i)
% accele
q_double                    (oe_v      v));

% velocity propogation
q_dot(:,i+1) = q_dot(:,i) + q_double_dot

% position update
```



```
g_v=g*[g
% fricti
Fr = [b1
     b2
% input
tau1 = 0
tau(:,i)
% accele
q_double                              (oe_      v));

% velocity propogation
q_dot(:,i+1) = q_dot(:,i) + q_double_dot

% position update
```

Just for demonstrating, I assume that the tau 1 is 0.1 Newton meter. So, what one can expect? So, the second system will remain same? No, because there is an interactive effect that you can see. So, I will you can it is very slow speed. You can like, not able to realize that explicitly. So, I will give that unit is slightly different. Because we made it very slow. So, I will just stop once it is done. You can see it is moving.

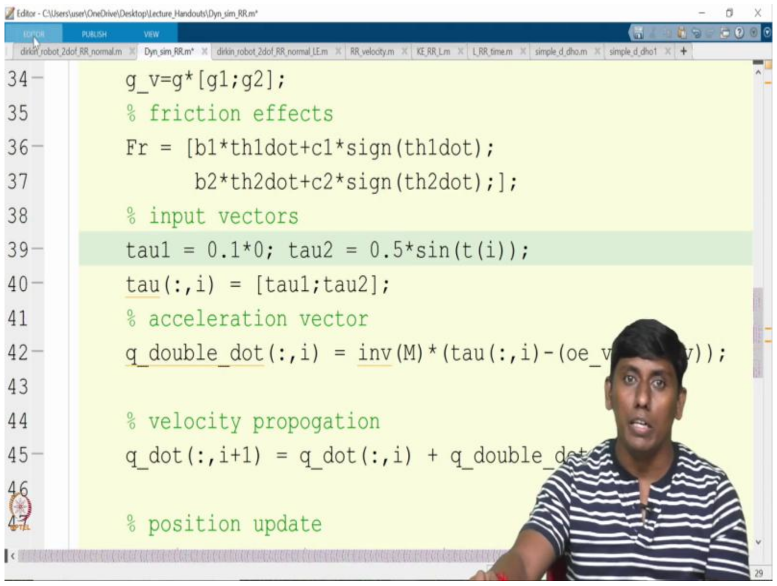
(Refer Slide Time: 22:30)



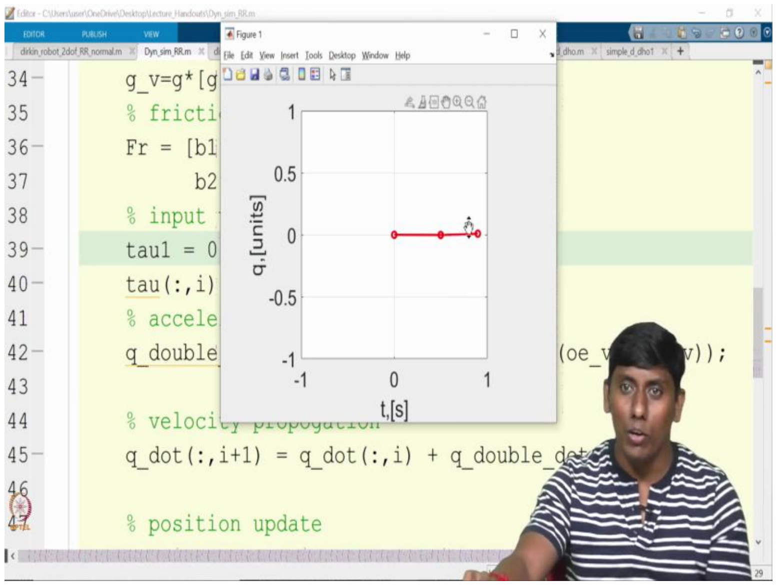
```
m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 + 2*a1*a2*m2*cos(th2
m21 = a2*m2*(a2 + a1*cos(th2));
m12 = a2*m2*(a2 + a1*cos(th2));
m22 = a2^2*m2;
M=[m11,m12;m21,m22];
%other effects
oe_v1 = -a1*a2*m2*th2dot*sin(th2)*(2*th1dot + th2dot);
oe_v2 = a1*a2*m2*th1dot^2*sin(th2);
oe_v=[oe_v1;oe_v2];
%gravity effects
g1 = a2*m2*cos(th1 + th2) + a1*m1*cos(th1)      *m2*cos
g2 = a2*m2*cos(th1 + th2);
g_v=g*[g1;g2];
% friction effects
```

But here you can exceed this particular term, the theta 2 dot is not generating at all. So, theta 2 dot is not generating that is why you are not getting it. But theta 1 dot is generating. So, that may influence some effect. So, that effect I want to see

(Refer Slide Time: 22:48)

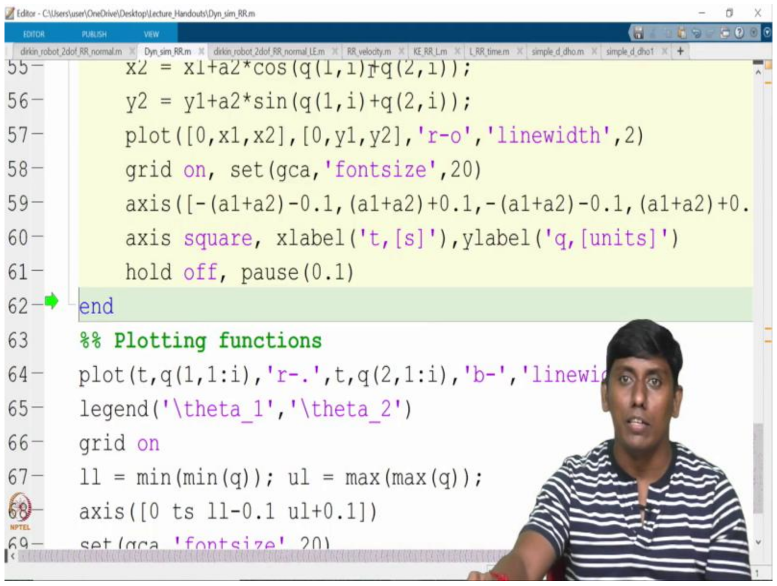


```
g_v=g*[g1;g2];
% friction effects
Fr = [b1*th1dot+c1*sign(th1dot);
      b2*th2dot+c2*sign(th2dot);];
% input vectors
tau1 = 0.1*0; tau2 = 0.5*sin(t(i));
tau(:,i) = [tau1;tau2];
% acceleration vector
q_double_dot(:,i) = inv(M)*(tau(:,i)-(oe_v      v));

% velocity propogation
q_dot(:,i+1) = q_dot(:,i) + q_double_d

% position update
```

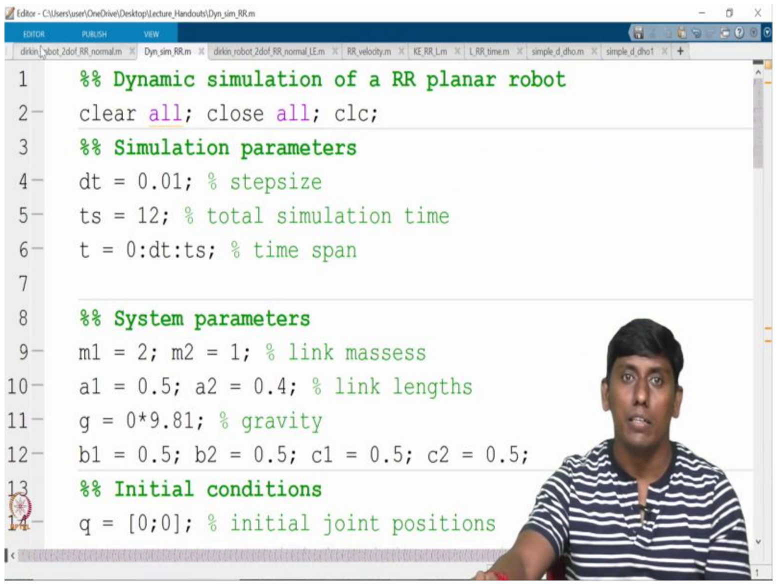


So, far that what I am trying to see even if I put this equal to 0 but I am giving a continuous signal which is varying in this manner. Because we have a very strong friction. So, that is why it is doing it that way. So, now you can see the friction is there that to Coulomb friction is there. It is oscillating but the friction is a avoiding because it is very strong. So, I will just make it you can see it is oscillating it is not so, visible.

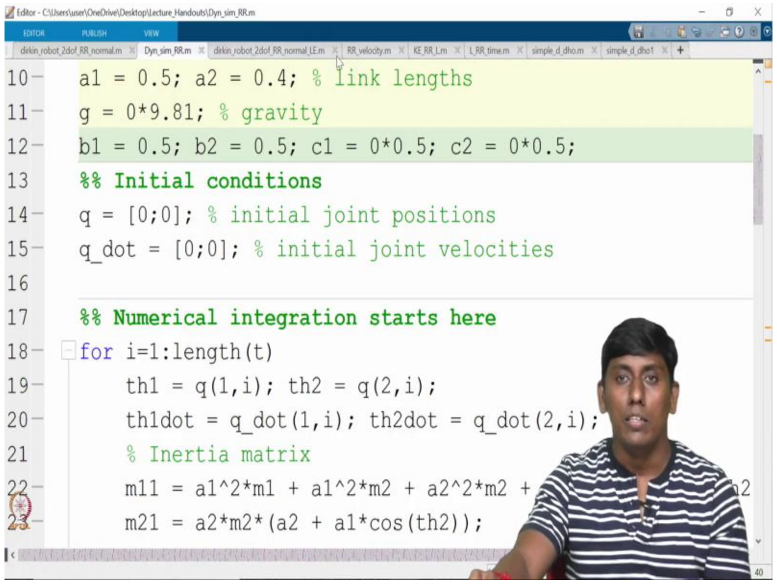
(Refer Slide Time: 23:26)



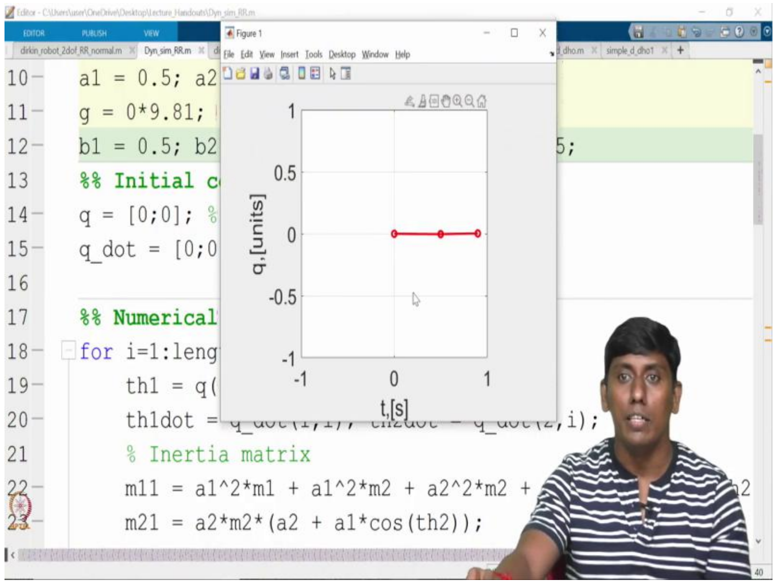
```
    x2 = x1+a2*cos(q(1,i)+q(2,i));
    y2 = y1+a2*sin(q(1,i)+q(2,i));
    plot([0,x1,x2],[0,y1,y2],'r-o','linewidth',2)
    grid on, set(gca,'fontsize',20)
    axis([-(a1+a2)-0.1,(a1+a2)+0.1,-(a1+a2)-0.1,(a1+a2)+0.
    axis square, xlabel('t,[s]'),ylabel('q,[units]')
    hold off, pause(0.1)
end
%% Plotting functions
plot(t,q(1,1:i),'r-.',t,q(2,1:i),'b-','linewi
legend('\theta_1','\theta_2')
grid on
ll = min(min(q)); ul = max(max(q));
axis([0 ts ll-0.1 ul+0.1])
set(gca,'fontsize',20)
```



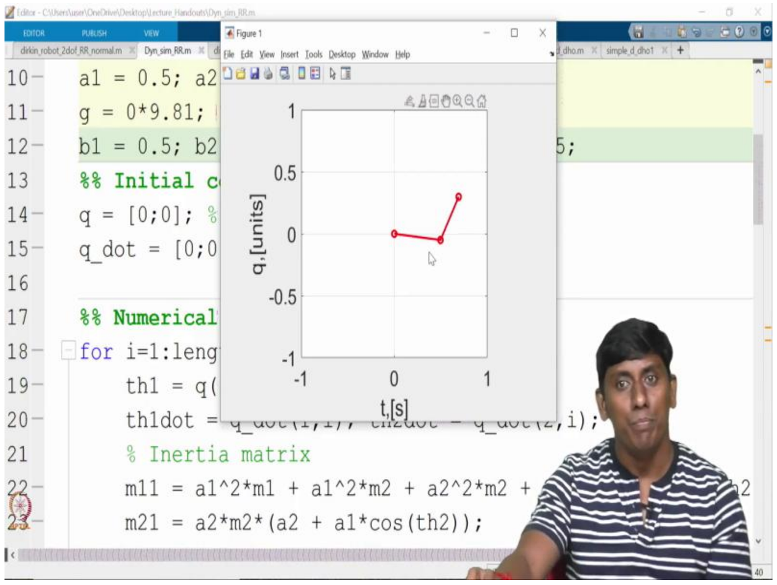
```
%% Dynamic simulation of a RR planar robot
clear all; close all; clc;
%% Simulation parameters
dt = 0.01; % stepsize
ts = 12; % total simulation time
t = 0:dt:ts; % time span

%% System parameters
m1 = 2; m2 = 1; % link massess
a1 = 0.5; a2 = 0.4; % link lengths
g = 0*9.81; % gravity
b1 = 0.5; b2 = 0.5; c1 = 0.5; c2 = 0.5;
%% Initial conditions
q = [0;0]; % initial joint positions
```


Editor - C:\Users\user\OneDrive\Desktop\Lecture_Handouts\Dyn_sim_RR.m

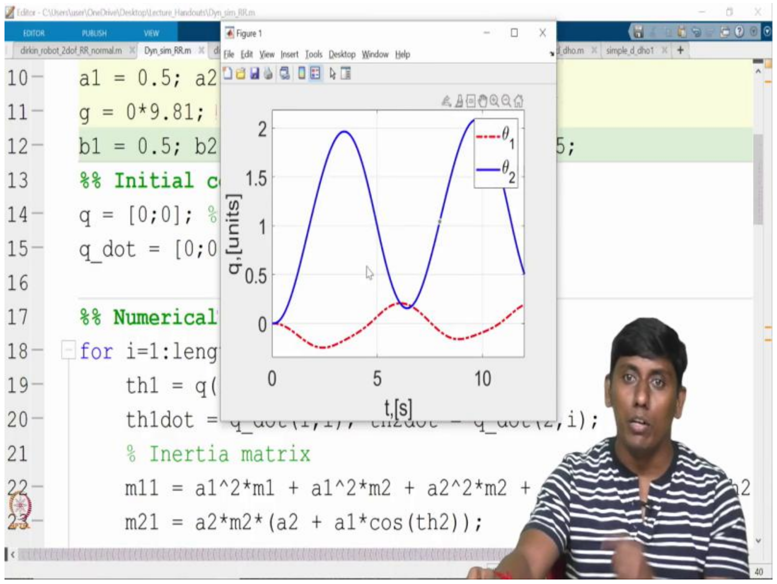
```
a1 = 0.5; a2 = 0.4; % link lengths
g = 0*9.81; % gravity
b1 = 0.5; b2 = 0.5; c1 = 0*0.5; c2 = 0*0.5;
%% Initial conditions
q = [0;0]; % initial joint positions
q_dot = [0;0]; % initial joint velocities

%% Numerical integration starts here
for i=1:length(t)
    th1 = q(1,i); th2 = q(2,i);
    th1dot = q_dot(1,i); th2dot = q_dot(2,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +                2
    m21 = a2*m2*(a2 + a1*cos(th2));
```



```
a1 = 0.5; a2
g = 0*9.81;
b1 = 0.5; b2                                  5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                                     ,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +                2
    m21 = a2*m2*(a2 + a1*cos(th2));
```



```
a1 = 0.5; a2
g = 0*9.81;
b1 = 0.5; b2                              5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                       ,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +           2
    m21 = a2*m2*(a2 + a1*cos(th2));
```

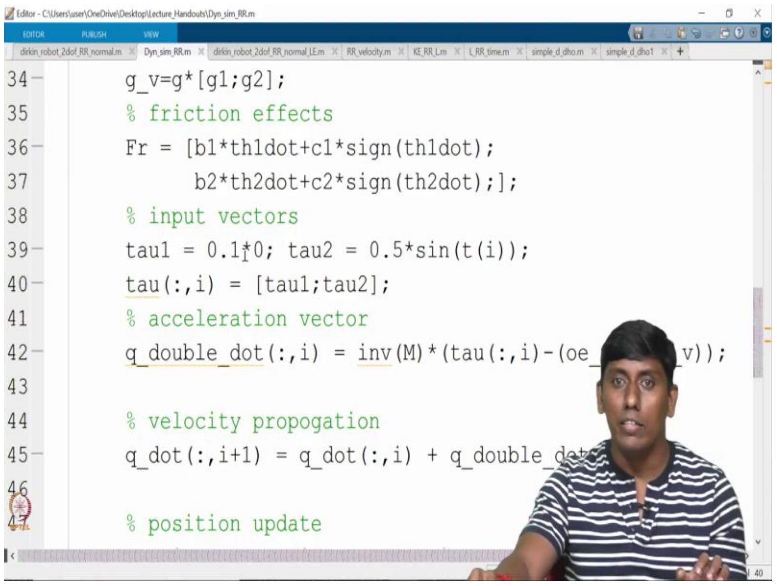


```
a1 = 0.5; a2
g = 0*9.81;
b1 = 0.5; b2                              5;
%% Initial c
q = [0;0]; %
q_dot = [0;0

%% Numerical
for i=1:leng
    th1 = q(
    th1dot =                       ,i);
    % Inertia matrix
    m11 = a1^2*m1 + a1^2*m2 + a2^2*m2 +           2
    m21 = a2*m2*(a2 + a1*cos(th2));
```

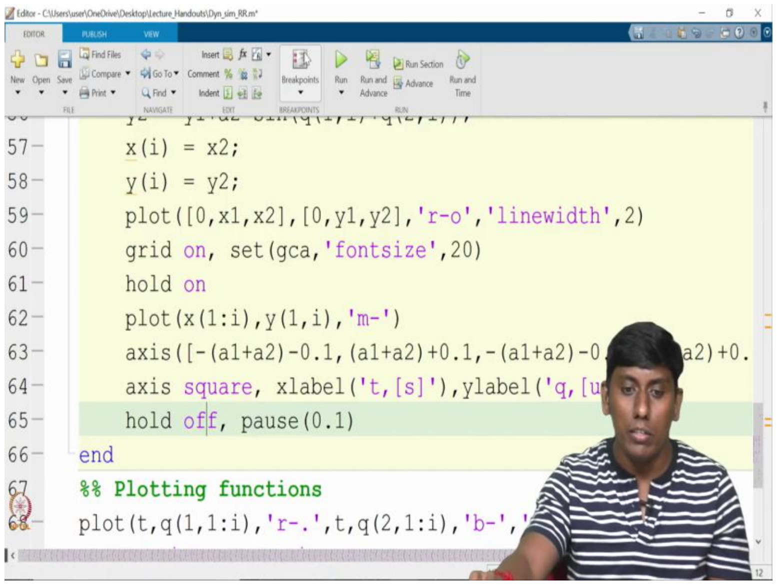
I will make it what you call that Coulomb friction is 0. So, then I will make it. So, here the coulomb friction is 0. Because the coulomb friction is constant force. Only the direction is getting changed based on the rotation. So, now you can see. So, this is start oscillating with the given input. And one not only this oscillating the second bar also getting oscillate because of the coupling effect.

So, that is what I want to show here. So, now you can see the same way if you apply even input. So, you can find a little more you can say, clarity, result. So, now you can see that this is oscillating. And this is also like getting oscillate.

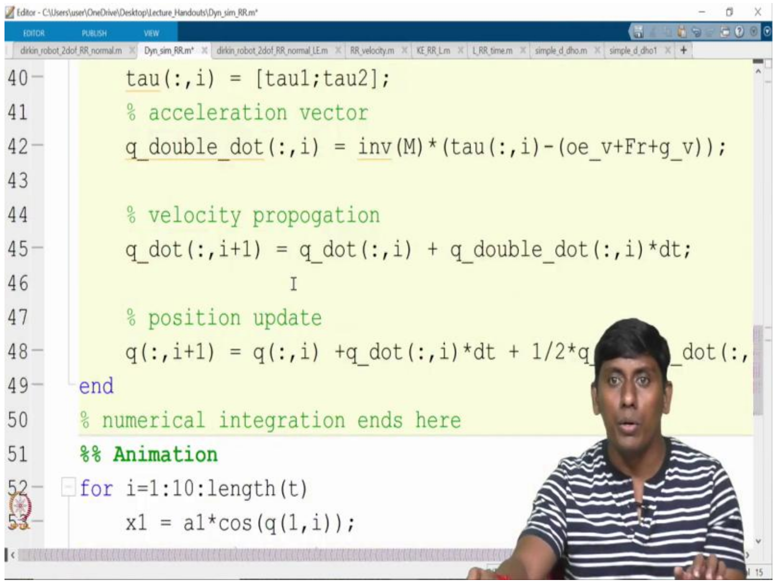
(Refer Slide Time: 24:15)



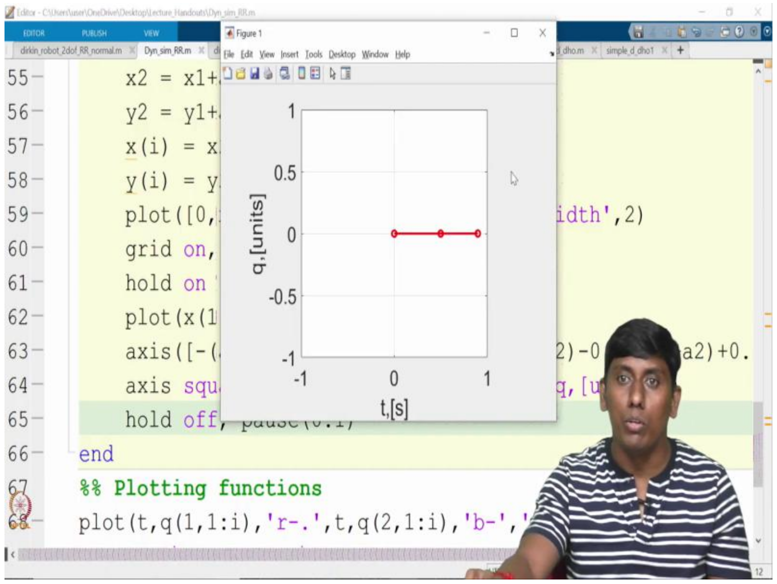
```
        g_v=g*[g1;g2];
        % friction effects
        Fr = [b1*th1dot+c1*sign(th1dot);
              b2*th2dot+c2*sign(th2dot);];
        % input vectors
        tau1 = 0.1*0; tau2 = 0.5*sin(t(i));
        tau(:,i) = [tau1;tau2];
        % acceleration vector
        q_double_dot(:,i) = inv(M)*(tau(:,i)-(oe_         v));

        % velocity propogation
        q_dot(:,i+1) = q_dot(:,i) + q_double_d

        % position update
```



```
        x(i) = x2;
        y(i) = y2;
        plot([0,x1,x2],[0,y1,y2],'r-o','linewidth',2)
        grid on, set(gca,'fontsize',20)
        hold on
        plot(x(1:i),y(1,i),'m-')
        axis([-(a1+a2)-0.1,(a1+a2)+0.1,-(a1+a2)-0        a2)+0.
        axis square, xlabel('t,[s]'),ylabel('q,[u
        hold off, pause(0.1)
    end
    %% Plotting functions
    plot(t,q(1,1:i),'r-.',t,q(2,1:i),'b-','
```



```
    tau(:,i) = [tau1;tau2];
    % acceleration vector
    q_double_dot(:,i) = inv(M)*(tau(:,i)-(oe_v+Fr+g_v));

    % velocity propogation
    q_dot(:,i+1) = q_dot(:,i) + q_double_dot(:,i)*dt;

    % position update
    q(:,i+1) = q(:,i) +q_dot(:,i)*dt + 1/2*q        dot(:,
end
% numerical integration ends here
%% Animation
for i=1:10:length(t)
    x1 = a1*cos(q(1,i));
```

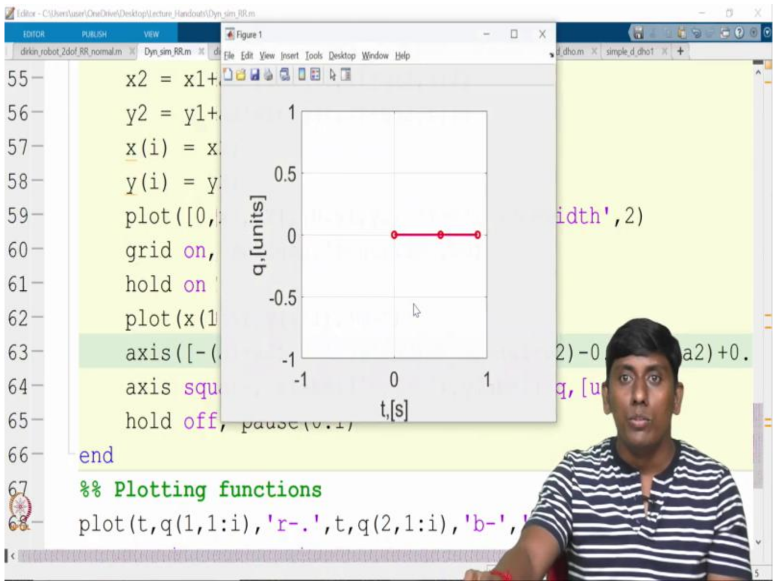


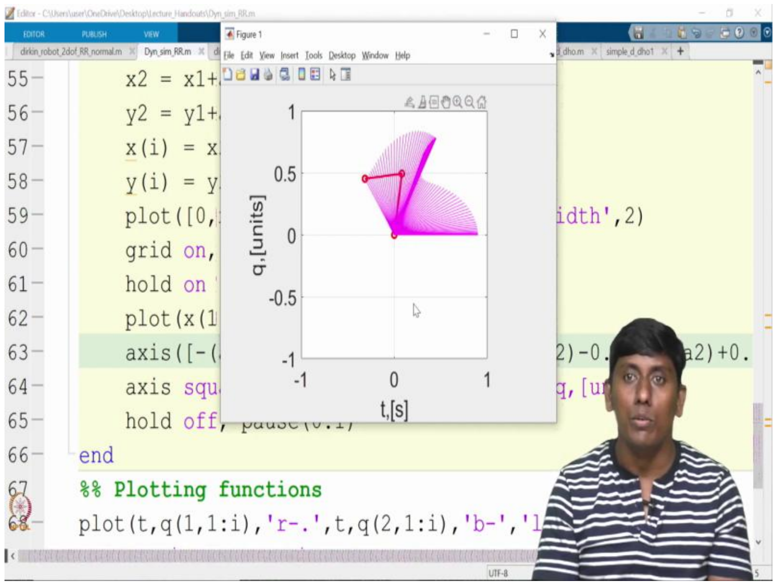
```
    x2 = x1+
    y2 = y1+
    x(i) = x
    y(i) = y
    plot([0,                                idth',2)
    grid on,
    hold on
    plot(x(1
    axis([-(                                2)-0        a2)+0.
    axis squ                                q, [u
    hold off, pause(0.1)
end
%% Plotting functions
plot(t,q(1,1:i),'r-.',t,q(2,1:i),'b-','
```

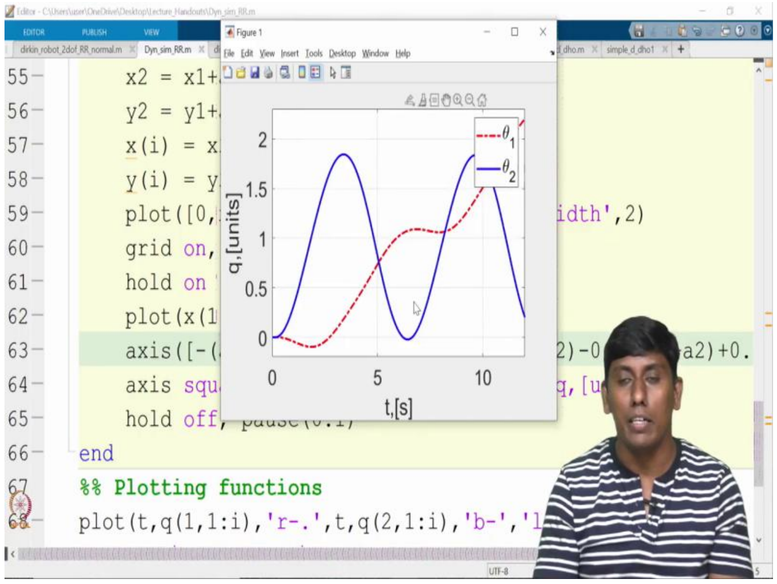
I just want to show that in more clarity. So, I will just give this is also one input. And here I am just adding one more. So, I am just to holding on then and just plotting you can say. So, I will just say that this is x equal to I will just put it x of i equal to x2 and y of i equal to y2. So, this I did not do it earlier. So, now you can see this. So, I just corrected this.

So, now I am plotting so, x of 1 to i. So, y of 1 to i in the sense that time history would be very clear. So, we can see that. So, for that only, I am just doing it here. So, you can see the profile also continuous. So, now I am holding you can say off itself. So, now I am just running this you can see the time history of the total thing in 2D plot. So, I hope that is I will just check it.

(Refer Slide Time: 25:38)


```
x2 = x1+
y2 = y1+
x(i) = x
y(i) = y
plot([0,                                  idth',2)
grid on,
hold on
plot(x(1
axis([-(                           2)-0       a2)+0.
axis squ                                  q,[u
hold off,
end
%% Plotting functions
plot(t,q(1,1:i),'r-.',t,q(2,1:i),'b-','
```


```
x2 = x1+
y2 = y1+
x(i) = x
y(i) = y
plot([0,                                  idth',2)
grid on,
hold on
plot(x(1
axis([-(                           2)-0.      a2)+0.
axis squ                                  q,[u
hold off,
end
%% Plotting functions
plot(t,q(1,1:i),'r-.',t,q(2,1:i),'b-','l
```


Editor - C:\Users\user\OneDrive\Desktop\Lecture_Handouts\Dyn_sim_RR.m
EDITOR
PUBLISH
VIEW
dirkin_robot_2dof_RR_normal.m
Dyn_sim_RR.m
d_dho.m
simple_d_dho1
Figure 1
File Edit View Insert Tools Desktop Window Help
θ_1
θ_2
q,[units]
t,[s]
0
0.5
1
1.5
2
0
5
10
UTF-8

```
        x2 = x1+
        y2 = y1+
        x(i) = x
        y(i) = y
        plot([0,                                  idth',2)
        grid on,
        hold on
        plot(x(1
        axis([-(                                  2)-0        a2)+0.
        axis squ                                  q, [u
        hold off, pause(0.1)
    end
    %% Plotting functions
    plot(t,q(1,1:i),'r-.',t,q(2,1:i),'b-','l
```

This is okay. So, once that is done. I will run it here I just wanted to show one addition. So, with that we can end this. So, I want to show the time history you can see. So, this is what I just wanted to show how the profile is varying. So, this gives how that link is getting into connected. So, even you want to make it much more clear.

So, even you want to make it much more clear. So, you can start from 0 or you just wanted to plot only those corresponding points you can still do it. So, that we can do it in the Simulink based simulation. So, in the sense the block diagram base. So, right now you can see this part is just a plotting option. So, where you can see that q of 1 and 2 we are plotted.

I hope now this is going to give a clarity how to do the dynamic simulation and how to change the parameter. Now you can change your mass and you can change the link, link length, and you can change the frictional coefficient, and then you can get to some kind of idea what your system all about. How your system will behave all those things you can find it. So, with that I am ending this particular lecture the next lecture we will see the same thing in a block diagram basis.

And followed with one important phenomenon called trajectory generation. Because we are trying to do the inverse dynamics without even controller. So, after that we will go to the motion controller. So, in that sense thank you and see you then bye. Take care.