

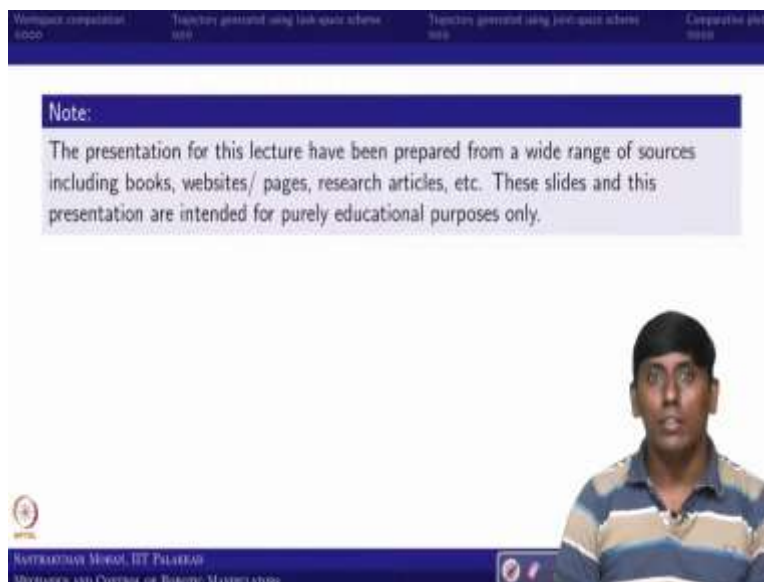
**Mechanics and Control of Robotic Manipulators**  
**Professor Santhakumar Mohan**  
**Department of Mechanical Engineering**  
**Indian Institute of Technology, Palakkad**  
**Lecture No 35**

**Trajectory generation for serial manipulators with workspace using MATLAB**

Hi, welcome back to Mechanics and Control of Robotic Manipulator. I am really thinking that you might have enjoyed the last lecture where it was showing something like very close to the real system where you would have seen some kind of animated where the manipulator is moving. Although I made it restricted to only 2R serial manipulator. Which is easy to show in you can say on a screen that is why we have made it.

However, it is not restricted only with the 2R serial or planar manipulator. It can be extended to even spatial so, you can do it that extension by your own, this particular lecture we are trying to show the workspace computation. How to make the workspace in MATLAB. So, why I have taken MATLAB because you can generate a point and the point you can make it as a cloud in a 2D plot or 3D plot. So, which will give the workspace environment then we can see how th;e you can say end effector trajectory is going one to another. So, in order to get that idea.

(Refer Slide Time: 1:15)



The image shows a screenshot of a presentation slide. At the top, there is a navigation bar with four tabs: "Workspace computation", "Trajectory generation using joint space scheme", "Trajectory generation using joint space scheme", and "Comparative plot". Below the navigation bar is a "Note:" section with the following text: "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." At the bottom of the slide, there is a video feed of Professor Santhakumar Mohan, who is wearing a blue and white striped shirt. The bottom of the slide also contains the text "PROFESSOR SANTHAKUMAR MOHAN, IIT PALAKKAD" and "MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS".

Workspace computation 0:00    Trajectory generated using task-space scheme 0:00    Trajectory generated using joint-space scheme 0:00    Comparative plots 0:00

## TRAJECTORY GENERATION FOR ROBOTIC MANIPULATORS USING MATLAB WITH WORKSPACE

- 1 Workspace computation
- 2 Trajectory generated using task-space scheme
- 3 Trajectory generated using joint-space scheme
- 4 Comparative plots



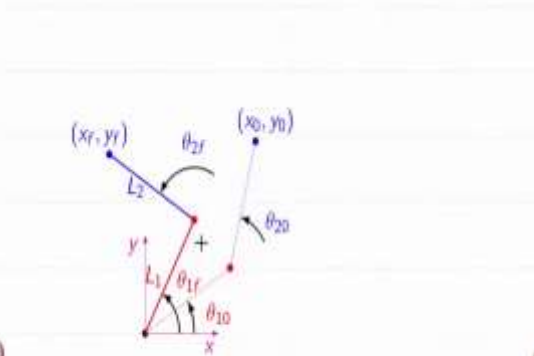


**RASTOGI ANAND MEHRA, IIT PALAKKAD**  
 MECHANICAL AND CONTROL OF ROBOTIC MANIPULATORS


So, we will take this particular lecture. So, this lecture is going to talk about workspace computation, then the trajectory generated using task space scheme for same 2R serial manipulator including workspace. Then we can see like the trajectory generated using joint space scheme. Then we will compare one to another how it is happening. So, we will take the same case and compare it how it is happening one to another.

(Refer Slide Time: 1:42)

Workspace computation 0:00    Trajectory generated using task-space scheme 0:00    Trajectory generated using joint-space scheme 0:00    Comparative plots 0:00

### Example: A planar RR serial manipulator


**RASTOGI ANAND MEHRA, IIT PALAKKAD**  
 MECHANICAL AND CONTROL OF ROBOTIC MANIPULATORS

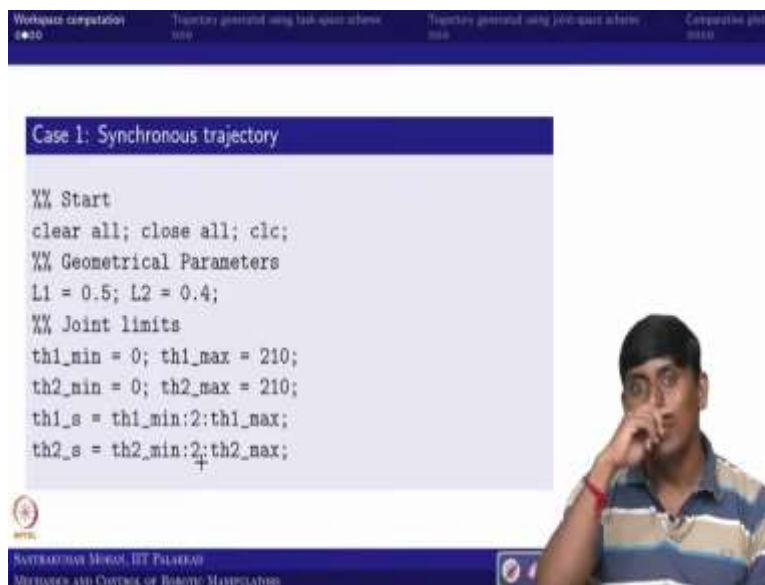
If that is the case. So, we will move to the generalize 2R serial manipulator situation where the initial position of the end effector and you can say final position of the end effector is given. In the other way if you have inverse kinematics solution. So, you know initial joint positions and

final joint positions are known provided  $l_1$  and  $l_2$  are known. So, in that case, what one can see first we will try to see.

So, what is the workspace so for that the  $\theta_1$  minimum to maximum, you should know the joint limit. Similarly, the  $\theta_2$  minimum and maximum you should know the joint limits of the  $\theta_1$  and  $\theta_2$ . So, once you know, we can a plot based on the forward kinematics, and we can generate the workspace.

So, provided you should know the forward kinematics solution. Since it is a 2R serial manipulator the forward kinematics solution is straightforward. The  $x$  can be written as  $l_1 \cos \theta_1 + l_2 \cos \theta_2$  or you can cross  $\theta_1 + \theta_2$ . So, the  $y$  is  $l_1 \sin \theta_1 + l_2 \sin \theta_2$ . So, this is we know.

(Refer Slide Time: 2:50)



```
Workspace calculation
4:30

Trajectory generation using task space scheme
1:55

Trajectory generation using joint space scheme
1:55

Trajectory plot
1:55

Case 1: Synchronous trajectory

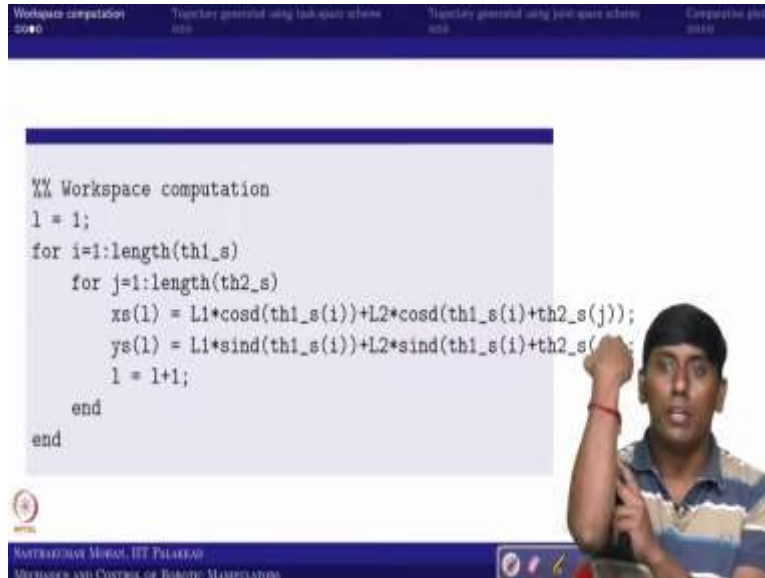
%% Start
clear all; close all; clc;
%% Geometrical Parameters
L1 = 0.5; L2 = 0.4;
%% Joint limits
th1_min = 0; th1_max = 210;
th2_min = 0; th2_max = 210;
th1_s = th1_min:2:th1_max;
th2_s = th2_min:2:th2_max;
```

So, that is what we are trying to do. So, for that we are taking it you can say the synchronous trajectory all the time. So, it needs not to be explicitly given. So, that is given here. So, now coming to the general case so, here the  $\theta_1$  minimum and maximum. So,  $\theta_2$  minimum and maximum is given. So, in the sense the  $\theta_1$  you can say simulation which is I want to show where is the workspace.

So, which start from  $\theta_1$  minimum to maximum with the interval of 2 degree because I just want to show only 2-degree interval of the points it need not to be. Further you please remember

if a here I have written as in degrees, so theta 1 and theta 2 all in degrees. So, whenever you write the equations, so you make sure that the unit as, units are matching.

(Refer Slide Time: 3:43)



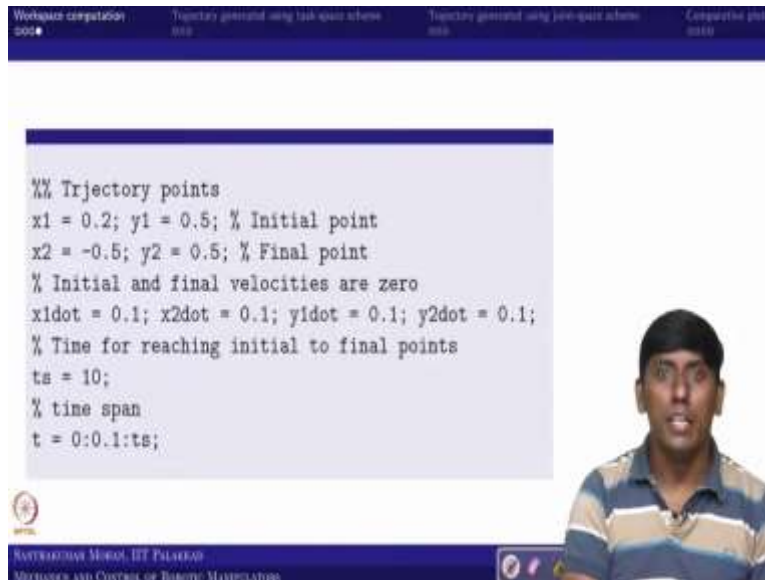
```
Workspace computation
%% Workspace computation
l = 1;
for i=1:length(th1_s)
    for j=1:length(th2_s)
        xs(l) = L1*cosd(th1_s(i))+L2*cosd(th1_s(i)+th2_s(j));
        ys(l) = L1*sind(th1_s(i))+L2*sind(th1_s(i)+th2_s(j));
        l = l+1;
    end
end
end
```

SRINIVASAN MURUGU, IIT PALAKKAD  
Mechatronics and Control of Robotic Manipulators

So, in that sense so, you can see the workspace I am trying to compute based on the forward kinematics solution. But what I am trying to do? So, I am trying to generate the x and y for the given range or the given range the theta 1 vary from 0 to or you can say minimum to maximum theta 2 also vary from minimum to maximum. So, that minimum to maximum I have already defined theta 1 s theta 2 s.

So, I am trying to run the loop where the theta 1 is running the first loop from minimum to maximum the second loop theta 2 which is running minimum to maximum. In the sense first I will keep the theta 1. So, you can look at it here. So, the theta 1 is coming here and the second is rotating from minimum to maximum once then you move another and again rotate from minimum to maximum like that you can do it in sequence. So, once this is done.

(Refer Slide Time: 4:42)



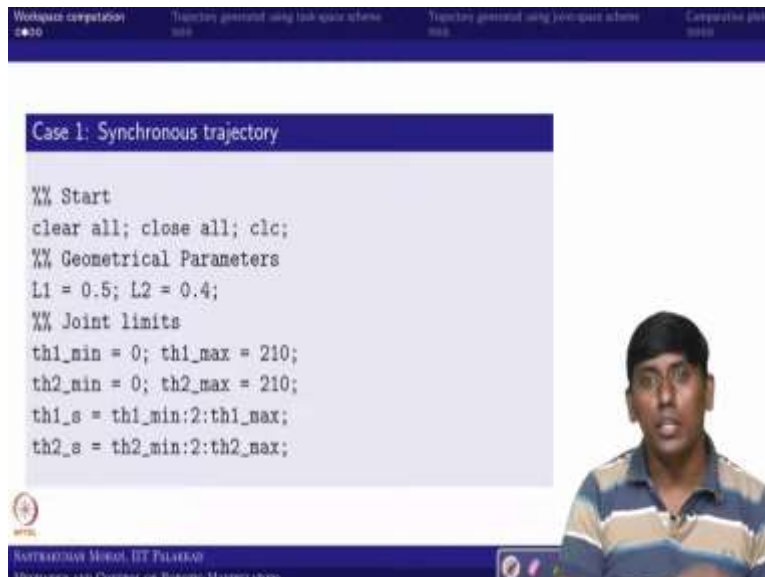
The screenshot shows a MATLAB workspace with a script editor containing the following code:

```
%% Trajectory points
x1 = 0.2; y1 = 0.5; % Initial point
x2 = -0.5; y2 = 0.5; % Final point
% Initial and final velocities are zero
xidot = 0.1; x2dot = 0.1; yidot = 0.1; y2dot = 0.1;
% Time for reaching initial to final points
ts = 10;
% time span
t = 0:0.1:ts;
```

The interface includes a top menu bar with options like 'Workspace', 'Trajectory generated using task space scheme', and 'Comparative plot'. A video feed of a presenter is visible on the right side of the screen.

So, what do you want to you want to plot. So, for plotting, we will come back in the later but right now I am trying to generate the trajectory. So, for generating trajectory what I know. So, I know already that trajectory points which are given to us. So, I am taking the same but here l 1 and l 2 I have taken a slightly different value.

(Refer Slide Time: 5:06)



The screenshot shows a MATLAB workspace with a script editor containing the following code:

```
Case 1: Synchronous trajectory

%% Start
clear all; close all; clc;
%% Geometrical Parameters
L1 = 0.5; L2 = 0.4;
%% Joint limits
th1_min = 0; th1_max = 210;
th2_min = 0; th2_max = 210;
th1_s = th1_min:2:th1_max;
th2_s = th2_min:2:th2_max;
```

The interface includes a top menu bar with options like 'Workspace', 'Trajectory generated using task space scheme', and 'Comparative plot'. A video feed of a presenter is visible on the right side of the screen.

```

%% Workspace computation
l = 1;
for i=1:length(th1_s)
    for j=1:length(th2_s)
        xs(l) = L1*cosd(th1_s(i))+L2*cosd(th1_s(i)+th2_s(j));
        ys(l) = L1*sind(th1_s(i))+L2*sind(th1_s(i)+th2_s(j));
        l = l+1;
    end
end
end

```

RAJESH KUMAR NIGAM, IIT PALAKA  
MECHATRONICS AND CONTROL OF ROBOTIC MANIPULATORS

So, you can see it is 0.5 and 0.4. So, I did not take it is equal so that I can see the workspace even if it rotates 0 to 360 it would be donut. So, that donut meaning you can say internal diameter is 0.1 meter and the external diameter is 0.9 meter. So, I just want to show but here it is not going to be a donut it would be a small patch. So, we can see that so before going to see that we can take the trajectory variable.

(Refer Slide Time: 5:37)

```

figure
for i=1:length(t)
    % Task-space trajectory
    x(i) = [1,t(i),t(i)^2,t(i)^3]*ax;
    y(i) = [1,t(i),t(i)^2,t(i)^3]*ay;
    % Inverse kinematic model (for calculating joint-space variables)
    c2 = (x(i)^2+y(i)^2-L1^2-L2^2)/(2*L1*L2); s2 = sqrt(1-c2^2);
    th1(i) = atan2(y(i),x(i))-atan2(L2*s2,L1+L2*c2);
    th2(i) = atan2(s2,c2);

    % workspace
    plot(xs,ys, '.', 'Color',[0.9 0.9 0.9])
    hold on, axis([-1 1 -1 1]); grid on, axis square;
end

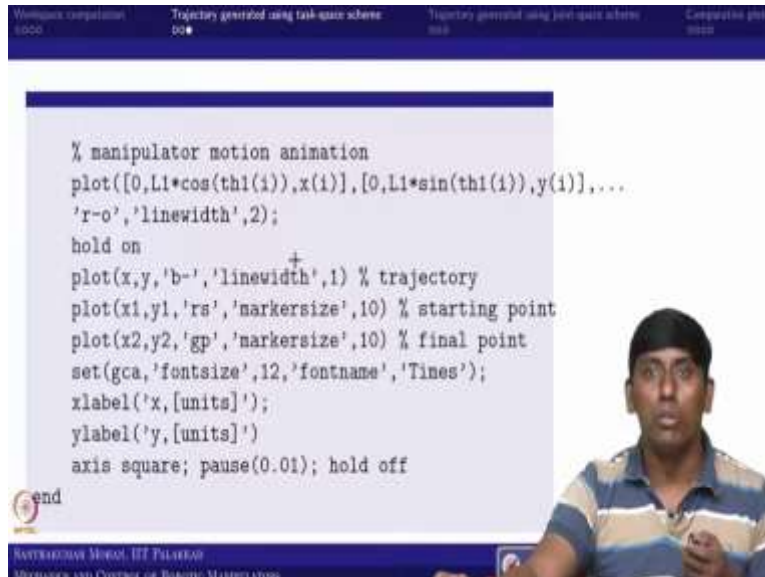
```

RAJESH KUMAR NIGAM, IIT PALAKA  
MECHATRONICS AND CONTROL OF ROBOTIC MANIPULATORS

So, we will just take the trajectory generated in the task space. So, we will take the  $a_x$  and  $a_y$  as given and  $x$  and  $y$  can be calculated. So here we are doing it independent trajectory generation. So, in that sense, the trajectory is generated  $x$  of  $i$  and  $y$  of  $i$  like this and

the inverse kinematic model we have taken so then we will go to the workspace. So here you can see the workspace I have plotted as a simple point across I hold on so that when I plot this all plot it is not that particular point.

(Refer Slide Time: 6:12)



```
% manipulator motion animation
plot([0,L1*cos(th1(i)),x(i)],[0,L1*sin(th1(i)),y(i)],...
'r-o','linewidth',2);
hold on
plot(x,y,'b-','linewidth',1) % trajectory
plot(x1,y1,'rs','markersize',10) % starting point
plot(x2,y2,'gp','markersize',10) % final point
set(gca,'fontsize',12,'fontname','Times');
xlabel('x,[units]');
ylabel('y,[units]')
axis square; pause(0.01); hold off
end
```

RAJESH K. MURUGAN, IIT PALAKKAD  
ANALYSIS AND CONTROL OF ROBOTS: MANIPULATORS

So, then I am ending the manipulator motion animation. So, I brought this you can say  $0 \times 1 \times 2$   $0 \times y_1 \times y_2$  and that I am making it as a line so that it would look like animation. So here it is 10 milliseconds as the delay and again the same thing. So, starting and ending point we, plotted and final trajectory also like we can generate. So, just to show that. So, we will first finish the entire thing then we will go to the MATLAB.



(Refer Slide Time: 6:45)

```
Workspace: workspace1  Trajectory generated using task-space scheme  Trajectory generated using joint-space scheme  Comparative plot
0000  000  000  0000

% saving the data for comparison
xt = x; yt = y; th1t = th1; th2t = th2;
clear x y;

% Initial and final joint-space coordinates
th10 = th1(1); th1f = th1(i); th20 = th2(1); th2f = th2(i);
J1 = [-L1*sin(th10)-L2*sin(th10+th20), -L2*sin(th10+th20);
      +L1*cos(th10)+L2*cos(th10+th20), +L2*sin(th10+th20)];
J2 = [-L1*sin(th1f)-L2*sin(th1f+th2f), -L2*sin(th1f+th2f);
      +L1*cos(th1f)+L2*cos(th1f+th2f), +L2*sin(th1f+th2f)];
th0dot = inv(J1)*[x1dot;y1dot]; thfdot = inv(J2)*[x2dot;y2dot];
```

So, the second case, so, we are trying to save this data the second case is we are taking the inverse kinematics. So then inverse differential kinematics where the theta 0 dot and theta final dot of individual cases are found.

(Refer Slide Time: 7:00)

```
Workspace: workspace1  Trajectory generated using task-space scheme  Trajectory generated using joint-space scheme  Comparative plot
0000  000  000  0000

figure
for i=1:length(t)
    % Joint-space trajectory
    th1(i) = [1,t(i)+t(i)^2,t(i)^3]*ath1;
    th2(i) = [1,t(i),t(i)^2,t(i)^3]*ath2;
    % Forward kinematic model (for calculating the task-space variables)
    x(i) = L1*cos(th1(i)) + L2 *cos(th1(i)+th2(i));
    y(i) = L1*sin(th1(i)) + L2 *sin(th1(i)+th2(i));
```



```

% saving the data for comparison
xt = x; yt = y; th1t = th1; th2t = th2;
clear x y;

% Initial and final joint-space coordinates
th10 = th1(1); th1f = th1(10); th20 = th2(1); th2f = th2(10);
J1 = [-L1*sin(th10)-L2*sin(th10+th20), -L2*sin(th10+th20);
      +L1*cos(th10)+L2*cos(th10+th20), +L2*sin(th10+th20)];
J2 = [-L1*sin(th1f)-L2*sin(th1f+th2f), -L2*sin(th1f+th2f);
      +L1*cos(th1f)+L2*cos(th1f+th2f), +L2*sin(th1f+th2f)];
th0dot = inv(J1)*[xidot;yidot]; thfdot = inv(J2)*[x2dot;y2dot];

```

```

figure
for i=1:length(t)
    % Joint-space trajectory
    th1(i) = [1,t(i),t(i)^2,t(i)^3]*ath1;
    th2(i) = [1,t(i),t(i)^2,t(i)^3]*ath2;
    % Forward kinematic model (for calculating the task-space variables)
    x(i) = L1*cos(th1(i)) + L2 *cos(th1(i)+th2(i));
    y(i) = L1*sin(th1(i)) + L2 *sin(th1(i)+th2(i));

    % workspace
    plot(xs,ys, '.', 'Color',[0.9 0.9 0.9])
    hold on, axis([-1 1 -1 1]);
    grid on, axis square

```

So, once these found we can generate the trajectory based on so and so thing. So, once these all found so you can see like here. So, once those things are found so we can calculate these all. So once these all calculated, so we can calculate the forward kinematic model just for comparison. So, then we can again plot the workspace.

(Refer Slide Time: 7:25)



```
% saving the data for comparison
xj = x; yj = y; th1j = th1; th2j = th2;

figure
subplot(2,2,1)
plot(xs,ys,'.','Color',[0.9 0.9 0.9])
hold on, axis([-1 1 -1 1]);, grid on, axis square
plot(xt,yt,'b-','linewidth',1)
plot(xt(1),yt(1),'rs','markersize',10)
plot(xt(i),yt(i),'gp','markersize',10)
set(gca,'fontsize',12,'fontname','Times');
xlabel('x,[units]'); ylabel('y,[units]');
```

And the plot the individual animation we can try to show finally we will end with a comparative plot where you can say both trajectories how it looked like one is joint space, the other one is task space. How it looked like that we are trying to compare.

(Refer Slide Time: 7:39)




```
subplot(2,2,2)
fill([th1_min th1_max th1_max th1_min th1_min]*pi/180,...
[th2_min th2_min th2_max th2_max th2_min]*pi/180,[0.9 0.9 0.9])
hold on, grid on
axis([th1_min*pi/180-pi/3 th1_max*pi/180+pi/3 ...
th2_min*pi/180-pi/3 th2_max*pi/180+pi/3]);
axis square; plot(th1t,th2t)
plot(th1t(1),th2t(1),'rs','markersize',10)
plot(th1t(i),th2t(i),'gp','markersize',10)
set(gca,'fontsize',12,'fontname','Times');
xlabel('\theta_1,[rad]'); ylabel('\theta_2,[rad]')
```

```

subplot(2,2,4)
fill([th1_min th1_max th1_max th1_min th1_min]*pi/180,...
[th2_min th2_min th2_max th2_max th2_min]*pi/180,[0.9 0.9 0.9])
hold on, grid on
axis([th1_min*pi/180-pi/3 th1_max*pi/180+pi/3 ...
th2_min*pi/180-pi/3 th2_max*pi/180+pi/3]);
axis square, plot(th1j,th2j)
plot(th1j(i),th2j(i),'rs','markersize',10)
plot(th1j(i),th2j(i),'gp','markersize',10)
set(gca,'fontsize',12,'fontname','Times');
xlabel('\theta_1,[rad]'); ylabel('\theta_2,[rad]');

```




So, for that we are trying to plot individual cases. So, and then we are ending it. So, for getting more clarity as it is a short video.

(Refer Slide Time: 7:52)


```

83- axis square
84- pause(0.01)
85- hold off
86- end
87-
88- % saving the data for comparison
89- xt = x; yt = y; th1t = th1; th2t = th2;
90-
91- clear x y;
92-
93- %% Case 2: Trajectory generated based on joint space sche
94-
95- % Initial and final joint-space coordin
96-


```



```
4
5 clear all; close all; clc;
6
7 %% Geometrical Parameters
8
9 L1 = 0.5; L2 = 0.4;
10
11 %% Joint limits
12 th1_min = 0; th1_max = 210;
13 th2_min = 0; th2_max = 210;
14 th1_s = th1_min:2:th1_max;
15 th2_s = th2_min:2:th2_max;
16
17 %% Workspace computation
```



```
7 %% Geometrical Parameters
8
9 L1 = 0.5; L2 = 0.4;
10
11 %% Joint limits
12 th1_min = 0; th1_max = 210;
13 th2_min = 0; th2_max = 210;
14 th1_s = th1_min:2:th1_max;
15 th2_s = th2_min:2:th2_max;
16
17 %% Workspace computation
18
19 l = 1;
```



```
10
11 %% Joint limits
12 th1_min = 0; th1_max = 210;
13 th2_min = 0; th2_max = 210;
14 th1_s = th1_min:2:th1_max;
15 th2_s = th2_min:2:th2_max;
16
17 %% Workspace computation
18
19 l = 1;
20
21 for i=1:length(th1_s)
22     for j=1:length(th2_s)
23         xs(l) = L1*cosd(th1_s(i))+L2*cosd(th1_s(i)+th2_s(j));
24         ys(l) = L1*sind(th1_s(i))+L2*sind(th1_s(i)+th2_s(j));
25         l = l+1;
26     end
27 end
```



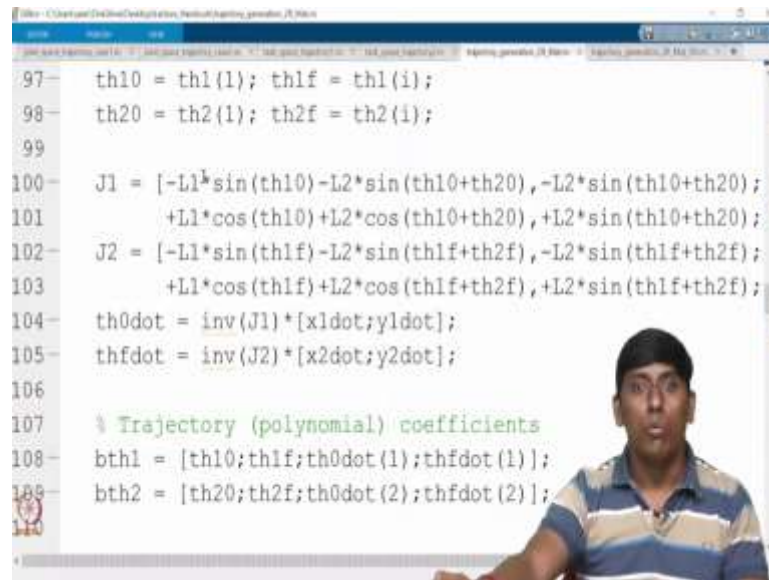
```
16
17 %% Workspace computation
18
19 l = 1;
20
21 for i=1:length(th1_s)
22     for j=1:length(th2_s)
23         xs(l) = L1*cosd(th1_s(i))+L2*cosd(th1_s(i)+th2_s(j));
24         ys(l) = L1*sind(th1_s(i))+L2*sind(th1_s(i)+th2_s(j));
25         l = l+1;
26     end
27 end
28 %% Trajectory points
```



```
43     1,ts,ts^2,ts^3;
44     0,1,0,0;
45     0,1,2*ts,3*ts^2];
46     %% Case 1: Trajectory generated based on task-space scheme
47     bx = [x1;x2;x1dot;x2dot];
48     by = [y1;y2;y1dot;y2dot];
49     ax = inv(A)*bx;
50     ay = inv(A)*by;
51
52     figure      I
53
54     for i=1:length(t)
55
56         % Task-space trajectory
```

```
64     th1(i) = atan2(y(i),x(i))-atan2(L2*s2,L1+L2*c2);
65     th2(i) = atan2(s2,c2);
66
67     % workspace
68     plot(xs,ys,'.','Color',[0.9 0.9 0.9])
69     hold on
70     axis([-1 1 -1 1]);
71     grid on
72     axis square
73     % manipulator motion animation
74     plot([0,L1*cos(th1(i)),x(i)], [0,L1*sin(th1(i)),y(i)],
75         'r-o','linewidth',2)
76     hold on
77     plot(x,y,'b-','linewidth',1) % trajectory
```





```
97- th10 = th1(1); th1f = th1(i);
98- th20 = th2(1); th2f = th2(i);
99
100- J1 = [-L1*sin(th10)-L2*sin(th10+th20), -L2*sin(th10+th20);
101-       +L1*cos(th10)+L2*cos(th10+th20), +L2*sin(th10+th20)];
102- J2 = [-L1*sin(th1f)-L2*sin(th1f+th2f), -L2*sin(th1f+th2f);
103-       +L1*cos(th1f)+L2*cos(th1f+th2f), +L2*sin(th1f+th2f)];
104- th0dot = inv(J1)*[x1dot;y1dot];
105- thfdot = inv(J2)*[x2dot;y2dot];
106
107- % Trajectory (polynomial) coefficients
108- bth1 = [th10;th1f;th0dot(1);thfdot(1)];
109- bth2 = [th20;th2f;th0dot(2);thfdot(2)];
```

So, for getting more clarity we will go to MATLAB. If anything will explain, then there. So, you can see like here, this is the starting point. So, we are as general we are clearing the; you can say a workspace closing all the figure windows and clearing the command history. And the geometry parameter here link 1 is the length of 0.5 meter and link 2 is 0.4 meter. And the joint limits 0 to 2 and 210 degrees.

And this is the simulation, and I am computing the MATLAB using for MATLAB for finding the workspace the workspace I have computed then the trajectory point details are given. So, I have calculated all those things. So, then I calculated. So, you can see this. And again, I have calculated the; what you call the bth and this one. Then I calculated the individual joint trajectory coefficients and the joint space trajectory I have generated it is synchronous.

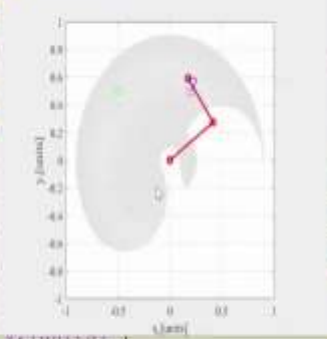


(Refer Slide Time: 8:55)

```
118- th1(i) = [1,t(i),t(i)^2,t(i)^3]*ath1;
119- th2(i) = [1,t(i),t(i)^2,t(i)^3]*ath2;
120- % Forward kinematic model (for calculating the
121- %task-space variables)
122- x(i) = L1*cos(th1(i)) + L2 *cos(th1(i)+th2(i));
123- y(i) = L1*sin(th1(i)) + L2 *sin(th1(i)+th2(i));
124-
125- % workspace
126- plot(xs,ys,'.','Color',[0.9 0.9 0.9])
127- hold on
128- axis([-1 1 -1 1]);
129- grid on
130- axis square
131- % manipulator motion animation

130- axis square
131- % manipulator motion animation
132- plot([0,L1*cos(th1(i)),x(i)],[0,L1*sin(th1(i)),y(i)],
133-      'r-o','linewidth',2)
134- hold on
135- plot(x,y,'b-','linewidth',1) % trajectory
136- plot(x1,y1,'rs','markersize',10) % starting point
137- plot(x2,y2,'gp','markersize',10) % final point
138- set(gca,'fontsize',12,'fontname','Times')
139- xlabel('x,[units]');
140- ylabel('y,[units]')
141- axis square
142- pause(0.01)
143- hold off
```

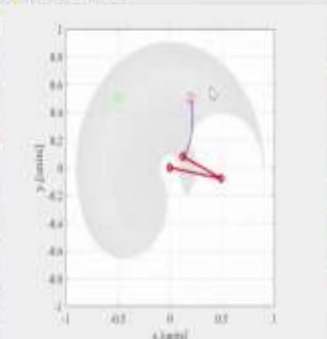
```
130 axis square
131 % manip
132 plot([0
133 'r-
134 hold on
135 plot(x,
136 plot(x1
137 plot(x2
138 set(gca
139 xlabel(
140 ylabel(
141 axis square
142 pause(0.01)
143 hold off
```



The plot shows a shaded gray region in the x-y plane. A red line segment connects two points, with a green dot at the starting point. The axes are labeled 'x [mm]' and 'y [mm]'.

in(th1(i)),y(i)],  
ctory  
starting point  
inal point  
mes')

```
130 axis sq
131 % manip
132 plot([0
133 'r-
134 hold on
135 plot(x,
136 plot(x1
137 plot(x2
138 set(gca
139 xlabel(
140 ylabel(
141 axis square
142 pause(0.01)
143 hold off
```



The plot shows a shaded gray region in the x-y plane. A red line segment connects two points, with a green dot at the starting point. The axes are labeled 'x [mm]' and 'y [mm]'.

in(th1(i)),y(i)],  
ctory  
starting point  
inal point  
mes')

```

130 axis square
131 % manip
132 plot([0
133 'r-
134 hold on
135 plot(x,
136 plot(x1
137 plot(x2
138 set(gca
139 xlabel(
140 ylabel(
141 axis square
142 pause(0.01)
143 hold off

```

And the other one is independent this is a synchronous and then I have tried to plot it. First, I will run this then we will go back wherever there is a clarification required. So first I run this. So, you can see like this is the workspace and based on the given condition, you can see like the task space trajectory and joint space trajectory is giving some kind of peculiar patch. So that is what we are trying to see. So, you can look at it this.

(Refer Slide Time: 9:30)

```

88 % saving the data for comparison
89 xt = x; yt = y; th1t = th1; th2t = th2;
90
91 clear x y;
92
93 %% Case 2: Trajectory generated based on joint-space scheme
94
95 % Initial and final joint-space coordinates
96
97 th10 = th1(1); th1f = th1(i);
98 th20 = th2(1); th2f = th2(i);
99
100 J1 = [-L1*sin(th10)-L2*sin(th10+th20), -L2*cos(th10+th20);
101       +L1*cos(th10)+L2*cos(th10+th20), -L2*sin(th10+th20)];

```

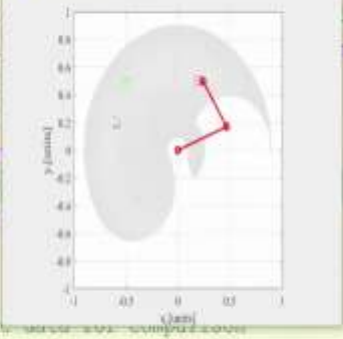
```
34 % Initial and final velocities are zero
35 xldot = 0.1; x2dot = 0.1; yldot = 0; y2dot = 0;
36
37 % Time for reaching initial to final points
38 ts = 10;
39 % time span
40 t = 0:0.1:ts;
41 %% Trajectory generation using cubic polynomial
42 A = [1,0,0,0;
43      1,ts,ts^2,ts^3;
44      0,1,0,0;
45      0,1,2*ts,3*ts^2];
46 %% Case 1: Trajectory generated based on
47 bx = [x1;x2;xldot;x2dot];
```

```
136 plot(x1,y1,'rs','markersize',10) % starting point
137 plot(x2,y2,'gp','markersize',10) % final point
138 set(gca,'fontsize',12,'fontname','Times');
139 xlabel('x,[units]');
140 ylabel('y,[units]')
141 axis square
142 pause(0.1)
143 hold off
144 end
145
146 % saving the data for comparison
147 xj = x; yj = y; th1j = th1; th2j = th2;
148
149 % Comparison plots
```

```

136 plot(x1)
137 plot(x2)
138 set(gca)
139 xlabel('x [mm]')
140 ylabel('y [mm]')
141 axis square
142 pause(0.5)
143 hold on
144 end
145
146 % saving the current state
147 xj = x; yj = y; th1j = th1; th2j = th2;
148
149 % Comparison plots

```



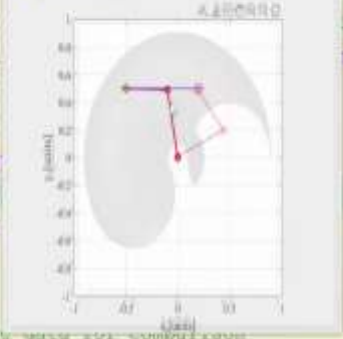
The plot shows a shaded gray region in the x-y plane. A red line segment connects two points within the region. A green dot is located at the top-left corner of the region. The axes are labeled 'x [mm]' and 'y [mm]' and are square.

Starting point  
Final point  
names');

```

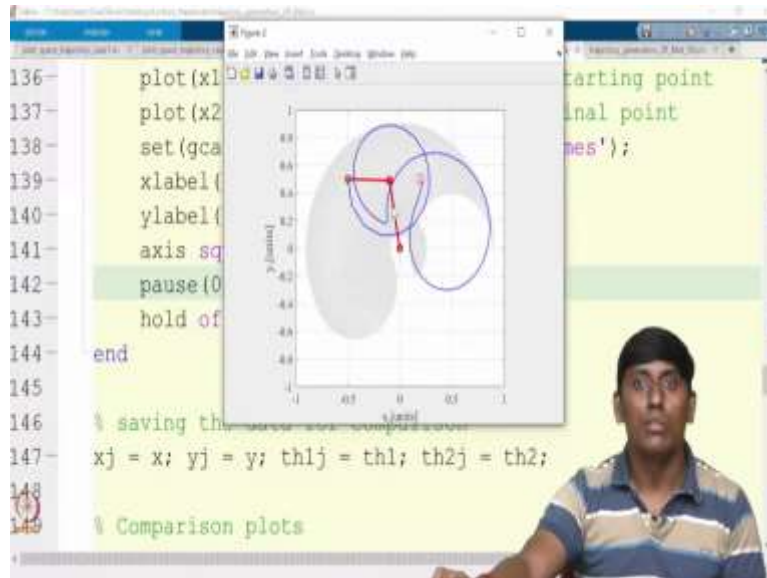
136 plot(x1)
137 plot(x2)
138 set(gca)
139 xlabel('x [mm]')
140 ylabel('y [mm]')
141 axis square
142 pause(0.5)
143 hold on
144 end
145
146 % saving the current state
147 xj = x; yj = y; th1j = th1; th2j = th2;
148
149 % Comparison plots

```



The plot is identical to the one in the first image, showing a shaded gray region with a red line segment and a green dot. The axes are labeled 'x [mm]' and 'y [mm]' and are square.

Starting point  
Final point  
names');



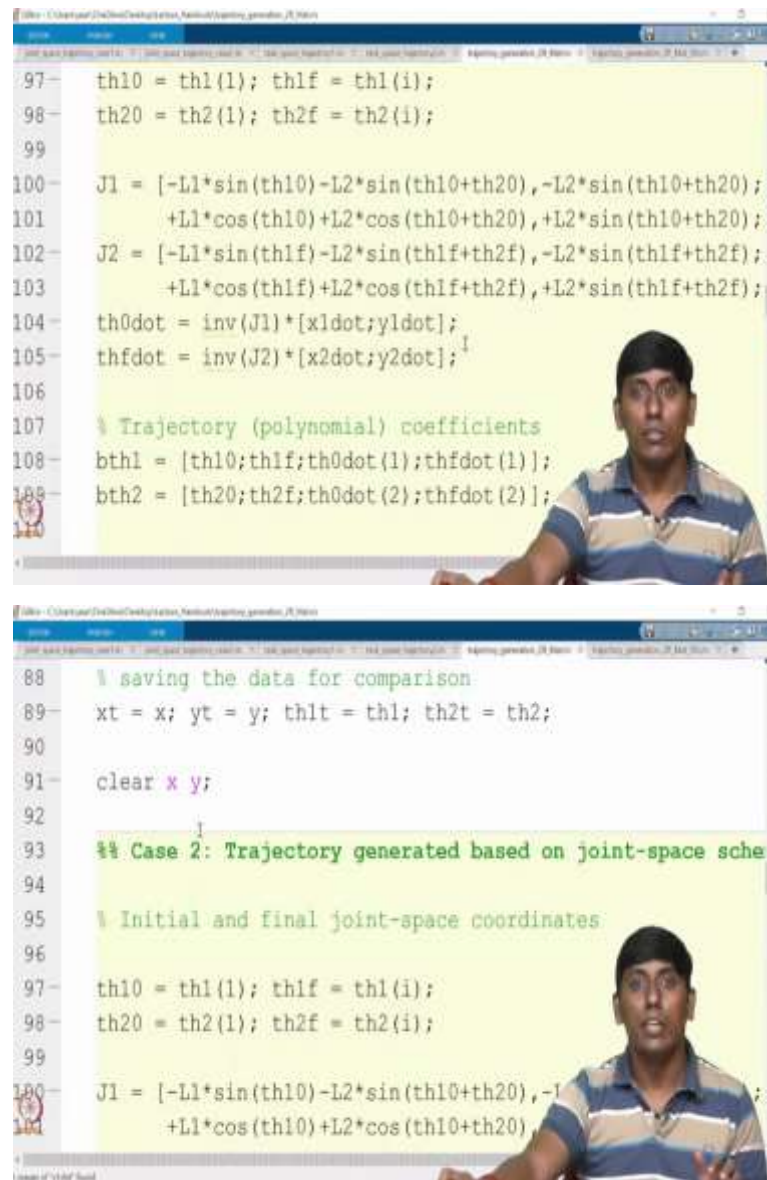
So first I will try to show this is the joint space trajectory. This is the task space trajectory. So, tasks space trajectory is there are based on this, the initial velocity and final velocity is making it a little unrealizable. So, I will just make it both 0. And in order to make it stand by just to give idea. So, I am making it 100 milliseconds for the step for one image to another image. So, now, I am trying to show here.

So, you can see this is the workspace the patch part which is the workspace. So, the first joint is rotate 0 to 210 degree second joint also rotate maximum 210 degree starting from 0. So, if I plot then this is a workspace will come within the workspace this is the initial point and this is the final point. So, now, it is from nonzero initial velocity to you can see finite velocity. So, you can see how it is so, complex.

Because, we have taken the inverse Jacobian, which is one such complex so, that is what it is trying to do it based on so-on-so case. So, here there is a peculiar thing we need to see it.



(Refer Slide Time: 10:51)



The image displays two screenshots of a MATLAB script, with a video overlay of a presenter in the bottom right corner of each. The script is divided into two sections by a comment line.

**Top Screenshot (Lines 97-108):**

```
97- th10 = th1(1); th1f = th1(i);
98- th20 = th2(1); th2f = th2(i);
99-
100- J1 = [-L1*sin(th10)-L2*sin(th10+th20), -L2*sin(th10+th20);
101-       +L1*cos(th10)+L2*cos(th10+th20), +L2*sin(th10+th20);
102- J2 = [-L1*sin(th1f)-L2*sin(th1f+th2f), -L2*sin(th1f+th2f);
103-       +L1*cos(th1f)+L2*cos(th1f+th2f), +L2*sin(th1f+th2f);
104- th0dot = inv(J1)*[x1dot;y1dot];
105- thfdot = inv(J2)*[x2dot;y2dot];1
106-
107- % Trajectory (polynomial) coefficients
108- bth1 = [th10;th1f;th0dot(1);thfdot(1)];
109- bth2 = [th20;th2f;th0dot(2);thfdot(2)];
```

**Bottom Screenshot (Lines 88-104):**

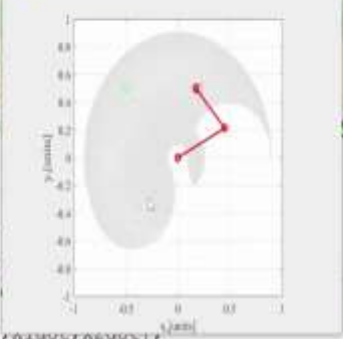
```
88 % saving the data for comparison
89 xt = x; yt = y; th1t = th1; th2t = th2;
90
91 clear x y;
92
93 %% Case 2: Trajectory generated based on joint-space sche
94
95 % Initial and final joint-space coordinates
96
97 th10 = th1(1); th1f = th1(i);
98 th20 = th2(1); th2f = th2(i);
99-
100- J1 = [-L1*sin(th10)-L2*sin(th10+th20), -L2*sin(th10+th20);
101-       +L1*cos(th10)+L2*cos(th10+th20), +L2*sin(th10+th20);
```

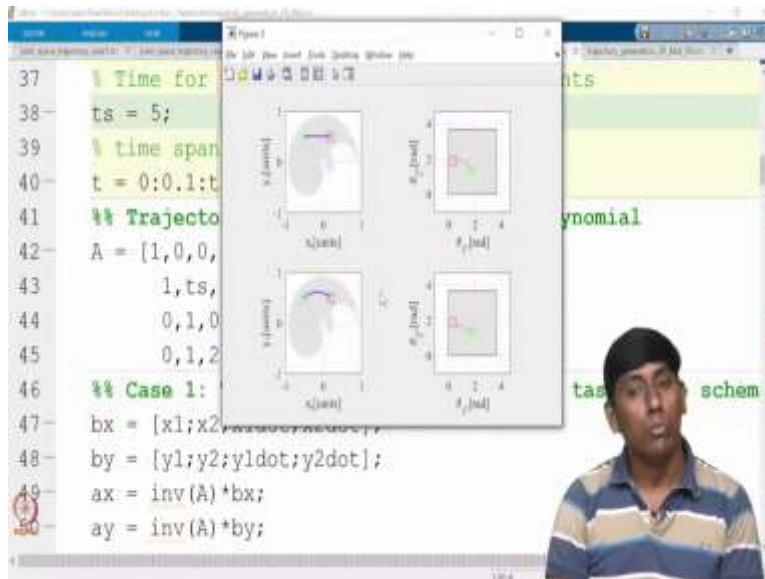
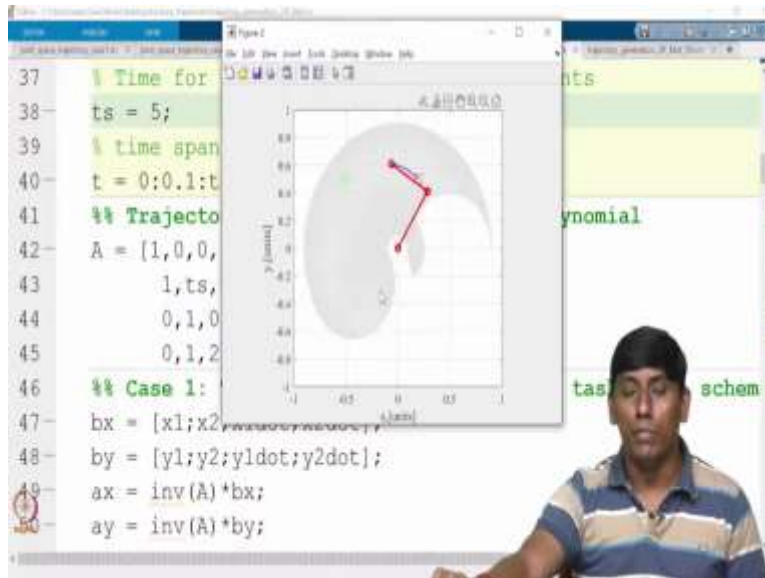


```
39 % time span
40 t = 0:0.1:ts;
41 %% Trajectory generation using cubic polynomial
42 A = [1,0,0,0;
43      1,ts,ts^2,ts^3;
44      0,1,0,0;
45      0,1,2*ts,3*ts^2];
46 %% Case 1: Trajectory generated based on task scheme
47 bx = [x1;x2;x1dot;x2dot];
48 by = [y1;y2;y1dot;y2dot];
49 ax = inv(A)*bx;
50 ay = inv(A)*by;
```



```
37 % Time for
38 ts = 5;
39 % time span
40 t = 0:0.1:t;
41 %% Trajectory
42 A = [1,0,0,
43      1,ts,
44      0,1,0,
45      0,1,2
46 %% Case 1:
47 bx = [x1;x2;x1dot;x2dot];
48 by = [y1;y2;y1dot;y2dot];
49 ax = inv(A)*bx;
50 ay = inv(A)*by;
```





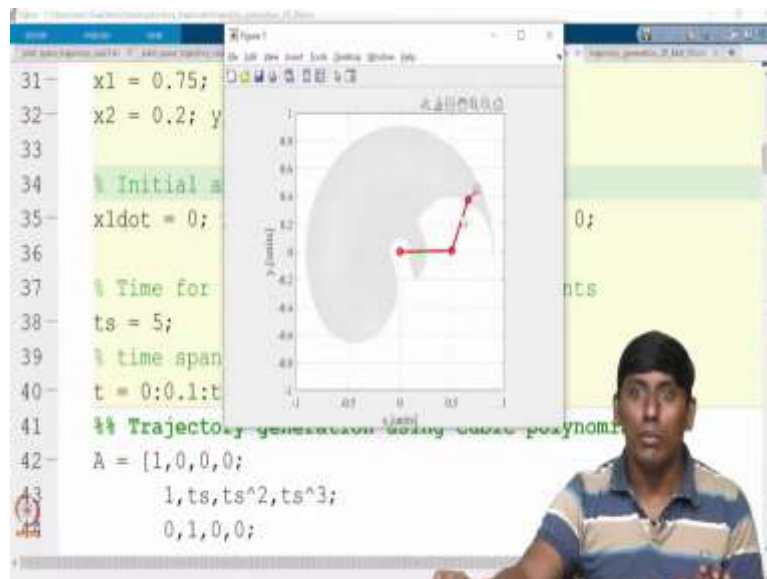
So, I like to like to try to show so, what happened here. So, here we are trying to calculate so and so this  $x_1$  dot and  $x_2$  dot so since we are clearing it there so, we are clearing everything clear  $x$   $y$ . So, these initial conditions, which we have calculated, you can see like these are still exist, so I make it initial and final velocity of both  $x$  and  $y$  are 0. And I just tried to show with a very simple 5 second. So, I just want to show it here. So, now I just you can see it.

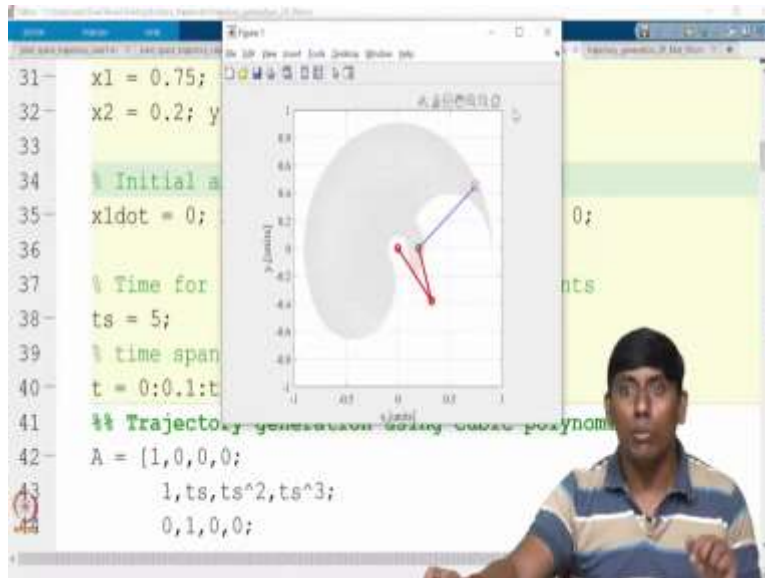
So, it takes this point to this point it is taking 5 second and it is running. So, this make it if initial and final velocities are 0 so you are getting much, much smoother profile. So now we can realistic. So, for example, I am taking one point here, so another point is somewhere here. So,

what will happen in the task space trajectory it will come here. So, this point, I am taking it 0.15 in the x and probably 0 in y I just take it.

(Refer Slide Time: 12:14)

```
31- x1 = 0.2; y1 = 0.5; % Initial point
32- x2 = 0.2; y2 = 0; % Final point
33-
34- % Initial and final velocities are zero
35- xldot = 0; x2dot = 0; yldot = 0; y2dot = 0;
36-
37- % Time for reaching initial to final points
38- ts = 5;
39- % time span
40- t = 0:0.1:ts;
41- %% Trajectory generation using cubic polynomials
42- A = [1,0,0,0;
43-      1,ts,ts^2,ts^3;
44-      0,1,0,0;
```





So, I will take the initial point is the same. So, the final point is 0.2 this is 0. I just so I just try to see the plot. So here I will be taking the initial point is something around probably, I will take it here. So, which is 0.75 and 0.45 I just randomly take 0.75 and 0.45 just to show that. So, the workspace since we have not restricted the joint here. So, the workspace will show, but the manipulator will show the animation says that it is going out of the workspace.

So, you can see like so you can see like it is going to go outside the workspace in the task space trajectory. This is what we have seen one of the constraint, which we were discussing one of the lecture. So, you can see it here. So, this is one of the solution we have taken. That is why it is coming this way. So now the solution, I am taking it the other solution.

(Refer Slide Time: 13:32)

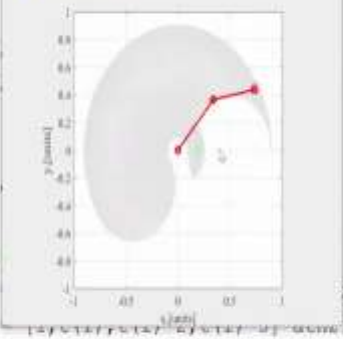
```
64- th1(i) = atan2(y(i),x(i))-atan2(L2*s2,L1+L2*c2);
65- th2(i) = atan2(s2,c2);
66-
67- % workspace
68- plot(xs,ys, '.', 'Color',[0.9 0.9 0.9])
69- hold on
70- axis([-1 1 -1 1]);
71- grid on
72- axis square
73- % manipulator motion animation
74- plot([0,L1*cos(th1(i)),x(i)], [0,L1*sin(th1(i)),y(i)],
75-      'r-o', 'linewidth',2)
76- hold on
77- plot(x,y, 'b-', 'linewidth',1) % trajectory
```

```
111- ath1 = inv(A)*bth1;
112- ath2 = inv(A)*bth2;
113-
114- figure
115- for i=1:length(t)
116-
117-     % Joint-space trajectory
118-     th1(i) = [1,t(i),t(i)^2,t(i)^3]*ath1;
119-     th2(i) = [1,t(i),t(i)^2,t(i)^3]*ath2;
120-     % Forward kinematic model (for calculating
121-     % task-space variables)
122-     x(i) = L1*cos(th1(i)) + L2*cos(th2(i));
```

```

109-  bth2 = [th2
110-
111-  ath1 = inv(
112-  ath2 = inv(
113-
114-  figure
115-  for i=1:len
116-
117-  % Joint
118-  th1(i)
119-  th2(i)
120-  % Forward kinematic model (for calculati
121-  %task-space variables)
122-  x(i) = L1*cos(th1(i)) + L2 *cos(th

```

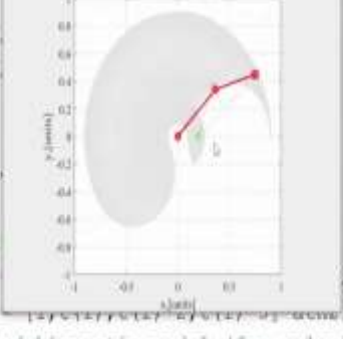


The figure window displays a 2-link planar robot arm in a 2D coordinate system. The x-axis is labeled 'x[meters]' and the y-axis is labeled 'y[meters]'. The origin is marked with '0'. The robot arm consists of two links: a base link of length L1 and a second link of length L2. The end effector is shown as a red dot. The plot shows the robot arm in a specific configuration, with the second link extending from the end of the first link.

```

109-  bth2 = [th2
110-
111-  ath1 = inv(
112-  ath2 = inv(
113-
114-  figure
115-  for i=1:len
116-
117-  % Joint
118-  th1(i)
119-  th2(i)
120-  % Forward kinematic model (for calculati
121-  %task-space variables)
122-  x(i) = L1*cos(th1(i)) + L2 *cos(th

```



The figure window displays a 2-link planar robot arm in a 2D coordinate system. The x-axis is labeled 'x[meters]' and the y-axis is labeled 'y[meters]'. The origin is marked with '0'. The robot arm consists of two links: a base link of length L1 and a second link of length L2. The end effector is shown as a red dot. The plot shows the robot arm in a specific configuration, with the second link extending from the end of the first link.



```

109-  bth2 = [th2
110
111-  ath1 = inv(
112-  ath2 = inv(
113
114-  figure
115-  for i=1:len
116
117-  % Joint
118-  th1(i)
119-  th2(i)
120
121-  % Forward kinematic model (for calculati
122-  %task-space variables)
123-  x(i) = L1*cos(th1(i)) + L2 *cos(th

```

We can see that other solution. So, what is the other solution? So, I am taking it this is  $s_2$  is minus sign. So, in that case so you can find. So, this would be still outside workspace. at least one solution is possible. This is also like going out. So probably I have taken some point from here to here. that would be clearly visible because both are outside the workspace even the input is outside the workspace in this case. So that is why it is not doing it.

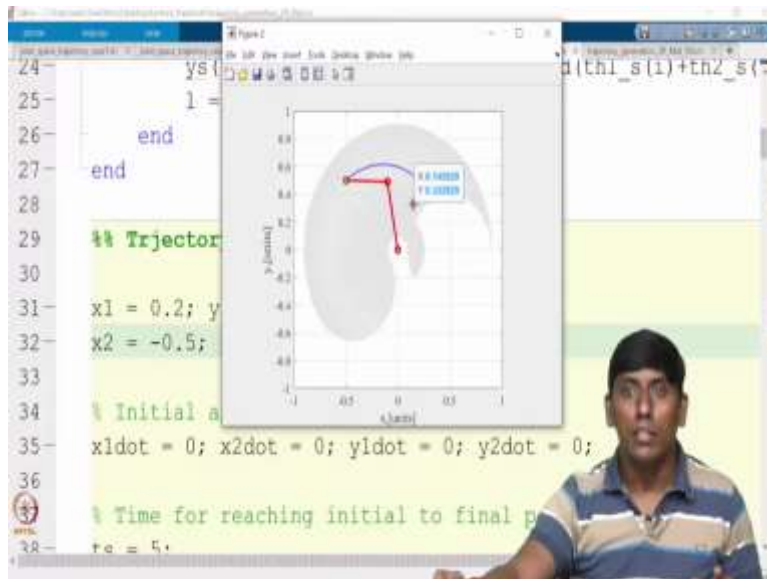
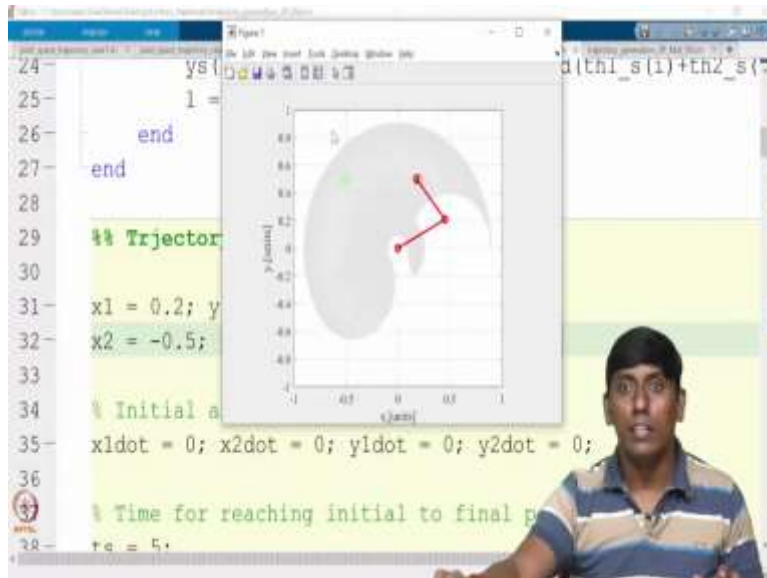
(Refer Slide Time: 14:18)

```

24-  ys(1) = L1*sind(th1_s(1))+L2*sind(th1_s(1)+th2_s(
25-  l = l+1;
26-  end
27- end
28
29-  % Trajectory points
30
31-  x1 = 0.2; y1 = 0.5; % Initial point
32-  x2 = -0.5; y2 = 0.5; % Final point
33
34-  % Initial and final velocities are zero
35-  x1dot = 0; x2dot = 0; y1dot = 0; y2dot = 0;
36
37-  % Time for reaching initial to final p
38-  te = 5;

```





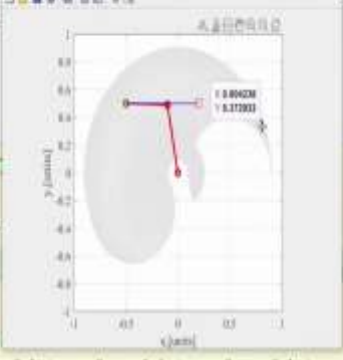
So, I will take it back. So, I will just run this. And we will modify that. So, I will take one point here and one point here. Then that would be within the workspace of both we can see at least. So probably I take 0.75 and 0.45 and here somewhere around 0.25 or something. So, I will just see that. So, in fact, I would have taken a grade as the point. So, this is the point 0.15 and 0.35 I take.

(Refer Slide Time: 15:03)

```
24     ys(i) = L1*sind(th1_s(i))+L2*sind(th1_s(i)+th2_s(i));
25     l = l+1;
26     end
27 end
28
29 %% Trajectory points
30
31 x1 = 0.2; y1 = 0.5; % Initial point
32 x2 = 0.15; y2 = 0.35; % Final point
33
34 % Initial and final velocities are zero
35 x1dot = 0; x2dot = 0; y1dot = 0; y2dot = 0;
36
37 % Time for reaching initial to final p
38 re = 5;
```



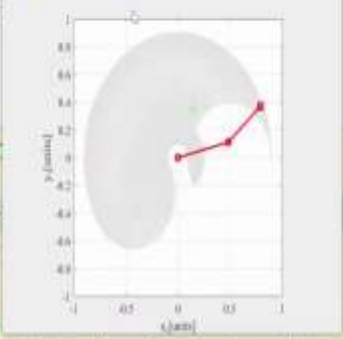
```
24     ys(i) = L1*sind(th1_s(i))+L2*sind(th1_s(i)+th2_s(i));
25     l = l+1;
26     end
27 end
28
29 %% Trajectory
30
31 x1 = 0.2; y1 = 0.5; % Initial point
32 x2 = 0.15; y2 = 0.35; % Final point
33
34 % Initial and final velocities are zero
35 x1dot = 0; x2dot = 0; y1dot = 0; y2dot = 0;
36
37 % Time for reaching initial to final p
38 re = 5;
```

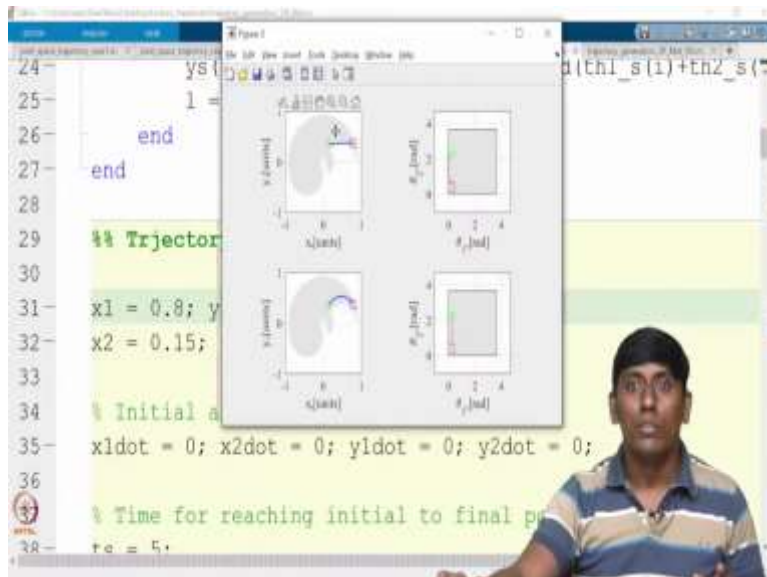
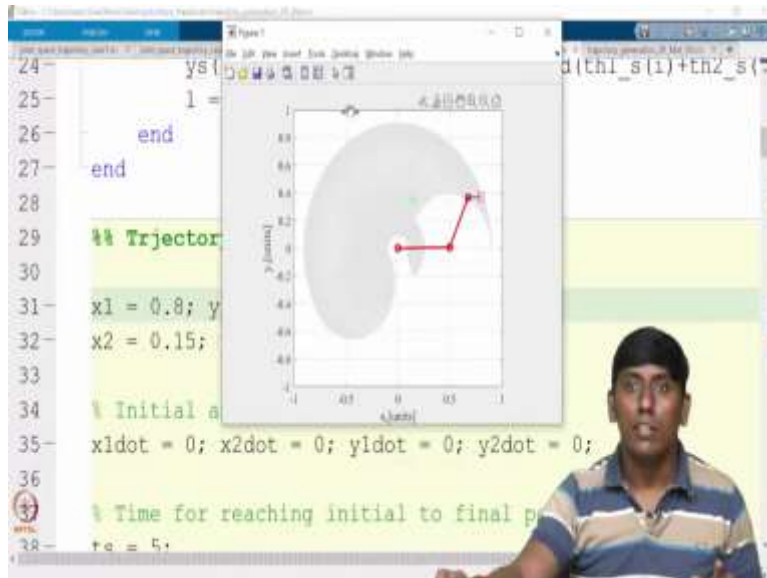


```
26     end
27 end
28
29 %% Trajectory points
30
31 x1 = 0.8; y1 = 0.37; % Initial point
32 x2 = 0.15; y2 = 0.35; % Final point
33
34 % Initial and final velocities are zero
35 x1dot = 0; x2dot = 0; y1dot = 0; y2dot = 0;
36
37 % Time for reaching initial to final p
38 re = 5;
```



```
24     ysl
25     l =
26     end
27 end
28
29 %% Trajectory
30
31 x1 = 0.8; y1 = 0.37; % Initial point
32 x2 = 0.15; y2 = 0.35; % Final point
33
34 % Initial and final velocities are zero
35 x1dot = 0; x2dot = 0; y1dot = 0; y2dot = 0;
36
37 % Time for reaching initial to final p
38 re = 5;
```





So, this is so this is 0.15 and so 0.35 and other point I am trying to take the input point somewhere here. So, 0.8 and 0.3 so 0.37 and 0.8 and just to taking a random point. So, I am just trying to demonstrate So, you can see it. So, it is going to make it outside but provided the joint space scheme is still following that profile. This is what we have seen as one simple case. So even I will take the other one.

(Refer Slide Time: 15:54)

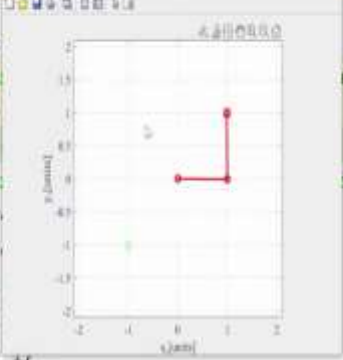
```
10- t = 0:0.1:tf; % time span
11- L1 = 1; L2 = 1;
12- A = [1,0,0,0;
13-      0,1,0,0;
14-      1,tf,tf^2,tf^3;
15-      0,1,2*tf,3*tf^2];
16- bx = [x0;xdot0;xf;xdotf];
17- tcx = inv(A)*bx;
18- for i=1:length(t)
19-     %% Task-space trajectory
20-     xt(i) = [1,t(i),t(i)^2,t(i)^3]*tcx;
21-     yt(i) = (yf-y0)/(xf-x0)*(xt(i)-xf)+yf;
22-     %% Inverse kinematics
23-     c2t = (xt(i)^2+yt(i)^2-L1^2-L2^2)/(2*
```

```
1 %% Start
2 clear all; close all; clc;
3 %% Initial and final end-effector positions
4 x0 = 1; y0 = 1;
5 xf = -1; yf = 1.5;
6 %% Initial and final end-effector velocities
7 xdot0 = 0.5; ydot0 = 0;
8 xdotf = 0; ydotf = 0.5;
9 tf = 10; % trajectory duration
10 t = 0:0.1:tf; % time span
11 L1 = 1; L2 = 1;
12 A = [1,0,0,0;
13      0,1,0,0;
14      1,tf,tf^2,tf^3;
```

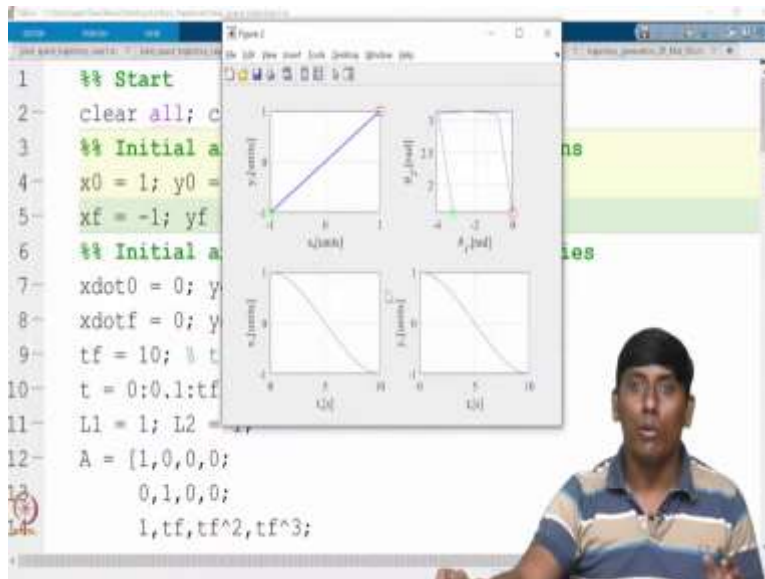
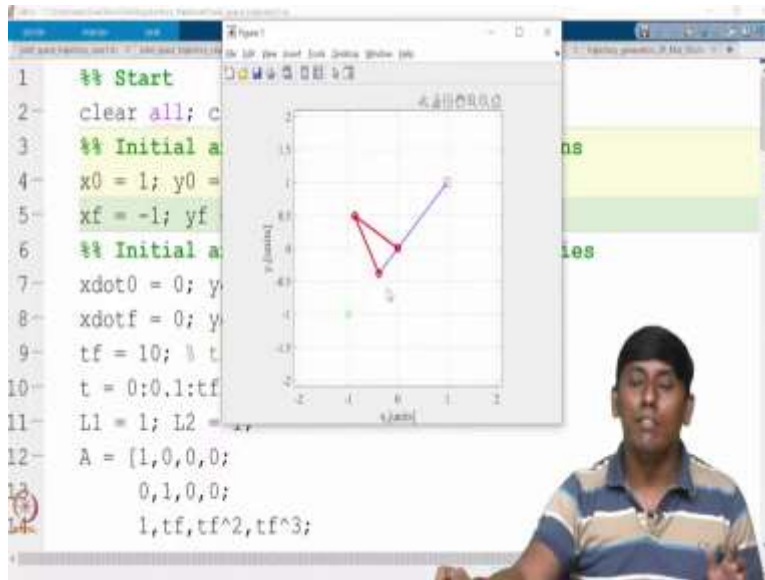
```
1 %% Start
2 clear all; close all; clc;
3 %% Initial and final end-effector positions
4 x0 = 1; y0 = 1;
5 xf = -1; yf = -1;
6 %% Initial and final end-effector velocities
7 xdot0 = 0; ydot0 = 0;
8 xdotf = 0; ydotf = 0;
9 tf = 10; % trajectory duration
10 t = 0:0.1:tf; % time span
11 L1 = 1; L2 = 1;
12 A = [1,0,0,0;
13      0,1,0,0;
14      1,tf,tf^2,tf^3;
```



```
1 %% Start
2 clear all; c
3 %% Initial a
4 x0 = 1; y0 =
5 xf = -1; yf
6 %% Initial a
7 xdot0 = 0; y
8 xdotf = 0; y
9 tf = 10; % t
10 t = 0:0.1:tf
11 L1 = 1; L2 =
12 A = [1,0,0,0;
13      0,1,0,0;
14      1,tf,tf^2,tf^3;
```







So, the task space trajectory itself. So, I will take it here. So, just the mirror image I will take. So, this I will take it 0. So, just to show that where it failed so it mirror image in the sense it is minus 1 you can feel it in a single simulation itself. You can feel it how this is going to be very tough; you can see it is going to fall in here single in the sense that it is going to fall on a singular point.

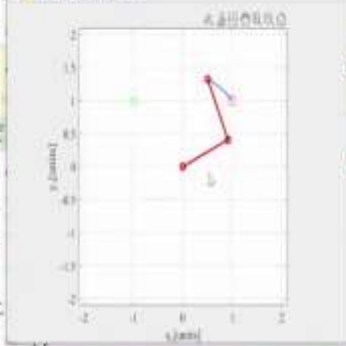
And you can see that it is giving the other solution. The task space which is giving the other solution. So, you can follow it. So, this is what I was saying if the solution of the same point, which is having multiple solution, the scheme can let go it in the end of this. So why it is it is just a mirror image of this. So that is why it is giving the other solution.

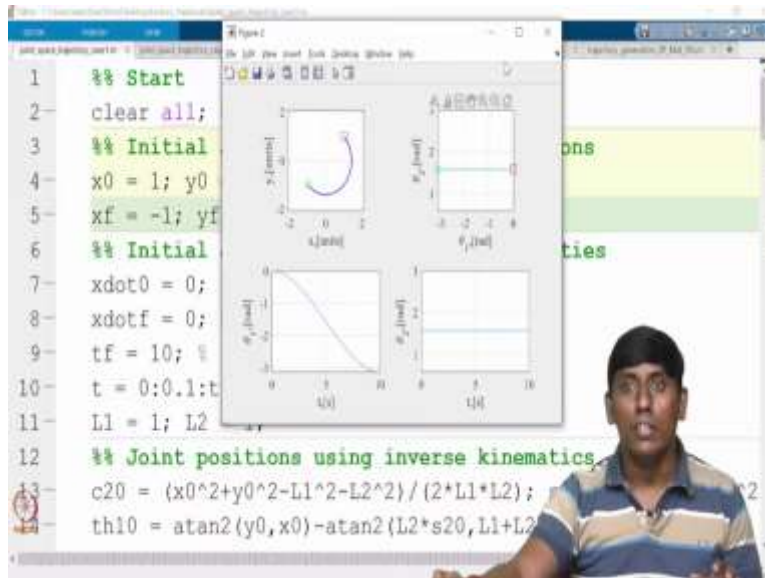


(Refer Slide Time: 16:54)

```
3 %% Initial and final end-effector positions
4 x0 = 1; y0 = 1;
5 xf = -1; yf = 1;
6 %% Initial and final end-effector velocities
7 xdot0 = 0; ydot0 = 0;
8 xdotf = 0; ydotf = 0;
9 tf = 10; % trajectory duration
10 t = 0:0.1:tf; % time span
11 L1 = 1; L2 = 1;
12 %% Joint positions using inverse kinematics
13 c20 = (x0^2+y0^2-L1^2-L2^2)/(2*L1*L2);
14 th10 = atan2(y0,x0)-atan2(L2*s20,L1+L2
```

```
1 %% Start
2 clear all;
3 %% Initial
4 x0 = 1; y0
5 xf = -1; yf
6 %% Initial
7 xdot0 = 0;
8 xdotf = 0;
9 tf = 10; %
10 t = 0:0.1:t
11 L1 = 1; L2
12 %% Joint positions using inverse kinematics
13 c20 = (x0^2+y0^2-L1^2-L2^2)/(2*L1*L2);
14 th10 = atan2(y0,x0)-atan2(L2*s20,L1+L2
```

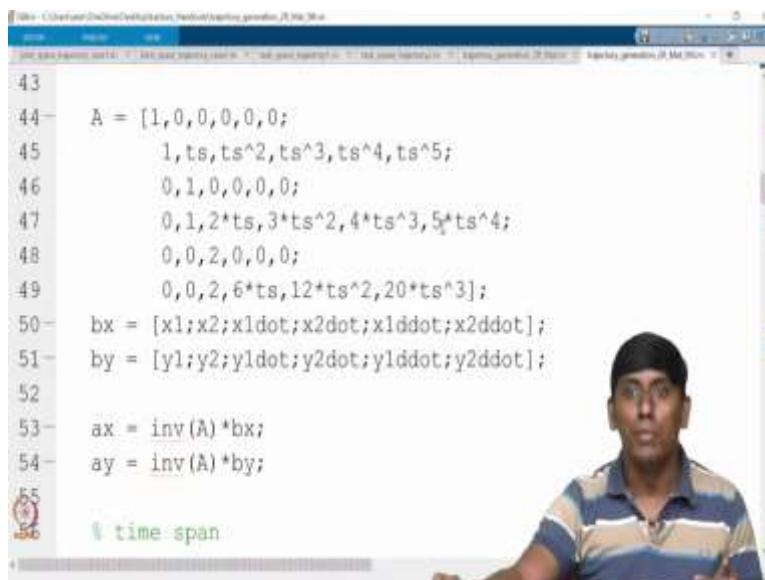




So now the same situation if you do it in joint space, we will try to see whether that is matching at least or not. So, with that, I like to like end it. So, this one and this one and we are trying to see. This is minus 1 I have to take. So, this is minus 1. So, I will try to see whether that consideration of near singular and other thing is restricted you can see it is overcome. So, this way as long the workspace is available even that solution is exists.

So, that is what we wanted to see. So, you can see like this point to this point in the task space is going with what you call overlap with a singular point. But the other case it is not happening it. So, these all we wanted to check.

(Refer Slide Time: 17:54)



```

115
116- ath1 = inv(A)*bth1;
117- ath2 = inv(A)*bth2;
118
119- figure
120- for i=1:length(t)
121
122     % Joint-space trajectory
123     th1(i) = [1,t(i),t(i)^2,t(i)^3,t(i)^4,t(i)^5]*ath1;
124     th2(i) = [1,t(i),t(i)^2,t(i)^3,t(i)^4,t(i)^5]*ath2;
125     % Forward kinematic model (for calculating the task-space)
126     x(i) = L1*cos(th1(i)) + L2*cos(th1(i)+th2(i));
127     y(i) = L1*sin(th1(i)) + L2*sin(th1(i)+th2(i));

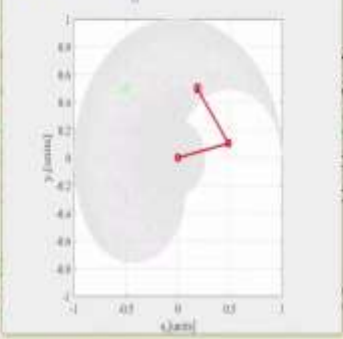
```



```

121
122     % Joint
123     th1(i)
124     th2(i)
125     % Forwa
126     x(i) =
127     y(i) =
128
129     % works
130     plot(xs
131     hold on
132     axis([-1 1 -1 1]);
133     grid on
134     axis square

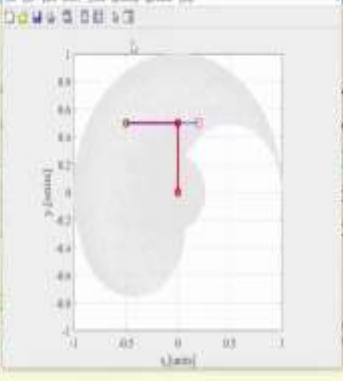
```



```

121
122 % Joint
123 th1(i)
124 th2(i)
125 % Forwa
126 x(i) =
127 y(i) =
128
129 % works
130 plot(xs
131 hold on
132 axis([-1 1 -1 1]);
133 grid on
134 axis square

```



```

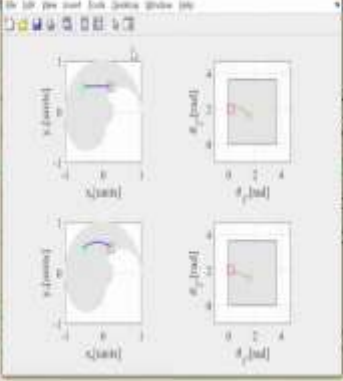
% ,t(i)^5]*ath1;
% ,t(i)^5]*ath2;
relating the task-s
(i)+th2(i));
(i)+th2(i));

```

```

121
122 % Joint
123 th1(i)
124 th2(i)
125 % Forwa
126 x(i) =
127 y(i) =
128
129 % works
130 plot(xs
131 hold on
132 axis([-1 1 -1 1]);
133 grid on
134 axis square

```



```

% ,t(i)^5]*ath1;
% ,t(i)^5]*ath2;
relating the task-s
(i)+th2(i));
(i)+th2(i));

```



So, the same thing we can even do for a fifth order. So, the fifth order you can see why only 1 change will come the A matrix and the coefficients would get changed. So, because of that, this you can say task space trajectory or joint space trajectory so that you can say values will change remaining all are same. So, for example, I just if you I run. So, the same thing we can realize it and it is going to happen.

So, now, I hope you are clear to this what you call how to find the workspace and how to get the task space trajectory and joint space trajectory and one if it is failing can we use the second one or which is beneficial and what instant it is beneficial all those things you can look at it. However, the task space trajectory is always better. However, like you can avoid certain constraint points because that will give more realistic.

Because that is going to give in real time. Sometime workspace means we always see only in the task space, but the workspace even we can realize it in the joint space. So here the workspace is minimum to maximum of each joint. So here are only 2 joints it is in a plane, if it is a 3 joint it would come as a volume. So similarly, if it is xyz that would come in a volume. So even we include orientation that would come in the higher order dimensional which is not directly visible to us.

So, with that I am closing this particular lecture. I hope this is giving some clarity on so what is trajectory and how to generate a trajectory? How we can imply or how we can employ these games in the you can say serial manipulator and serial manipulator what is the difference

between joint space and task space these all you realize it. So, with that, I am ending this particular lecture.

And the next lecture we will be talking about control we will initially start with open loop, then we will go to feedback which we call closed loop. Then we will go the nonlinear scheme, then we can like close the subject with the advanced topic. With that see you then, thank you bye, take care.