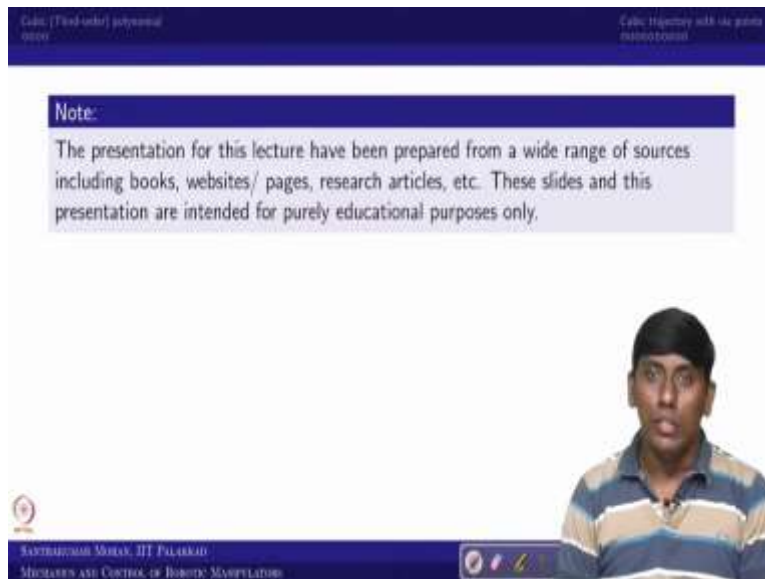


Mechanics and Control of Robotic Manipulators
Professor Santhakumar Mohan
Department of Mechanical Engineering
Indian Institute of Technology, Palakkad
Lecture No 32
Trajectory generation using MATLAB part 1

Welcome back to Mechanics and Control of Robotic Manipulator. So, far last class till now so, what we have covered is basically like a trajectory generation. So, the trajectory can be generated using several you can see generation schemes even there are several methods of generating in robotic manipulator. So, this particular class we are going to see how we can generate a trajectory by using different you can say polynomial function or cycloidal function or all. So, that to like with the help of MATLAB we are trying to do.

(Refer Slide Time: 0:46)



The screenshot shows a presentation slide with a dark blue header and footer. The header contains the text "Cubic (Third-order) polynomial" on the left and "Cubic Trajectory with six points" on the right. The main content area features a "Note:" box 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." In the bottom right corner, there is a video inset showing Professor Santhakumar Mohan, who is wearing a blue and white striped shirt and a black cap. The footer of the slide includes the IIT Palakkad logo, the text "SANTHAKUMAR MOHAN, IIT PALAKKAD" and "MECHANICS AND CONTROL OF ROBOTIC MANIPULATORS", and a set of navigation icons.

Cubic (Third-order) polynomial
#000

Cubic Trajectory with via points
#00000000

TRAJECTORY GENERATION USING MATLAB

- 1 Cubic (Third-order) polynomial
- 2 Cubic trajectory with via points



SARABJIT SINGH MOHAN, IIT PALAKHEDI
MECHATRONICS AND CONTROL OF ROBOTS: MANIPULATORS

So, in that sense we are trying to see in this particular short lecture we are going to see third order polynomial. So, where the cubic polynomial function as the trajectory generator and in that we would be seeing additionally like with the via point even the via point we have seen like the via point velocity specified or not how that can be. So, if that is the case, we will go to the you can say scenario.

(Refer Slide Time: 1:10)


Cubic (Third-order) polynomial
#000

Cubic Trajectory with via points
#00000000

The cubic (third-order) polynomial function:

$$\begin{aligned} x(t) &= a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3 \\ \dot{x}(t) &= a_1 + 2a_2(t - t_0) + 3a_3(t - t_0)^2 \\ \ddot{x}(t) &= 2a_2 + 6a_3(t - t_0) \end{aligned} \quad (1)$$

The general situation, where $t_0 = 0$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ x_f \\ \dot{x}_f \end{bmatrix} \quad (2)$$


SARABJIT SINGH MOHAN, IIT PALAKHEDI
MECHATRONICS AND CONTROL OF ROBOTS: MANIPULATORS

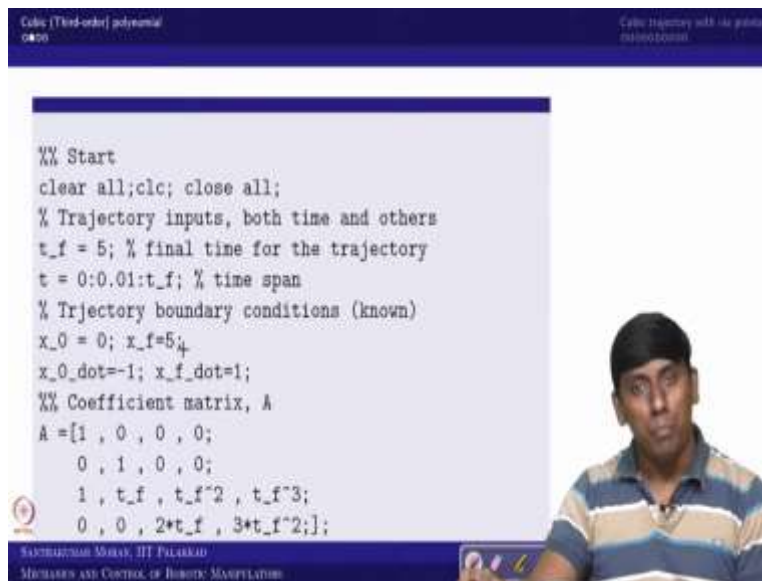
So, where the cubic polynomial function or the third order polynomial function will give the displacement velocity and acceleration in this form. So, where you can see this is the displacement and this is the velocity and acceleration. So, these all can be written in this way.

So, where t_{naught} conventionally we assumed as to be 0 even the t_{naught} is something. So, we will add the t_{naught} whenever there is a need.

So, in that sense, we will rewrite this equation if t_{naught} is 0. So, we can rewrite this equation in a matrix and vector form. So, where the; a_{naught} a_1 a_2 a_3 are unknown. So, the initial you can say and final conditions both displacement and velocity all are known to us. So, in the sense these are known input, and these are unknown and these are the coefficient again in the coefficient matrix are known which would be function of t_f or the constants.

So, now once you know this all. So, what we can do we can write a MATLAB code and we try to generate and then we can try to generate whatever we have written as the cubic or third order polynomial function and try to see whatever we have seen in the lecture is following or not.

(Refer Slide Time: 2:25)



```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
t_f = 5; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trajectory boundary conditions (known)
x_0 = 0; x_f=5;
x_0_dot=-1; x_f_dot=1;
%% Coefficient matrix, A
A=[1, 0, 0, 0;
    0, 1, 0, 0;
    1, t_f, t_f^2, t_f^3;
    0, 0, 2*t_f, 3*t_f^2];
```

FAUZIYAH MOHAMMAD, IIT PALAKKAD
MEMBERS AND CONTROL OF ROBOTS, MANIPULATORS

The cubic (third-order) polynomial function:

$$\begin{aligned}x(t) &= a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3 \\ \dot{x}(t) &= a_1 + 2a_2(t - t_0) + 3a_3(t - t_0)^2 \\ \ddot{x}(t) &= 2a_2 + 6a_3(t - t_0)\end{aligned}\quad (1)$$

The general situation, where $t_0 = 0$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ x_f \\ \dot{x}_f \end{bmatrix}\quad (2)$$

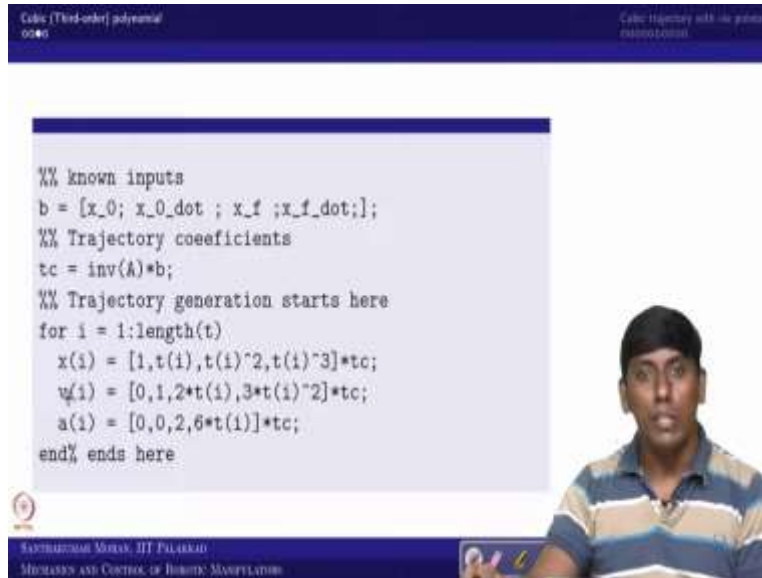


So, for that we are writing the MATLAB code. So, we know like the initial is straightforward syntax. So, they clear all you can say clear the command history or close all these are standard because we are going to run you can say figure windows and all so, better we need to close. So, then you can see that the trajectory inputs we need to view where the trajectory inputs are 1 is the finite time and we always assume that t naught is 0 in this case.

So, then the timespan which start from t naught to t the t naught we assumed as 0. So, this is what the trajectory input of the time. So, further you know like x of 0 and x of t_f and the x dot of t_0 and x dot of t_f all are given. So, now these all we assumed to be known. So, now these are actually given and assumed to be known. Then we can write the coefficient matrix what we derived here. So, that coefficient matrix we have written here.

So, now what else we need to like talk about what is the; you can say this known value that I will write as a b vector and this I am going to call us t_c vector. So, in the sense trajectory coefficient vector.

(Refer Slide Time: 3:39)



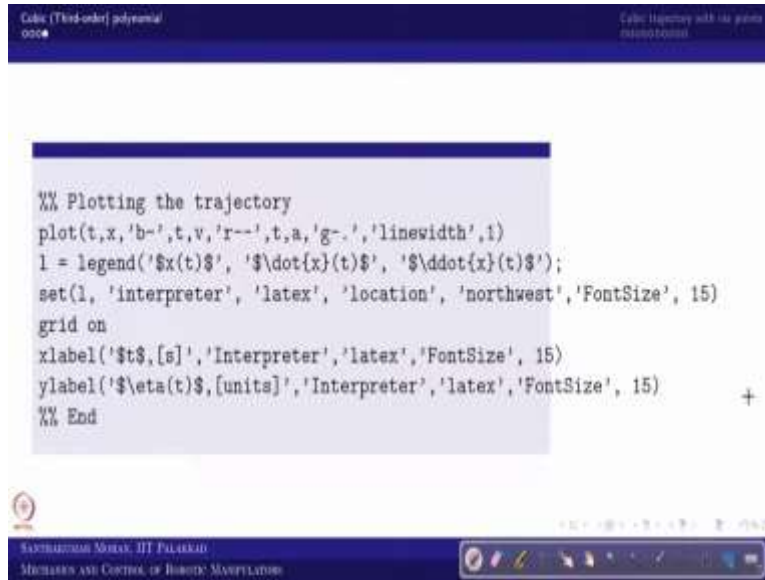
```
%% Cubic (Third-order) polynomial
%% Calc trajectory with six points
%%
%% known inputs
b = [x_0; x_0_dot ; x_f ;x_f_dot];
%% Trajectory coefficients
tc = inv(A)*b;
%% Trajectory generation starts here
for i = 1:length(t)
    x(i) = [1,t(i),t(i)^2,t(i)^3]*tc;
    v(i) = [0,1,2*t(i),3*t(i)^2]*tc;
    a(i) = [0,0,2,6*t(i)]*tc;
end% ends here
```

SANDEEP MOHAN, IIT PALAKKI
MECHANICS AND CONTROL OF ROBOTS MANIPULATORS

So, now in that sense you can see like this is the known input which is written as a b mat you can say b vector. So, which is initial you can say position initial velocity final position and final velocity which we have written in the standard syntax.

So, after that the trajectory coefficient can be found. So, there is a f is missing so, like this is co f it is f is double. So, anyhow it is comment it is not a problem. Then that trajectory coefficient can be taken in this form. So, once that is done. So, then you can see like x x dot and x double dot written in this way. So, here I do not want to write x dot x double dot I have written x is position v is the velocity and a is the acceleration. So, this is what the equation which we obtain. So, once these all done. So, what we can do?

(Refer Slide Time: 4:29)



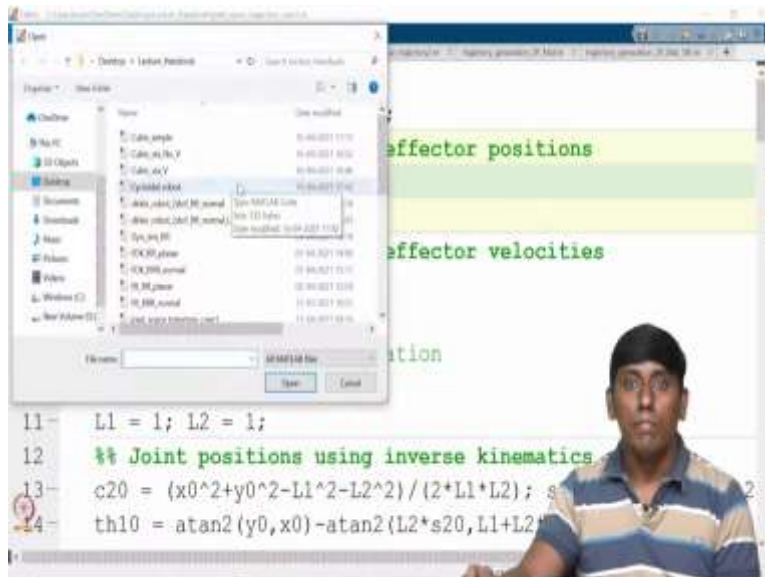
Cubic (Third-order) polynomial
Cubic trajectory with its points
numerically

```
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-','linewidth',1)
l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{x}(t)$');
set(l,'interpreter','latex','location','northwest','FontSize',15)
grid on
xlabel('$t$, [s]','Interpreter','latex','FontSize',15)
ylabel('$\eta(t)$, [units]','Interpreter','latex','FontSize',15)
%% End
```

SANDESH MISHRA, IIT PALAKHAR
MECHATRONICS AND CONTROL OF ROBOTS MANIPULATORS

We can plot the trajectory. So, where we can see the x trajectory v trajectory a trajectory we can plot and then see it. So, in that sense, we can go to the MATLAB and try to see how this can be done.

(Refer Slide Time: 4:47)



Workspace: effector positions, effector velocities, position

```
11- L1 = 1; L2 = 1;
12- % Joint positions using inverse kinematics
13- c20 = (x0^2+y0^2-L1^2-L2^2)/(2*L1*L2); s20 = sqrt(1-c20^2);
14- th10 = atan2(y0,x0)-atan2(L2*s20,L1+L2*c20);
```

SANDESH MISHRA, IIT PALAKHAR
MECHATRONICS AND CONTROL OF ROBOTS MANIPULATORS

```
1 %% Start
2 clear all;clc; close all;
3 % Trajectory inputs, both time and others
4 t_f = 5; % final time for the trajectory
5 t = 0:0.01:t_f; % time span
6 % Trajectory boundary conditions (known)
7 x_0 = 0; x_f=5;
8 x_f_dot=1; x_0_dot=-1;
9 %% Coefficient matrix, A
10 A=[1, 0, 0, 0;
11     0, 1, 0, 0;
12     1, t_f, t_f^2, t_f^3;
13     0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
```



```
7 x_0 = 0; x_f=5;
8 x_f_dot=1; x_0_dot=-1;
9 %% Coefficient matrix, A
10 A=[1, 0, 0, 0;
11     0, 1, 0, 0;
12     1, t_f, t_f^2, t_f^3;
13     0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
16 %% Trajectory coefficients
17 tc = inv(A)*b;
18 %% Trajectory generation starts here
19 for i = 1:length(t)
20     x(i) = [1,t(i),t(i)^2,t(i)^3]*tc;
21     v(i) = [0,1,2*t(i),3*t(i)^2]*tc;
```




```

13     0, 0, 2*t_f, 3*t_f^2;];
14     %% known inputs
15     b = [x_0; x_0_dot; x_f; x_f_dot];
16     %% Trajectory coefficients
17     tc = inv(A)*b;
18     %% Trajectory generation starts here
19     for i = 1:length(t)
20         x(i) = [1,t(i),t(i)^2,t(i)^3]*tc;
21         v(i) = [0,1,2*t(i),3*t(i)^2]*tc;
22         a(i) = [0,0,2,6*t(i)]*tc;
23     end % ends here
24     %% Plotting the trajectory
25     plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth',2);
26     legend('$x(t)$', '$\dot{x}(t)$', '$a(t)$');

```

```

13     0, 0, 2*t_f, 3*t_f^2;];
14     %% known inputs
15     b = [x_0; x_0_dot; x_f; x_f_dot];
16     %% Trajectory coefficients
17     tc = inv(A)*b;
18     %% Trajectory generation starts here
19     for i = 1:length(t)
20         x(i) = [1,t(i),t(i)^2,t(i)^3]*tc;
21         v(i) = [0,1,2*t(i),3*t(i)^2]*tc;
22         a(i) = [0,0,2,6*t(i)]*tc;
23     end % ends here
24     %% Plotting the trajectory
25     plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth',2);
26     legend('$x(t)$', '$\dot{x}(t)$', '$a(t)$');

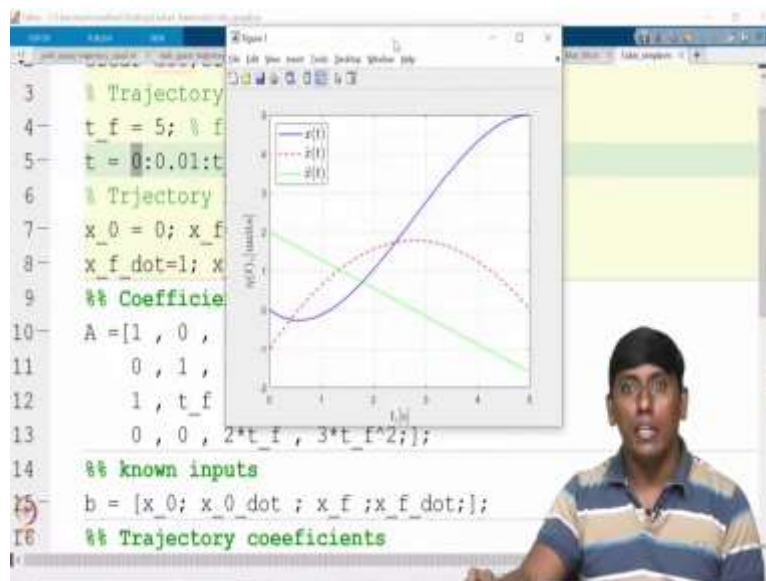
```

So, for that I am opening the MATLAB. So, you can see like this is what we have written. So, the initial position is you can see instead of giving theta naught and theta you can say the case. So, I will just check that. So, So, cubic sample. So, we can see that instead of giving that we are giving the x naught and x f and x f dot and x naught dot as given. So, these all standard. So, the only thing is we can make it.

So, now we are taking this b and then t c we have done and this matrix whatever we have obtained this a from there you can see this is what the time base. So, now this we can convert into t of i. So, that is what the relation and this way we can write velocity and acceleration and we can plot in this way. So, in that sense if I run this code.

(Refer Slide Time: 5:51)

```
4 t_f = 5; % final time for the trajectory
5 t = 0:0.01:t_f; % time span
6 % Trajectory boundary conditions (known)
7 x_0 = 0; x_f=5;
8 x_f_dot=1; x_0_dot=-1;
9 %% Coefficient matrix, A
10 A = [1, 0, 0, 0;
11      0, 1, 0, 0;
12      1, t_f, t_f^2, t_f^3;
13      0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
16 %% Trajectory coefficients
```



So, we can see like it is starting from you can say 0 and it goes 5 and the velocity is starting from you can see like a minus 1 to end with 1 units. So, the total finite time is which start from 0 to 5. So, we can see all those things in the MATLAB you can see running. So, I have run this, and you can see like and now there are three things. So, you can see like this is the x . So, x of time this is actual time versus the η of t η of can be written as x x dot and x double dot.

So, now, you can see the blue curve is x . So, x of t is very from 0 to 5 in a cubic polynomial form. So, this curve is very bent because there is a nonzero initial velocity which is minus 1 you can say units So, that is why it is going in that curve manner. Similarly, the acceleration you can

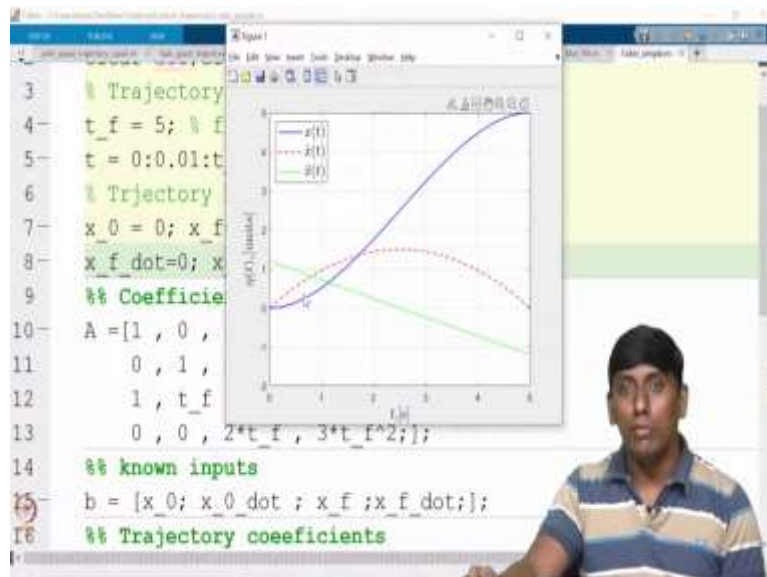
see like acceleration we do not have control, but the acceleration is finite at the initial and as well as finite at the end in the sense constant acceleration and deceleration in the initial and final. That way we can see in the sense nonzero initial and final acceleration. So, now in order to get the more clarity.

(Refer Slide Time: 7:07)

```

3  % Trajectory inputs, both time and others
4  t_f = 5; % final time for the trajectory
5  t = 0:0.01:t_f; % time span
6  % Trajectory boundary conditions (known)
7  x_0 = 0; x_f=5;
8  x_f_dot=0; x_0_dot=0;
9  %% Coefficient matrix, A
10 A = [1, 0, 0, 0;
11       0, 1, 0, 0;
12       1, t_f, t_f^2, t_f^3;
13       0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
16 %% Trajectory coefficients

```



So, for example, I am just making the initial and final velocities are 0 just for understanding. So, you can see like now, the curve would be a little different. You can see like it is smooth, and again you can see the maximum velocity is happening at the middle which is 2.5 here and the acceleration is actually 0 at that point. So, that way we can look at. Now, the curve is you can see

it is going much more smooth, the other one is look like a non-minimum phase system. So, I do not want to bring that unnecessarily complicated term. So, this is what the idea.

(Refer Slide Time: 7:45)

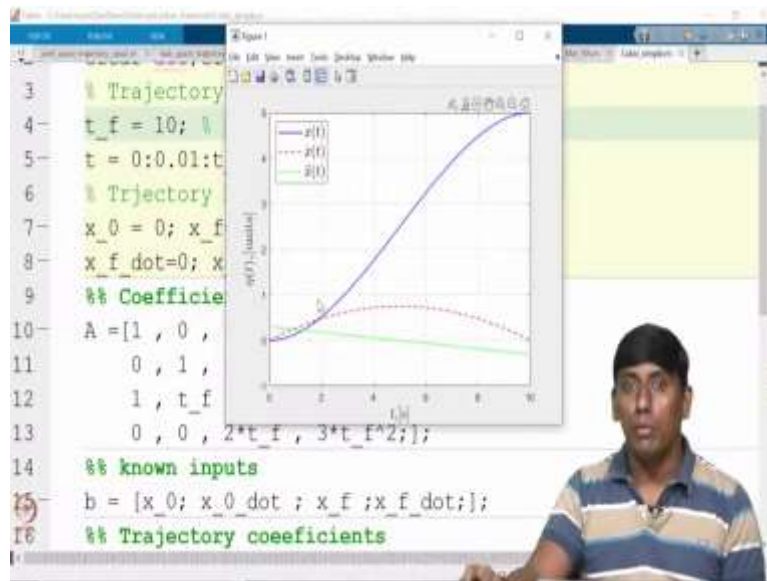
```
3 % Trajectory inputs, both time and others
4 t_f = 2; % final time for the trajectory
5 t = 0:0.01:t_f; % time span
6 % Trajectory boundary conditions (known)
7 x_0 = 0; x_f=5;
8 x_f_dot=0; x_0_dot=0;
9 %% Coefficient matrix, A
10 A = [1, 0, 0, 0;
11       0, 1, 0, 0;
12       1, t_f, t_f^2, t_f^3;
13       0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
16 %% Trajectory coefficients
```

The plot shows three curves over time t from 0 to 2. The x-axis is labeled 't' and the y-axis is labeled 'x(t)'. The legend indicates: a solid blue line for x(t), a dashed red line for x_dot(t), and a solid green line for x_double_dot(t). The x(t) curve starts at 0 and ends at 5. The x_dot(t) curve starts at 0 and ends at 0. The x_double_dot(t) curve starts at 0 and ends at 0.

So, now even if I change the time you can see that I am just giving only t f is 2. So, now you can see like it is compressed. So, now the concept you can see it is modified within 2 seconds it is starting from 0 to 5 it is maintained. But the correspondingly you can see the acceleration increased because you have reduced the timespan and the velocity also, like increase now it is approximately 4 units that what you can see approximately.

(Refer Slide Time: 8:16)

```
3  % Trajectory inputs, both time and others
4  t_f = 10; % final time for the trajectory.
5  t = 0:0.01:t_f; % time span
6  % Trajectory boundary conditions (known)
7  x_0 = 0; x_f=5;
8  x_f_dot=0; x_0_dot=0;
9  %% Coefficient matrix, A
10 A = [1, 0, 0, 0;
11      0, 1, 0, 0;
12      1, t_f, t_f^2, t_f^3;
13      0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
16 %% Trajectory coefficients
```



So, now you can even do it to probably higher, then you can see that the acceleration and velocity would be getting decreased. However, 0 to 5 you can see it. So, these are all variations you can try.

(Refer Slide Time: 8:29)

The top screenshot shows a MATLAB script with the following code:

```
4 t_f = 5; % final time for the trajectory
5 t = 0:0.01:t_f; % time span
6 % Trajectory boundary conditions (known)
7 x_0 = 2; x_f=5;
8 x_f_dot=0; x_0_dot=0;
9 %% Coefficient matrix, A
10 A = [1, 0, 0, 0;
11       0, 1, 0, 0;
12       1, t_f, t_f^2, t_f^3;
13       0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
16 %% Trajectory coefficients
```

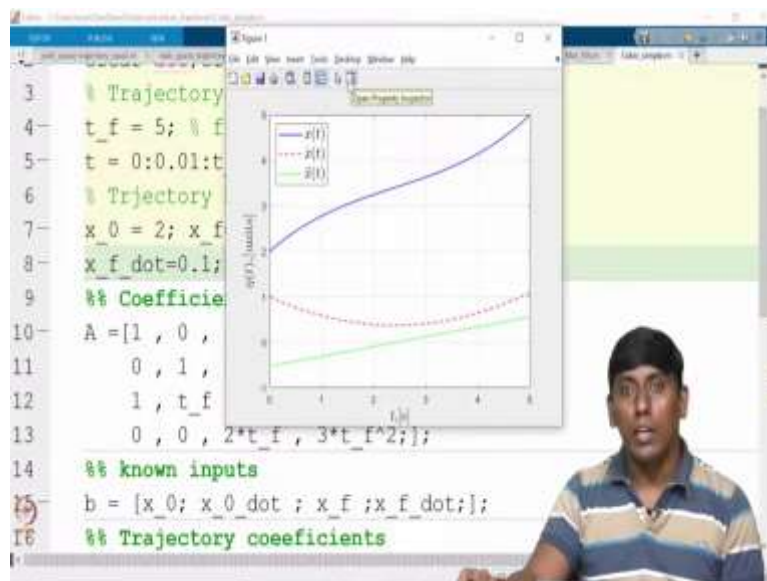
The bottom screenshot shows the same script running, with a plot of the trajectory. The plot displays three variables over time t from 0 to 5:

- $x(t)$ (solid blue line): Starts at 2 and increases to 5.
- $\dot{x}(t)$ (dotted red line): Starts at 0, increases to a peak, and then decreases to 0.
- $x(t)$ (solid green line): Starts at 2 and decreases to 0.

So, instead of this if I to try to give you for example, this is starting from 2. So, and I am making it this is 5 second. So, even you can feel it. It is starting from 2 and it end at 5. So, now you can see how the velocity and acceleration is going on. So, now you can see like obviously there is no control over the acceleration only maximum you can do the velocity.

(Refer Slide Time: 8:55)

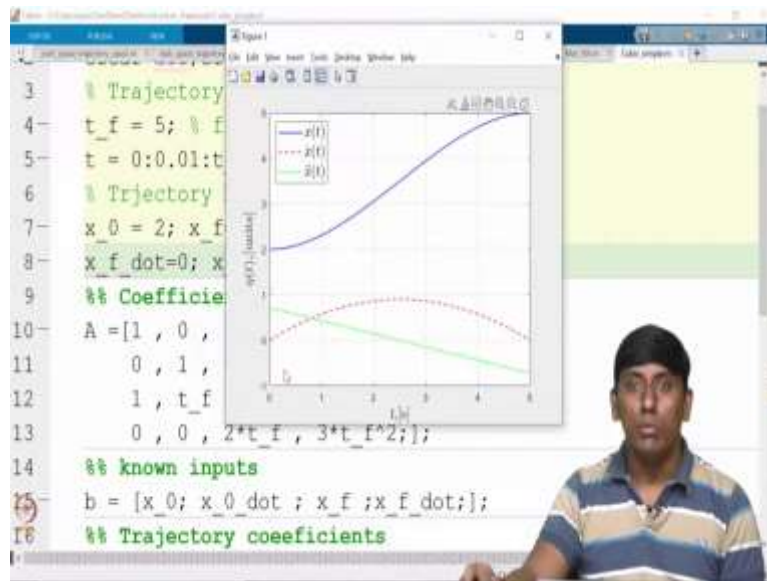
```
3 % Trajectory inputs, both time and others
4 t_f = 5; % final time for the trajectory
5 t = 0:0.01:t_f; % time span
6 % Trajectory boundary conditions (known)
7 x_0 = 2; x_f=5;
8 x_f_dot=0.1; x_0_dot=1;
9 %% Coefficient matrix, A
10 A=[1, 0, 0, 0;
11     0, 1, 0, 0;
12     1, t_f, t_f^2, t_f^3;
13     0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
16 %% Trajectory coefficients
```



For example, now this is 0.1 when it started and the end of it, I wanted 1 meter or you can say 1 unit. So, now I am trying to rotate sorry I am trying to run you can see like it is starting from you can see final is 0.1 and the initial is you can see something around what you call z 1. So, that is what it is trying to happen. So, now this all happening. So, now you can see like it is starting from 1 and it is a move to 1.1 because. So, you have a starting from 2 to 5. So, that is the case. So, you can get it.

(Refer Slide Time: 9:39)

```
3 % Trajectory inputs, both time and others
4 t_f = 5; % final time for the trajectory
5 t = 0:0.01:t_f; % time span
6 % Trajectory boundary conditions (known)
7 x_0 = 2; x_f=5;
8 x_f_dot=0; x_0_dot=0;
9 %% Coefficient matrix, A
10 A=[1, 0, 0, 0;
11     0, 1, 0, 0;
12     1, t_f, t_f^2, t_f^3;
13     0, 0, 2*t_f, 3*t_f^2];
14 %% known inputs
15 b = [x_0; x_0_dot; x_f; x_f_dot];
16 %% Trajectory coefficients
```



So, now in order to make it clear. So, I will just make it this 0 and this is also 0. So, now you can see it is much more smoother than the earlier. This is what one such. So, you can see it this all. So, now we will go back to what you call the MATLAB before that I will just open it.

(Refer Slide Time: 10:03)

```
1 %% Start
2 clear all;clc; close all;
3 % Trajectory inputs, both time and others
4 t_v = 2; % time for first segment
5 t_f = 5; % final time for the trajectory
6 t = 0:0.1:t_f; % time span
7 % Trajectory boundary conditions (known)
8 x_0 = 0;
9 x_v = 2;
10 x_f=5;
11 x_f_dot=0;
12 x_0_dot=0;
13 %% Coefficient matrix, A
14 A = [1, 0, 0, 0, 0, 0, 0, 0, 0;
15      0, 1, 0, 0, 0, 0, 0, 0, 0;
```

So, what is called a cubic. These all just better open. So, that it would be easy for us to come back.

(Refer Slide Time: 10:15)

Cubic (Third order) polynomial
Cubic trajectory with via point

The cubic (third-order) polynomial function with via point. In this case, the trajectory has two segments.

For $t = t_0$ to t_v

$$\begin{aligned} x(t) &= a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3 \\ \dot{x}(t) &= a_1 + 2a_2(t - t_0) + 3a_3(t - t_0)^2 \\ \ddot{x}(t) &= 2a_2 + 6a_3(t - t_0) \end{aligned} \quad (3)$$

For $t = t_v$ to t_f

$$\begin{aligned} x(t) &= b_0 + b_1(t - t_v) + b_2(t - t_v)^2 + b_3(t - t_v)^3 \\ \dot{x}(t) &= b_1 + b_2(t - t_v) + b_3(t - t_v)^2 \\ \ddot{x}(t) &= b_2 + b_3(t - t_v) \end{aligned} \quad (4)$$

AMRITHAN MURAN, IIT PALAKKAD
MECHATRONICS AND CONTROL OF ROBOTS: MANIPULATORS

So, now I am actually going back to the slide. So, you can see like, the second thing is what we are trying to cover is cubic trajectories with via point. So, in the sense it is a segment. So, for understanding here I have taken only 2 segments in the sense there is a via point which will denote correspondingly the time is t_v . So, the profile or the trajectory start from t_0 to t_v which is the first segment then the t_v to t_f is the second segment.

So, in that case so that trajectory can be written in this form. This is the first segment and the second segment. Now, there are two cases can come. So, where the; you can say via point velocity is specified. So, then it would be giving 8 equation and 8 unknowns. So, invariantly we can consider each segment separately because the final velocity of the first segment would be the initial velocity of the second segment. However, there would be a slight transition happening between you can say acceleration that is what we have seen in the lecture.

(Refer Slide Time: 11:22)

Cubic (third-order) polynomial

Cubic trajectory with via point

Case 1: The cubic (third-order) polynomial function with via point's velocity specified:
The general situation, where $t_0 = 0$.

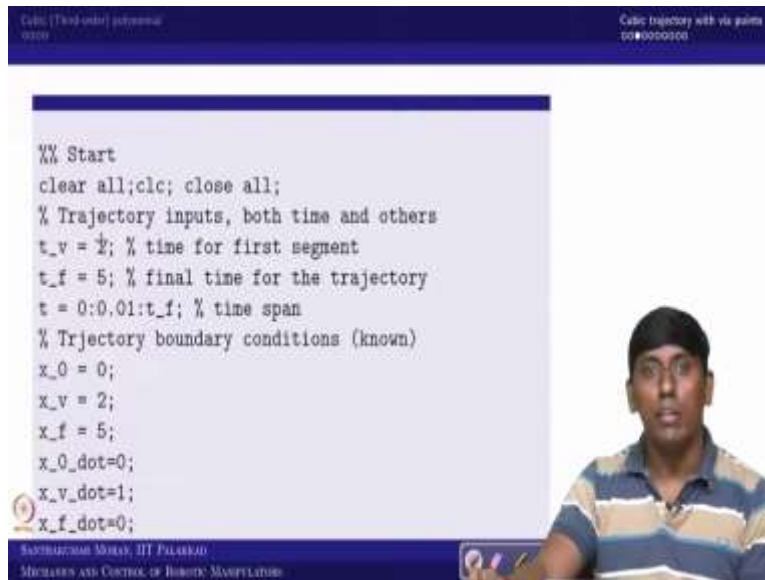
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & t_v & t_v^2 & t_v^3 \\ 0 & 1 & 2t_v & 3t_v^2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & (t_f - t_v) & (t_f - t_v)^2 & (t_f - t_v)^3 \\ 0 & 1 & 2(t_f - t_v) & 3(t_f - t_v)^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ x_v \\ \dot{x}_v \\ x_f \\ \dot{x}_f \end{bmatrix} \quad (5)$$

SANKARANARAYANAN MOHAN, IIT PALARU
MECHATRONICS AND CONTROL OF ROBOTS/MANIPULATORS

So, in that sense of 1 is with the specification the via point velocity is specified however we assume that t_{naught} is 0. So, in that case, the equation will give 8 equations. So, I assume that the first segment is a a_0 to a_3 as the coefficient, the second segment start from b_0 to b_3 . So, you can see like the x_v and \dot{x}_v . So, these 2 are the second segment initial and initial position and velocity.

So, in the sense that invariantly it is going to you 6 valid equations. But that 6 valid equations we can make it as 8 valid equation with 8 unknowns. So, in that sense, we can calculate this. So, based on this what one can look at it.

(Refer Slide Time: 12:04)



Cubic (Third-order) polynomial
00000

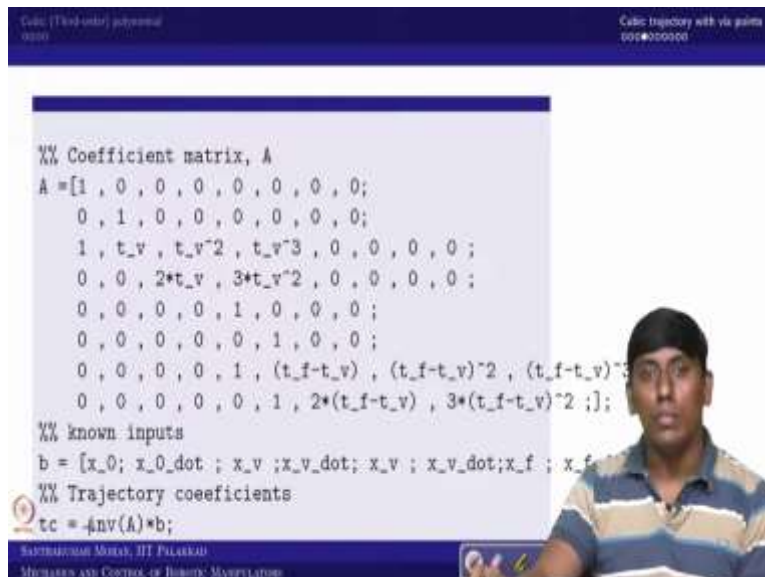
Cubic trajectory with via points
0000000000

```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
t_v = 2; % time for first segment
t_f = 5; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trajectory boundary conditions (known)
x_0 = 0;
x_v = 2;
x_f = 5;
x_0_dot=0;
x_v_dot=1;
x_f_dot=0;
```

SUBRAMANIAM MOHAN, IIT PALARU
MECHANICS AND CONTROL OF ROBOTS MANIPULATORS

So, the code we can rewrite. So, now, we are writing this via point. So, this we have brought it So, via point, you can say position and velocity we have added further than the other one. So, now the via point time also we have given. So, these are the changes which we made from the existing code.

(Refer Slide Time: 12:23)



Cubic (Third-order) polynomial
00000

Cubic trajectory with via points
0000000000

```
%% Coefficient matrix, A
A = [1, 0, 0, 0, 0, 0, 0, 0;
     0, 1, 0, 0, 0, 0, 0, 0;
     1, t_v, t_v^2, t_v^3, 0, 0, 0, 0;
     0, 0, 2*t_v, 3*t_v^2, 0, 0, 0, 0;
     0, 0, 0, 0, 1, 0, 0, 0;
     0, 0, 0, 0, 0, 1, 0, 0;
     0, 0, 0, 0, 1, (t_f-t_v), (t_f-t_v)^2, (t_f-t_v)^3;
     0, 0, 0, 0, 0, 1, 2*(t_f-t_v), 3*(t_f-t_v)^2];

%% known inputs
b = [x_0; x_0_dot; x_v; x_v_dot; x_v; x_v_dot; x_f; x_f_dot];

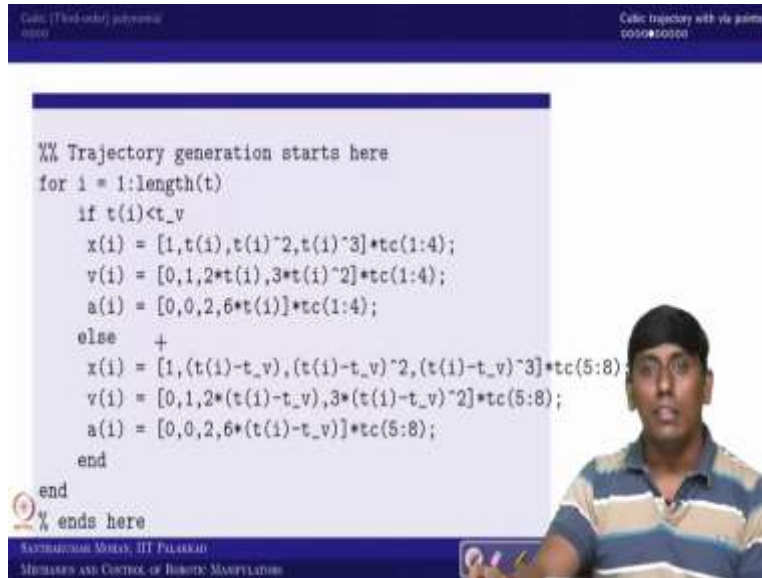
%% Trajectory coefficients
tc = inv(A)*b;
```

SUBRAMANIAM MOHAN, IIT PALARU
MECHANICS AND CONTROL OF ROBOTS MANIPULATORS

And then the coefficient matrix earlier we have seen it is only 4 cross 4. But in this case, I am trying to make it to everything in a single segment. So, it is going to be 8 cross 8. So, further the b vector would be again 8 cross 1. So, that we have added, and we are calculating the coefficient.

Now, the first 4 coefficient would be equivalent to what you call the first segment you can say fifth the term to eighth term would be related to second segment.

(Refer Slide Time: 12:51)



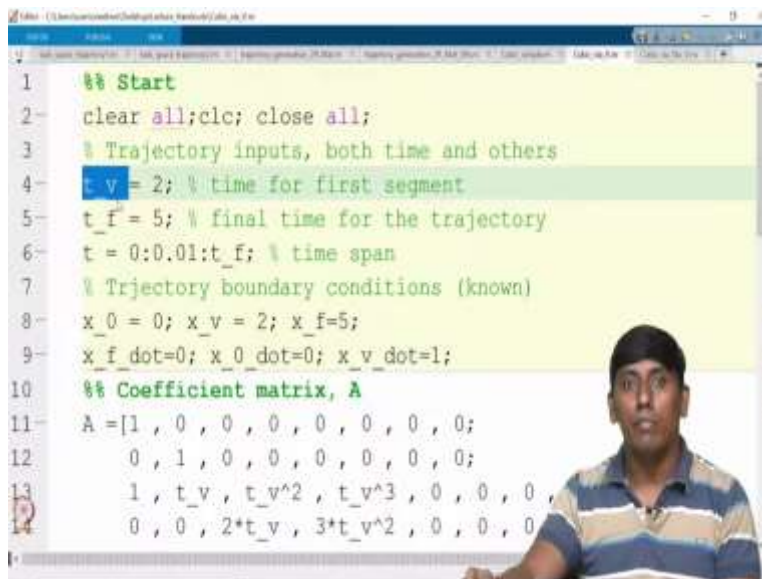
Cubic (Third order) polynomial
Cubic trajectory with via points

```
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<t_v
        x(i) = [1,t(i),t(i)^2,t(i)^3]*tc(1:4);
        v(i) = [0,1,2*t(i),3*t(i)^2]*tc(1:4);
        a(i) = [0,0,2,6*t(i)]*tc(1:4);
    else +
        x(i) = [1,(t(i)-t_v),(t(i)-t_v)^2,(t(i)-t_v)^3]*tc(5:8);
        v(i) = [0,1,2*(t(i)-t_v),3*(t(i)-t_v)^2]*tc(5:8);
        a(i) = [0,0,2,6*(t(i)-t_v)]*tc(5:8);
    end
end
%% ends here
```

ANANTHARAM MOHAN, IIT PALAKKAD
MECHATRONICS AND CONTROL OF ROBOTIC MANIPULATORS

So, that is what you can see the first segment is up to t_v , the second segment is after t_v till t_f . So, the coefficient is 1 to 4 for the first one and 5 to 8 for the second one. So, the remaining all are same. So, the plotting functions and all same.

(Refer Slide Time: 13:12)



1 %% Start
2 clear all;clc; close all;
3 % Trajectory inputs, both time and others
4 t_v = 2; % time for first segment
5 t_f = 5; % final time for the trajectory
6 t = 0:0.01:t_f; % time span
7 % Trajectory boundary conditions (known)
8 x_0 = 0; x_v = 2; x_f=5;
9 x_f_dot=0; x_0_dot=0; x_v_dot=1;
10 %% Coefficient matrix, A
11 A = [1, 0, 0, 0, 0, 0, 0, 0;
12 0, 1, 0, 0, 0, 0, 0, 0;
13 1, t_v, t_v^2, t_v^3, 0, 0, 0, 0;
14 0, 0, 2*t_v, 3*t_v^2, 0, 0, 0, 0

```
10 %% Coefficient matrix, A
11 A = [1, 0, 0, 0, 0, 0, 0, 0;
12      0, 1, 0, 0, 0, 0, 0, 0;
13      1, t_v, t_v^2, t_v^3, 0, 0, 0, 0;
14      0, 0, 2*t_v, 3*t_v^2, 0, 0, 0, 0;
15      0, 0, 0, 0, 1, 0, 0, 0;
16      0, 0, 0, 0, 0, 1, 0, 0;
17      0, 0, 0, 0, 1, (t_f-t_v), (t_f-t_v)^2, (t_f-t_v)^3;
18      0, 0, 0, 0, 0, 1, 2*(t_f-t_v), 3*(t_f-t_v)^2];
19 %% known inputs
20 b = [x_0; x_0_dot; x_v; x_v_dot; x_v; x_v_dot; x_f; x_f_dot];
21 %% Trajectory coefficients
22 tc = inv(A)*b;
23 %% Trajectory generation starts here
```

```
13      1, t_v, t_v^2, t_v^3, 0, 0, 0, 0;
14      0, 0, 2*t_v, 3*t_v^2, 0, 0, 0, 0;
15      0, 0, 0, 0, 1, 0, 0, 0;
16      0, 0, 0, 0, 0, 1, 0, 0;
17      0, 0, 0, 0, 1, (t_f-t_v), (t_f-t_v)^2, (t_f-t_v)^3;
18      0, 0, 0, 0, 0, 1, 2*(t_f-t_v), 3*(t_f-t_v)^2];
19 %% known inputs
20 b = [x_0; x_0_dot; x_v; x_v_dot; x_v; x_v_dot; x_f; x_f_dot];
21 %% Trajectory coefficients
22 tc = inv(A)*b;
23 %% Trajectory generation starts here
24 for i = 1:length(t)
25     if t(i)<t_v
26         x(i) = [1,t(i),t(i)^2,t(i)^3]*tc
```



```
22- tc = inv(A)*b;
23- %% Trajectory generation starts here
24- for i = 1:length(t)
25-     if t(i)<t_v
26-         x(i) = [1,t(i),t(i)^2,t(i)^3]*tc(1:4);
27-         v(i) = [0,1,2*t(i),3*t(i)^2]*tc(1:4);
28-         a(i) = [0,0,2,6*t(i)]*tc(1:4);
29-     else
30-         x(i) = [1,(t(i)-t_v),(t(i)-t_v)^2,(t(i)-t_v)^3]*tc(5:8);
31-         v(i) = [0,1,2*(t(i)-t_v),3*(t(i)-t_v)^2]*tc(5:8);
32-         a(i) = [0,0,2,6*(t(i)-t_v)]*tc(5:8);
33-     end
34- end % ends here
35- %% Plotting the trajectory
```



```
28-         a(i) = [0,0,2,6*t(i)]*tc(1:4);
29-     else
30-         x(i) = [1,(t(i)-t_v),(t(i)-t_v)^2,(t(i)-t_v)^3]*tc(5:8);
31-         v(i) = [0,1,2*(t(i)-t_v),3*(t(i)-t_v)^2]*tc(5:8);
32-         a(i) = [0,0,2,6*(t(i)-t_v)]*tc(5:8);
33-     end
34- end % ends here
35- %% Plotting the trajectory
36- plot(t,x,'b-',t,v,'r--',t,a,'g-','linewidth',2);
37- l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{x}(t)$');
38- set(l,'interpreter','latex','location','northwest','FontSize',14);
39- grid on
40- xlabel('$t$, [s]', 'Interpreter','latex','FontSize',14);
41- ylabel('$\eta(t)$, [units]', 'Interpreter','latex','FontSize',14);
```



```

25     if t(i)<t_v
26         x(i) = [1,t(i),t(i)^2,t(i)^3]*tc(1:4);
27         v(i) = [0,1,2*t(i),3*t(i)^2]*tc(1:4);
28         a(i) = [0,0,2,6*t(i)]*tc(1:4);
29     else
30         x(i) = [1,(t(i)-t_v),(t(i)-t_v)^2,(t(i)-t_v)^3]*tc
31         v(i) = [0,1,2*(t(i)-t_v),3*(t(i)-t_v)^2]*tc(5:8);
32         a(i) = [0,0,2,6*(t(i)-t_v)]*tc(5:8);
33     end
34 end% ends here
35 %% Plotting the trajectory
36 plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth',1);
37 l = legend('$x(t)$','$\dot{x}(t)$','$\ddot{x}(t)$');
38 set(l,'interpreter','latex','location','northwest');

```

```

29     else
30         x(i) = [1,(t(i)-t_v),(t(i)-t_v)^2,(t(i)-t_v)^3]*tc
31         v(i) = [0,1,2*(t(i)-t_v),3*(t(i)-t_v)^2]*tc(5:8);
32         a(i) = [0,0,2,6*(t(i)-t_v)]*tc(5:8);
33     end
34 end% ends here
35 %% Plotting the trajectory
36 plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth',1);
37 l = legend('$x(t)$','$\dot{x}(t)$','$\ddot{x}(t)$');
38 set(l,'interpreter','latex','location','northwest','fontSize',14);
39 grid on
40 xlabel('$t$, [s]','Interpreter','latex','FontSize',14);
41 ylabel('$\eta(t)$, [units]','Interpreter','latex','FontSize',14);
42 %% End

```

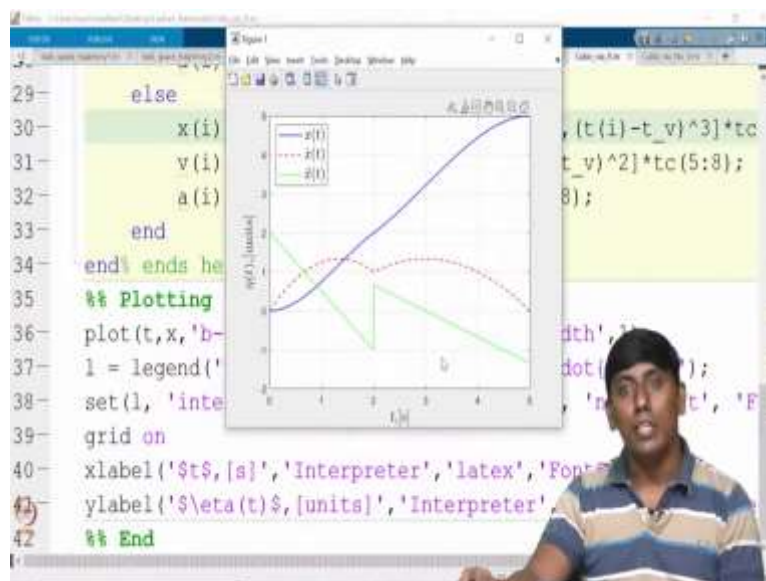
So, now, in that case so, we will take the velocity specified. So, we can see this is the function which we have written in the even slide. So, you can see now the via point you can say given. So, in that sense the via point time or you can say when the first segment supposed to be end. So, that point is given, and the second segment is end 5 second from 0. So, then you can see like we have added the via point position and via point velocity.

So, here I just want to give you can say nonzero velocity you can see how it comes. So, after that you can see like this is the coefficient a matrix which we have shown in the slide, and this is a b vector. So, which is again we have written and the coefficient we have calculated. And now, you

can see the first segment is coefficient of a 0 to a 3 which is t c 1 to 4 and the second segment is b 0 to b 3 which is t c 5 to 8.

So, here the second segment start from t v till t f. So, that is why the t f of i minus t v would be equivalent of that second segment. So, indirectly you can see this is first segment separately we have taken a cubic polynomial and the second segment again another cubic polynomial and we stitch that is what we are done. So, in that sense if we run this code so, you can get clear so, we will run this.

(Refer Slide Time: 14:40)



So, it starts from you can say 0 and 5, but there is a via point. So, you can see like this is with velocity specified. So, where the velocity is specified to 1 meter per second for example, if we consider as position linear position this is equal into 1 meter per second. So, the correspondingly it moved 2 meter and the second segment it is moved next 3 meter. So, the velocity is modified. So, initial and final velocity we have given 0 you can see like there is a jump in the velocity however, it is smooth.

But if you look at the acceleration it is sudden you can say jerk in this sense it is non smooth. So, the first segment end with acceleration is somewhere here and the second segment the acceleration starts from here. So, this is unexpected that is why we said it is non smooth. If your actuator is capable enough then we can go with this otherwise we have to go for something like with the no velocity specified.

(Refer Slide Time: 15:42)

The top screenshot shows a MATLAB script with the following code:

```
1 %% Start
2 clear all;clc; close all;
3 % Trajectory inputs, both time and others
4 t_v = 3; % time for first segment
5 t_f = 5; % final time for the trajectory
6 t = 0:0.01:t_f; % time span
7 % Trajectory boundary conditions (known)
8 x_0 = 0; x_v = 2; x_f=5;
9 x_f_dot=0; x_0_dot=0; x_v_dot=0.5;
10 %% Coefficient matrix, A
11 A=[1, 0, 0, 0, 0, 0, 0, 0;
12     0, 1, 0, 0, 0, 0, 0, 0;
13     1, t_v, t_v^2, t_v^3, 0, 0, 0, 0;
14     0, 0, 2*t_v, 3*t_v^2, 0, 0, 0, 0;
```

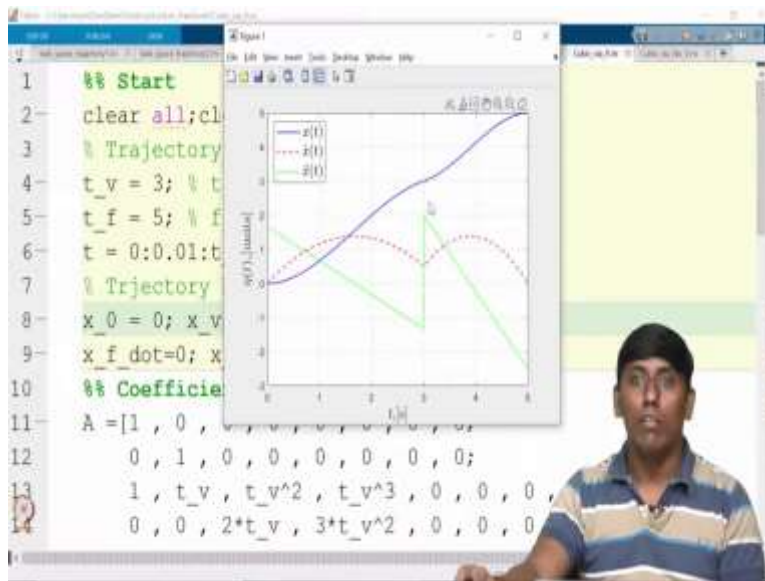
The bottom screenshot shows the same script with a plot of position, velocity, and acceleration over time. The plot has three y-axes: position (m) on the left, velocity (m/s) on the right, and acceleration (m/s^2) on the right. The x-axis is time (s) from 0 to 5. The legend indicates: $x(t)$ (solid blue line), $\dot{x}(t)$ (dashed red line), and $\ddot{x}(t)$ (dotted green line). The position curve starts at 0, increases to 2 at $t=3$, and then increases to 5 at $t=5$. The velocity curve starts at 0, increases to 0.5 at $t=3$, and then increases to 0 at $t=5$. The acceleration curve starts at 0, increases to a peak at $t=3$, and then decreases to 0 at $t=5$.

So, even just for iteration we can modify. For example, this via is I modified to 3 and the velocity at the via point is I put it 0.1 and the via point is only 1 just I am giving an idea. So, in the sense first you can say 3 second it moved only 1 meter distance and the next 2 second it moved up to like 4 meter it is quite unrealistic still we can see how it goes you can see the velocity, or you can say the cubic polynomial it is fast 3 second it is a cubic very smooth.

Then it is another smooth the velocity also like going like this and then going however, the acceleration you can see it is very small, but suddenly jerked because you have given the second profile. So, now even for more realistic.

(Refer Slide Time: 16:36)

```
1 %% Start
2 clear all;clc; close all;
3 % Trajectory inputs, both time and others
4 t_v = 3; % time for first segment
5 t_f = 5; % final time for the trajectory
6 t = 0:0.01:t_f; % time span
7 % Trajectory boundary conditions (known)
8 x_0 = 0; x_v = 3; x_f=5;
9 x_f_dot=0; x_0_dot=0; x_v_dot=0.5;
10 %% Coefficient matrix, A
11 A=[1, 0, 0, 0, 0, 0, 0, 0;
12     0, 1, 0, 0, 0, 0, 0, 0;
13     1, t_v, t_v^2, t_v^3, 0, 0, 0, 0;
14     0, 0, 2*t_v, 3*t_v^2, 0, 0, 0, 0;
```



So, I am just putting it this is a 3. And now again I am running this. So, now you can see that this would be giving. So, little different you can see like this giving the segment very explicitly visible. So, this is what we call with via point that to like via point velocity is specified.

(Refer Slide Time: 17:03)

Cubic (Third order) polynomial
0000

Cubic trajectory with via points
0000000000

Case 2: The cubic (third-order) polynomial function with via point's velocity not specified:

$$a_0 = x_0$$

$$a_1 = \dot{x}_0$$

$$a_0 + a_1(t_v - t_0) + a_2(t_v - t_0)^2 + a_3(t_v - t_0)^3 = x_v$$

$$b_0 = x_v \quad (6)$$

$$b_0 + b_1(t_f - t_v) + b_2(t_f - t_v)^2 + b_3(t_f - t_v)^3 = x_f$$

$$b_1 + 2b_2(t_f - t_v) + 3b_3(t_f - t_v)^2 = \dot{x}_f$$

$$a_1 + 2a_2(t_v - t_0) + 3a_3(t_v - t_0)^2 = b_1$$

$$2a_2 + 6a_3(t_v - t_0) = 2b_2$$

SANTOSH K. MOHAN, IIT PALARU
MECHATRONICS AND CONTROL OF ROBOTS MANIPULATORS

So, now coming back to the slide where the via point velocity is not specified what would be the scenario. So, in that case so this would be having 3 you can say consideration. Where you can see like x_0 would be given \dot{x}_0 would be given which is initial position and velocity and via point a position only would be known final you can say position and velocity will be obtainable.

So, from that you will get a 6 equation. But the three you can say consideration which is involved as indirect mode the b_0 equal to x_v it just straightforward. However, the b_1 and b_2 we need to calculate for that what we take the first segment final velocity would be equivalent to the second segment initial velocity in that case if we assume that the second segment is going to give initial velocity, we start from 0 and the first one is end with the t_v .

So, then you can see like you will end up with the equation like this. So, now this equation is added. Similarly, the first segment final acceleration equal to the second segment you can say initial acceleration then the $2b_2$ can be equate like this. So, now what happened the earlier equation is modified. So, now still 8 unknowns and 8 equations.

(Refer Slide Time: 18:25)


Cubic (Third-order) polynomial

Cubic trajectory with via points

Case 2: The cubic (third-order) polynomial function with via point's velocity not specified:
The general situation, where $t_0 = 0$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_v & t_v^2 & t_v^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & (t_r - t_v) & (t_r - t_v)^2 & (t_r - t_v)^3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2(t_r - t_v) & 3(t_r - t_v)^2 \\ 0 & 1 & 2t_v & 3t_v^2 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 6t_v & 0 & 0 & \pm 2 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ x_v \\ \dot{x}_v \\ x_f \\ \dot{x}_f \\ 0 \\ 0 \end{bmatrix} \quad (7)$$

SANTOSH K. MOHAN, IIT PALAKKAD
MECHATRONICS AND CONTROL OF ROBOTS MANIPULATORS



So, we can bring this a matrix slightly modified. So, in this case the via point velocity is not specified we have chosen based on the smoothness of the acceleration.


(Refer Slide Time: 18:38)

Cubic (Third-order) polynomial

Cubic trajectory with via points

```
%%
clear all;clc; close all;
% Trajectory inputs, both time and others
t_v = 2; % time for first segment
t_f = 5; % final time for the trajectory
t = 0:0.1:t_f; % time span
% Trajectory boundary conditions (known)
x_0 = 0;
x_v = 2;
x_f = 5;
x_f_dot = 0;
x_0_dot = 0;
```

SANTOSH K. MOHAN, IIT PALAKKAD
MECHATRONICS AND CONTROL OF ROBOTS MANIPULATORS



So, in that sense what we have done so, these all not changed. So, only thing the via point velocity we have removed from this.


```
13 %% Coefficient matrix, A
14 A = [1, 0, 0, 0, 0, 0, 0, 0;
15      0, 1, 0, 0, 0, 0, 0, 0;
16      1, t_v, t_v^2, t_v^3, 0, 0, 0, 0;
17      0, 0, 0, 0, 1, 0, 0, 0;
18      0, 0, 0, 0, 1, (t_f-t_v), (t_f-t_v)^2, (t_f-t_v)^3;
19      0, 0, 0, 0, 0, 1, 2*(t_f-t_v), 3*(t_f-t_v)^2;
20      0, 1, 2*t_v, 3*t_v^2, 0, -1, 0, 0;
21      0, 0, 2, 6*t_v, 0, 0, -2, 0];
22 %% known inputs
23 b = [x_0; x_0_dot; x_v; x_v; x_f; x_f_dot];
24 %% Trajectory coefficients
25 tc = inv(A)*b;
26 %% Trajectory generation starts here
```

```
19      0, 0, 0, 0, 0, 1, 2*(t_f-t_v), 3*(t_f-t_v)^2;
20      0, 1, 2*t_v, 3*t_v^2, 0, -1, 0, 0;
21      0, 0, 2, 6*t_v, 0, 0, -2, 0];
22 %% known inputs
23 b = [x_0; x_0_dot; x_v; x_v; x_f; x_f_dot; 0; 0];
24 %% Trajectory coefficients
25 tc = inv(A)*b;
26 %% Trajectory generation starts here
27 for i = 1:length(t)
28     if t(i) < t_v
29         x(i) = [1, t(i), t(i)^2, t(i)^3]*tc(1:4);
30         v(i) = [0, 1, 2*t(i), 3*t(i)^2]*tc(1:4);
31         a(i) = [0, 0, 2, 6*t(i)]*tc(1:4);
32     else
```



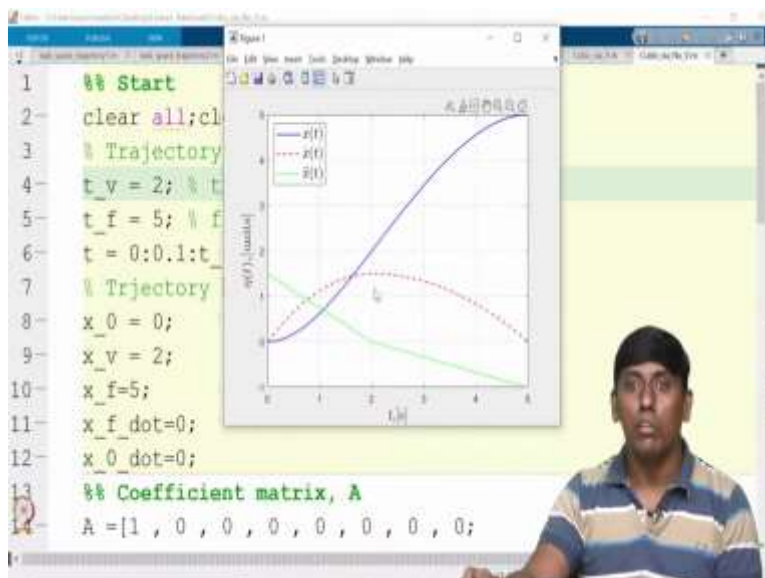
```

25- tc = inv(A)*b;
26- %% Trajectory generation starts here
27- for i = 1:length(t)
28-     if t(i)<t_v
29-         x(i) = [1,t(i),t(i)^2,t(i)^3]*tc(1:4);
30-         v(i) = [0,1,2*t(i),3*t(i)^2]*tc(1:4);
31-         a(i) = [0,0,2,6*t(i)]*tc(1:4);
32-     else
33-         x(i) = [1,(t(i)-t_v),(t(i)-t_v)^2,(t(i)-t_v)^3]*tc
34-         v(i) = [0,1,2*(t(i)-t_v),3*(t(i)-t_v)^2]*tc(5:8);
35-         a(i) = [0,0,2,6*(t(i)-t_v)]*tc(5:8);
36-     end
37- end
38- %% ends here

```

So, then you can find it so, this is what the no velocity specified. So, you can see like now the velocity is not specified for the via point. So, now, A matrix is calculated based on this. So, then the b vector and the trajectory coefficients are actually calculated in this manner. And then we can end up with so, these all. So, now if we run the same thing what we have seen earlier case. So, we say that the final velocity is 0 initial velocity also 0 and it is the middle point is 2 meter is the displacement and it takes for 2 seconds. So, we will see how this happening.

(Refer Slide Time: 19:40)

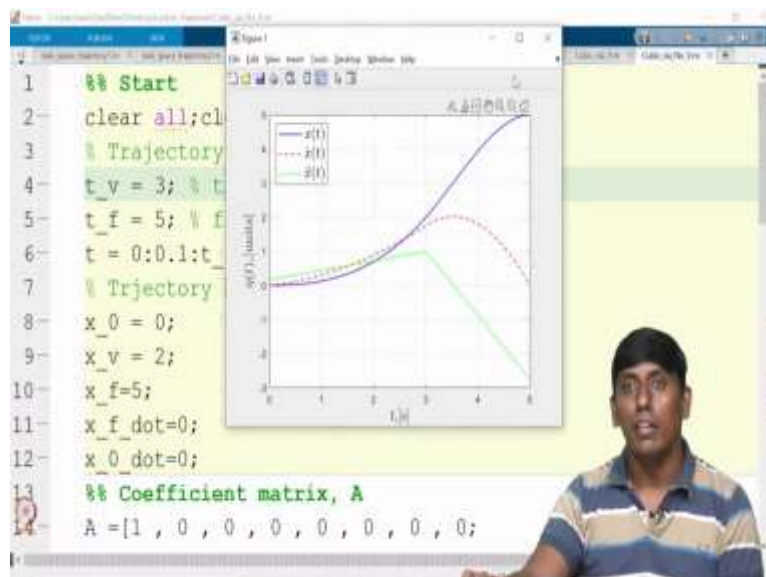


So, now you can see like the acceleration is smooth, earlier you would have seen it is a; you can see arbitrarily mode but in this case, it is very smooth. So, now this is what we have actually

tried to try to do. Again, here you can see the velocities you can see the first segment end velocity. Which is equal to the second segment initial velocity that is what you can look at it.

(Refer Slide Time: 20:07)

```
1 %% Start
2 clear all;clc; close all;
3 % Trajectory inputs, both time and others
4 t_v = 3; % time for first segment
5 t_f = 5; % final time for the trajectory
6 t = 0:0.1:t_f; % time span
7 % Trajectory boundary conditions (known)
8 x_0 = 0;
9 x_v = 2;
10 x_f=5;
11 x_f_dot=0;
12 x_0_dot=0;
13 %% Coefficient matrix, A
14 A =[1, 0, 0, 0, 0, 0, 0, 0, 0;
```



So, just for giving clarity I just make it the second is 3. So, that I can show the; you can say clearly distinguish. So, now you can see it right? So, now the velocity is it like at 3 second you can see like that is the initial and the final.

(Refer Slide Time: 20:25)

The image displays two screenshots of a MATLAB script and its execution. The top screenshot shows the script defining initial conditions and a coefficient matrix A. The bottom screenshot shows the same script with a plot of the trajectory x(t) overlaid.

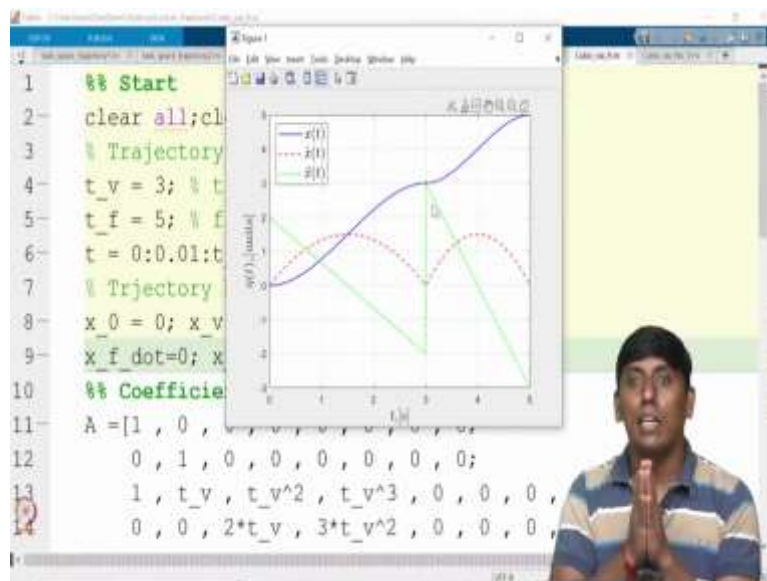
```
9 x_v = 3;
10 x_f=5;
11 x_f_dot=0;
12 x_0_dot=0;
13 %% Coefficient matrix, A
14 A = [1, 0, 0, 0, 0, 0, 0, 0, 0;
15       0, 1, 0, 0, 0, 0, 0, 0, 0;
16       1, t_v, t_v^2, t_v^3, 0, 0, 0, 0, 0;
17       0, 0, 0, 0, 1, 0, 0, 0, 0;
18       0, 0, 0, 0, 1, (t_f-t_v), (t_f-t_v)^2, (t_f-t_v)^3, 0;
19       0, 0, 0, 0, 0, 1, 2*(t_f-t_v), 3*(t_f-t_v)^2, 0;
20       0, 1, 2*t_v, 3*t_v^2, 0, -1, 0, 0, 0];
```

The bottom screenshot shows the same script with a plot of the trajectory x(t) overlaid. The plot shows three curves: x(t) (solid blue line), x_dot(t) (dashed red line), and x_double_dot(t) (dotted green line). The x-axis is labeled 't(s)' and ranges from 0 to 5. The y-axis is labeled 'x(t), x_dot(t), x_double_dot(t)' and ranges from -2 to 6. The x(t) curve starts at 0 and increases to approximately 5.5 at t=5. The x_dot(t) curve starts at 3 and decreases to approximately 0.5 at t=5. The x_double_dot(t) curve starts at 0 and decreases to approximately -1 at t=5.

So, even I will just make it this. So, I will just do all iterations. So, we can see how this is happening. So, now I gave 3 seconds, and you can see like there is no you can say segment is happening. So, it is much more smooth everything is smooth. So, this is what we call with you can say via point but no velocity specified. So, in order to compare even you can run the same condition.

(Refer Slide Time: 20:57)

```
1 %% Start
2 clear all;clc; close all;
3 % Trajectory inputs, both time and others
4 t_v = 3; % time for first segment
5 t_f = 5; % final time for the trajectory
6 t = 0:0.01:t_f; % time span
7 % Trajectory boundary conditions (known)
8 x_0 = 0; x_v = 3; x_f=5;
9 x_f_dot=0; x_0_dot=0; x_v_dot=0;
10 %% Coefficient matrix, A
11 A=[1, 0, 0, 0, 0, 0, 0, 0;
12     0, 1, 0, 0, 0, 0, 0, 0;
13     1, t_v, t_v^2, t_v^3, 0, 0, 0, 0;
14     0, 0, 2*t_v, 3*t_v^2, 0, 0, 0, 0;
```



```

1 %% Start
2 clear all;clc; close all;
3 % Trajectory inputs, both time and others
4 t_v = 3; % time for first segment
5 t_f = 5; % final time for the trajectory
6 t = 0:0.1:t_f; % time span
7 % Trajectory boundary conditions (known)
8 x_0 = 0;
9 x_v = 3;
10 x_f=5;
11 x_f_dot=0;
12 x_0_dot=0;
13 %% Coefficient matrix, A
14 A =[1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0;

```

```

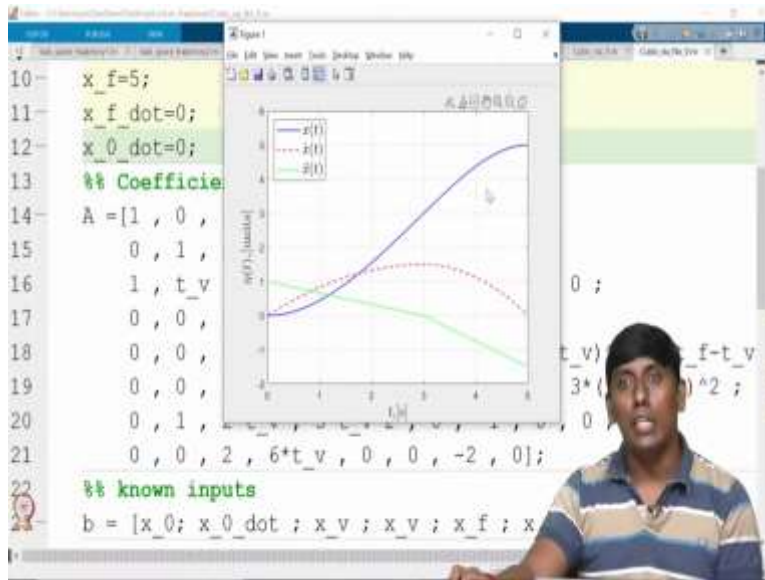
10 x_f=5;
11 x_f_dot=0;
12 x_0_dot=0; I
13 %% Coefficient matrix, A
14 A =[1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0;
15     0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0;
16     1 , t_v , t_v^2 , t_v^3 , 0 , 0 , 0 , 0 , 0 ;
17     0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 ;
18     0 , 0 , 0 , 0 , 1 , (t_f-t_v) , (t_f-t_v)^2 , (t_f-t_v)^3 , 0 ;
19     0 , 0 , 0 , 0 , 0 , 1 , 2*(t_f-t_v) , 3*(t_f-t_v)^2 ;
20     0 , 1 , 2*t_v , 3*t_v^2 , 0 , -1 , 0 , 0 , 0 ;
21     0 , 0 , 2 , 6*t_v , 0 , 0 , -2 , 0];
22 %% known inputs
23 b = [x_0; x_0_dot ; x_v ; x_v ; x_f ; x_f_dot ;

```

So, but only issue is you do not have any velocity specified. So, just for benefit even I assume that this velocity is 0 it is very tricky. So, because velocity 0 means it will come 0 and 0. So, we can try so I am not taking away. So, we can see how it goes. So, you can see like it start from 0 and the end again in the first segment 0 and the second segment is actually again 0. So, you can see like it is very clearly say that it is explicitly two segments.

So, now, we say the same case here. So, which is like the 3 is the segment and you can see there is no via point velocity the initial and final velocity only we assume 0. So, now, in that case you can see like the profile is more close like a single cubic polynomial.

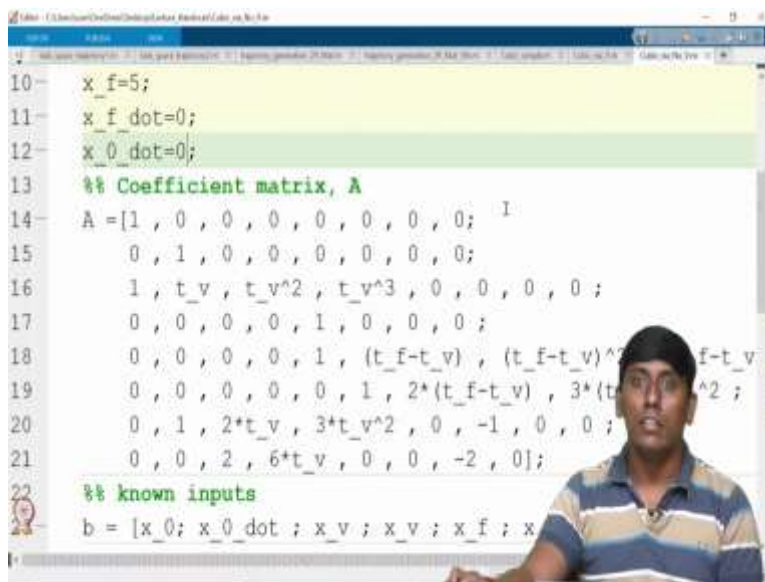
(Refer Slide Time: 21:46)



The top image shows a MATLAB script and its corresponding plot. The script defines a system with a step input $x_f=5$ and initial conditions $x_{f_dot}=0$ and $x_{0_dot}=0$. The coefficient matrix A is defined as:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_v & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & (t_f - t_v) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2(t_f - t_v) & 3(t_f - t_v)^2 \\ 0 & 1 & 2t_v & 3t_v^2 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 6t_v & 0 & 0 & -2 & 0 \end{bmatrix}$$

The known inputs are defined as $b = [x_0; x_{0_dot}; x_v; x_v; x_f; x_v; x_v; x_v]$. The plot shows the response of the system over time, with three curves: $x(t)$ (solid blue line), $\dot{x}(t)$ (dashed red line), and $\ddot{x}(t)$ (solid green line).



The bottom image shows a MATLAB script defining a system with a polynomial input. The coefficient matrix A is defined as:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & t_v & t_v^2 & t_v^3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & (t_f - t_v) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2(t_f - t_v) & 3(t_f - t_v)^2 \\ 0 & 1 & 2t_v & 3t_v^2 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 6t_v & 0 & 0 & -2 & 0 \end{bmatrix}$$

The known inputs are defined as $b = [x_0; x_{0_dot}; x_v; x_v; x_f; x_v; x_v; x_v]$.

```
19     0, 0, 0, 0, 0, 1, 2*(t_f-t_v), 3*(t_f-t_v)^2 ;
20     0, 1, 2*t_v, 3*t_v^2, 0, -1, 0, 0 ;
21     0, 0, 2, 6*t_v, 0, 0, -2, 0];
22 %% known inputs
23 b = [x_0; x_0_dot; x_v; x_v; x_f; x_f_dot; 0; 0];
24 %% Trajectory coefficients
25 tc = inv(A)*b;
26 %% Trajectory generation starts here
27 for i = 1:length(t)
28     if t(i)<t_v
29         x(i) = [1,t(i),t(i)^2,t(i)^3]*tc(1:4);
30         v(i) = [0,1,2*t(i),3*t(i)^2]*tc(1:4);
31         a(i) = [0,0,2,6*t(i)]*tc(1:4);
32     else
```

So, you can see like there is no segment is visible. So, that is what we can feel it. So, this is what we call you can see cubic polynomial with via point even the same thing can be done for fifth order or some other thing. However, the restriction here we have put it as actually a cubic polynomial is easy. And that is a more commonly used profile generation for the you call robotic manipulator and all.

So, with that we close this particular content here. So, the next lecture we are going to talk about the other profile. Which we have not covered in this particular lecture like higher order polynomial, cycloidal polynomial you can say you call you can say parabolic blend and all. That we would be seeing in the next lecture. So, in that sense so, I hope you are able to follow this. So, with that I am saying thank you and see you then in the next lecture bye, bye.