Mechanics and Control of Robotic Manipulators
Professor Santhakumar Mohan
Department of Mechanical Engineering
Indian Institute of Technology, Palakkad
Lecture – 39
MATLAB Simulation on Kinematic Control

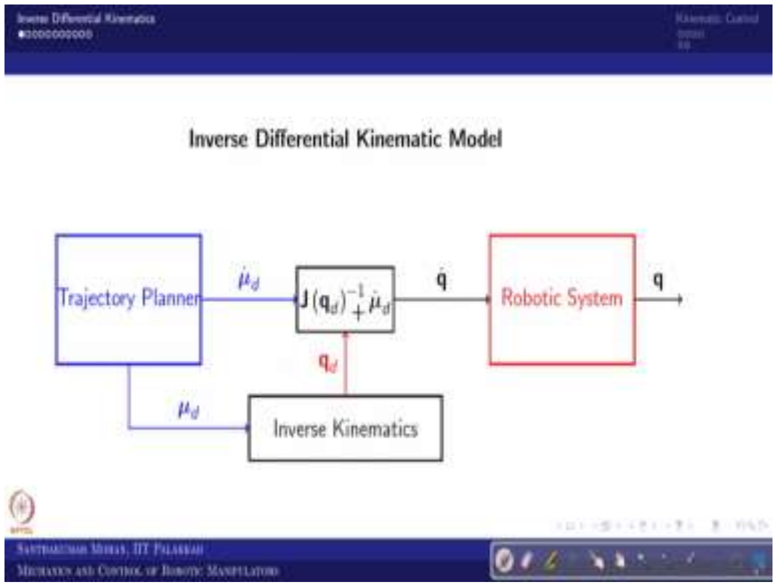
Welcome back to mechanics and control of robotic manipulator. In the last class we have seen how to do a kinematic control, starting from inverse differential kinematics; that is also we can call as a kinematic control because it is an open loop. So, in this class we are trying to see the kinematic control including inverse differential kinematics in the MATLAB simulated form.

(Refer Slide Time: 00:39)

So, whatever we have derived the equation, those things we are trying to do. So, you can see like the inverse differential kinematics both joint space or task space; so, how one complement other that we can see. So, finally we will end up with a kinematic control; we will see joint scheme. So, there we talk about computed velocity control, where we are computing the velocity in the form of kinematic control scheme.

So, where feed forward term and feedback term will be there. So, even if we assume that the feed forward term is not there; so, then we can consider as a proportional control. Then there would be a steady state error, then the steady state error we can rectify with; so, you can see a proportional integral control. These all we have seen in the last class; so, in this particular lecture we are trying to see that the same thing in MATLAB.

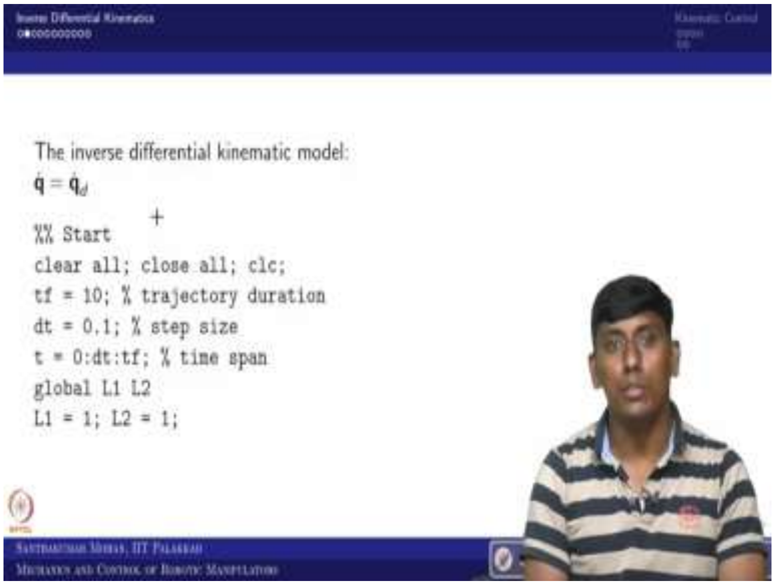
(Refer Slide Time: 01:30)



So, for that we are taking a simple inverse differential kinematic model which we have seen in the last class. So, we would be assuming that the mu desired dot and mu desired are known. Then what we can see, so we need the q dot; the q dot we can write as J of q desired inverse into mu desired dot; where this q desired can be obtained from the inverse kinematics, so we can do it.

So, even if the trajectory planner is giving straight away the joint space; so, q dot desired and q dot, then that is straight forward, where the q dot can be written as q dot desired. So, that is why that picture I have not plotted here; however, we can see in MATLAB, so how we can do it.

(Refer Slide Time: 02:18)

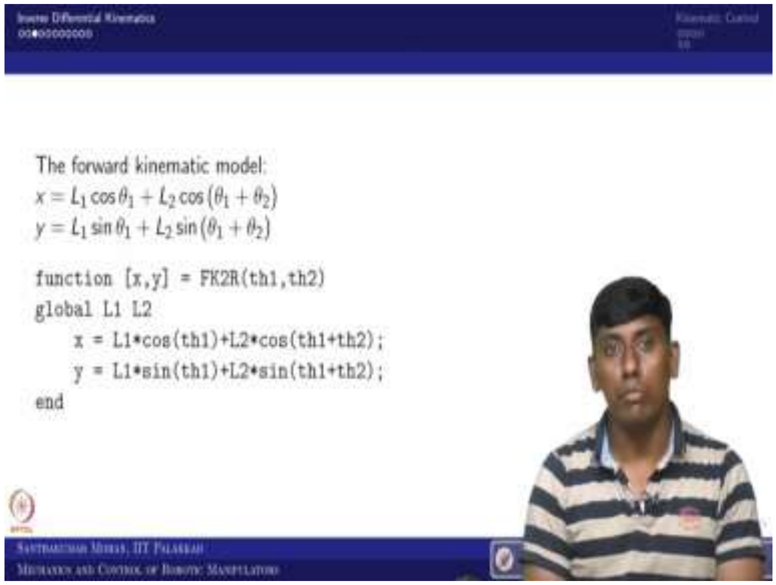


So, for that we are saying the first one is in the joint space, where q dot can be considered as q dot desired; this is the inverse differential kinematic model. So, here we assume that it is straight forward, so where the desired is known and we are assuming that is the actual; so, then we can write the MATLAB code in this way.

So, since it is a differential equation, so we are trying to solve the q of time by numerically integrating. So, again we will bring the Euler integration; so here you can see the 10 is the total duration of the trajectory planner; or probably you can say simulation. So, then the step size is here we have given us dt, and then the time span goes.

So, here I am doing only one addition because I am going to create a sub function. In fact, in last class itself I told we can use a sub function; so, any how I thought of introducing in this kinematic control itself. So, we are talking about global L1 and L2, where link length L1 and L2 I would be using in sub functions also. So, that is why we are taking it that way.

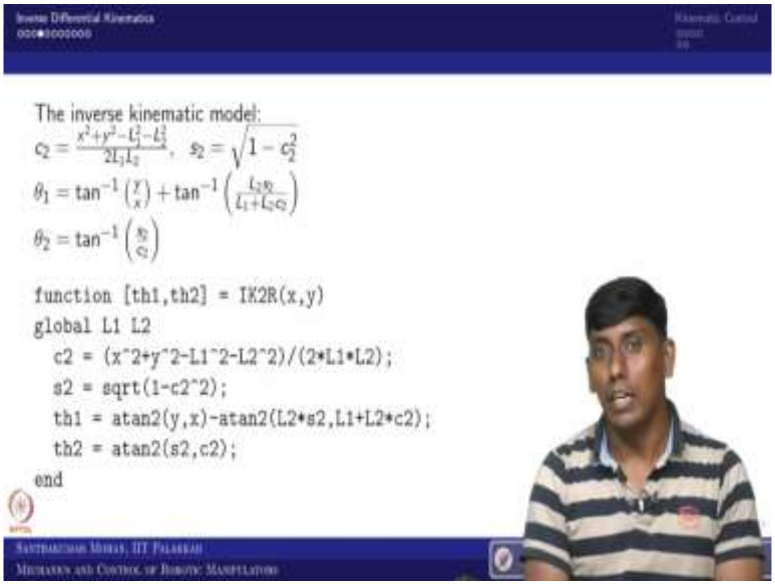
(Refer Slide Time: 03:22)



So, what we are trying to do the forward kinematic model, we want to use it; so x and y we want to write it, so, the x and y write it in this form. So, the same thing I can write as a one of the sub functions, where function x comma y basically this particular function would be returning x and y, for given theta1 and theta2.

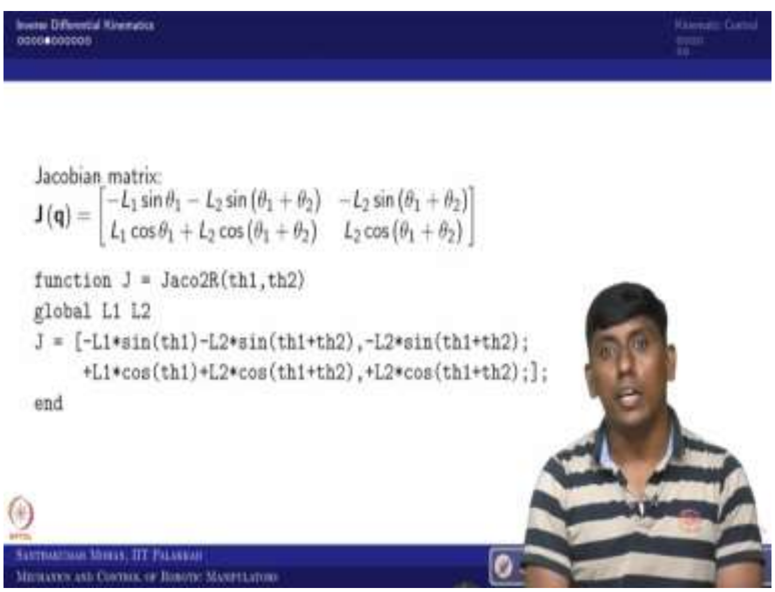
So, even if you are going further and further, you can straight away write mu and you can say q. So, that also we can do where q of 1 would be related to theta1, and q of 2 related to theta2; and similarly, mu of 1 is belongs to x, mu of 2 is belongs to y that also we can use. Since, this is beginning, so we will use independent variable.

(Refer Slide Time: 04:09)



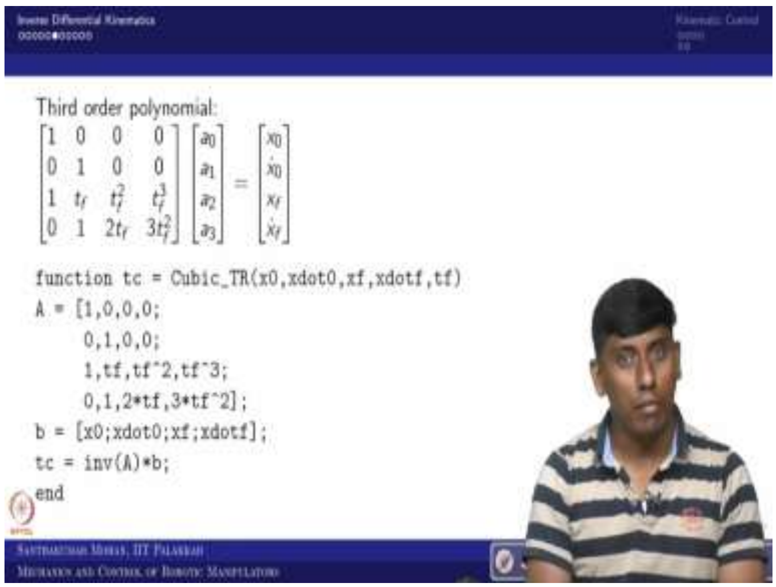
So, the same way we want to know like what is inverse kinematic model; because any of q desired, we would be taking from the mu desired. So, in that sense we need to know this; so, we are taking a inverse kinematic model which we have derived in one of the lecture, the same model we are taking it in MATLAB function. So, you can see this is going to return theta1 and theta2, for given x and y right. So, now these two functions are sufficient; so, then we need to know Jacobian.

(Refer Slide Time: 04:36)



So, the Jacobian matrix we derived 2 cross 2 in the given form; the same thing we can write as a sub function. So, Jaco2R is like Jacobian of the two are serial manipulator, for given theta1 and theta2 the Jacobian matrix would be written in this particular sub function; so, these all the sub function. So, further what we want? We want trajectory because so here we assume that the mu desired and mu desired dot would be there. So, in that case one of the easiest ways is we can take a cubic polynomial.

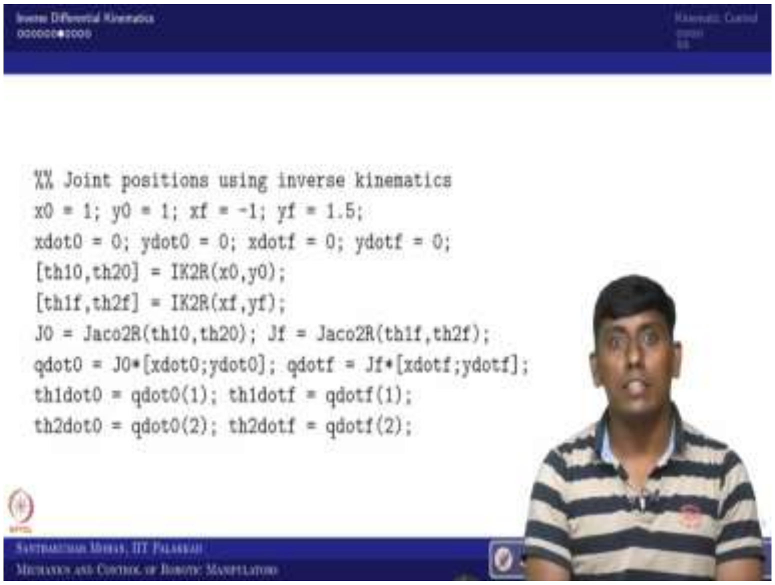
(Refer Slide Time: 05:10)



So, we are taking the third order polynomial or cubic polynomial; so, we assume that tf is given and t0 we have assumed as 0. So, in the sense general we take x as the variable; so, x of t I can derive based on this third order polynomial. So, I can find the trajectory coefficients tc for given input x naught, x dot naught, then xf, x dot of f and tf is given; then I can find the trajectory coefficient based on this.

Once I found the trajectory coefficient what I can do? I can go with x of t; so that is what we are trying to do here. So, first case what we are trying to do? So, only joint space, the q desired and q desired dot are given.

(Refer Slide Time: 05:57)



So, in that case, so we assume that the initial position and final position of the end effector is given, and the time also given already tf; so, which is 10 second. So, now we assume that it is a 2R serial manipulator, where L1 and L2 also given.

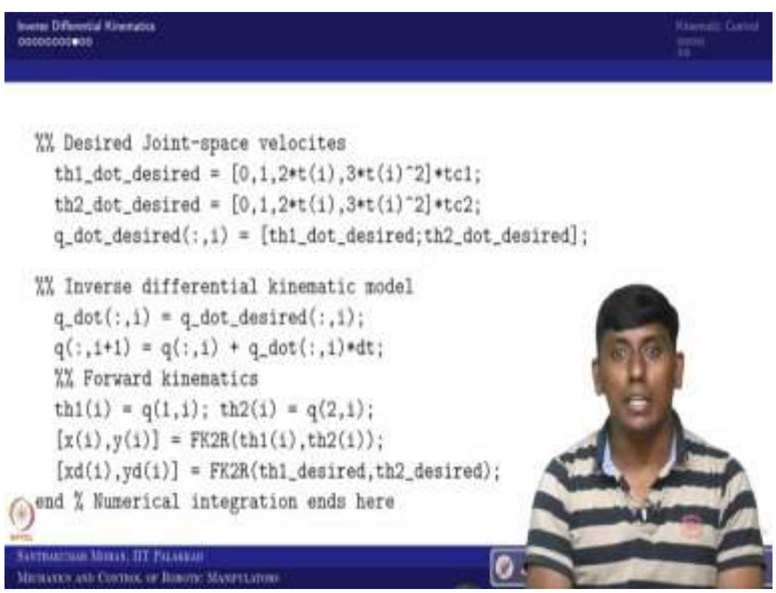
So, now what we can do through the inverse kinematics? We can find theta1 initial and theta2 initial; similarly, theta1 final and theta2 final we can get. Some further extend you want to find the initial velocities, so we can use Jacobian; because here we have, we know the end effector velocities.

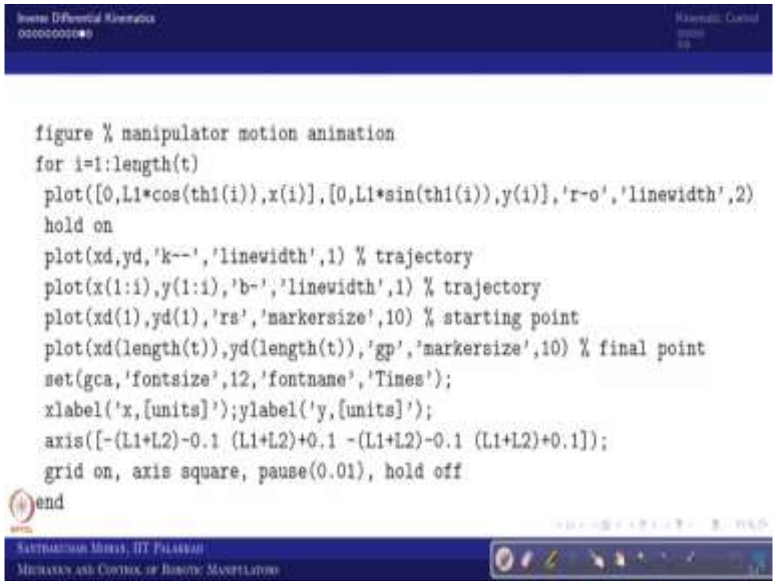
So, we can find the joint space velocity by the Jacobian matrix; so that is what we can do it. So, here we take inverse Jacobian so that anyhow, in this particular case would not be that beneficial; so, that we can correct it in the MATLAB code, so, what exactly we wanted. So, then theta1 dot all those things we defined.

(Refer Slide Time: 06:58)

Then we send it to the cubic polynomial sub function; then we will get the trajectory coefficient. So, then we assume that the initial conditions are not equal to the same; so, then you can find the difference, why we need kinematic control. So, for that we assume that there is a point one radian is the error in both cases. So, then we are going for a numerical integration, there first we define the theta1 of time; in the sense the desired we are trying to do. So, theta1 decide and theta2 desired we have taken as this form.

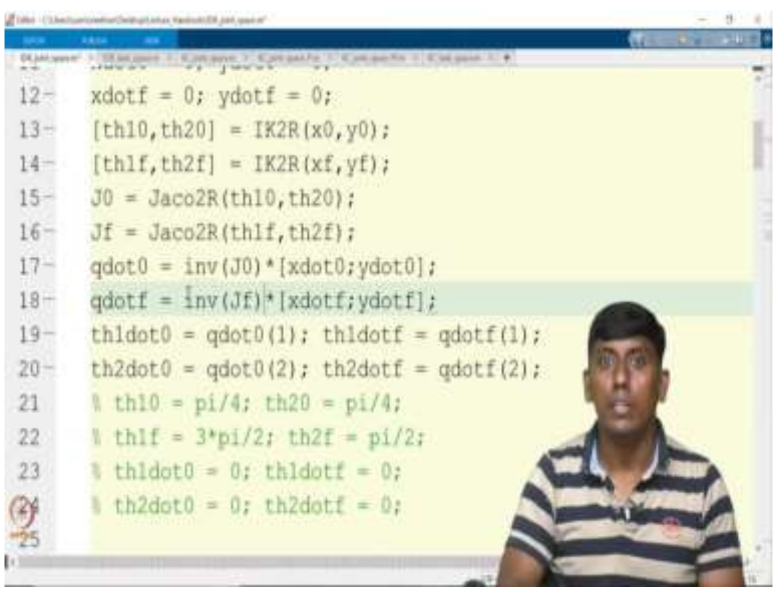
(Refer Slide Time: 07:34)





So, then we are going with q dot desired; so that also we can derive it from here; so, here there is no issue. So, then we are doing the inverse differential kinematic model, where so q dot has a q dot desired. So, then we are making it this and then we are trying to find out the forward kinematic model; so that x desired and x we can find. So, then we can plot it as a manipulator motion animation. So, now we will go to the MATLAB; so, we can see the inverse differential kinematics for joint space, so, this we are doing it here.

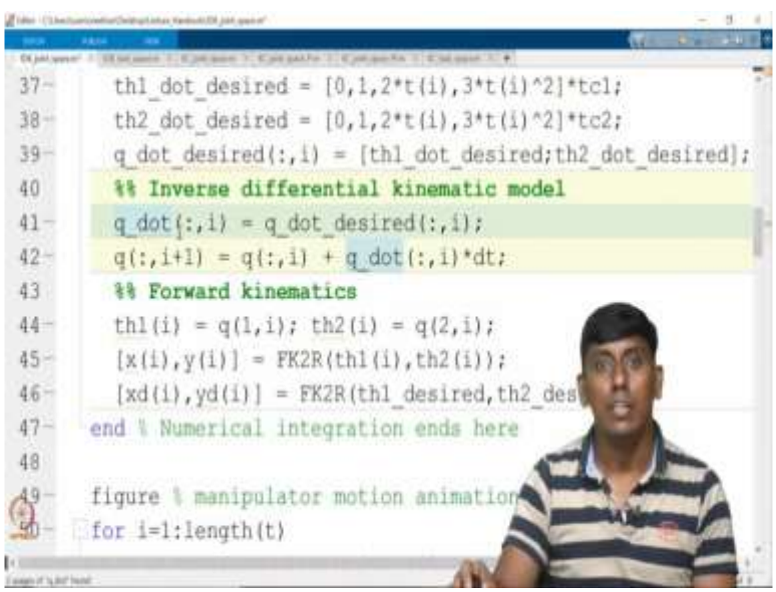
(Refer Slide Time: 08:16)



So, here I will just so correct this, so this is one thing which; because mu is like mu dot equal to J of q into q dot. So, we are trying to find out q dot; so obviously inverse of this. So, now these are the two things we have modified. So, now we have taken L1 and L2 are one meter each or one unit each; so, then the tf is 10 second.

So, the total simulation also like 10 second that is what we have taken. So, now this is the initial, so we are trying to do the inverse kinematics so that we can find theta1 zero and theta2 zero; so, based on that we have found trajectory coefficients. So, if you do not want straight away for example you are giving theta1 initial and final, you have joint space coordinate straight away.

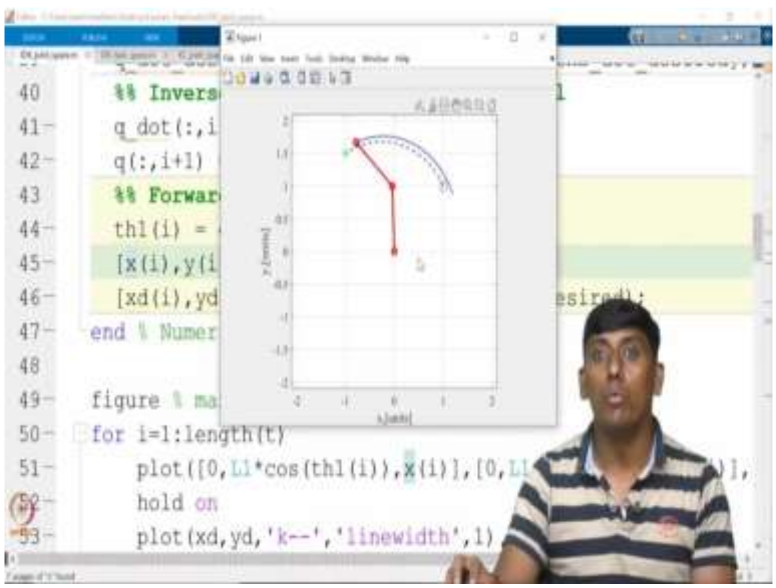
(Refer Slide Time: 09:09)

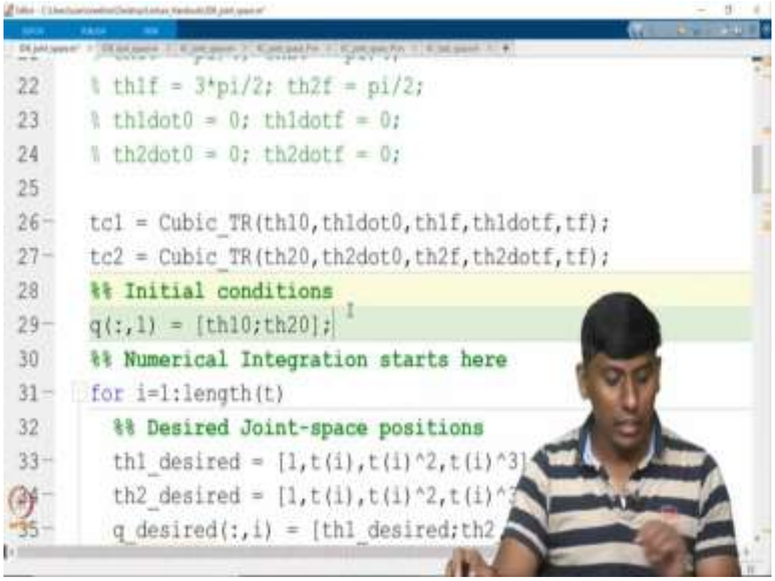


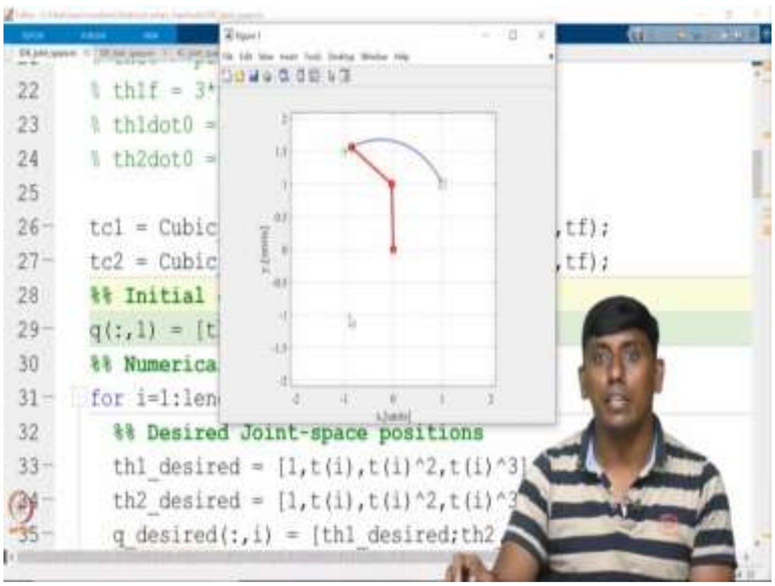
So, then we can use this, so instead of using the inverse differential kinematics and inverse kinematics; we can directly use this, so let us go to the initial condition. Now, we are taking as an open loop control, so theta1 desired and theta2 desired we have derived; and theta1 dot desired and theta2 dot desired also we have derived. And then we are using this inverse differential kinematic model what it says?

So, q dot is q dot desired; so, we are trying to find out. So, we assume that x and y are the actual, and x desired and y desired are the desired task space or end effector positions. So, now we are trying to plot this. At the end we are trying to compare the results, or just I want to plot it this; this is what we have used. And these are the sub functions which we have seen in the slide also. So, now if I run this, now if I run this; so, I am using a shortcut F5.

(Refer Slide Time: 10:08)



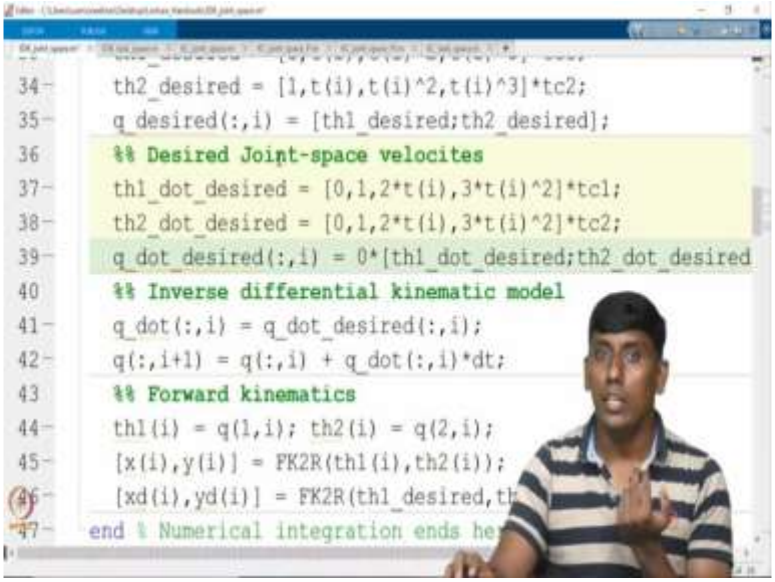


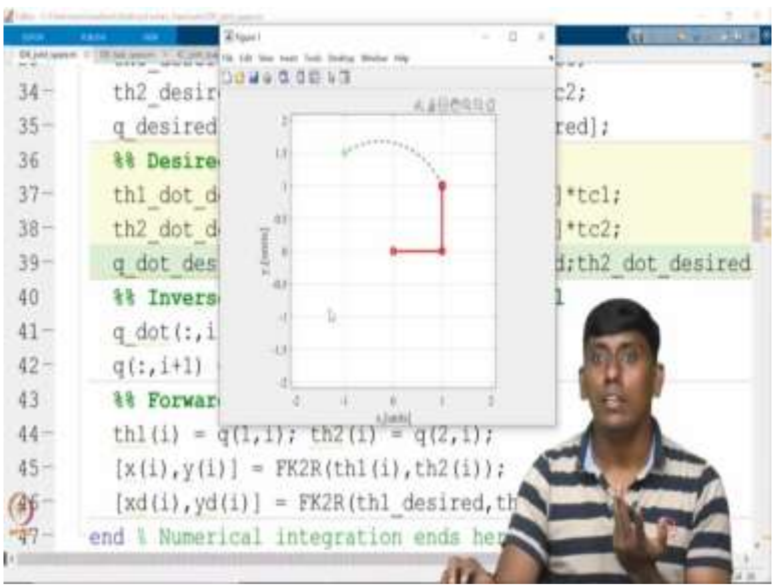


If I run this so what you can see this is the desired, and you can see like the actual is different; because it is an open loop control; it does not know how to correct it. So, in order to make it that clear for example, the initial conditions are same as the desired initial conditions. So, then you can see this would be giving the same trajectory following you can find it already; so, this is what we can see this particular plot is just for making more beneficial.

So, in fact this is not really required because we are trying to see whether the trajectory is following or not. So, now I already said so this is the case. So, now we assumed that the desired is like zero, or you can say q dot desired is zero; so, this will not even propagate.

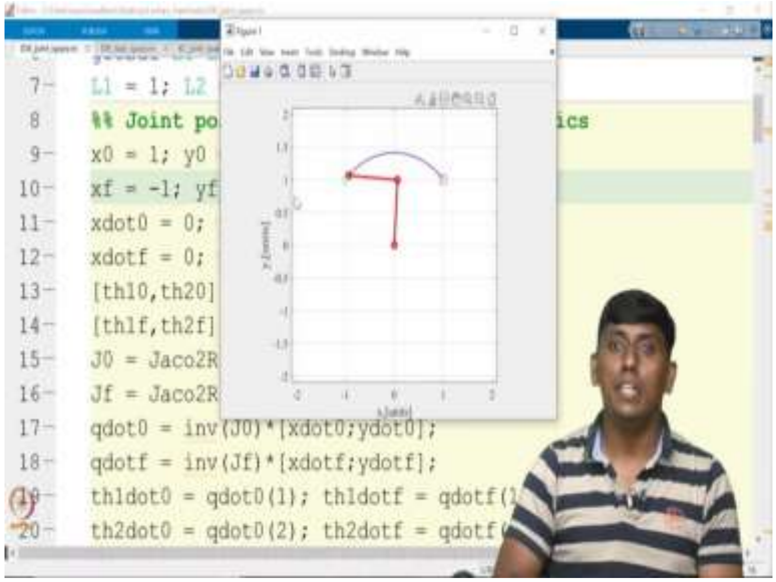
(Refer Slide Time: 10:56)





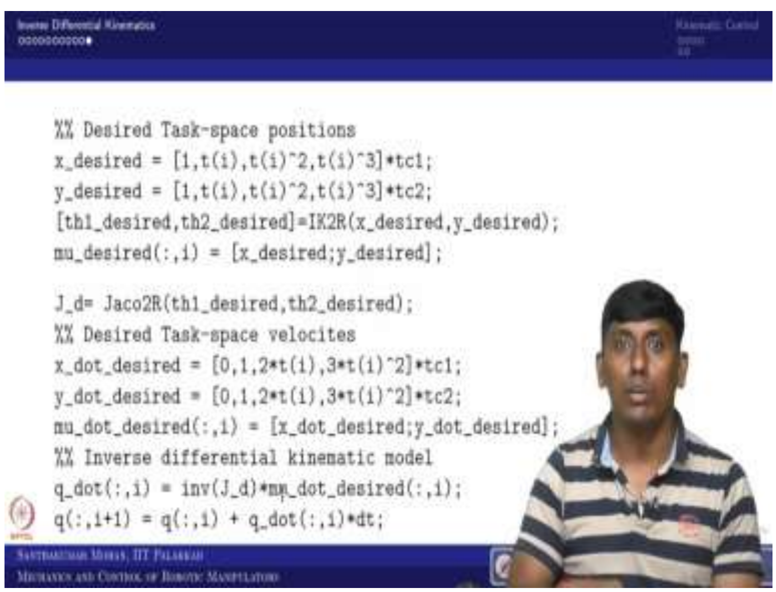
So, I assume that this is like a zero; it one it actually like will not go. So, in order to understand this is the shortcut which I have used. So, it is not propagating why? Because we do not have any push. So, that is why I said this inverse differential kinematic model call open loop control will work or feed forward control will work when the q dot is desired is non-zero. And the initial conditions of both desired and actual are same; so that is what we have seen. So, now even you want to change this.

(Refer Slide Time: 11:32)



For example, I am trying to change this probably so 1; so, I am just changed this. So, you can see it is trying to follow; so, this is what we did in the joint space inverse differential kinematic. So, the same thing we can do it even in the task space. So, there would be a small change in the code. So, that is what we are trying to see here.

(Refer Slide Time: 11:58)

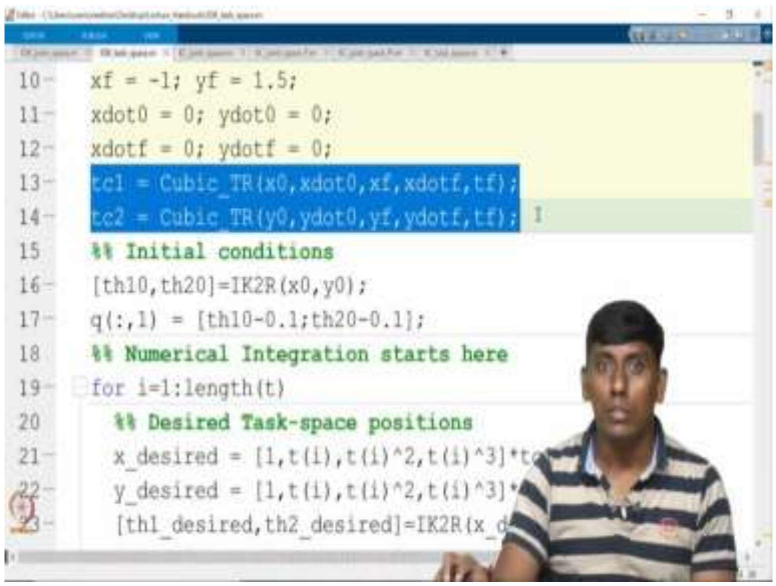


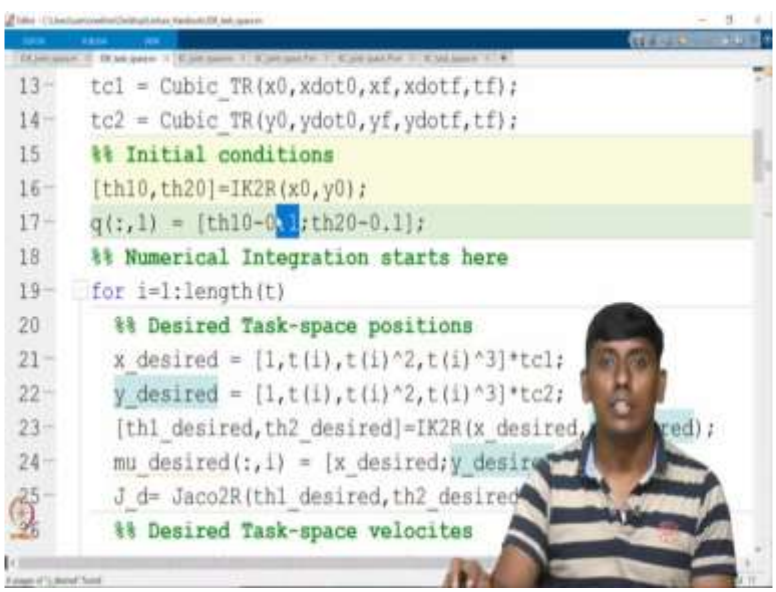
So, this is the task space, so now what change you can expect? So, the theta1 desired and theta2 desired become x desired and y desired. So, now theta1 desired and theta2 desired you want;

then you can use inverse kinematics; so that is what we have done. And here mu desired and mu desired dot would be found; so, this is what we have get it.

You can see mu desired and mu desired dot we have obtained; so, then so the q dot is inverse of Jacobian multiply with mu dot desired, so, this is what we have obtained. So, the code; the prior of this and after this we are not going to change; so, only this content is going to change. In fact, I want to show it here.

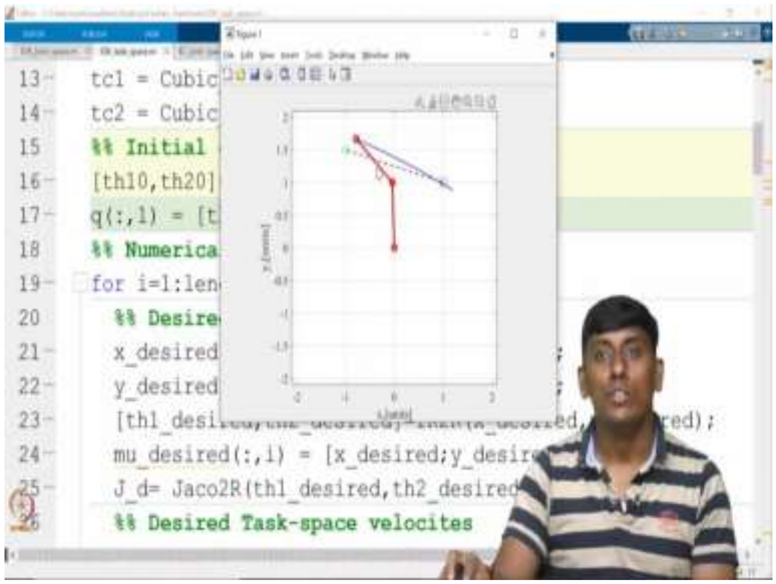
(Refer Slide Time: 12:44)

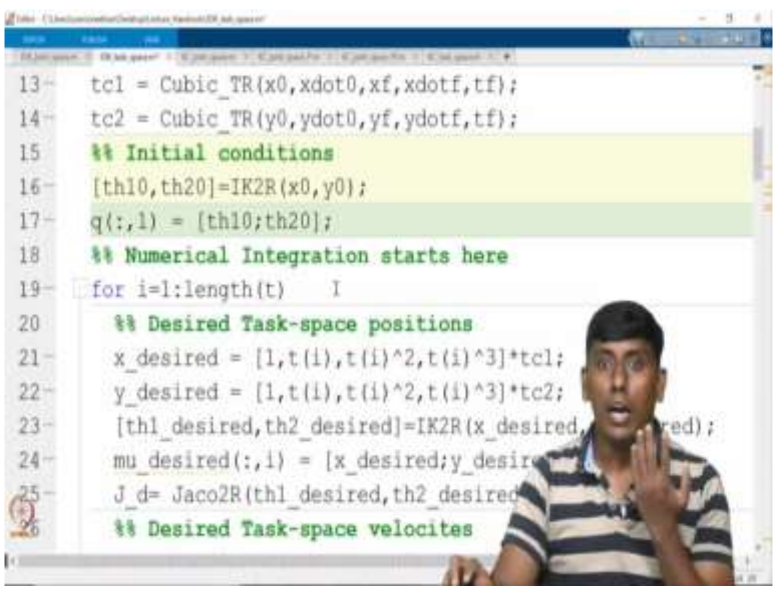




So, this is the inverse differential kinematics of the task space. You can see like this segment are same and the input whatever we have taken is same. So, only thing the trajectory coefficients we are calculating for x and y; so, because of that the x desired and y desired we are getting. Similarly, x desired dot and y desired dot we are getting. The same profile we have given and here also we have given induced this error.

(Refer Slide Time: 13:13)



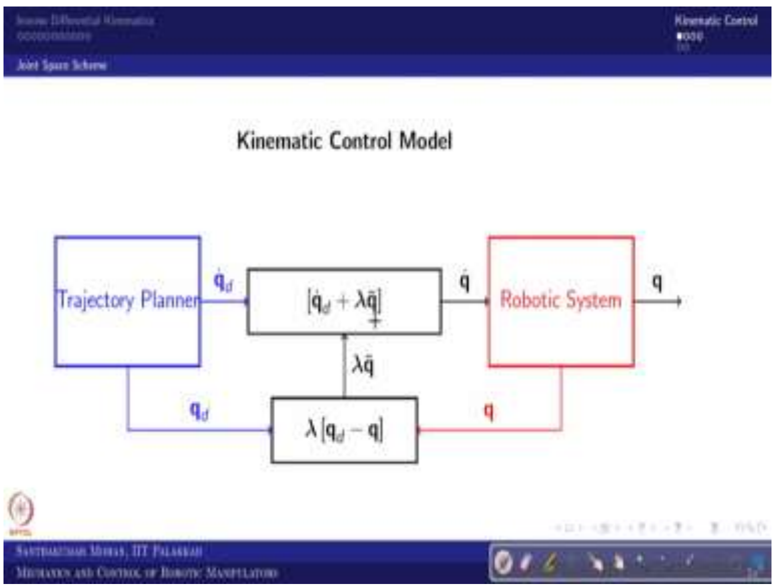


So, you can see that so now, so you can see like the profile is actually straight line; so earlier it was not. So, now if you are look at it this, you can see like it is non-zero or non-zero error I suppose to say; or you can say the desired initial position and desired or actual initial positions are not same. So, this is the desired initial position, and this is the actual initial position; these two are same, so that is why the profile is not supposed to be followed here.

So, now in order to make it that the condition what you have seen is this. If the velocity is non-zero and the initial actual and desired are same; so, then you will find that this would be a good control. That is why it is actually we call open loop control, because it is a feed forward.

So, there is no compensation for the actual case. So, now even I introduce a small friction or some kind of uncertainty; this will not be following it. So, that is what we want to get it here; so, will go to the slide. So, then we can see how we can use this as a kinematic control.

(Refer Slide Time: 14:25)



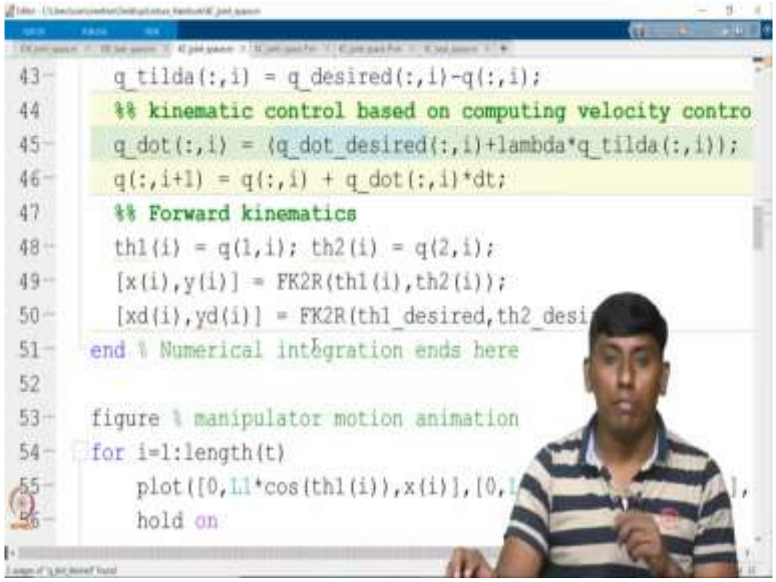
So, for the kinematic control what we have taken? So, one of the conditions we have taken that. So, if the first order error dynamics is going to be stable; so, for that what we have taken? So, lambda is the one of the positive constants we have considered. So, then this is the kinematic control model; so, this is going to be a feed forward term, which we have seen inverse differential kinematics. Now, this is the feedback term, this is what we are going to call as a proportional control.

(Refer Slide Time: 14:56)

So, what code would be changing? So, the initial thing all are same. So, only thing we are adding the control parameter, where we have seen the lambda, that lambda is coming here; so I have taken as 4. And further these all same; so only added thing is the joint position error we have added. And you call the q dot change to q dot desired plus lambda into q tilde; the q tilde is q desired minus q.

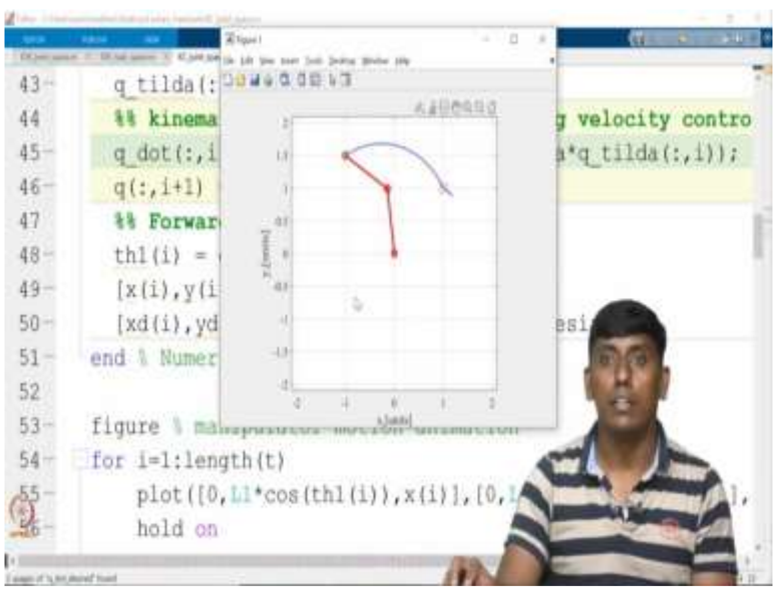
So, now this is the change which we expected; so now based on that what we can look at it. So, the code would be getting change; so, we can see this is the, so kinematic control we are trying to see. So, what we can see here? So, these all same, whatever we have done earlier.

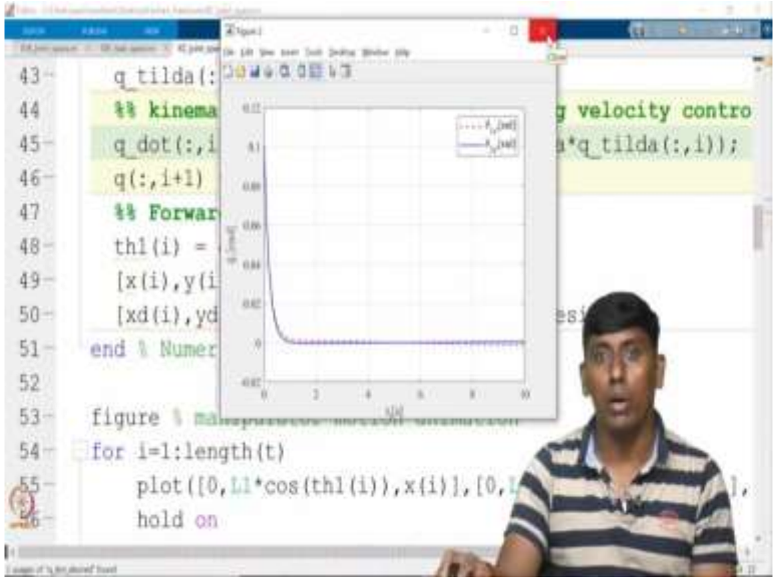
(Refer Slide Time: 15:49)



So, even we have end up with the initial actual position is different from the desired position; and he lambda we have taken as 4. We will see if we change what will happen; and the joint position error is coming here, and the kinematic control is changed here.

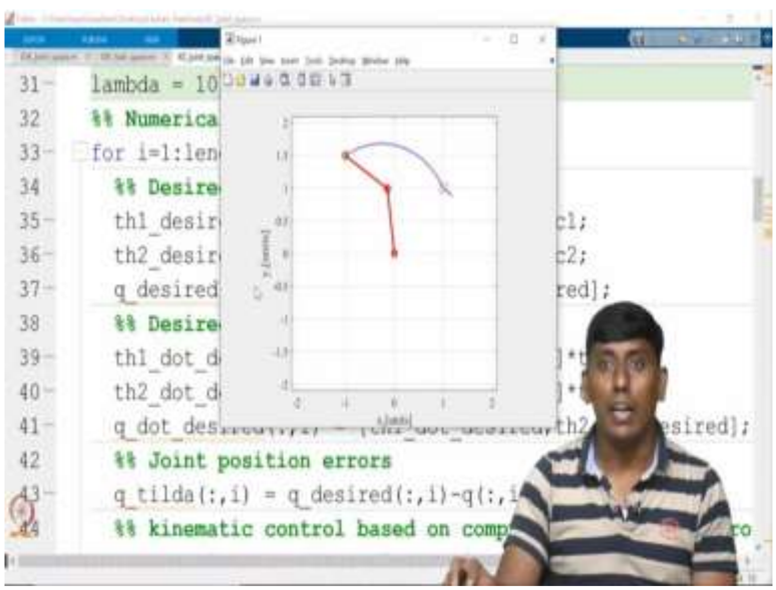
(Refer Slide Time: 16:06)

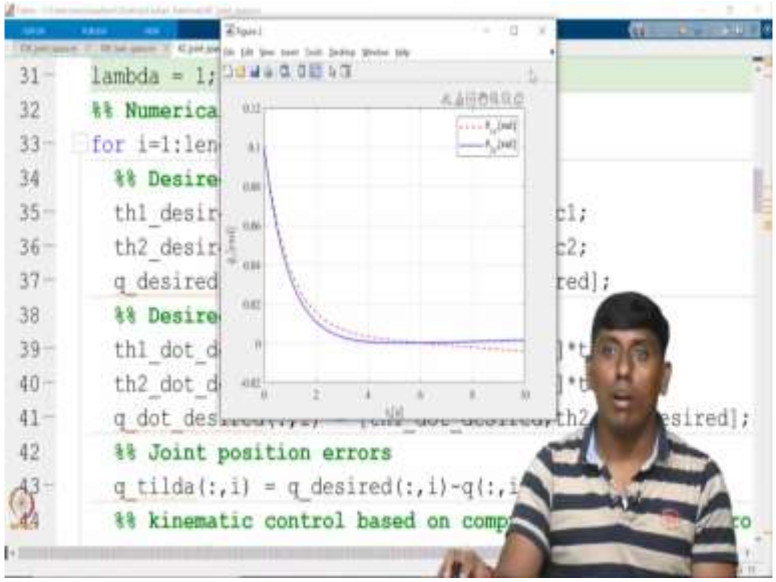


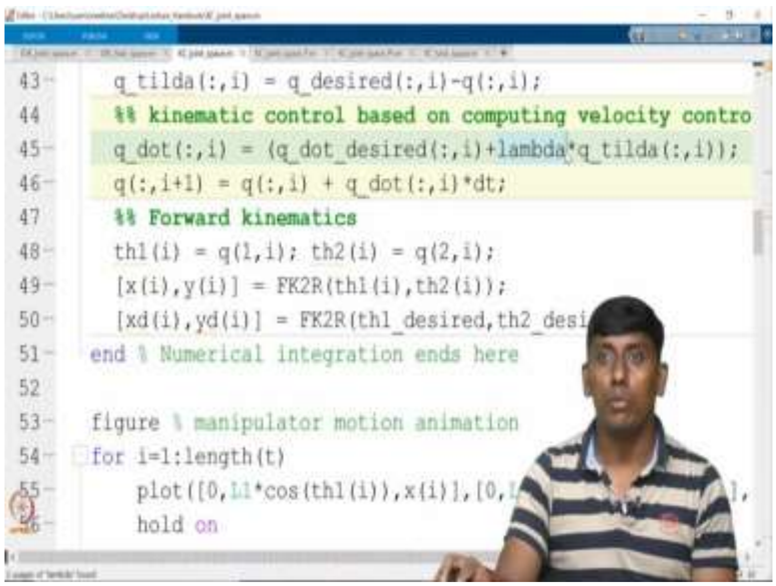


So, now if I run this, you can see that even though the initial is change; but you can see it is trying to follow it. So, that is what we can look at it. So, if you look at the error, initially it is 0.1 radian in both theta1 and theta2 that is converge to 0. But it is having a small error, so that error can be neglected by introducing some integral control. So, now if I increase this lambda or decreased my lambda, so what will happen if I increase?

(Refer Slide Time: 16:40)



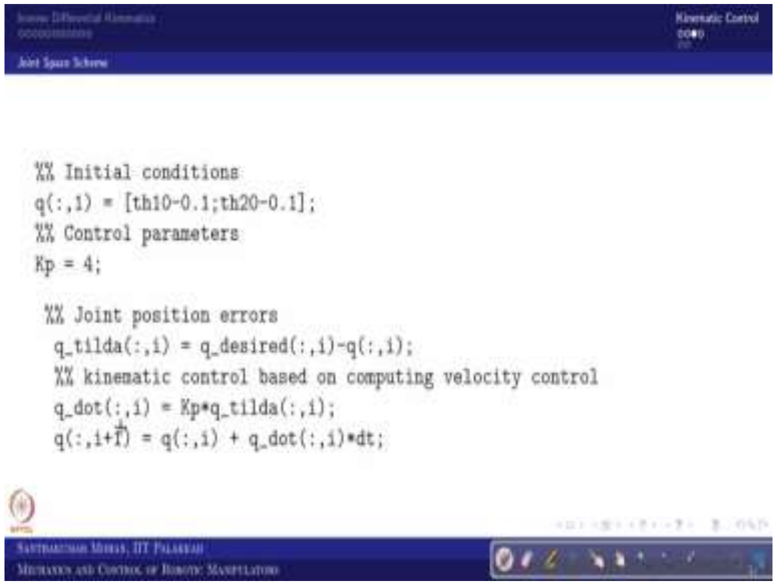


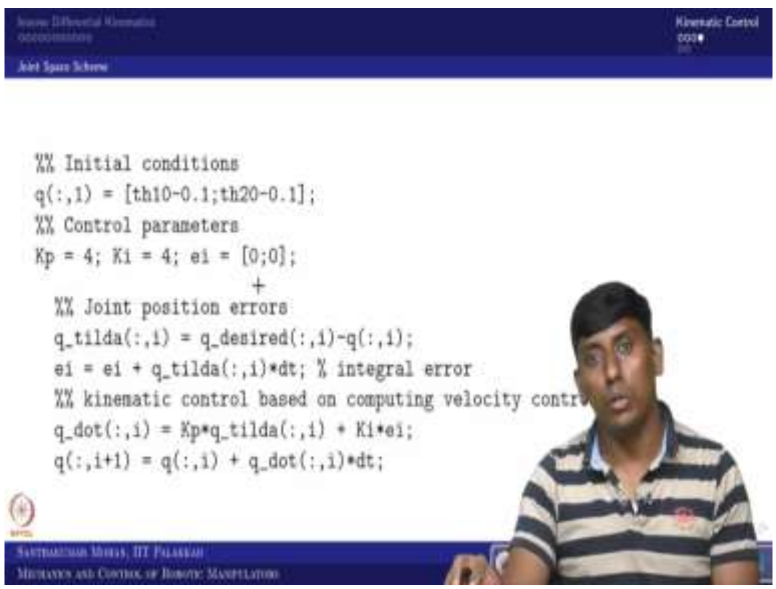


So, you see it is like much faster; and as well as that the error would be eliminated very fast. Earlier it was going here, but now it is coming and error magnitude also very much reduced. So, like that you can take it, so instead of that lambda I consider as a 1. So, then also you can see the error would be taking; it is very significant, it is not followed.

And the time taken for the error also converges; in fact, it is not even converged; if I by 10 second, some error is there. So, these all what we have seen as a kinematic control. So, what we have changed in the inverse differential kinematics? We have added the proportional control, where you have taken error multiplied with one of the positive constants; that is what we have done. So, now we will go to the slide; so, where we have considered only proportional control.

(Refer Slide Time: 17:40)

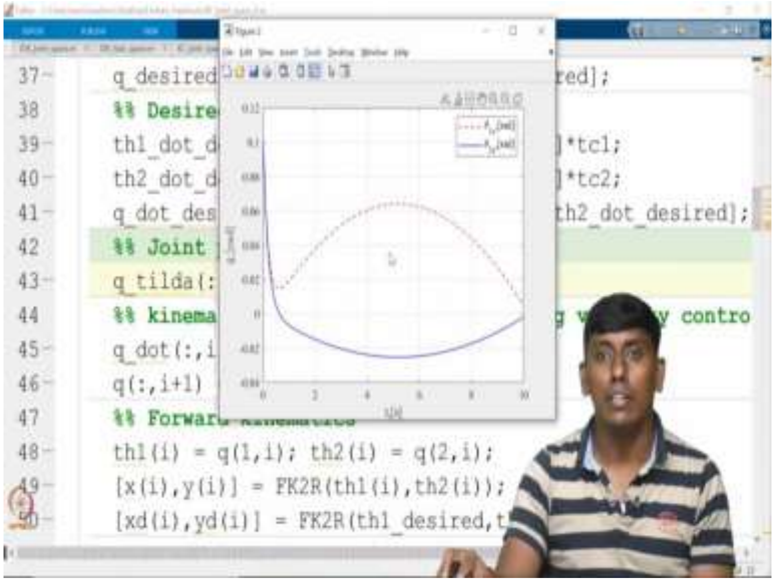




You are assuming that the q dot desired is unknown to us. So, then what happened the Kp we have taken; and Kp q tilde is the q dot. So, the q dot desired will not be coming into a picture; so, this would be having what we have seen in the last lecture. This would be having some steady state error; so that steady state error can be overcome with the integral control.

So, where we have brought the integral control term, and then you can see this is compensated with both the terms. So, here we need to know the integral error. So, initially we assume that the integral error is ei equal to 0, 0; then it would be propagated. So, in order to get understand this will go to again MATLAB window.

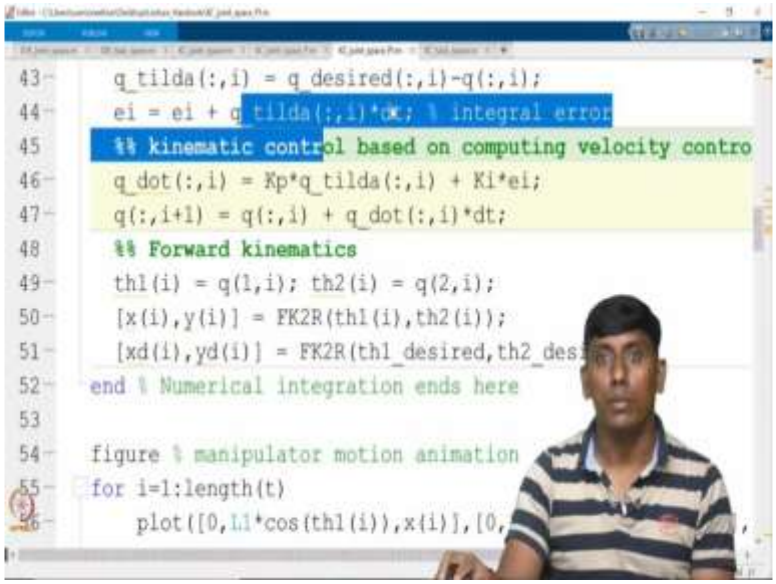
(Refer Slide Time: 18:26)

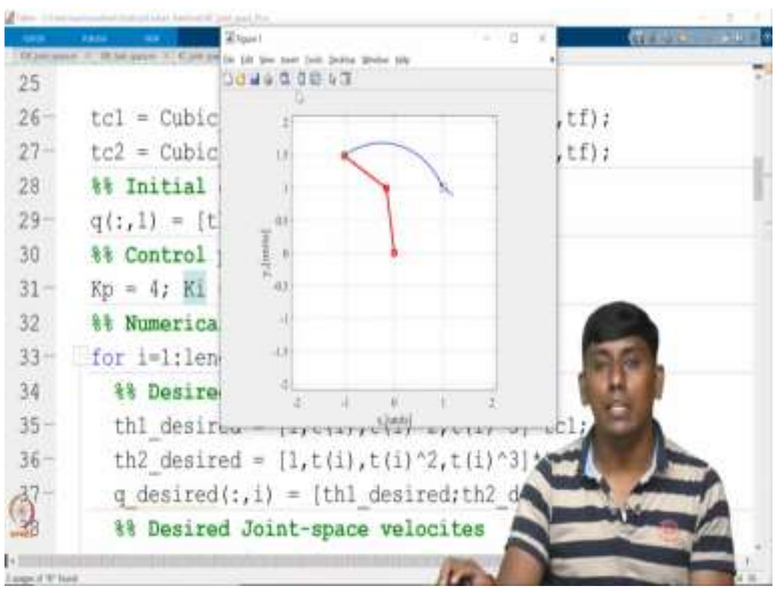


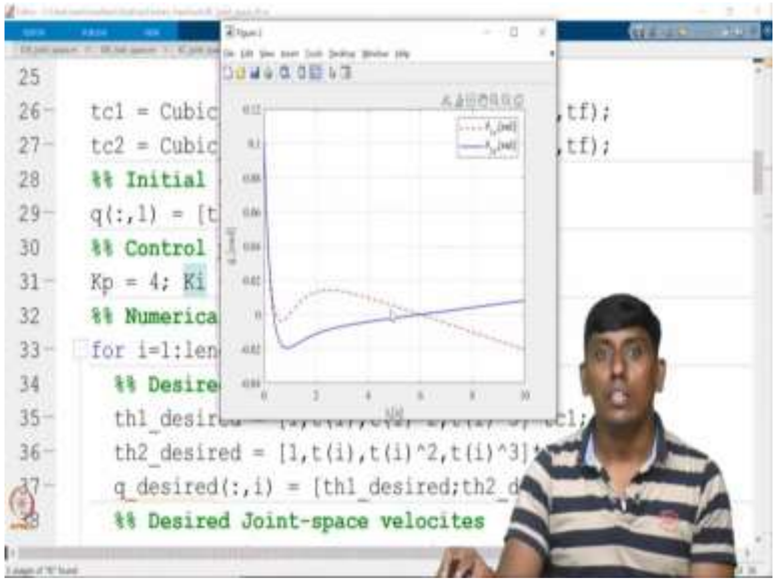
You can see this is the proportional control; so, where you can see Kp equal to 4. And this is the same thing what we did, only thing the q dot desired is taken away. So, now so since this is cubic polynomial, where q dot desired is there; but you have not considered. So, that there would be a error which would be some what visible to us.

So, since it is fast moving, so there is a error which is significant. So, this error you want to eliminate even without q dot desired; so, then we can go with the integral error. Or, you can say integral error with integral control. So, this is what we can see; so now what we have added.

(Refer Slide Time: 19:10)



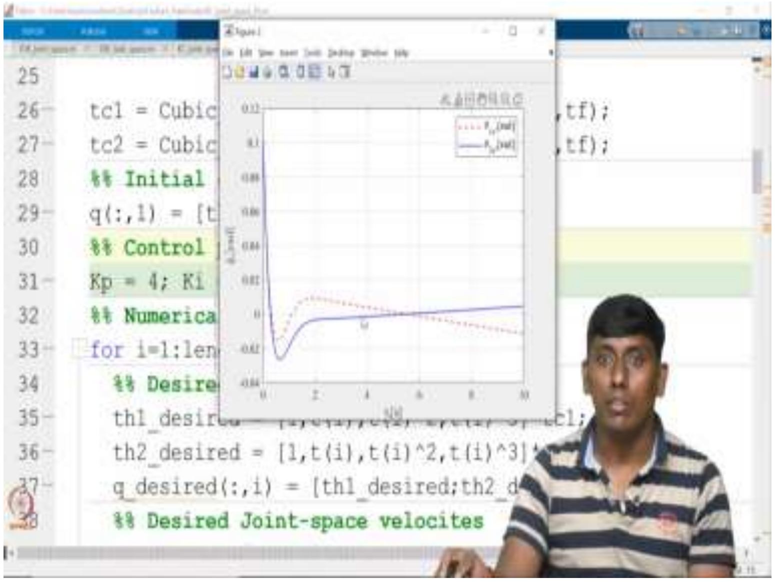




So, we have added Ki into ei; so, for that we have taken ei. Ei is ei of previous plus q tilde integrated; so, this is the integral error. So, now if I add Ki some value, so here we have taken as 4. If I added this what you can see the same condition; so, you can see like the error would be reduced. So, when t tends to infinity, it would be like see.

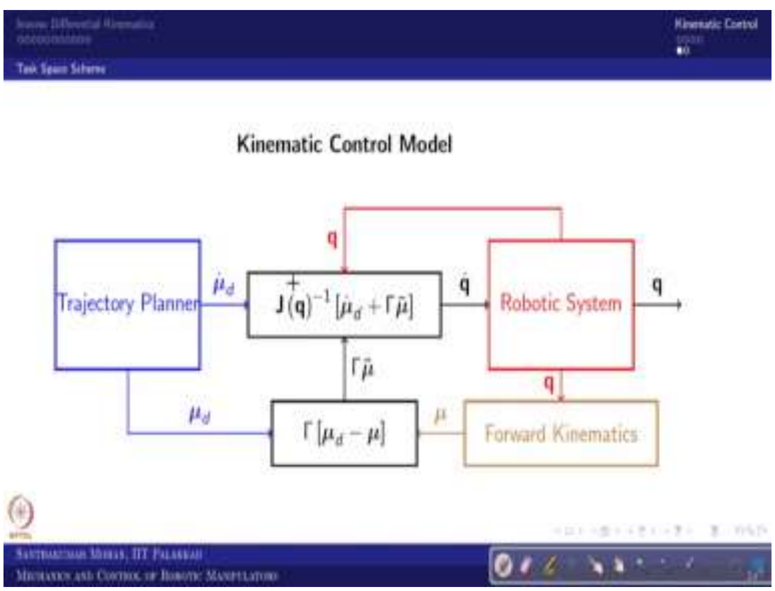
So, it is like reduced significantly; so earlier it was somewhere here; so now it is reduced. Even if we increase Ki further, so then this would be converged somewhere closer. So, if you want, we can check it; so I am just putting 8 just for understanding, just to see whether that you can see it is already somewhat it is faster. So, that is what we can look at it here.

(Refer Slide Time: 20:10)



So, you can see it is trying to converge to 0; so, these all the benefit of proportional and proportional integral. So, in that sense what is desired and what is actual and then you can compensate. So, let us move the final part, where we are seeing the same thing, where the desired trajectory is given in the task space.

(Refer Slide Time: 20:29)

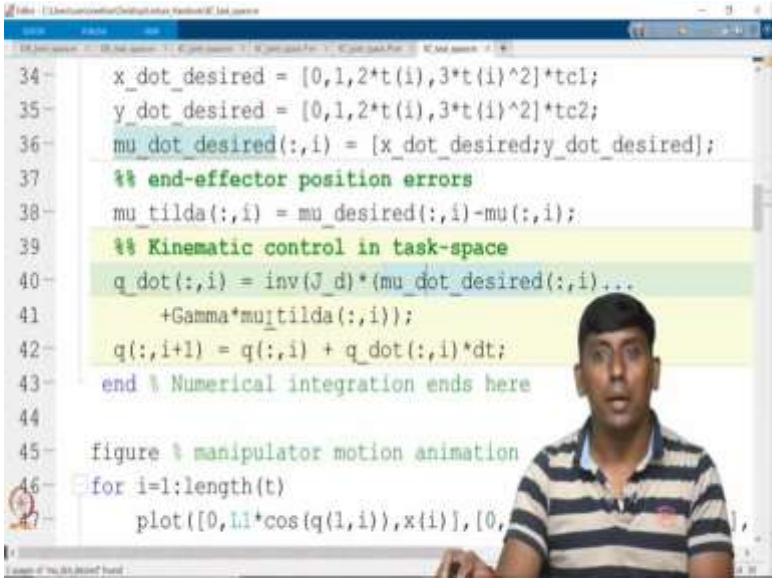


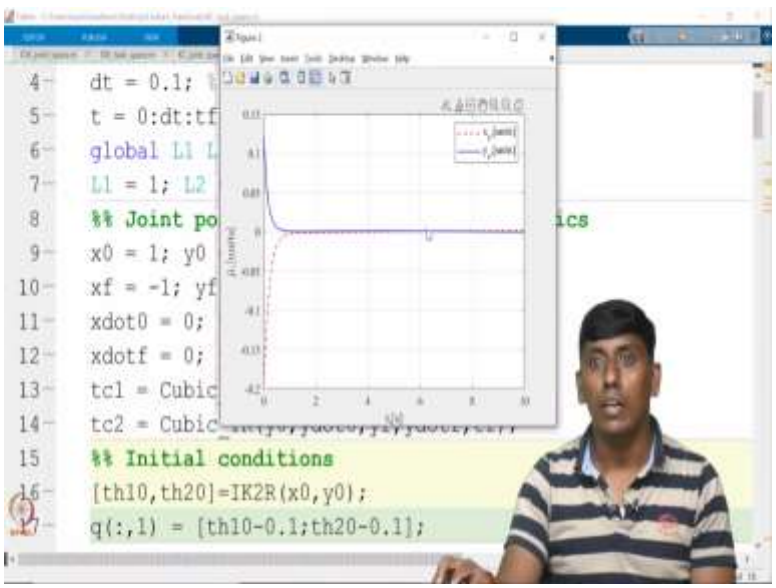
So, then we can do the computed velocity control, where the J of q inverse is coming. So, this is one peculiar because some cases the J of q can be probably infinity; so, where for example in two are serial manipulator. So, if the theta2 tends to be very close to 0 or 180 degree or even 0 and 180; so, this J of q inverse may be tends to infinity. So, that is why we always avoid this kind of cases; but this is what the one additional input, where this control law we can do it.

(Refer Slide Time: 21:05)

So, for that what we are doing? So, we are taking one simple addition. Just for comparison we are taking q; so, then we are taking a gamma and mu dot desired plus gamma into mu tilde; and inverse of Jd. So, this is what we have taken, in fact instead of taking Jd, even we can take J so both will be the same result. Just to get that idea, we will go go to the kinematic control in the task space. So, this is the change which we were saying.

(Refer Slide Time: 21:39)

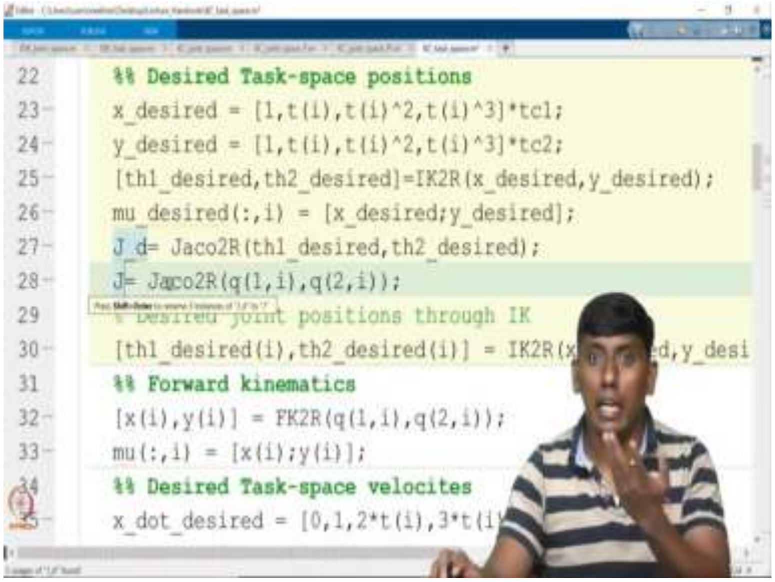


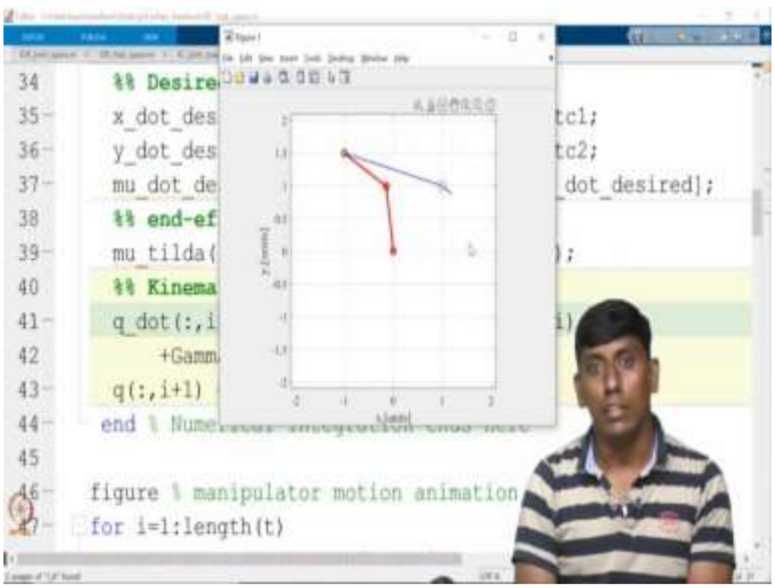


So, the mu tilde we calculate mu desired minus mu. So, then the q dot is inverse of Jd or inverse of J, multiply with mu dot desired plus gamma into mu tilde. So, now again we see that he initial is having error and this is a profile we want to follow, which is a straight line. We will see whether this is following it or not.

So, it is like following it, whereas in the simple inverse differential kinematics it was not followed; and the error is almost 0 in the mu tilde, x error and y error almost 0. So, now even we want to check whether you want to use this only q dot, sorry J desired; it can be even can be use J. So, in that case I will take it, so this like theta1 and theta2. So, or I have to like to use it.

(Refer Slide Time: 22:38)





So, I will just write J, so which means so I just see this; so I will just take it. This is a J, which is from the q; so, this is; so, q of 1 comma i comma q of 2 comma i; in the sense theta1 and theta2. So, now I calculated J, so J I calculated; and then this is going to give. So, now instead of this J if I do it, so the nature will not be changing; so that is what I wanted to show it here.

So, I hope now you are clear what is kinematic control and what is inverse differential kinematics, where we can use proportional control; and where we can use proportional integral. And here also we can take it simple proportional in task space, and proportional integral these all same cases. So, I hope you are clear on the kinematic control and inverse differential kinematic of a serial manipulator.

The next class we will see what is dynamic control? and in the dynamic control extension we will see the dual loop; where the outer loop will be doing in a kinematic level and the inner loop would be in dynamic level. That is what we are going to see in upcoming lecture. So, this lecture is on kinematic control with MATLAB simulation. I hope you have enjoyed; and see you then thank you.