Mechanics and Control of Robotic Manipulators

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

Lecture No 33

Trajectory generation using MATLAB part 2

Welcome back to Mechanics and Control of Robotic Manipulator. I hope you would have enjoyed in the last lecture where these simulations all come. So, it is actually giving little more intuition. So, I hope you are able to even extract the MATLAB code from the handout and you would have tried at your own.

So, now, in this particular lecture, we are continuing that part where we left. So, we left that we have seen, what is cubic polynomial? How to generate the cubic polynomial in MATLAB even with via point. So, this particular lecture we are going to see how to generate a higher order polynomial with the MATLAB function and similarly cycloidal lines straight line interpolation with parabolic blend.

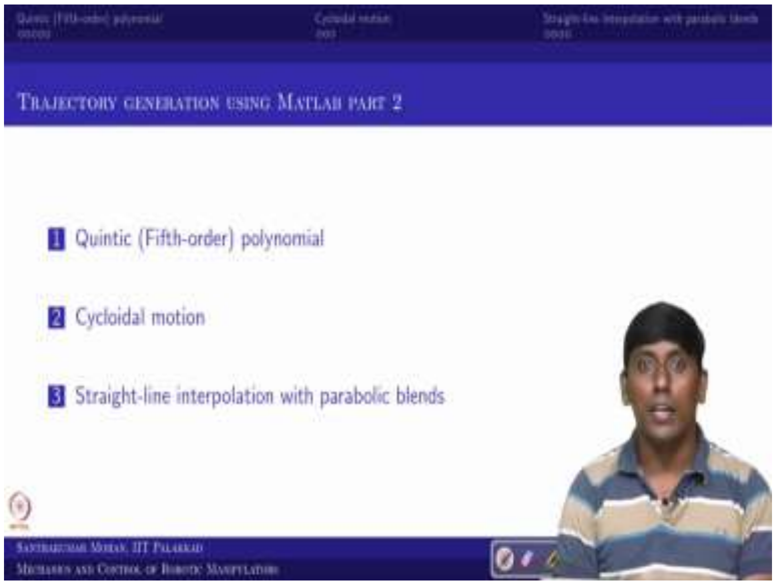
(Refer Slide Time: 0:54)

Quintic (Fifth-order) polynomial
Cycloidal motion
Straight-line interpolation with parabolic blends

Note:

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

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators


Quintic (Fifth-order) polynomial | Cycloidal motion | Straight-line interpolation with parabolic blends

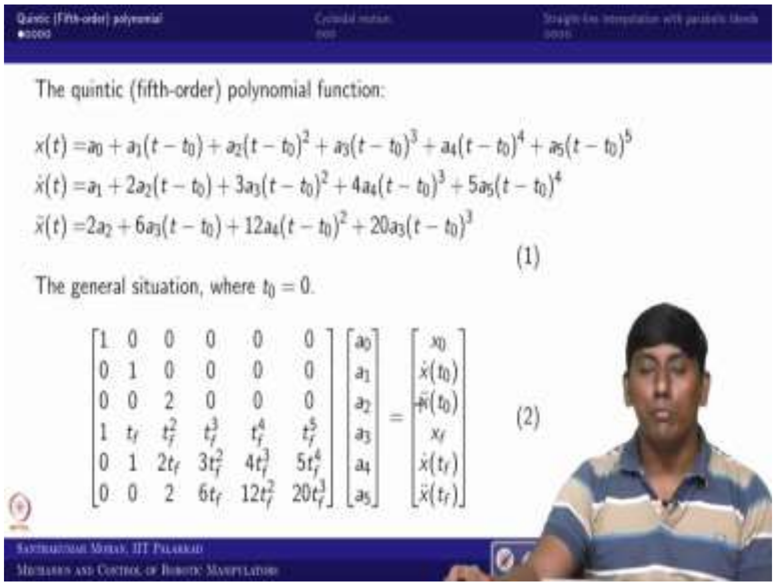
Trajectory generation using Matlab part 2

1. Quintic (Fifth-order) polynomial
2. Cycloidal motion
3. Straight-line interpolation with parabolic blends

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, in that case we will try to see these are the three points. So, where the fifth order polynomial we call quintic. So, which is a Latin word, so, Latin word quintic means fifth. So, similarly the cubic means third. So, this is a way we also like try to see.

(Refer Slide Time: 1:09)


Quintic (Fifth-order) polynomial | Cycloidal motion | Straight-line interpolation with parabolic blends

The quintic (fifth-order) polynomial function:

$$\begin{aligned} x(t) &= a_0 + a_1(t-t_0) + a_2(t-t_0)^2 + a_3(t-t_0)^3 + a_4(t-t_0)^4 + a_5(t-t_0)^5 \\ \dot{x}(t) &= a_1 + 2a_2(t-t_0) + 3a_3(t-t_0)^2 + 4a_4(t-t_0)^3 + 5a_5(t-t_0)^4 \\ \ddot{x}(t) &= 2a_2 + 6a_3(t-t_0) + 12a_4(t-t_0)^2 + 20a_3(t-t_0)^3 \end{aligned} \quad (1)$$

The general situation, where $t_0 = 0$.

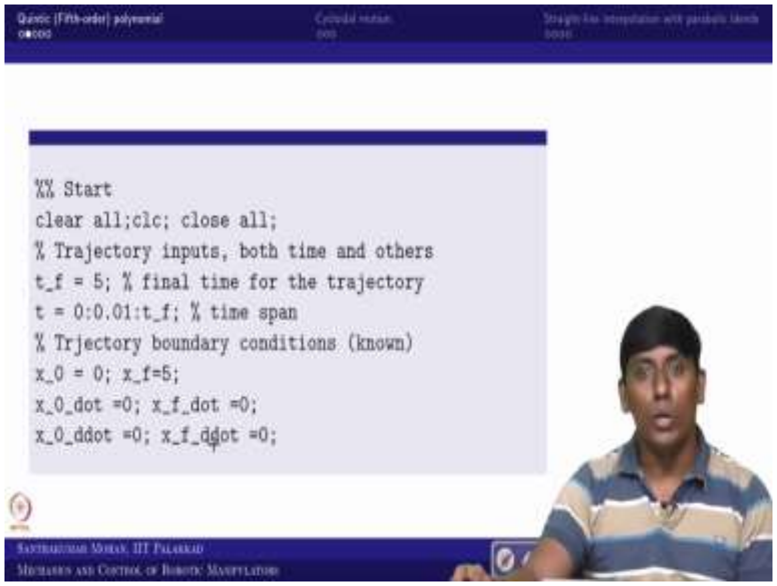
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} x_0 \\ \dot{x}(t_0) \\ \ddot{x}(t_0) \\ x_f \\ \dot{x}(t_f) \\ \ddot{x}(t_f) \end{bmatrix} \quad (2)$$

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, in that sense we will see what equation we have derived in the regular lecture. So, these are the equations which we have derived in the regular lecture. So, this can be portrayed with the initial and final condition you can say here the acceleration also controllable. So, the initial acceleration and final acceleration if you equate. So, there would be 6 unknowns and 6 equation we can generate those 6 equations and 6 unknown we can write it in a matrix and vector form.

So, where a naught to a 5 are unknown. So, x naught 2 x double dot of t f are known, and this is coefficient matrix where inside all the terms are known to us. So, in that sense, we can take this as a and this is the x and this is b. So, we can take x is A inverse of b so, the same principle we will try to do it in MATLAB.

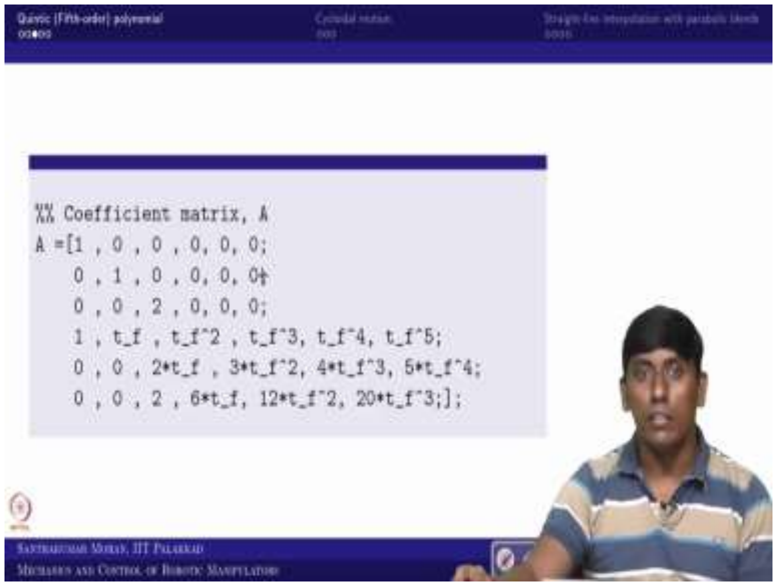
(Refer Slide Time: 1:59)



```
%% 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
% Trjectory boundary conditions (known)
x_0 = 0; x_f=5;
x_0_dot =0; x_f_dot =0;
x_0_ddot =0; x_f_ddot =0;
```

So, for that we are taking the same code. So, we are modifying a certain parameter. So, you can see like this is what we have taken for the cubic polynomial we have taken as a t f 5 second and it is varying as a span where t starting from 0 to t f and the initial condition and final you can say initial position and final position initial velocity and final velocity are known in cubic polynomial. But in this case, so, even the acceleration also known. So, here I have written as d d dot or you can say d dot. So, which means a double dot so these also like given.

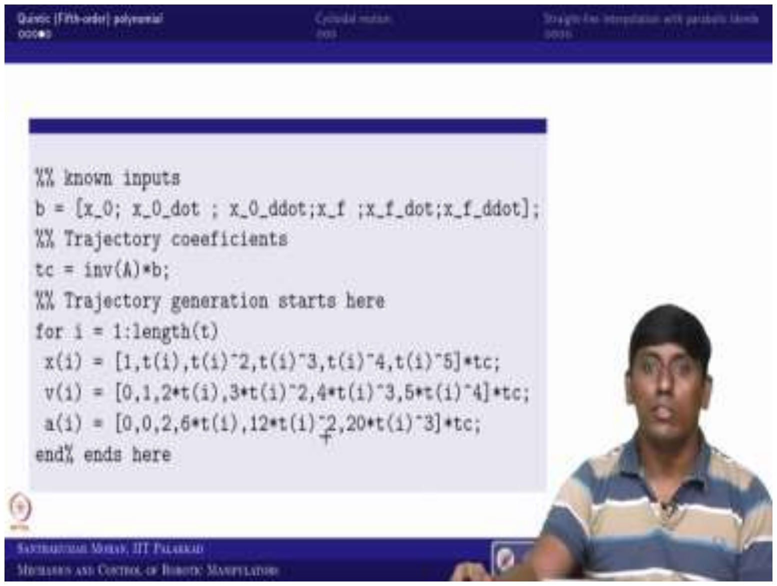
(Refer Slide Time: 2:33)



```
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*t_f^4;
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f^3;];
```

So, based on this we can rewrite the coefficient matrix earlier case it was 4 cross 4. So, now, this is 6 cross 6. So, we have written this.

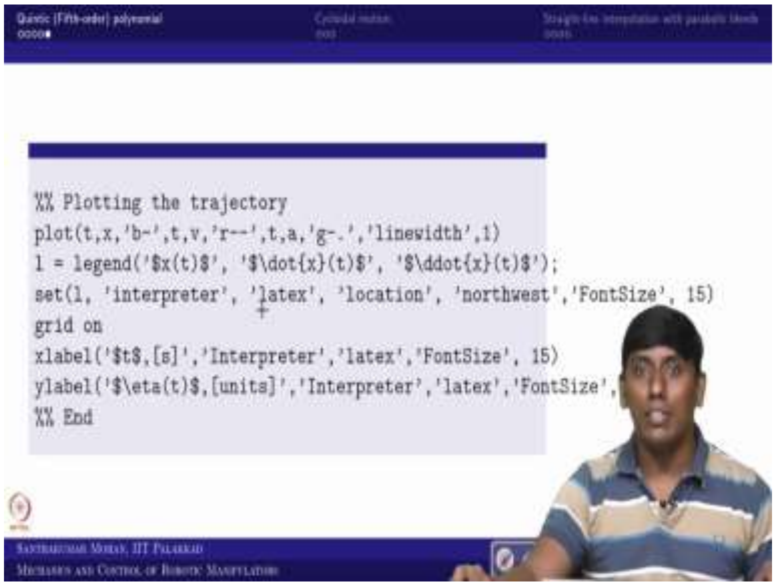
(Refer Slide Time: 2:44)



```
%% known inputs
b = [x_0; x_0_dot ; x_0_ddot;x_f ;x_f_dot;x_f_ddot];
%% Trajectory coeeficients
tc = inv(A)*b;
%% Trajectory generation starts here
for i = 1:length(t)
 x(i) = [1,t(i),t(i)^2,t(i)^3,t(i)^4,t(i)^5]*tc;
 v(i) = [0,1,2*t(i),3*t(i)^2,4*t(i)^3,5*t(i)^4]*tc;
 a(i) = [0,0,2,6*t(i),12*t(i)^2,20*t(i)^3]*tc;
end% ends here
```

So, now based on this so, we can find the coefficients, so, where the b matrix sorry b vector can be given. Then the trajectory coefficient can be calculated. So, then there is all standard protocol. So, you can see like this is the; you can say position this is the velocity and acceleration.

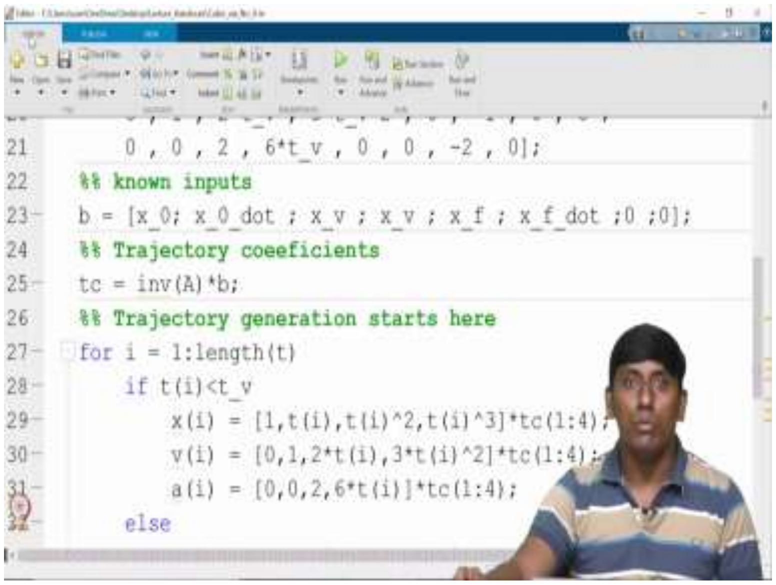
(Refer Slide Time: 3:04)


Quintic (Fifth-order) polynomial
Cycloidal motion
Straight-line interpolation with parabolic blends
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

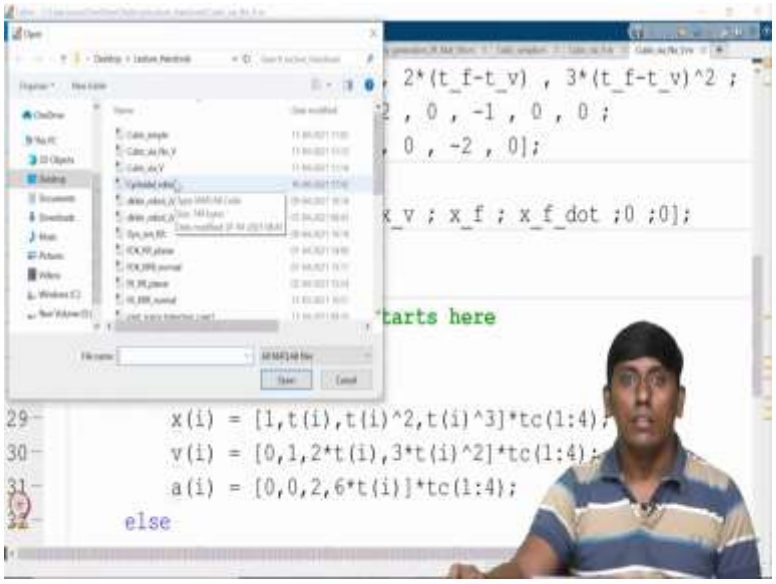
```
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth',1)
l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{x}(t)$');
set(l, 'interpreter', 'latex', 'location', 'northwest','FontSize', 15)
grid on
xlabel('$t$,[s]','Interpreter','latex','FontSize', 15)
ylabel('$\eta(t)$,[units]','Interpreter','latex','FontSize',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

And we can use the same MATLAB function how we have written the plotting function. So, the same thing we can use it. So, now, in order to understand this. So, we are going to the MATLAB window.

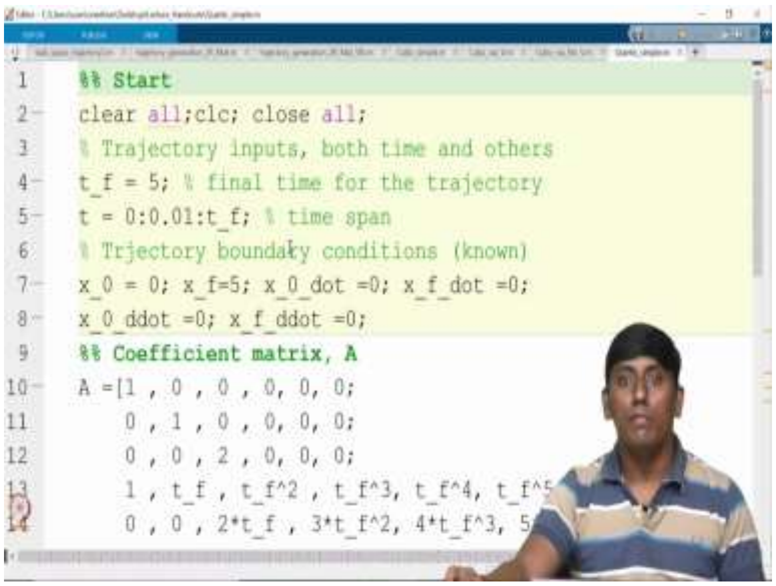
(Refer Slide Time: 3:14)



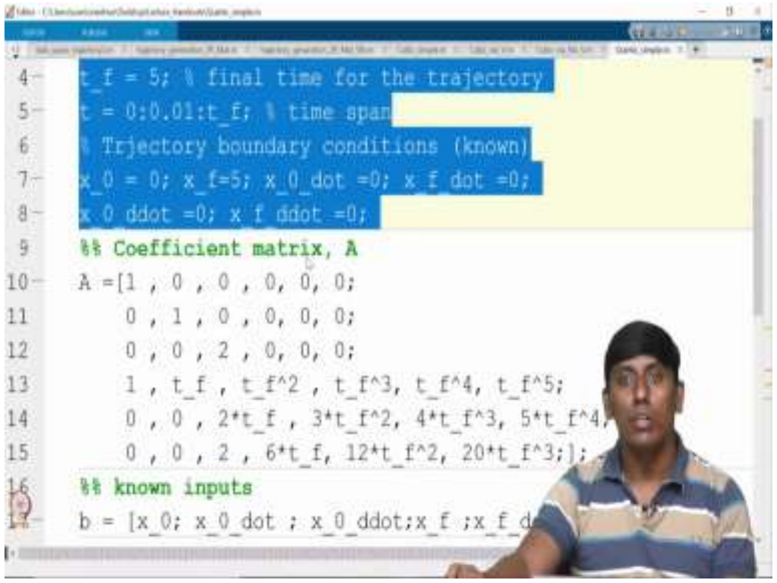
```
        0 , 0 , 2 , 6*t_v , 0 , 0 , -2 , 0];
%% known inputs
b = [x_0; x_0_dot ; x_v ; x_v ; x_f ; x_f_dot ;0 ;0];
%% Trajectory coeeficients
tc = inv(A)*b;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<t_v
        x(i) = [1,t(i),t(i)^2,t(i)^3]*tc(1:4);
        v(i) = [0,1,2*t(i),3*t(i)^2]*tc(1:4);
        a(i) = [0,0,2,6*t(i)]*tc(1:4);
    else
```



```
, 2*(t_f-t_v) , 3*(t_f-t_v)^2 ;
2 , 0 , -1 , 0 , 0 ;
, 0 , -2 , 0];

x_v ; x_f ; x_f_dot ;0 ;0];

tarts here

        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
```



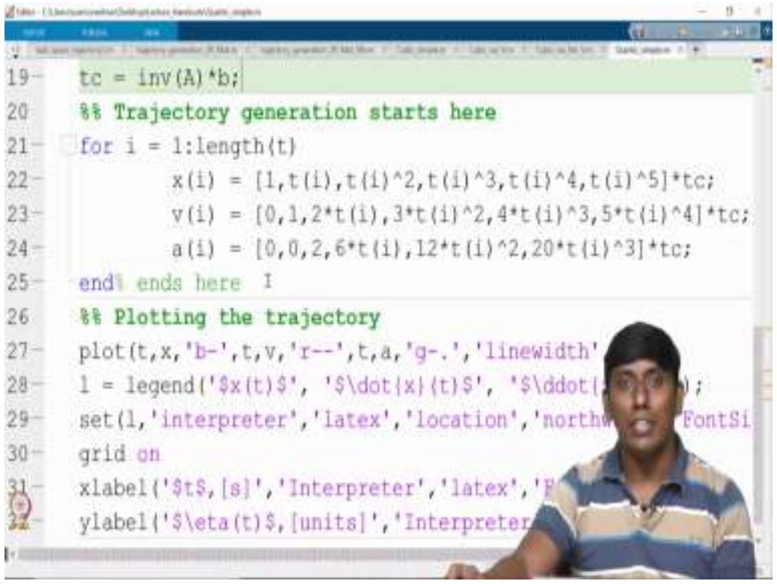
```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
t_f = 5; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=5; x_0_dot =0; x_f_dot =0;
x_0_ddot =0; x_f_ddot =0;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5
```



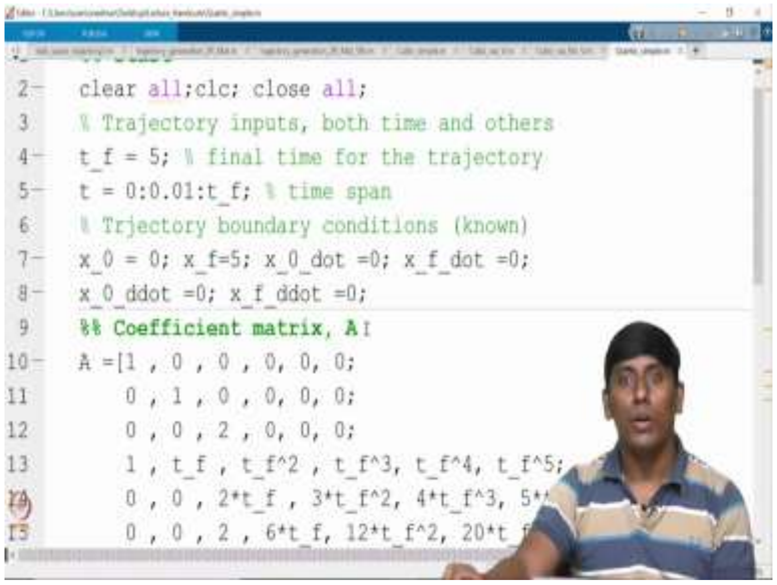
```
t_f = 5; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=5; x_0_dot =0; x_f_dot =0;
x_0_ddot =0; x_f_ddot =0;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*t_f^4;
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f^3;];
%% known inputs
b = [x_0; x_0_dot ; x_0_ddot;x_f ;x_f_d
```

And I am trying to show that so, I think the fifth order polynomial. So, this is the fifth order polynomial. So, where you can see like these all; what we have written in the you can say handout. And this is A matrix we have written in the same you can same handout so, we have not changed and b vector we have assumed to be known these all initial and final conditions including position velocity and acceleration all known. So, we have calculated the coefficients.

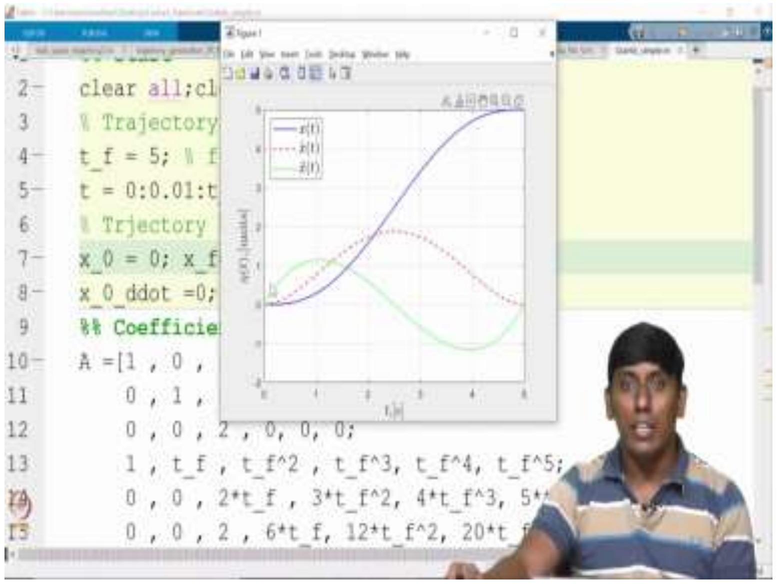
(Refer Slide Time: 3:59)



```
tc = inv(A)*b;
%% Trajectory generation starts here
for i = 1:length(t)
        x(i) = [1,t(i),t(i)^2,t(i)^3,t(i)^4,t(i)^5]*tc;
        v(i) = [0,1,2*t(i),3*t(i)^2,4*t(i)^3,5*t(i)^4]*tc;
        a(i) = [0,0,2,6*t(i),12*t(i)^2,20*t(i)^3]*tc;
end% ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth'
l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{
set(l,'interpreter','latex','location','northw
grid on
xlabel('$t$,[s]','Interpreter','latex','F
ylabel('$\eta(t)$,[units]','Interpreter
```



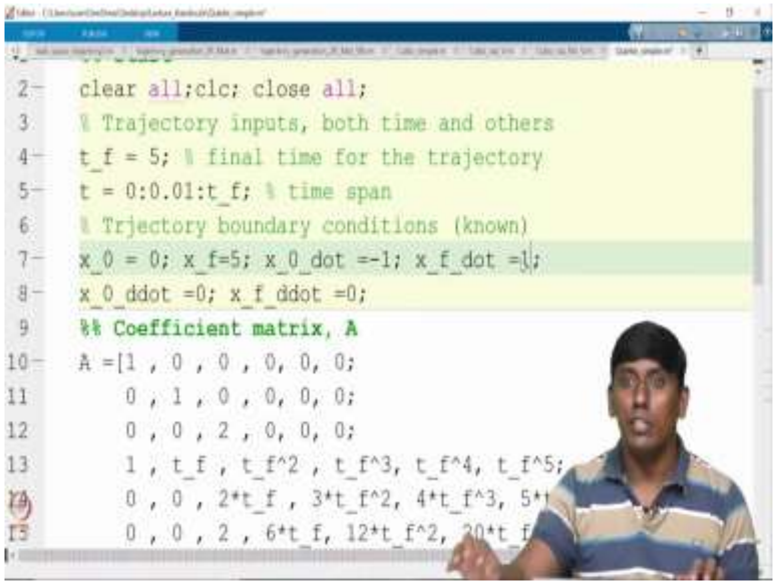
```
clear all;clc; close all;
% Trajectory inputs, both time and others
t_f = 5; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=5; x_0_dot =0; x_f_dot =0;
x_0_ddot =0; x_f_ddot =0;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_
```



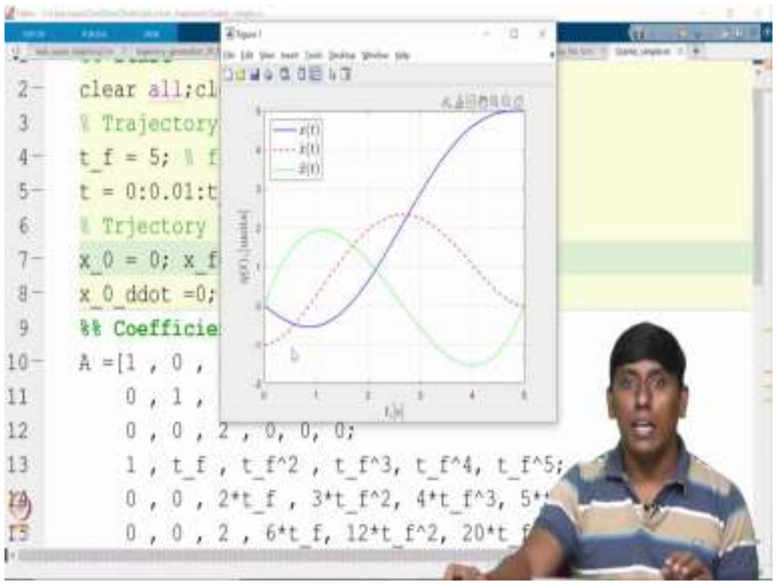
```
clear all;cl
% Trajectory
t_f = 5; % f
t = 0:0.01:t
% Trjectory
x_0 = 0; x_f
x_0_ddot =0;
%% Coefficie
A =[1 , 0 ,
    0 , 1 ,
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_
```

So, then we have generated the trajectory and we are actually trying to plot. So, now, we will take the same case. So, where it is start from 0 and it ends with the 5 with the 5 second. So, we can try to see. So, how it is going it is much more smooth. You can see the acceleration earlier case it was you can see a nonzero initial and final acceleration. But in this case, I have given 0 0 initial and final acceleration. So, you can see like it is a very smooth curve. So, even you want to make it much more interesting fact.

(Refer Slide Time: 4:35)



```
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
% Trjectory boundary conditions (known)
x_0 = 0; x_f=5; x_0_dot =-1; x_f_dot =1;
x_0_ddot =0; x_f_ddot =0;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```

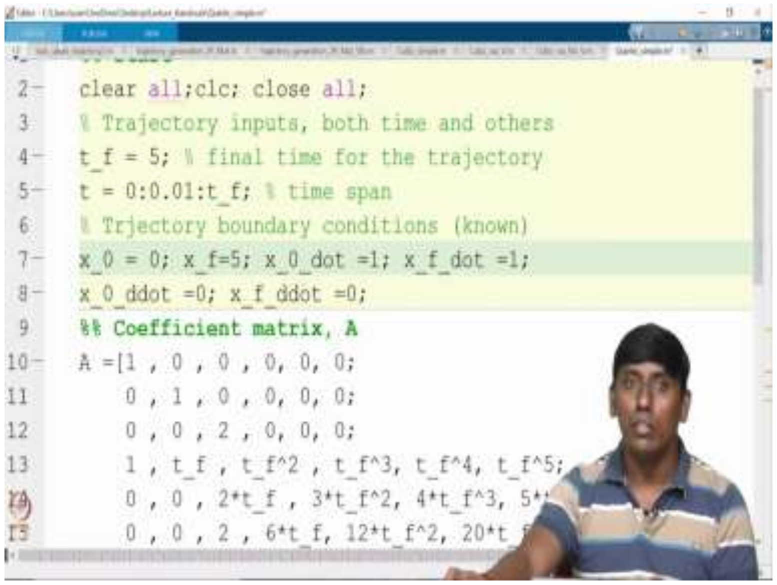


```
clear all;cl
% Trajectory
t_f = 5; % f
t = 0:0.01:t
% Trjectory
x_0 = 0; x_f
x_0_ddot =0;
%% Coefficie
A =[1 , 0 ,
    0 , 1 ,
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```

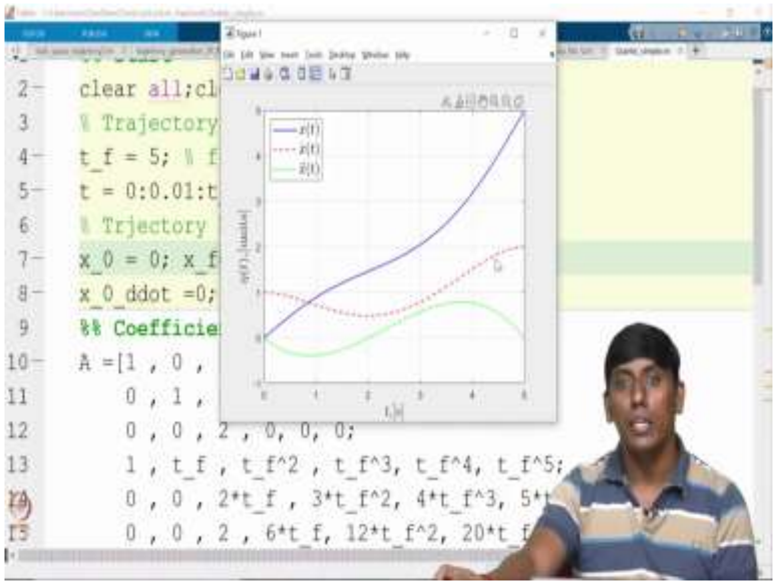
For example, I said that this is so minus 1 is the initial velocity and final velocity is 1 which we have tried in the cubic polynomial. The same thing I am trying to show here. So, you can see like it is start still the acceleration profile is smooth it is more close to a sinusoidal curve. It is not strictly sinusoidal; it is like a smooth it is very close to sinusoidal. So, that is what the sinusoidal anyhow we are going to see in cycloidal.

So, this is what the curve since it is nonzero initial velocity that is why it is going the way so, that to like it is starting from what you call minus 1 meter per second. So that is why it is going down and go. However, you can see like the maximum velocity it is coming the same midpoint which is 2.5 second. So, that particular point you can see the acceleration also like 0.

(Refer Slide Time: 5:33)

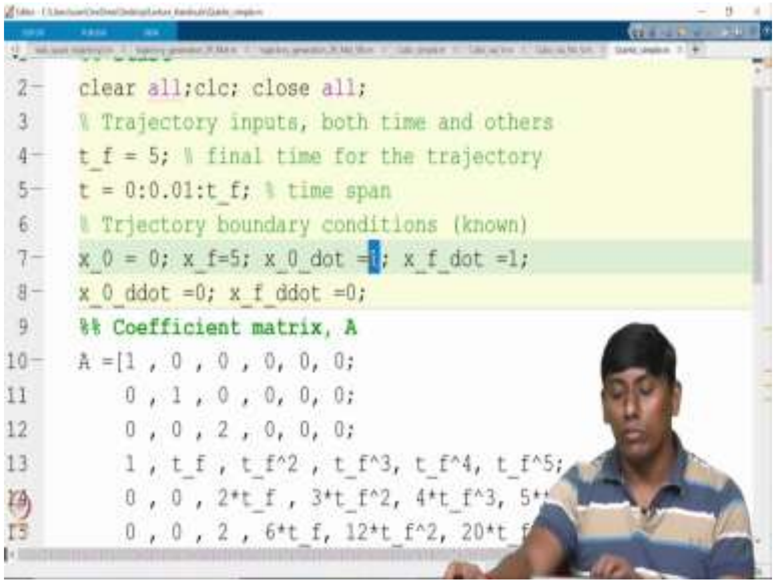


```
clear all;clc; close all;
% Trajectory inputs, both time and others
t_f = 5; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=5; x_0_dot =1; x_f_dot =1;
x_0_ddot =0; x_f_ddot =0;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*t
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```

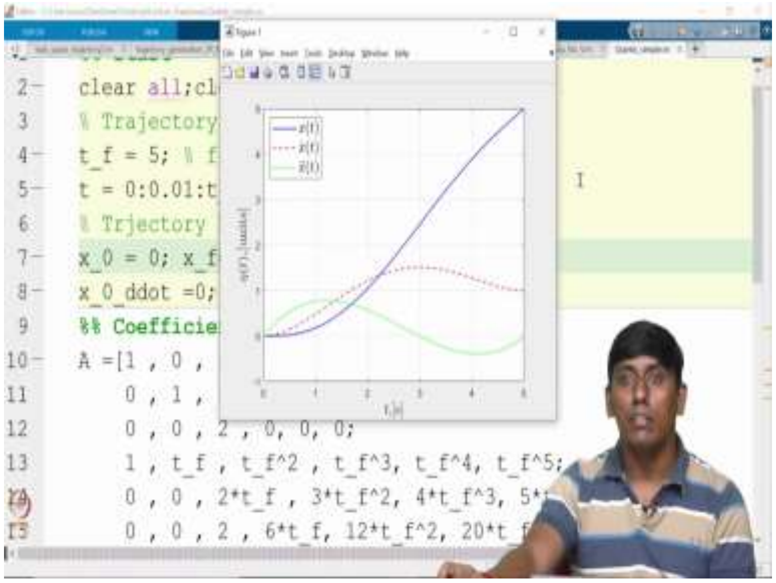


So, even you want to see little more you can see inside I just put it initial and final velocity is the 1. So, you may expect that it would be come as a constant it is not it is again like a go as a smooth So, we can try to see that. So, you can see like it is starting from 1 and from 1 to 1 it is added. So, because we assume that initial velocity is 0, so that way it is making it addition.

(Refer Slide Time: 6:07)

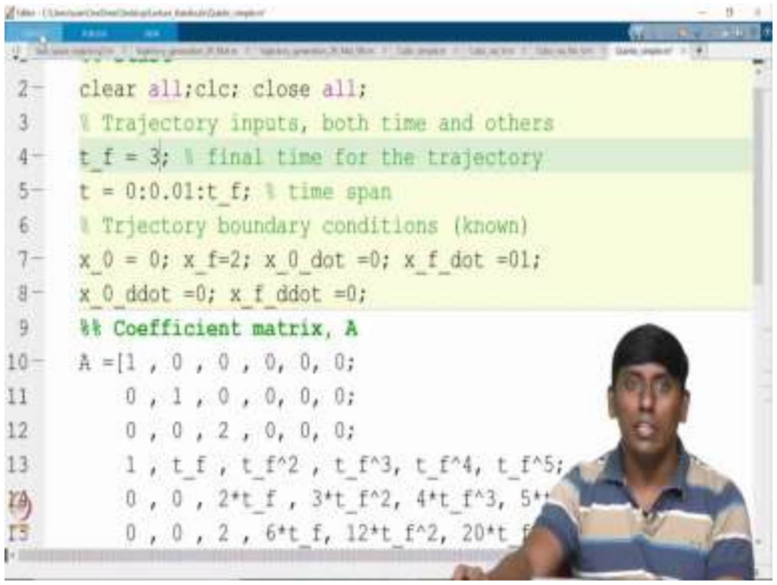


```
clear all;clc; close all;
% Trajectory inputs, both time and others
t_f = 5; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=5; x_0_dot =1; x_f_dot =1;
x_0_ddot =0; x_f_ddot =0;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```

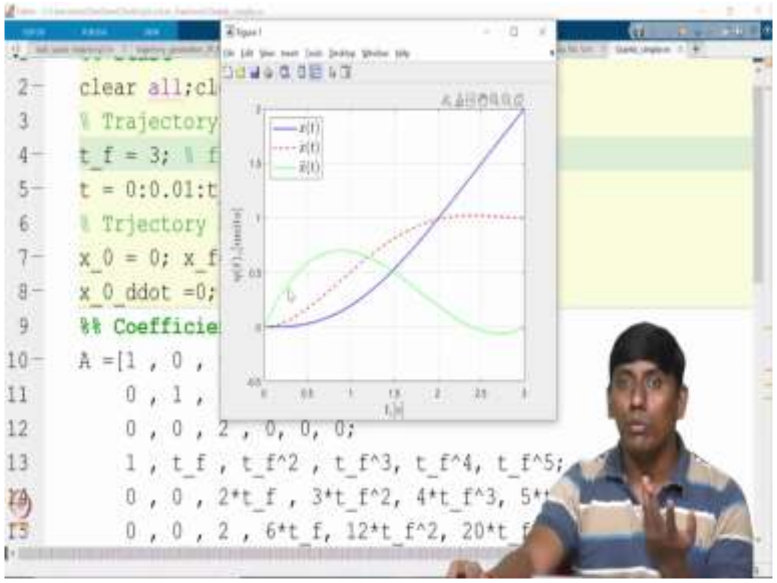


So, now in that case, we can see this aspect how it is going so, this is added from the initial so that way we can see. So now I am putting this as 0. So, you can feel it is ending with 1.

(Refer Slide Time: 6:22)



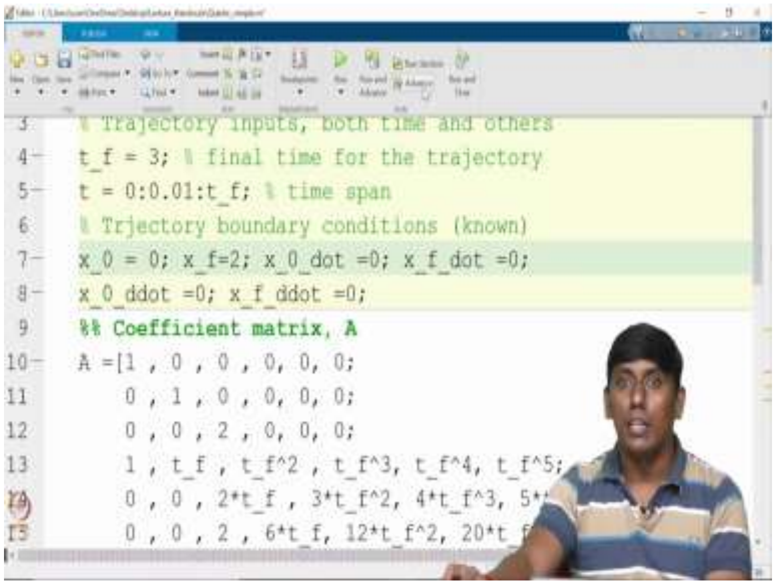
```
clear all;clc; close all;
% Trajectory inputs, both time and others
t_f = 3; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=2; x_0_dot =0; x_f_dot =01;
x_0_ddot =0; x_f_ddot =0;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```



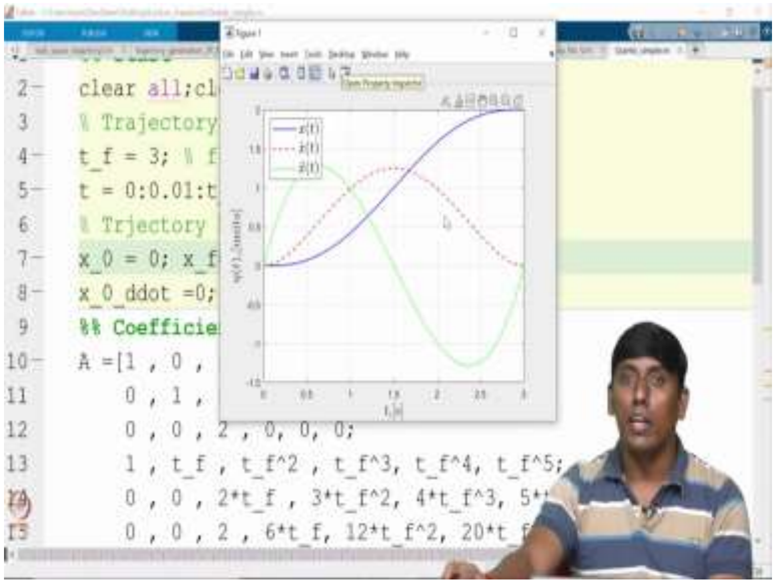
```
clear all;cl
% Trajectory
t_f = 3; % f
t = 0:0.01:t
% Trjectory
x_0 = 0; x_f
x_0_ddot =0;
%% Coefficie
A =[1 , 0 ,
    0 , 1 ,
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```

So, now this I am putting it 0 and this is I am putting it 2 and this is I am putting it 3 you can see that variation whatever variation you can do it these all we can do it. Now you can see it is much much different. So, these all we can try to see and get it done.

(Refer Slide Time: 6:44)



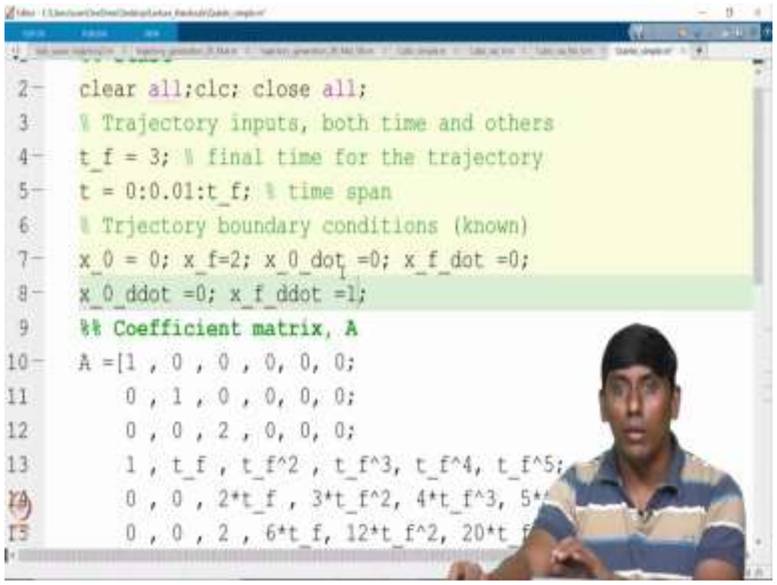
```
% Trajectory inputs, both time and others
t_f = 3; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=2; x_0_dot =0; x_f_dot =0;
x_0_ddot =0; x_f_ddot =0;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```



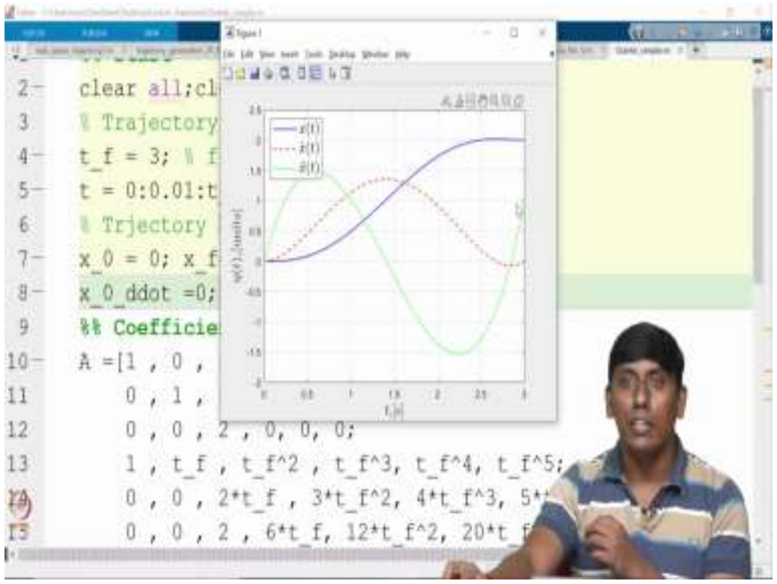
```
clear all;cl
% Trajectory
t_f = 3; % f
t = 0:0.01:t
% Trjectory
x_0 = 0; x_f
x_0_ddot =0;
%% Coefficie
A =[1 , 0 ,
    0 , 1 ,
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```

So, here I did not make it here. So, now we can see even if it is 0 final velocity, it is end.

(Refer Slide Time: 6:56)



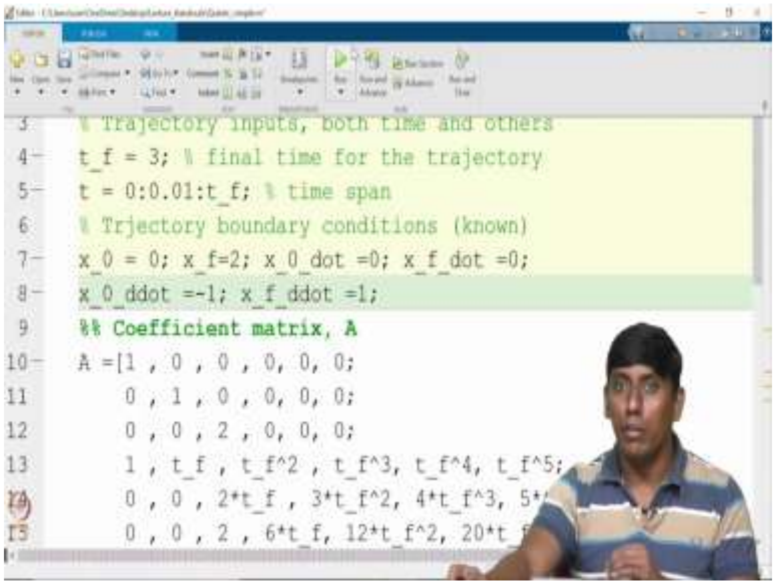
```
clear all;clc; close all;
% Trajectory inputs, both time and others
t_f = 3; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=2; x_0_dot =0; x_f_dot =0;
x_0_ddot =0; x_f_ddot =1;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```



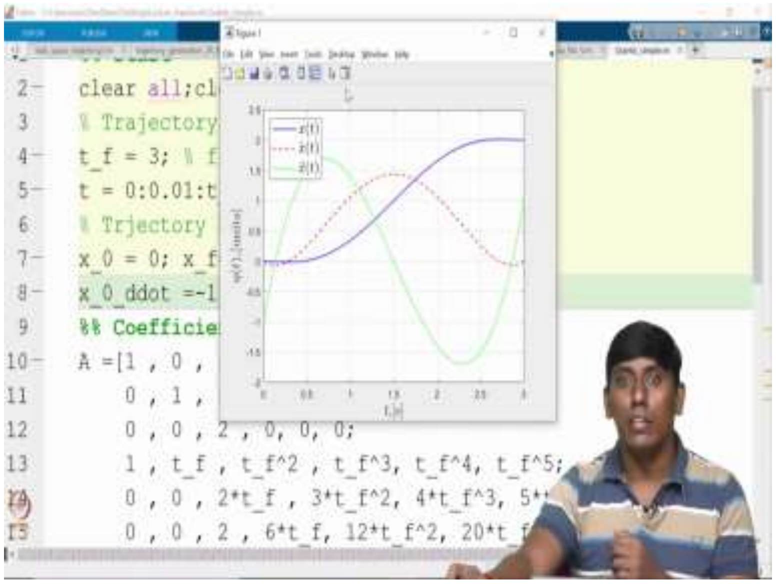
```
clear all;cl
% Trajectory
t_f = 3; % f
t = 0:0.01:t
% Trjectory
x_0 = 0; x_f
x_0_ddot =0;
%% Coefficie
A =[1 , 0 ,
    0 , 1 ,
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_f
```

So, now I hope how the fifth order polynomial is working. So, and what other parameter you can vary. So, the one is the timespan which is how work and how you can make the acceleration that is also like consideration. for example, now, the final acceleration supposed to end with point one. So, you can see like it is end with point 1. So, in order to give a little more, you can say explicitly the visible. So, I am putting it as like 1 meter per second squared. So, you can see it, it is ending. So, these all can be done with you can say fifth order polynomial.

(Refer Slide Time: 7:32)



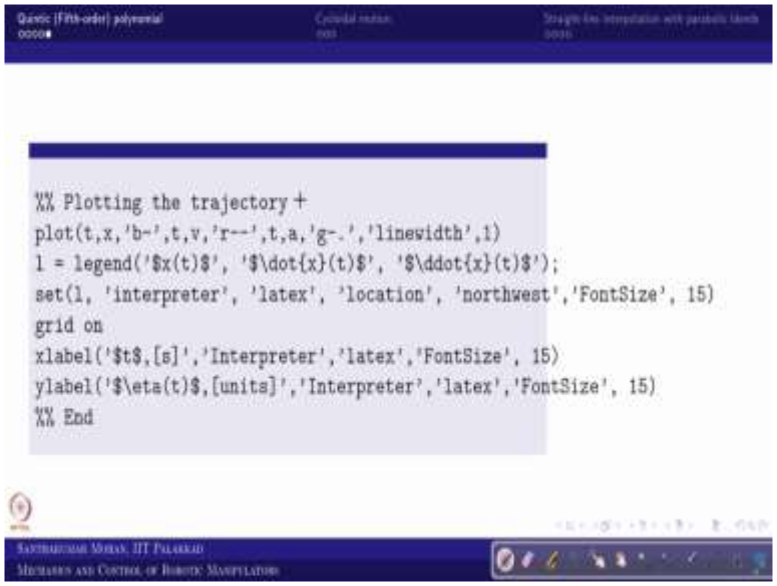
```
% Trajectory inputs, both time and others
t_f = 3; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=2; x_0_dot =0; x_f_dot =0;
x_0_ddot =-1; x_f_ddot =1;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_
```



```
clear all;cl
% Trajectory
t_f = 3; % f
t = 0:0.01:t
% Trjectory
x_0 = 0; x_f
x_0_ddot =-1
%% Coefficie
A =[1 , 0 ,
    0 , 1 ,
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_
```

And even you can give initial acceleration is a minus 1 meter per second squared, which is starting from different profile it ends with a force which is in a negative direction you can see. So, all those things are coming into picture. However, the profile is not changing only thing the velocity profile and the acceleration profile is changing. So here in the sense, it is not changing in the sense starting and ending point is not changing, it is following the smoothness. So, that is what I mean to say it is not changing.

(Refer Slide Time: 8:08)



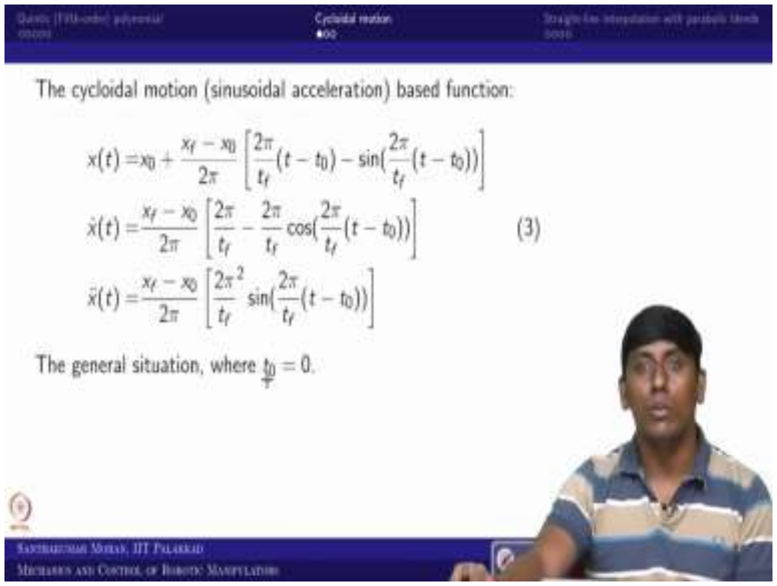
Quintic (Fifth-order) polynomial | Cycloidal motion | Straight-line interpolation with parabolic blends

```
%% 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
```

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So now we will actually go back to the slide. So, where we will be seeing about what is cycloidal motion.

(Refer Slide Time: 8:12)



Quintic (Fifth-order) polynomial | Cycloidal motion | Straight-line interpolation with parabolic blends

The cycloidal motion (sinusoidal acceleration) based function:

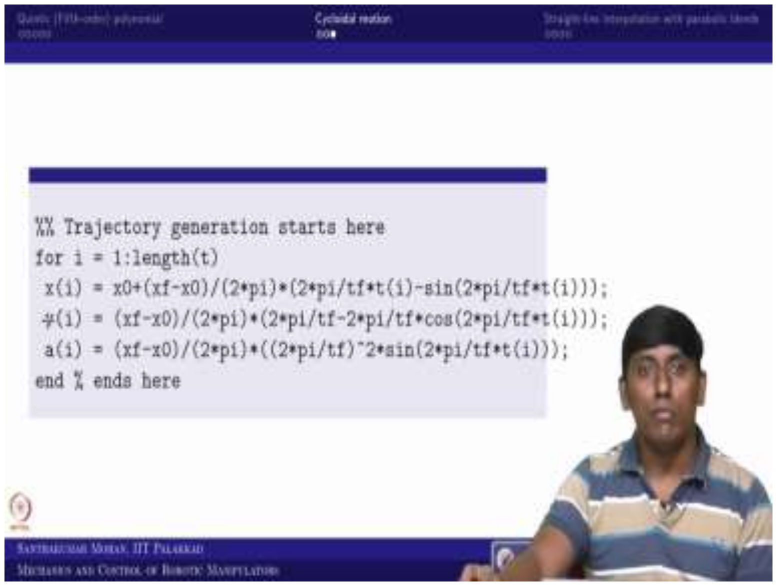
$$\begin{aligned} x(t) &= x_0 + \frac{x_f - x_0}{2\pi}\left[\frac{2\pi}{t_f}(t-t_0) - \sin\left(\frac{2\pi}{t_f}(t-t_0)\right)\right] \\ \dot{x}(t) &= \frac{x_f - x_0}{2\pi}\left[\frac{2\pi}{t_f} - \frac{2\pi}{t_f}\cos\left(\frac{2\pi}{t_f}(t-t_0)\right)\right] \qquad (3) \\ \ddot{x}(t) &= \frac{x_f - x_0}{2\pi}\left[\frac{2\pi^2}{t_f}\sin\left(\frac{2\pi}{t_f}(t-t_0)\right)\right] \end{aligned}$$

The general situation, where $t_0 = 0$.

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

So, the cycloidal motion is we can rewrite in terms of position velocity and acceleration in this form. I do not want to cancel this two pi everywhere. Because if I cancel then I have to bring it here 1 by 2 pi, then that would be all the time residues. So that is why I have written in this general form. So where while we are calculating we assume that t 0 is 0. So, in that case, so, how the MATLAB function will look like the initial stage everything is same.

(Refer Slide Time: 8:41)



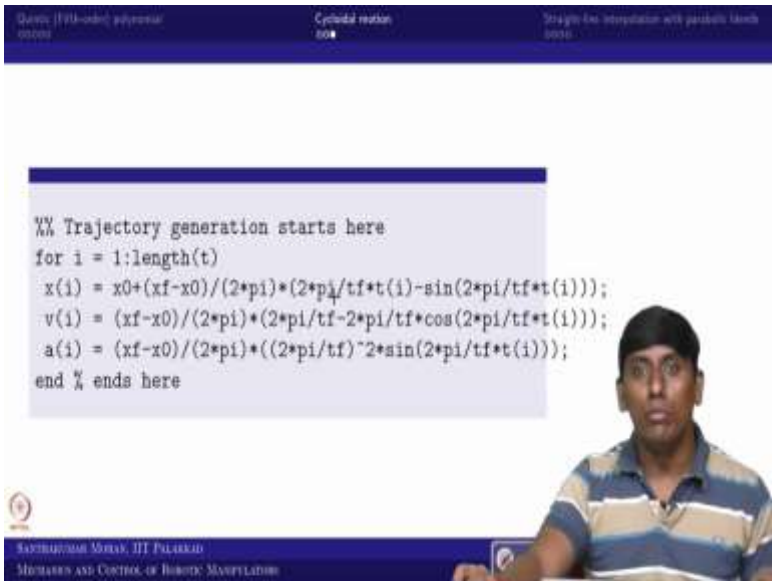
Quintic (Fifth-order) polynomial | Cycloidal motion | Straight-line interpolation with parabolic blends

```
%% Trajectory generation starts here
for i = 1:length(t)
 x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin(2*pi/tf*t(i)));
 v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*cos(2*pi/tf*t(i)));
 a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*pi/tf*t(i)));
end % ends here
```

Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

In the sense these all same. So, the only thing here x naught and the x f are the input because the cycloidal we assume that it is starting from rest and ending with again it rests in the sense it is going to be stopped. So, no further motion. So, it is starting from rest and going to end that is what the segment says.

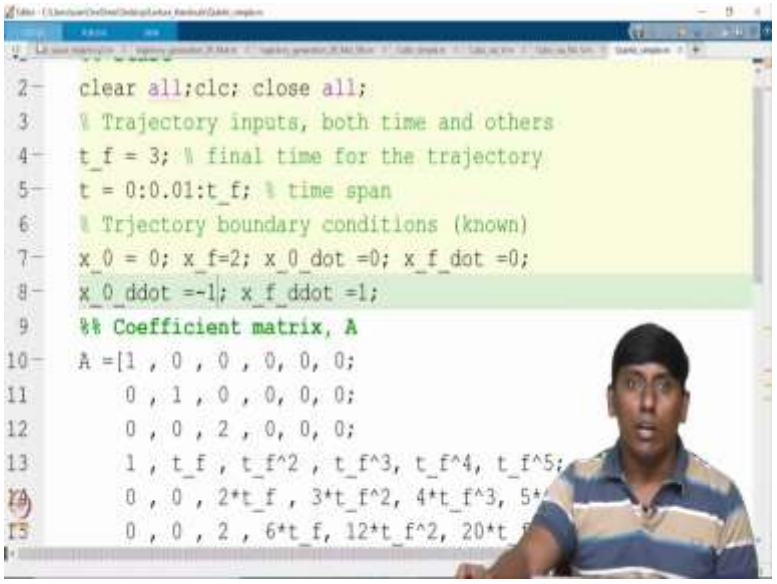
(Refer Slide Time: 9:00)



Quintic (Fifth-order) polynomial | Cycloidal motion | Straight-line interpolation with parabolic blends

```
%% Trajectory generation starts here
for i = 1:length(t)
 x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin(2*pi/tf*t(i)));
 v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*cos(2*pi/tf*t(i)));
 a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*pi/tf*t(i)));
end % ends here
```

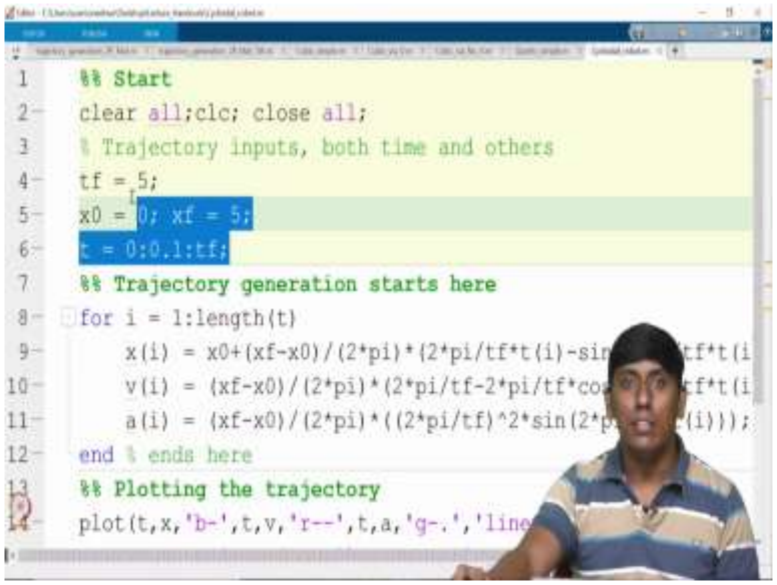
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators



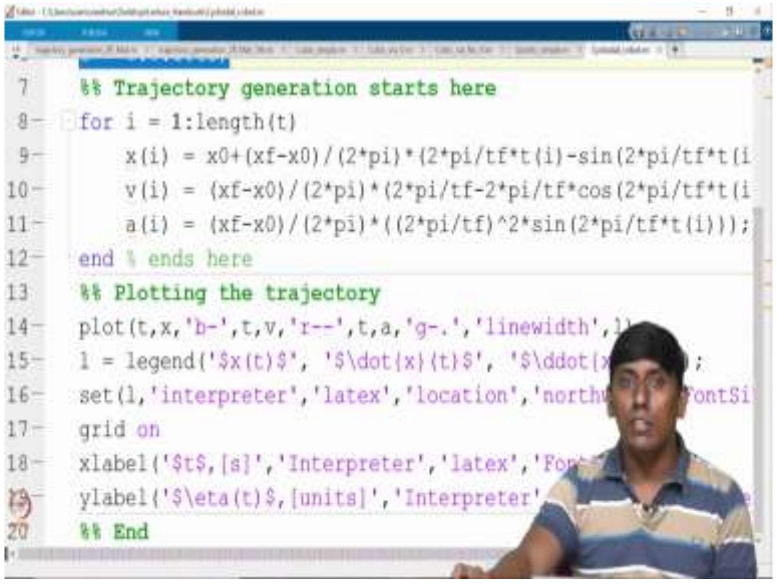
```
clear all;clc; close all;
% Trajectory inputs, both time and others
t_f = 3; % final time for the trajectory
t = 0:0.01:t_f; % time span
% Trjectory boundary conditions (known)
x_0 = 0; x_f=2; x_0_dot =0; x_f_dot =0;
x_0_ddot =-1; x_f_ddot =1;
%% Coefficient matrix, A
A =[1 , 0 , 0 , 0, 0, 0;
    0 , 1 , 0 , 0, 0, 0;
    0 , 0 , 2 , 0, 0, 0;
    1 , t_f , t_f^2 , t_f^3, t_f^4, t_f^5;
    0 , 0 , 2*t_f , 3*t_f^2, 4*t_f^3, 5*
    0 , 0 , 2 , 6*t_f, 12*t_f^2, 20*t_
```

So, in that sense, we can generate the position velocity and acceleration in this form. So now, this is the cycloidal profile. And the same thing we can use as for plotting and all I am not showing the plotting here. So now we will open the file which we make it so cycloidal, I hope.

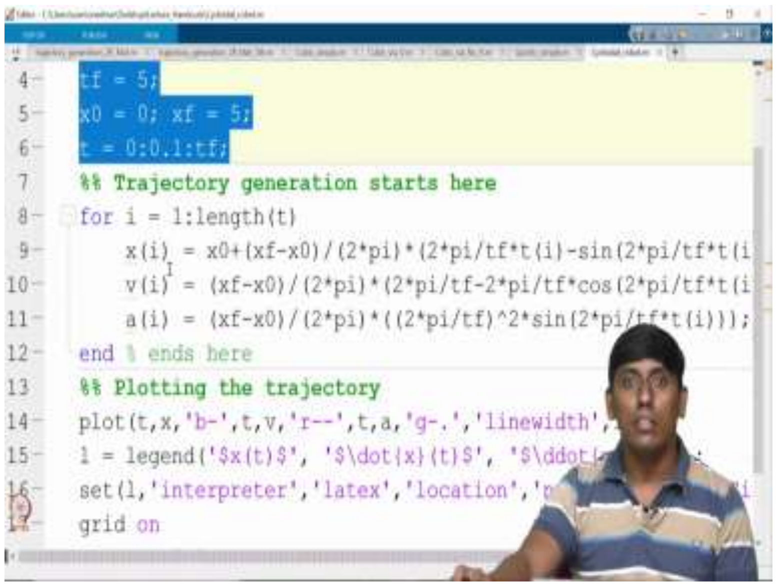
(Refer Slide Time: 9:20)



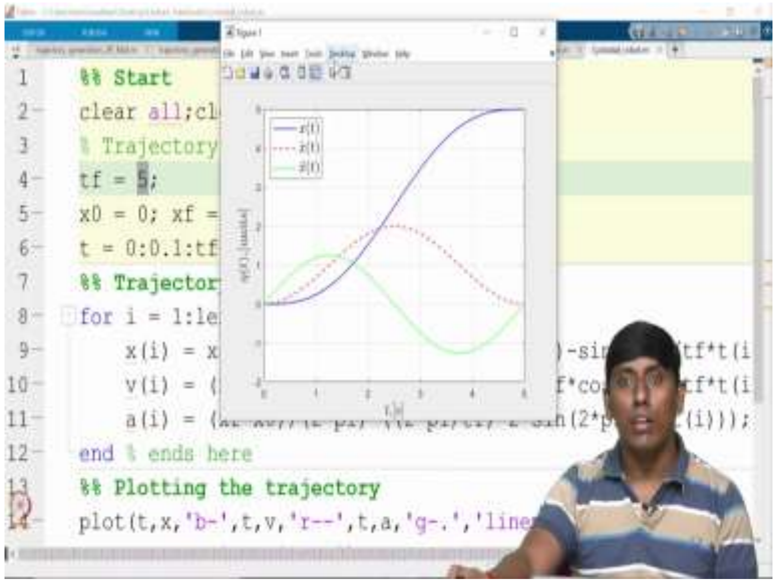
```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
tf = 5;
x0 = 0; xf = 5;
t = 0:0.1:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin          tf*t(i
    v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*co           f*t(i
    a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*p          (i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','line
```



```
%% Trajectory generation starts here
for i = 1:length(t)
    x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin(2*pi/tf*t(i
    v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*cos(2*pi/tf*t(i
    a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*pi/tf*t(i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth',1)
l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{x          );
set(l,'interpreter','latex','location','northw        FontSi
grid on
xlabel('$t$,[s]','Interpreter','latex','Fo
ylabel('$\eta(t)$,[units]','Interpreter'
%% End
```



```
tf = 5;
x0 = 0; xf = 5;
t = 0:0.1:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin(2*pi/tf*t(i
    v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*cos(2*pi/tf*t(i
    a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*pi/tf*t(i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth',
l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{
set(l,'interpreter','latex','location','n
grid on
```



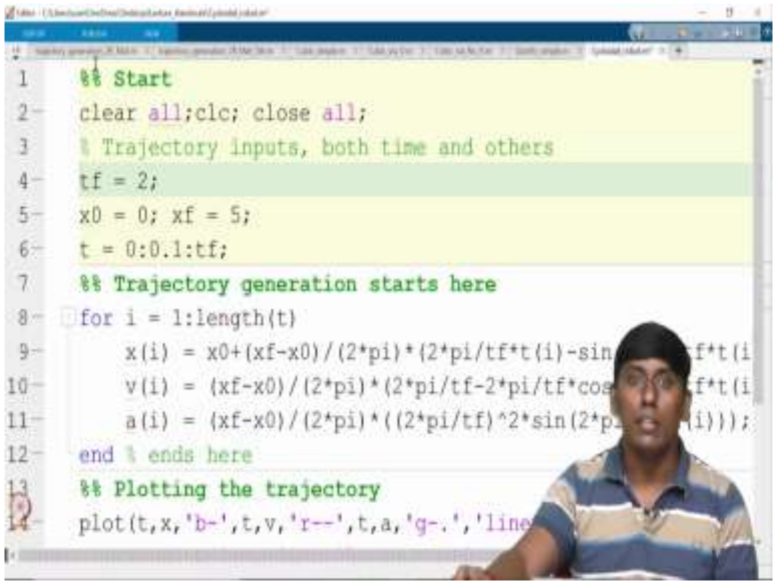
```
%% Start
clear all;cl
% Trajectory
tf = 5;
x0 = 0; xf =
t = 0:0.1:tf
%% Trajector
for i = 1:le
    x(i) = x                                  )-si      tf*t(i
    v(i) = (                                  f*co      tf*t(i
    a(i) = (                            sin(2*p      (i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','line
```

So, now you can see this is the command which we have given in the you can say handout, and this is the trajectory generation and this plotting we have used earlier the same thing I have tried. So now the same case we are trying. So, it takes 5 seconds to reach 0 to 5 in a smoother sense. So, we can try to see by running this. So, you can see it, so it is very smooth. But here we do not have any control over initial and final velocity initial and final acceleration.

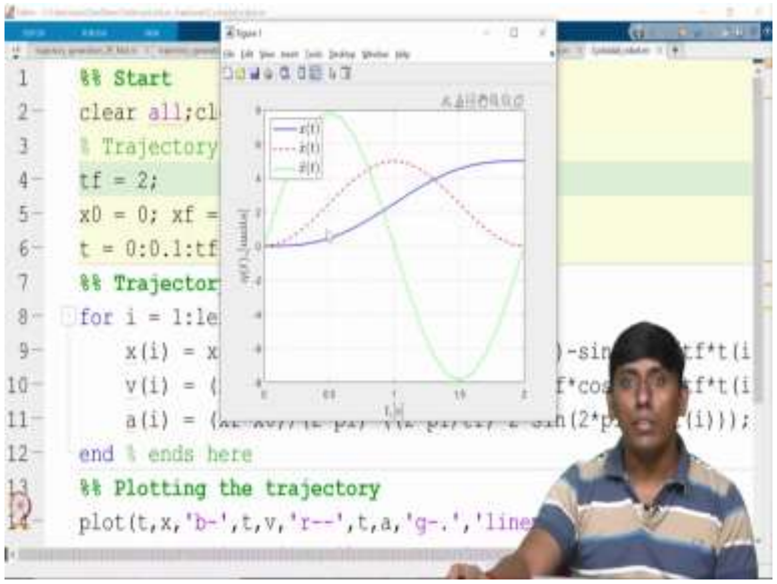
However, it starts from 0 and the end also at 0. In that case, acceleration velocity all starts from 0 and end at also 0. In that sense you have much, much smoother. And you can always use this kind of profile for like a complex system like a humanoid or even a complex manipulator and all we can use this. That is why people try to use for example the legged locomotion for example, the biped motion.

So, one foot to another footstep, so, you need to make a cycloidal motion that would be very much beneficial it start from rest and the end also at standby there is no further motion. So, then the next step will start that way we can do it.

(Refer Slide Time: 10:33)



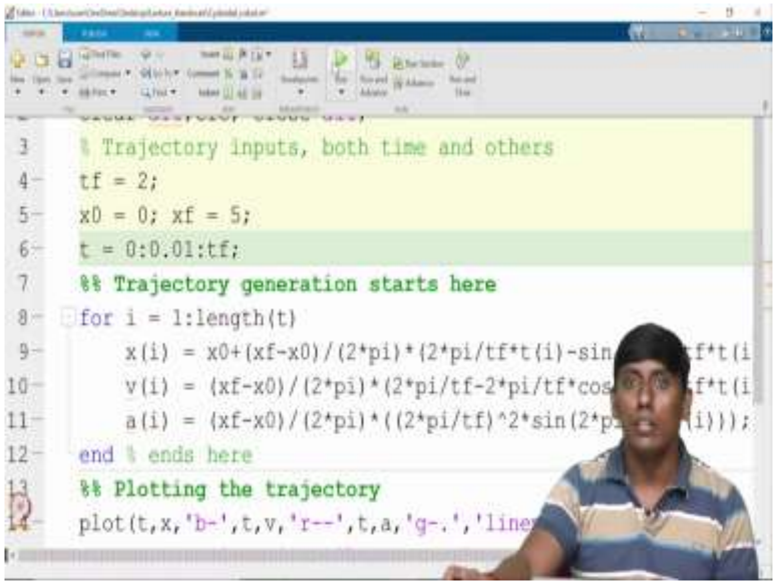
```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
tf = 2;
x0 = 0; xf = 5;
t = 0:0.1:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin      f*t(i
    v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*cos      f*t(i
    a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*p      (i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','line
```



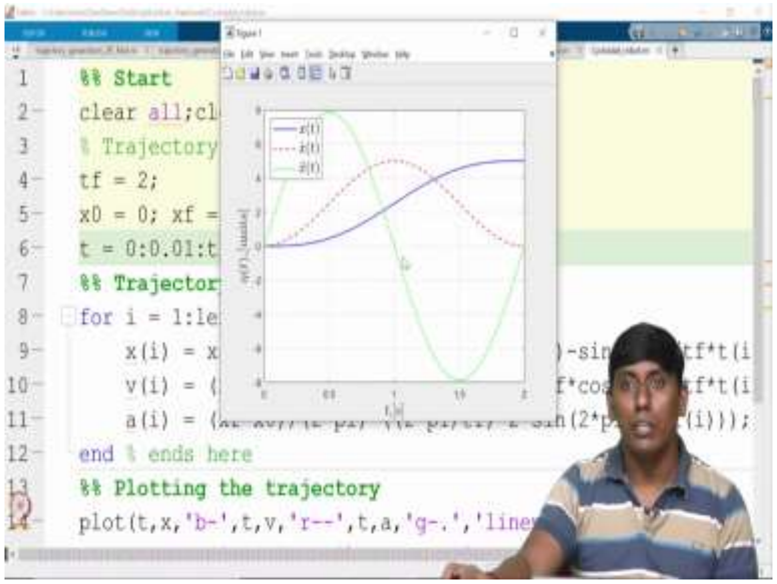
```
%% Start
clear all;cl
% Trajectory
tf = 2;
x0 = 0; xf =
t = 0:0.1:tf
%% Trajector
for i = 1:le
    x(i) = x                     )-sin      f*t(i
    v(i) = (                     f*cos      f*t(i
    a(i) = (                     n(2*p      (i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','line
```

So, now if I change this. For example, I am saying that it is run with only 2 seconds. So, you can see like the variation would be very significant. But the one thing you can clearly see that the acceleration is pure sinusoidal acceleration. So here I make it 0.1 is the step size. So that is why it is look like very, you can say segment basis.

(Refer Slide Time: 10:59)



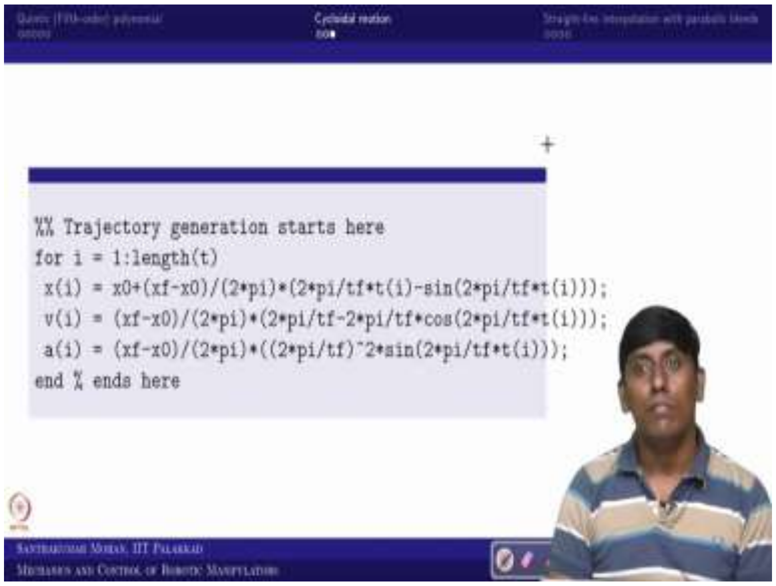
```
% Trajectory inputs, both time and others
tf = 2;
x0 = 0; xf = 5;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin    f*t(i
    v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*cos    f*t(i
    a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*p     i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','line
```



```
%% Start
clear all;cl
% Trajectory
tf = 2;
x0 = 0; xf =
t = 0:0.01:t
%% Trajector
for i = 1:le
    x(i) = x                )-sin    tf*t(i
    v(i) = (                f*cos    f*t(i
    a(i) =                    n(2*p    (i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','line
```

So, for increasing that you can make it smooth. So, now I just wanted to show that it is sinusoidal profile you can say like now it is smooth. So that is I make it only for plotting what segment I want to give. So, I take every 0.1 second in the sense 100 millisecond, but in this case, so I have taken 10 milliseconds, so to show the smoothness. So here the acceleration is because sinusoidal acceleration even some people call the cycloidal profile called sinusoidal acceleration curve. So, this is what the cycloidal.

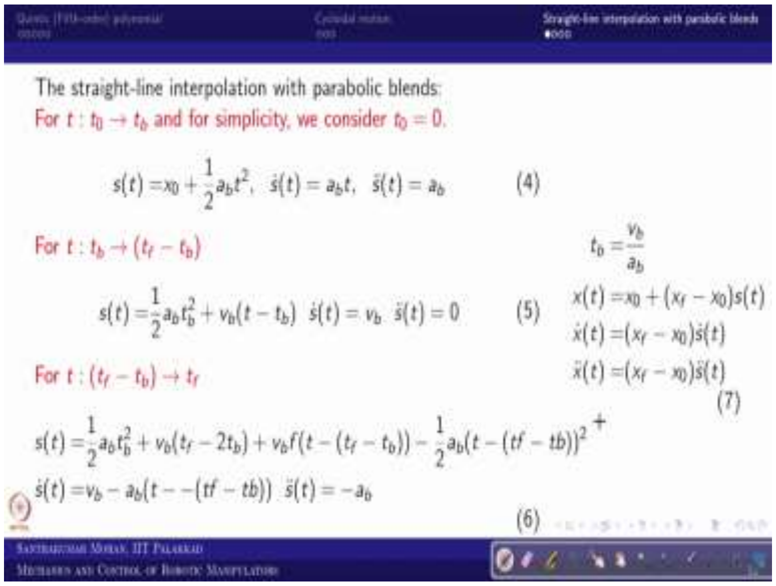
(Refer Slide Time: 11:36)



```
%% Trajectory generation starts here
for i = 1:length(t)
 x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin(2*pi/tf*t(i)));
 v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*cos(2*pi/tf*t(i)));
 a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*pi/tf*t(i)));
end % ends here
```

So, we will see the final one on this particular lecture is what you call straight line interpolation with the parabolic blend. So, you know straight line interpolation means starting to end we just make a straight line and the slope we will take and multiply. But that is non smooth because the acceleration at infinite at initial and final, which is not possible in the real time the actuator will not be substantially support that.

(Refer Slide Time: 11:58)



The straight-line interpolation with parabolic blends:

For $t : t_0 \rightarrow t_b$ and for simplicity, we consider $t_0 = 0$.

$$s(t) = x_0 + \frac{1}{2}a_b t^2, \quad \dot{s}(t) = a_b t, \quad \ddot{s}(t) = a_b \qquad (4)$$

For $t : t_b \rightarrow (t_f - t_b)$

$$s(t) = \frac{1}{2}a_b t_b^2 + v_b(t - t_b) \quad \dot{s}(t) = v_b \quad \ddot{s}(t) = 0 \qquad (5)$$

For $t : (t_f - t_b) \rightarrow t_f$

$$s(t) = \frac{1}{2}a_b t_b^2 + v_b(t_f - 2t_b) + v_b f(t - (t_f - t_b)) - \frac{1}{2}a_b(t - (tf - tb))^2$$

$$\dot{s}(t) = v_b - a_b(t - -(tf - tb)) \quad \ddot{s}(t) = -a_b \qquad (6)$$

$$t_b = \frac{v_b}{a_b}$$

$$x(t) = x_0 + (x_f - x_0)s(t)$$
$$\dot{x}(t) = (x_f - x_0)\dot{s}(t)$$
$$\ddot{x}(t) = (x_f - x_0)\ddot{s}(t) \qquad (7)$$

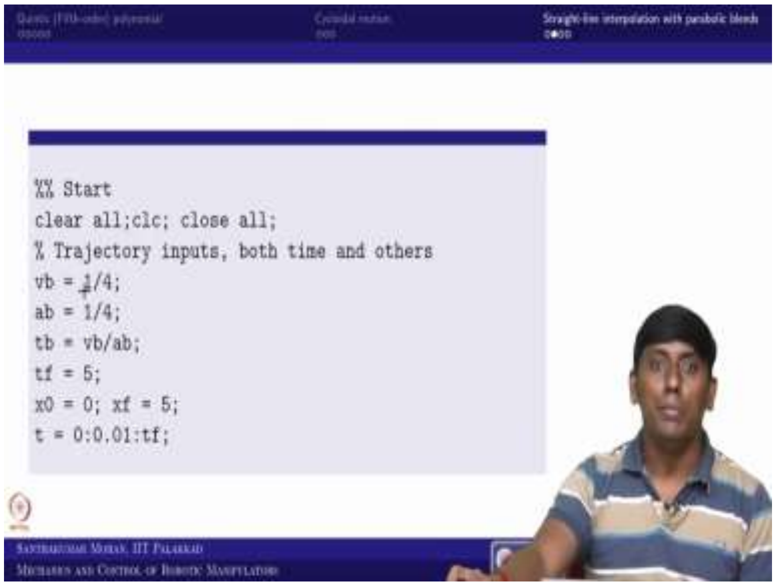
So, for that only we are doing the blending. So, what we are trying to blend? So, we are taking the same straight line into three segments, the initial and finally will add the parabolic segment.

So, that is what we are trying to do. So, you can see that the t b the parabolic blend. So, where that segment time so, now, you can see that t 0 to t b is the parabolic. So, the parabolic can be written in this form. So, then after that it is a straight line till so, t f minus t b then t f you can see where the t is start from t f minus t b to t f.

So, there you will be using again the parabolic. So, in that sense the profile is much smoother, and it is faster to use. However, the algorithm is much much complicated. So, you have to bring this v b and the a b in such a way that the profile will fulfill. So, far that we are bringing this consideration. So, where this v b and a b we have to choose accordingly the t b will come. However, while choosing so, you had to make sure that the v b squared by a b should be always less than or equal to 1.

So, if you take this further consideration is you can take that the capital or you can say t f t f should be take at least three times of t b. So, then only you can make a blend. So, these are all complex. However, what we can do we can take this particular consideration and we can make as a s of t so then we can multiply with your original x of 0. So, why? because this would be run for 0 to 1. So, then we will multiply whatever the segment you wanted. So, that is why you can see like x f minus x 0 multiply with the s and s dot everything is coming as in the sense we are multiplying that many times.

(Refer Slide Time: 13:55)

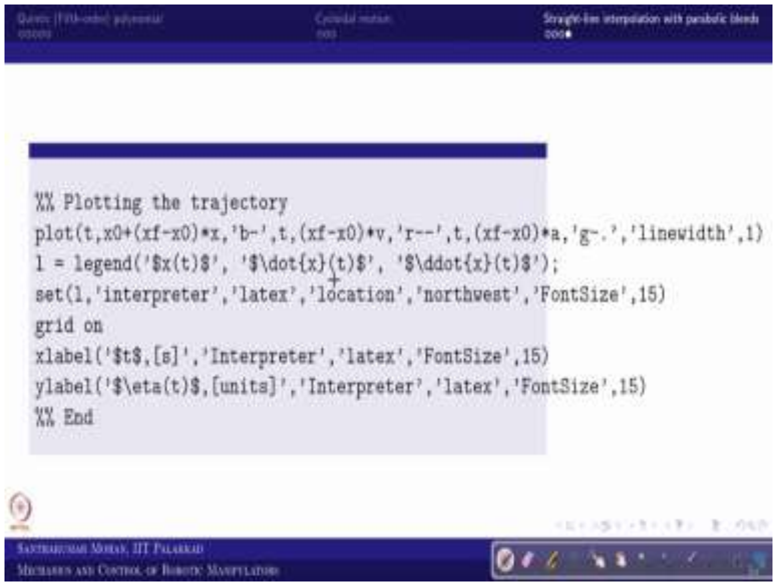


Quintic (Fifth-order) polynomial
Cycloidal motion
Straight-line interpolation with parabolic blends
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
vb = 1/4;
ab = 1/4;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 5;
t = 0:0.01:tf;
```

So for that we are writing the code. So, here we need to define v b a b similarly, the t b is defined based on the v b divided by a b and we have taken this. So, here you can see again 5 second. So, we are taking first 1 second for parabolic the next a 3 second is linear, then final 1 second is for parabolic that way we make a blend. So, because we take a v b and a b are 1 by 4 each. So, in that sense the t b is 1 second.

(Refer Slide Time: 14:24)

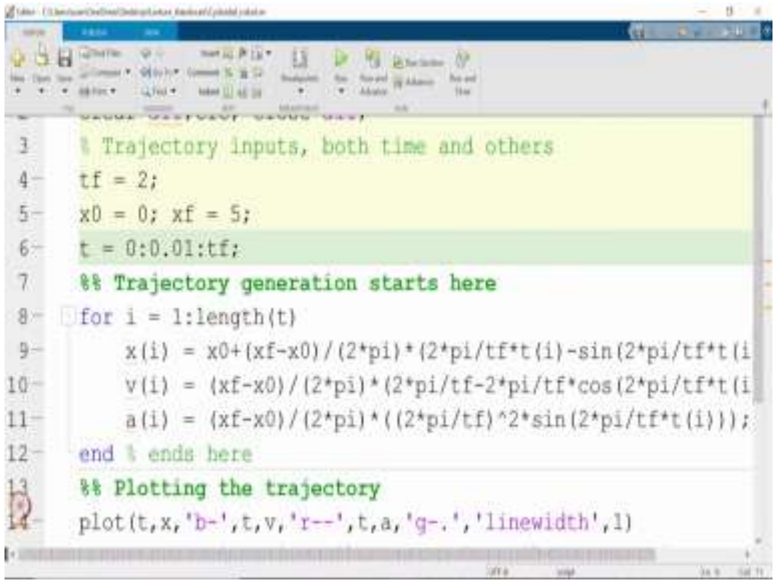


Quintic (fifth-order) polynomial
Cycloidal motion
Straight-line interpolation with parabolic blends
Santhakumar Mohan, IIT Palakkad
Mechanics and Control of Robotic Manipulators

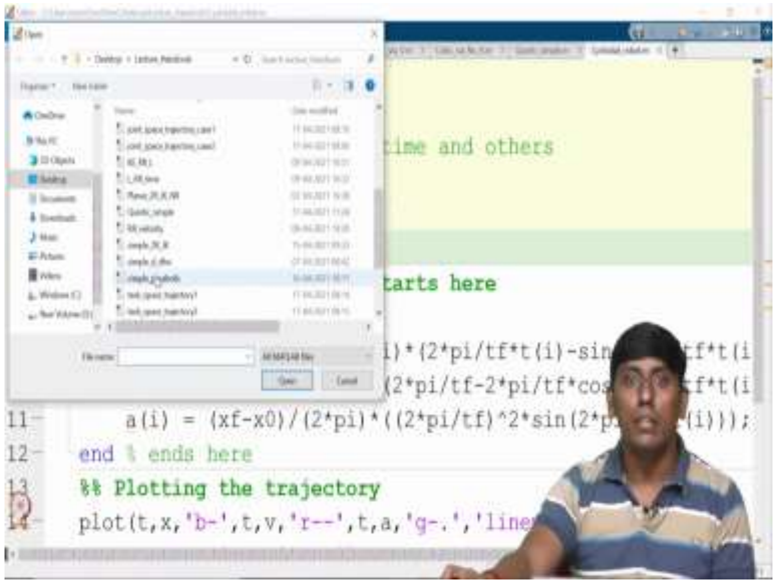
```
%% Plotting the trajectory
plot(t,x0+(xf-x0)*x,'b-',t,(xf-x0)*v,'r--',t,(xf-x0)*a,'g-.','linewidth',1)
l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{x}(t)$');
set(l,'interpreter','latex','location','northwest','FontSize',15)
grid on
xlabel('$t$,[s]','Interpreter','latex','FontSize',15)
ylabel('$\eta(t)$,[units]','Interpreter','latex','FontSize',15)
%% End
```

So, based on that we have written the code whatever we have written in the you can say syntax. The same thing we have written. So now you can see x v and a would be based on you can say a b and v b. So that we have obtained this is a parabolic then straight line another parabolic. So, this way we have written and we use the simple plotting function but now the plotting function is modified because we have derived only s of t but the s of t need to be modified as a x of t. So, we need to multiply this particular value. So that is why we have modified this. So, now we will actually go to the MATLAB.

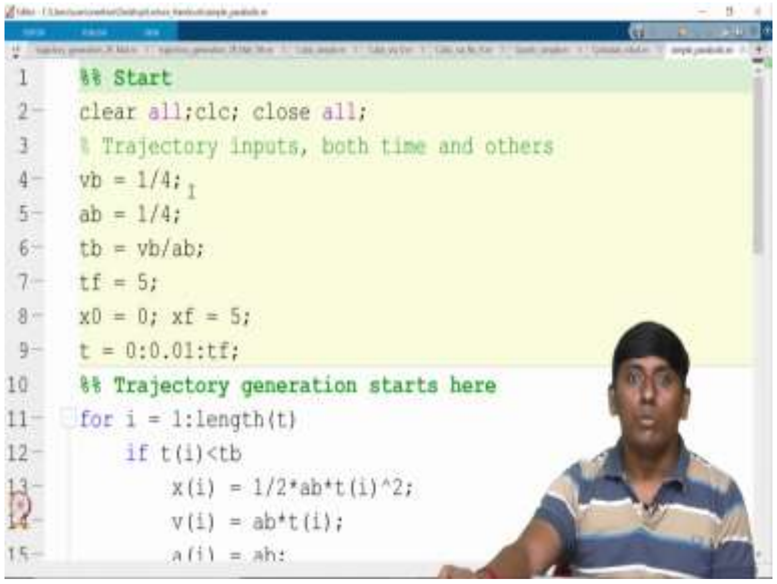
(Refer Slide Time: 15:07)



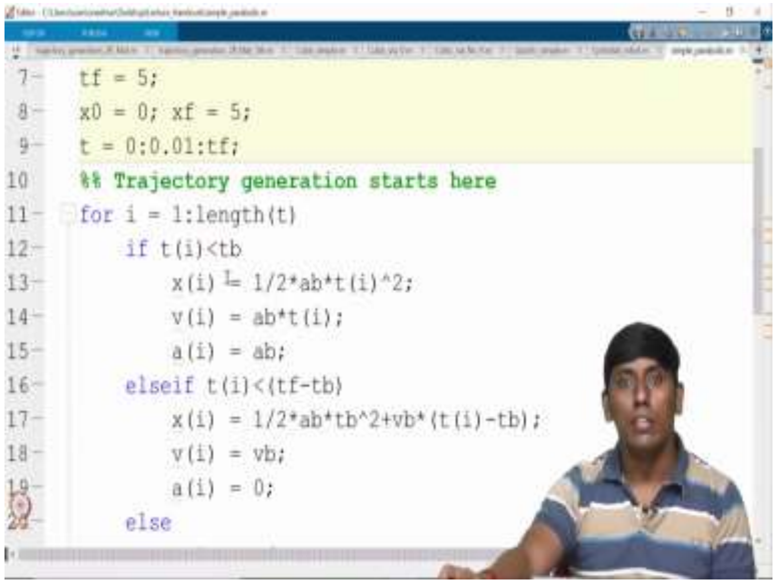
```
% Trajectory inputs, both time and others
tf = 2;
x0 = 0; xf = 5;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    x(i) = x0+(xf-x0)/(2*pi)*(2*pi/tf*t(i)-sin(2*pi/tf*t(i
    v(i) = (xf-x0)/(2*pi)*(2*pi/tf-2*pi/tf*cos(2*pi/tf*t(i
    a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*pi/tf*t(i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','linewidth',1)
```



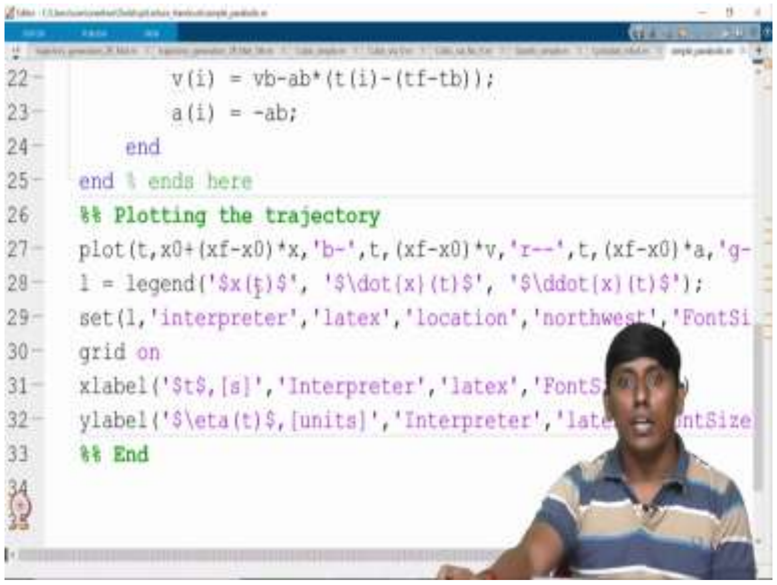
```
time and others
tarts here
i)*(2*pi/tf*t(i)-sin        tf*t(i
(2*pi/tf-2*pi/tf*cos        tf*t(i
    a(i) = (xf-x0)/(2*pi)*((2*pi/tf)^2*sin(2*p        (i)));
end % ends here
%% Plotting the trajectory
plot(t,x,'b-',t,v,'r--',t,a,'g-.','line
```



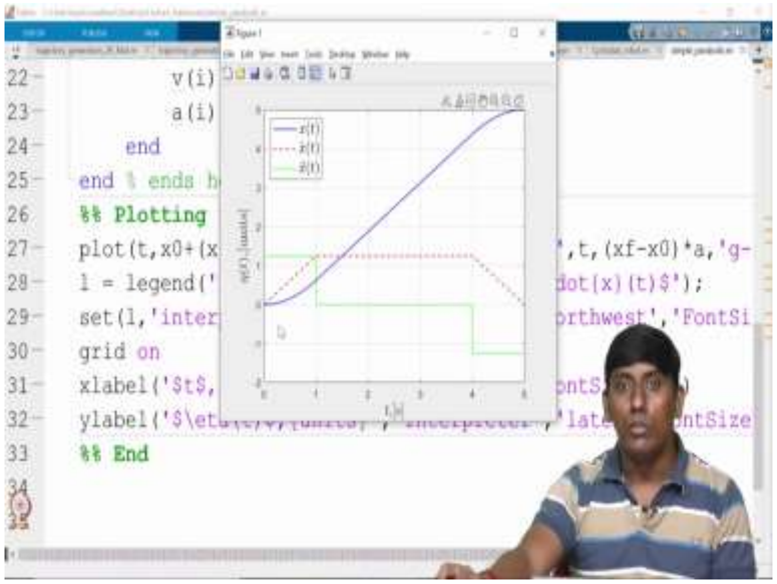
```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
vb = 1/4;
ab = 1/4;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 5;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
        a(i) = ab;
```



```
tf = 5;
x0 = 0; xf = 5;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
        a(i) = ab;
    elseif t(i)<(tf-tb)
        x(i) = 1/2*ab*tb^2+vb*(t(i)-tb);
        v(i) = vb;
        a(i) = 0;
    else
```



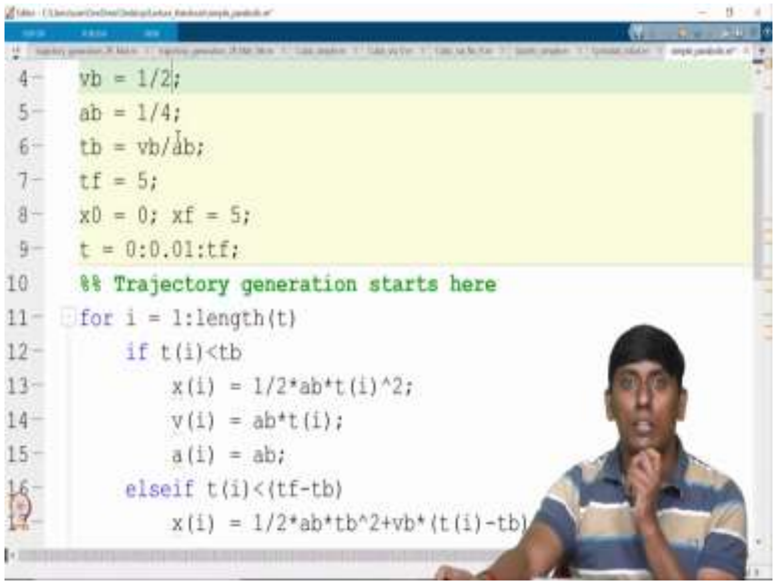
```
        v(i) = vb-ab*(t(i)-(tf-tb));
        a(i) = -ab;
    end
end % ends here
%% Plotting the trajectory
plot(t,x0+(xf-x0)*x,'b-',t,(xf-x0)*v,'r--',t,(xf-x0)*a,'g-
l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{x}(t)$');
set(l,'interpreter','latex','location','northwest','FontSi
grid on
xlabel('$t$,[s]','Interpreter','latex','FontS
ylabel('$\eta(t)$,[units]','Interpreter','late      ntSize
%% End
```



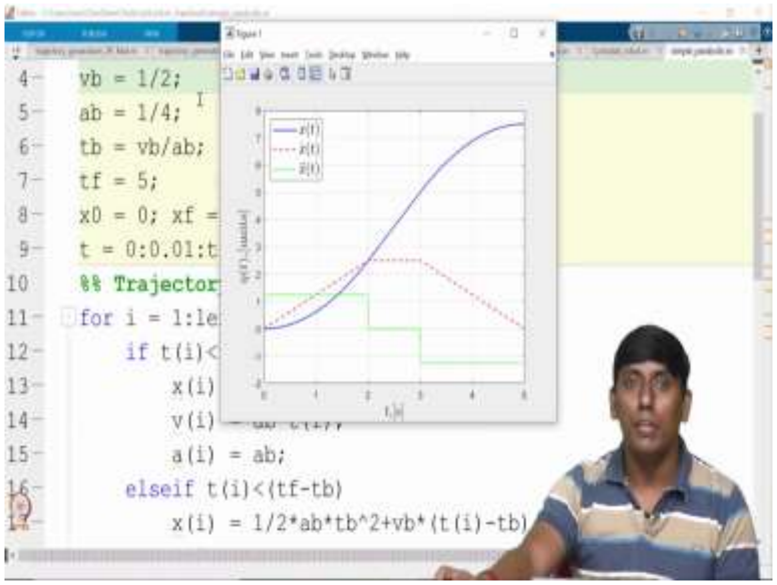
And we can see how that will work. So, for that we are taking the parabolic blend. So now you can see that v b and a b we have taken as 1 by 4 each. And in that case, you can see this. So, we are taking into this consideration. So, now I am modifying the equation earlier we have written that I modified into three segments, and then I am plotting it. So, in that sense, what we can do? So, if we run this. So, you can feel it what exactly happening so, you can see.

So, this is the constant acceleration stage, and this is constant deceleration stage where the velocity would be starting 0 to increase to maximum. Then the constant deceleration maximum to 0 it comes. So, in the middle segment you can see that it is a straight line where the velocity is constant. So, like that we can keep playing it. So now, this will give you much, much better aspect. So now you want to increase to 2.

(Refer Slide Time: 16:12)



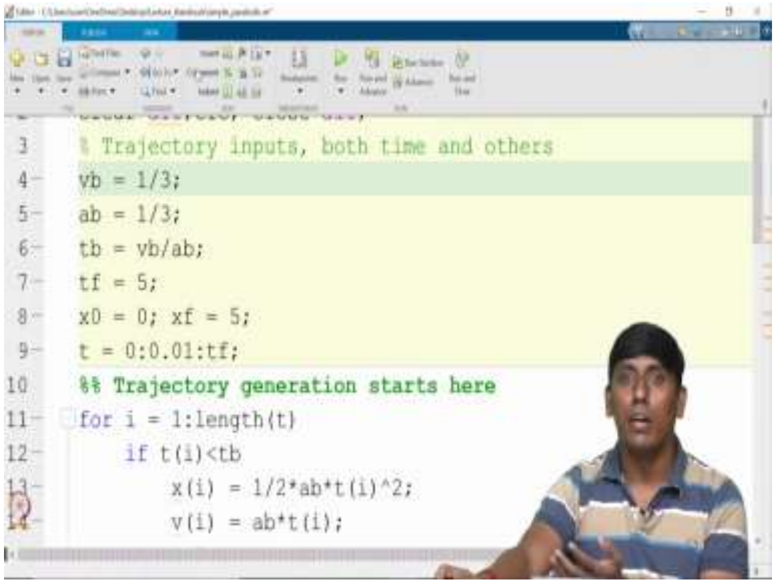
```
vb = 1/2;
ab = 1/4;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 5;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
        a(i) = ab;
    elseif t(i)<(tf-tb)
        x(i) = 1/2*ab*tb^2+vb*(t(i)-tb)
```



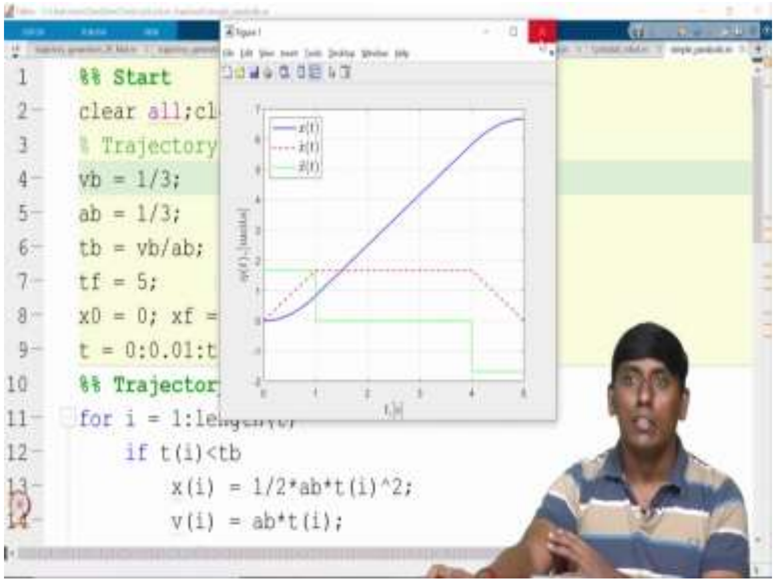
```
vb = 1/2;
ab = 1/4;
tb = vb/ab;
tf = 5;
x0 = 0; xf =
t = 0:0.01:t
%% Trajector
for i = 1:le
    if t(i)<
        x(i)
        v(i)
        a(i) = ab;
    elseif t(i)<(tf-tb)
        x(i) = 1/2*ab*tb^2+vb*(t(i)-tb)
```

I say that is 5 second, but I want to increase to 2. So, what I am trying to make it this is 1 by 2. So now I am keeping this way. So, in that sense, you can see the t b become 2 I will just check. So, I did not try so I will just check. So now you can see like it is make it but now the proposition is change. So now you can see like it is supposed to end with 5 but the proposition has changed. So, we see here what we made.

(Refer Slide Time: 16:47)



```
clear all;clc; close all;
% Trajectory inputs, both time and others
vb = 1/3;
ab = 1/3;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 5;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```

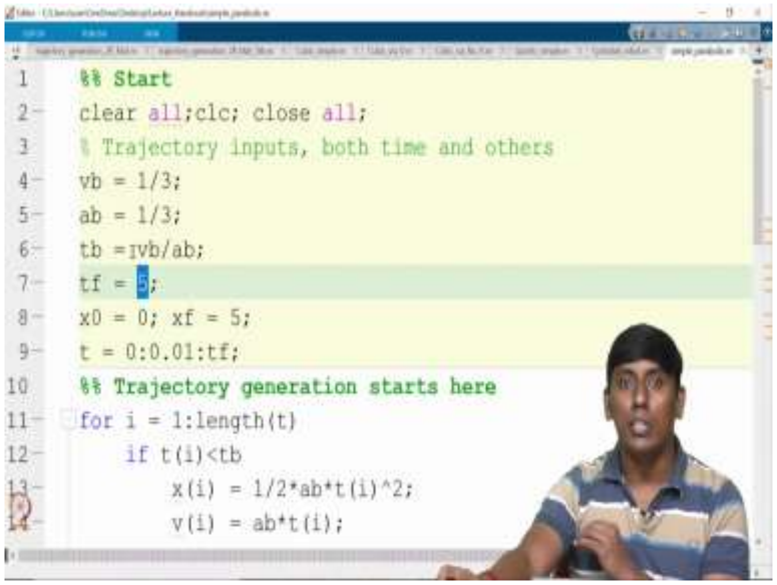


```
%% Start
clear all;cl
% Trajectory
vb = 1/3;
ab = 1/3;
tb = vb/ab;
tf = 5;
x0 = 0; xf =
t = 0:0.01:t
%% Trajector
for i = 1:len
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```

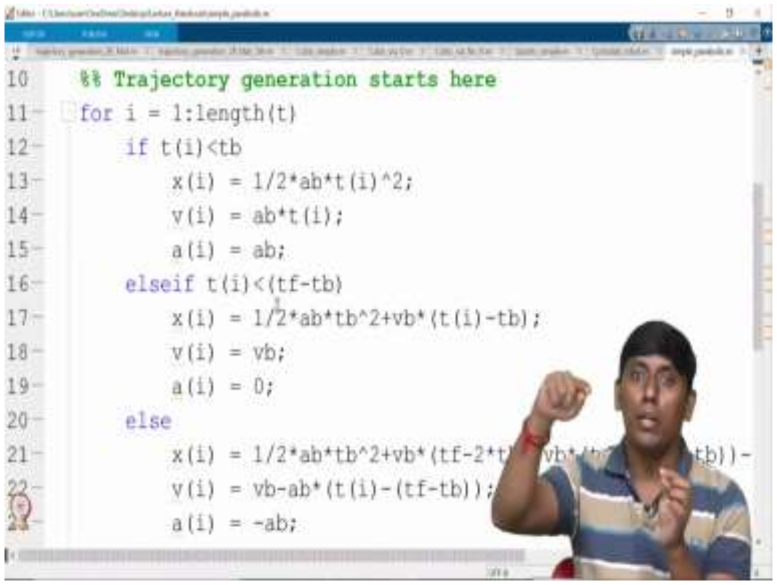
So, this is modified into some consideration of v b. So, now in that case we are modify. So, that is why I said this what you call parabolic blend is very difficult to make it because you have 3 equations, or you can say you have 4 equations and 3 unknowns it is very difficult to fulfill. So, that is what the consideration it is. So, now, if I make it, this is the other way around. So, now I modified this into a slight consideration.

So, now, you can see that this is still 1 but the velocity and acceleration is modified. So, now the acceleration and this one is modified now, the position is shifted according to that. So, like that you can make it. So, even when you wanted to do it a constant acceleration and deceleration.

(Refer Slide Time: 17:45)



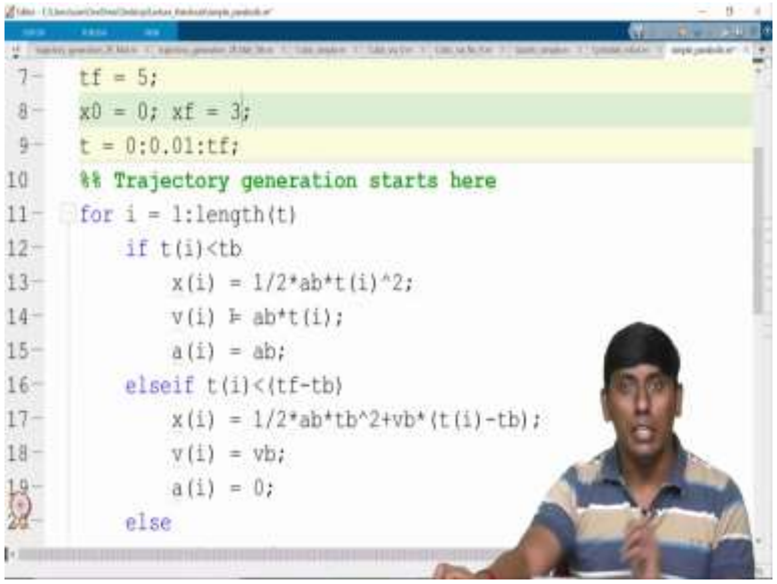
```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
vb = 1/3;
ab = 1/3;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 5;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```



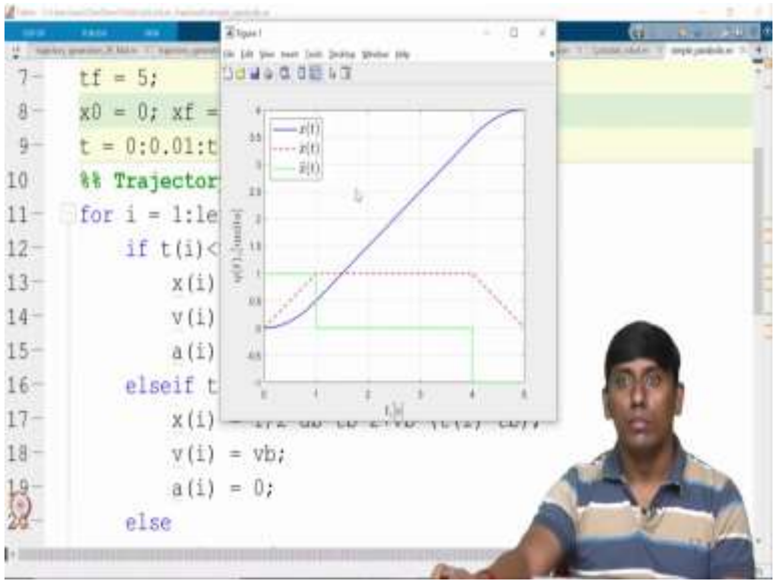
```
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
        a(i) = ab;
    elseif t(i)<(tf-tb)
        x(i) = 1/2*ab*tb^2+vb*(t(i)-tb);
        v(i) = vb;
        a(i) = 0;
    else
        x(i) = 1/2*ab*tb^2+vb*(tf-2*t[illegible]vb*(t[illegible]tb))-
        v(i) = vb-ab*(t(i)-(tf-tb));
        a(i) = -ab;
```

Then you can assume that you just assume that so, your what do you call this t f is still 5 but the t b is 2.5 so, then you will get only two parabolic blend that make so which we call a constant acceleration and deceleration. So, this we have seen already in the you can say lecture. So, now this way we can play. So, I already said this is impose with some additional constraint.

(Refer Slide Time: 18:13)



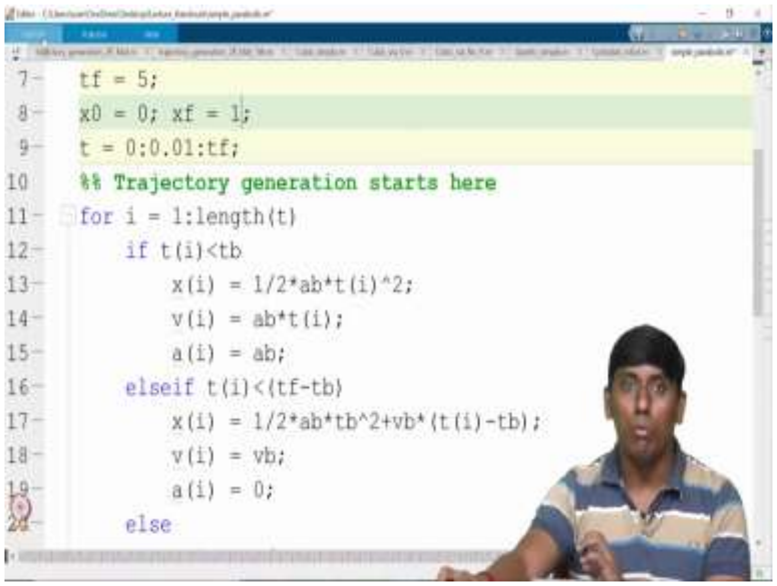
```
tf = 5;
x0 = 0; xf = 3;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
        a(i) = ab;
    elseif t(i)<(tf-tb)
        x(i) = 1/2*ab*tb^2+vb*(t(i)-tb);
        v(i) = vb;
        a(i) = 0;
    else
```



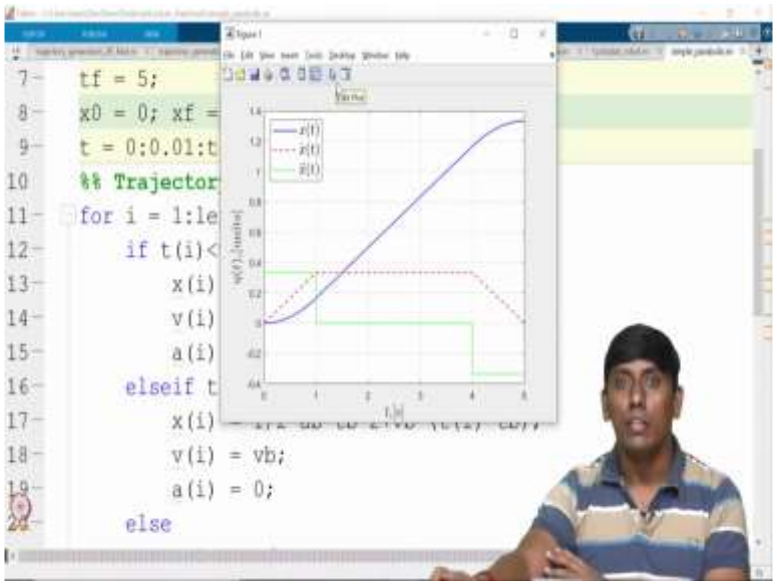
```
tf = 5;
x0 = 0; xf =
t = 0:0.01:t
%% Trajector
for i = 1:le
    if t(i)<
        x(i)
        v(i)
        a(i)
    elseif t
        x(i)
        v(i) = vb;
        a(i) = 0;
    else
```

So, for example, I am putting this into 3. So, we can see whether that is following it you can see. So, it is not exactly matching because you can see if you modified that step is varying.

(Refer Slide Time: 18:29)

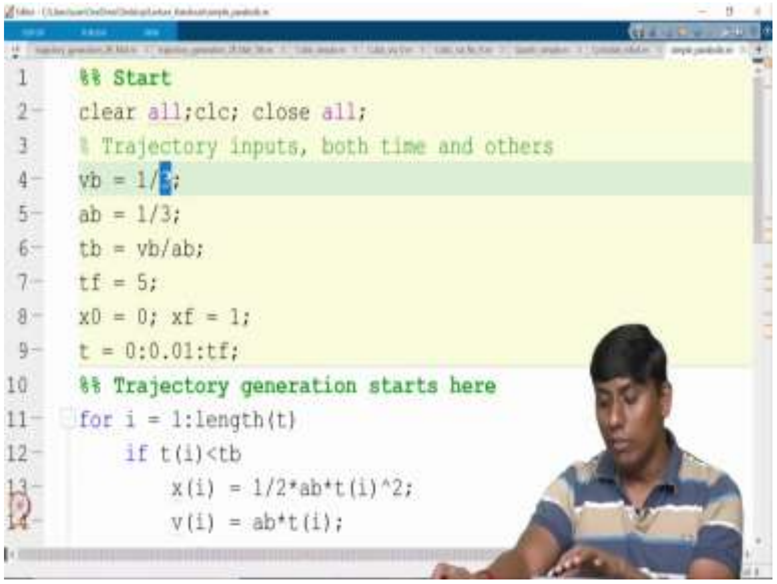


```
tf = 5;
x0 = 0; xf = 1;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
        a(i) = ab;
    elseif t(i)<(tf-tb)
        x(i) = 1/2*ab*tb^2+vb*(t(i)-tb);
        v(i) = vb;
        a(i) = 0;
    else
```

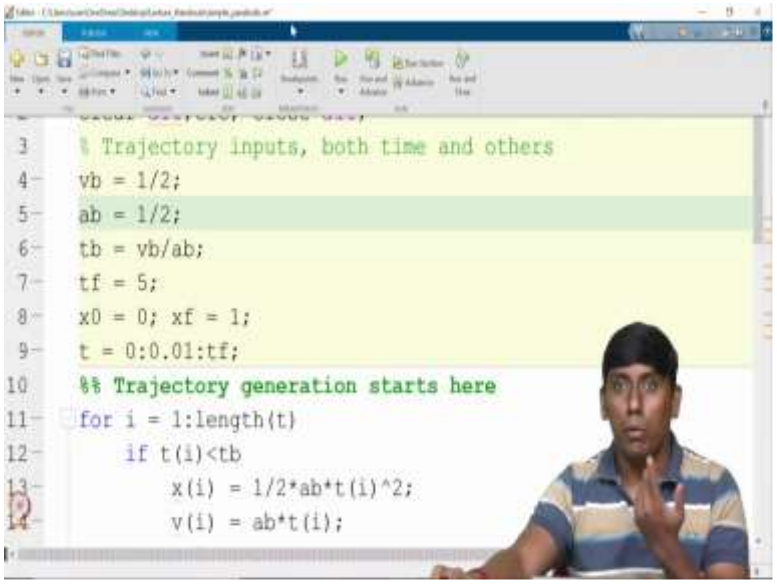


So, instead of that if I keep it this 1 is to 1 so, you will see that this would be following it. So, now, you can see now. So, what is the proposition it is making? So, it is 1.3 or something accordingly you have to make a proposition earlier we have seen it is so 1 by 2 each. So, in the sense it is 0.5 and 0.5 which is equal to you call 1. So, now it is; you can say 0.67 plus 0.67. So that way it is added.

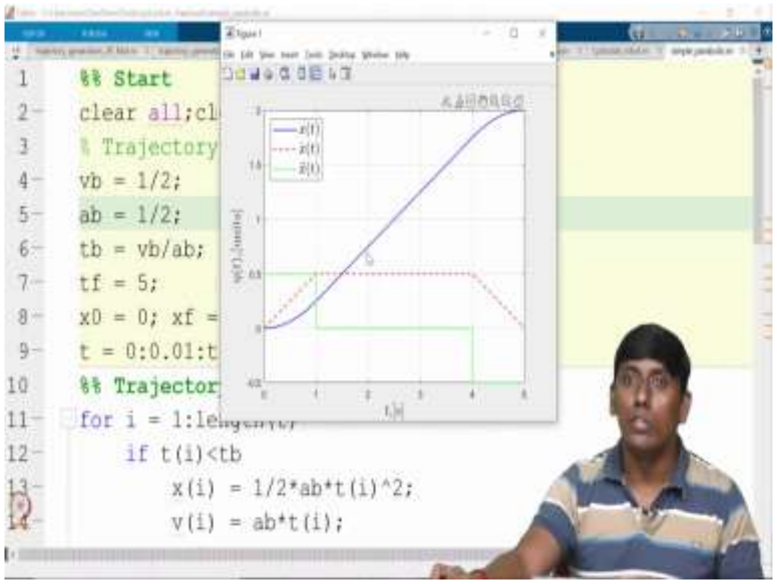
(Refer Slide Time: 19:04)



```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
vb = 1/3;
ab = 1/3;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 1;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```



```
% Trajectory inputs, both time and others
vb = 1/2;
ab = 1/2;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 1;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```

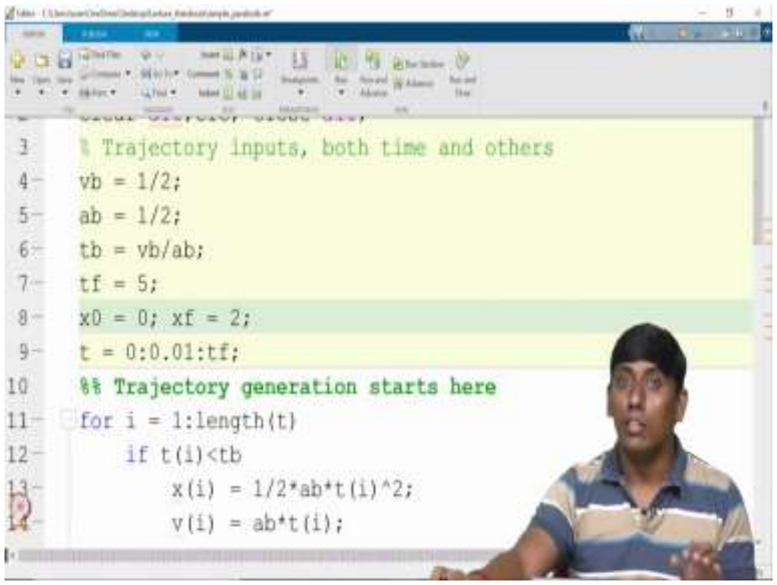


```
%% Start
clear all;cl
% Trajectory
vb = 1/2;
ab = 1/2;
tb = vb/ab;
tf = 5;
x0 = 0; xf =
t = 0:0.01:t
%% Trajector
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```

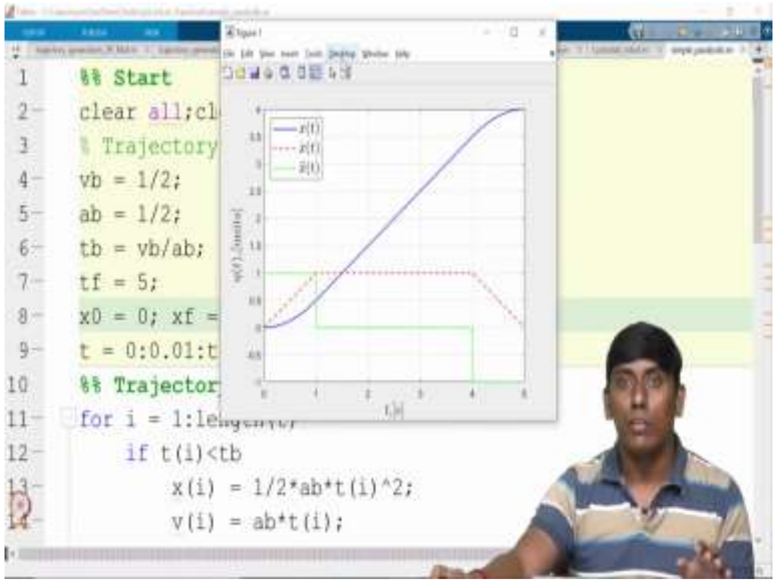
So, what I mean to say so, for example, earlier case we have taken is 1 by 2, 1 by 2 which gives you can say total addition is 1. So, but now here is 1 by 3 and 1 by 3. So, in that sense what would be the x f, so x f would be like a 2 by 3. So that multiplication would come or the other way around, you can make it so you can say this particular proposition you would be adding it.

So, that way we can see it. So, in the sense we can add that. So now we will keep it this as it is. So, that proposition you are attached to like multiply. So that is why the parabolic blend is not so common to most of us. So, because that you little a trickier one. So, now you can feel it this is happening.

(Refer Slide Time: 20:11)



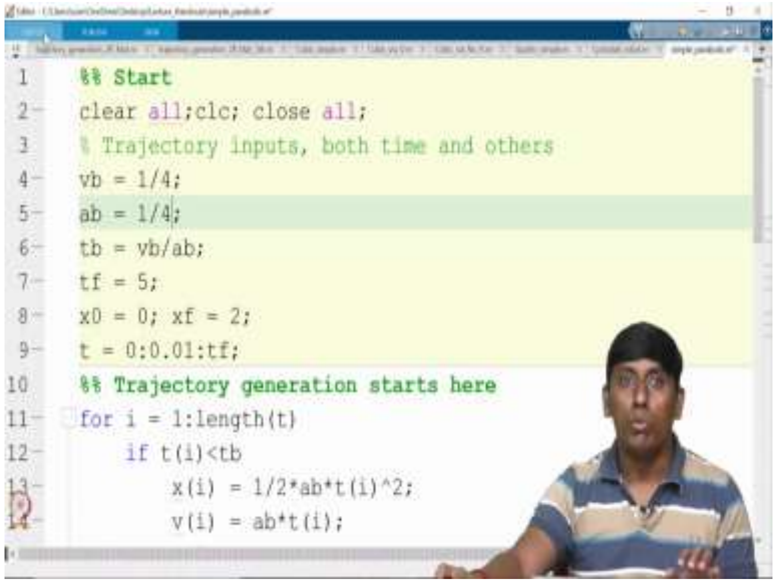
```
% Trajectory inputs, both time and others
vb = 1/2;
ab = 1/2;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 2;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```



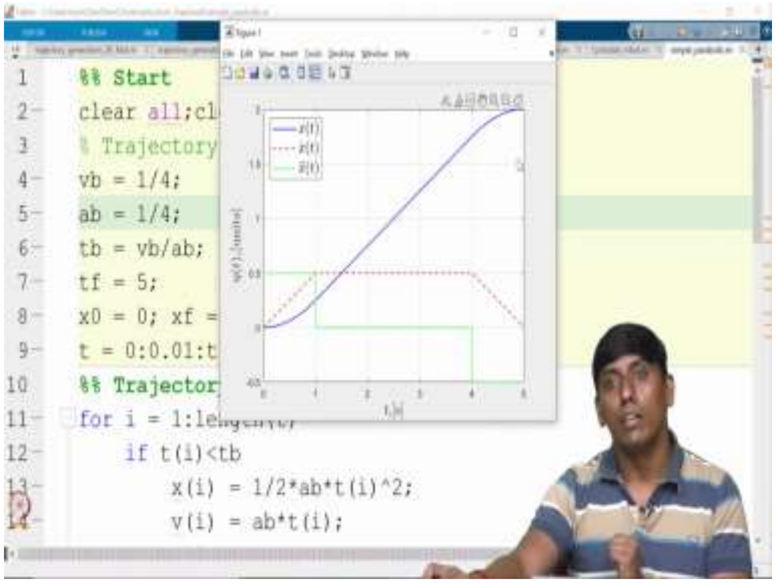
```
%% Start
clear all;cl
% Trajectory
vb = 1/2;
ab = 1/2;
tb = vb/ab;
tf = 5;
x0 = 0; xf =
t = 0:0.01:t
%% Trajector
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```

So, now I just make it this. So, we can see it this will go to 4.

(Refer Slide Time: 20:20)



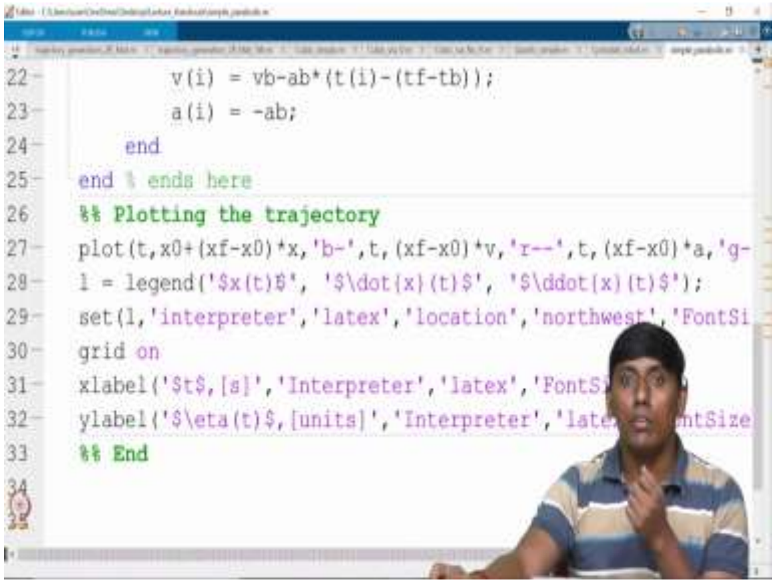
```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
vb = 1/4;
ab = 1/4;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 2;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```



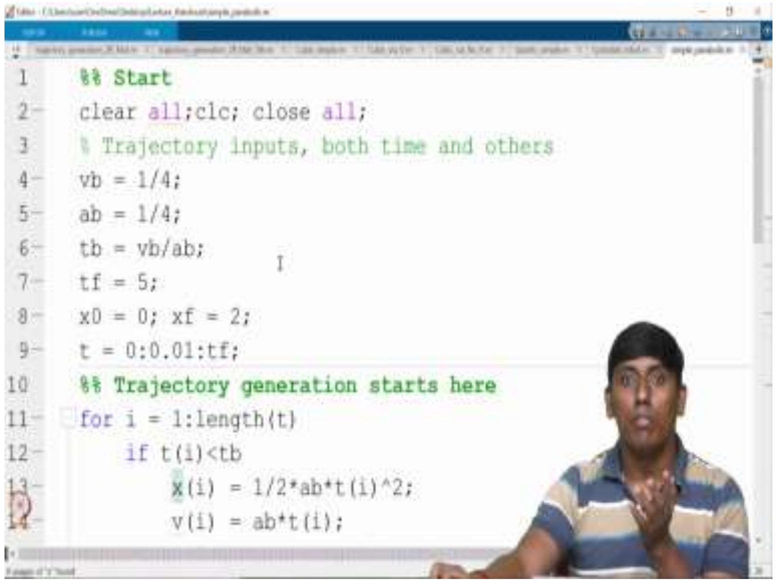
```
%% Start
clear all;cl
% Trajectory
vb = 1/4;
ab = 1/4;
tb = vb/ab;
tf = 5;
x0 = 0; xf =
t = 0:0.01:t
%% Trajector
for i = 1:le
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```

So, now if I make it this is 1 by 4 so, it will remain what you call that aspect you can see it is remain to meter per second or you can say 2 meter. So, these are all what we want to say in a parabolic blend. So, now whatever you are getting this is so, the addition of this, so, v b plus a b whatever addition is coming so, that multiply with the twice so that much would be your proposition.

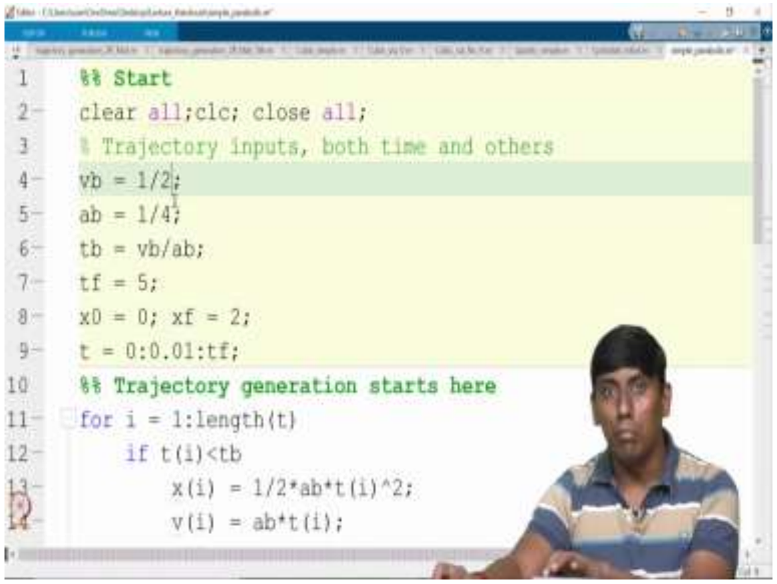
(Refer Slide Time: 20:53)



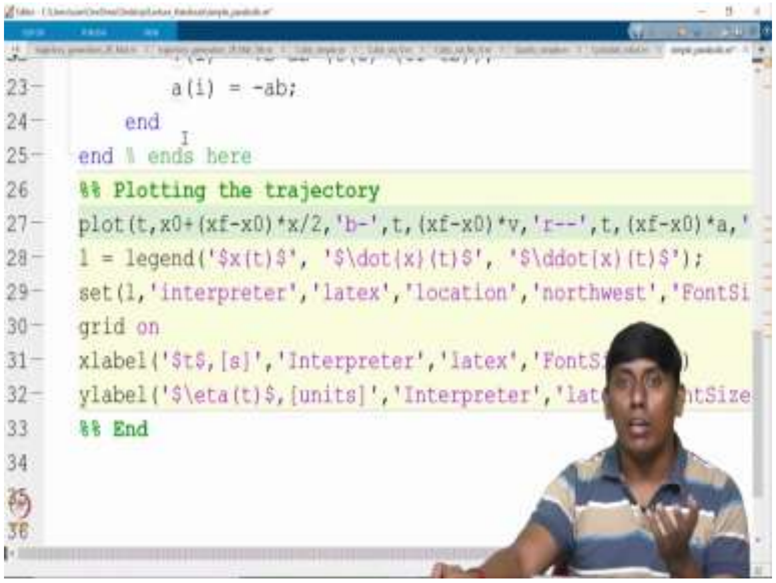
```
        v(i) = vb-ab*(t(i)-(tf-tb));
        a(i) = -ab;
    end
end % ends here
%% Plotting the trajectory
plot(t,x0+(xf-x0)*x,'b-',t,(xf-x0)*v,'r--',t,(xf-x0)*a,'g-
l = legend('$x(t)$', '$\dot{x}(t)$', '$\ddot{x}(t)$');
set(l,'interpreter','latex','location','northwest','FontSi
grid on
xlabel('$t$,[s]','Interpreter','latex','FontS
ylabel('$\eta(t)$,[units]','Interpreter','late    ntSize
%% End
```



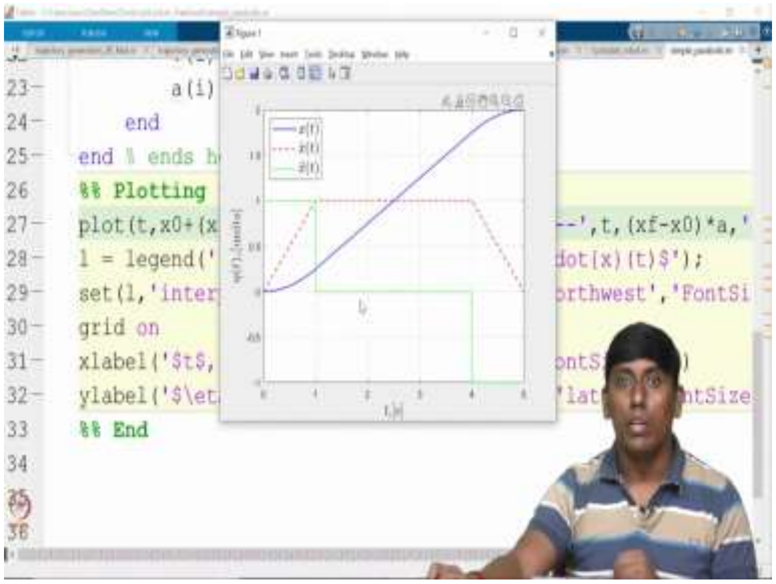
```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
vb = 1/4;
ab = 1/4;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 2;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```



```
%% Start
clear all;clc; close all;
% Trajectory inputs, both time and others
vb = 1/2;
ab = 1/4;
tb = vb/ab;
tf = 5;
x0 = 0; xf = 2;
t = 0:0.01:tf;
%% Trajectory generation starts here
for i = 1:length(t)
    if t(i)<tb
        x(i) = 1/2*ab*t(i)^2;
        v(i) = ab*t(i);
```



```
            a(i) = -ab;
        end
end % ends here
%% Plotting the trajectory
plot(t,x0+(xf-x0)*x/2,'b-',t,(xf-x0)*v,'r--',t,(xf-x0)*a,'
l = legend('$x(t)$', '$\dot(x)(t)$', '$\ddot(x)(t)$');
set(l,'interpreter','latex','location','northwest','FontSi
grid on
xlabel('$t$,[s]','Interpreter','latex','FontS          )
ylabel('$\eta(t)$,[units]','Interpreter','lat         ntSize
%% End
```



```
            a(i)
        end
end % ends h
%% Plotting
plot(t,x0+(x                              --',t,(xf-x0)*a,'
l = legend('                              dot(x)(t)$');
set(l,'inter                              orthwest','FontSi
grid on
xlabel('$t$,                              ontS          )
ylabel('$\et                              'lat          ntSize
%% End
```

So, in that sense, so, here whatever you are going to get so, that this x would be that many times. So, now, if I say this, so, if I make it this is 1 by 2. So, then I can see that this would be so divided by something like so 2. So, if it is 3 then it divided by you can say 2 by 3. So, it is in the other way around, you have to multiply 3 by 2. So, that kind of cases all will come. So, that is what we wanted to see. So, now I just run this and end this.

So, now you can see it. So, these all by experience only will come so that is why most of the industrial manipulator they go with you call cubic polynomial or cycloidal or even you want to control the acceleration they go for you can see the fifth order quintic one. So, with that, I am ending this particular lecture. So, we will see in the next lecture, how we can generate the; you can see manipulator trajectory.

So, for that we will take a simple example of to our serial manipulator, and we will see how the joint space and task space schemes can be generated and realized in the MATLAB. So, with that, see you and thank you. Bye.