Week 10 – Trajectory Planning

Advanced Robotic Systems – MANU2453

Dr Ehsan Asadi, School of Engineering
RMIT University, Victoria, Australia
Email: ehsan.asadi@rmit.edu.au

RMIT UNIVERSITY

Lectures

Wk	Date	Lecture (NOTE: video recording)	Maths Difficulty	Hands-on Activity	Related Assessment
1	24/7	• Introduction to the Course • Spatial Descriptions & Transformations			
2	31/7	• Spatial Descriptions & Transformations • Robot Cell Design			Robot Cell Design Assignment
3	7/8	• Forward Kinematics • Inverse Kinematics			
4	14/8	• ABB Robot Programming via Teaching Pendant • ABB RobotStudio Offline Programming		ABB RobotStudio Offline Programming	Offline Programming Assignment
5	21/8	• Jacobians: Velocities and Static Forces			
6	28/8	• Manipulator Dynamics			
7	11/9	• Manipulator Dynamics		MATLAB Simulink Simulation	
8	18/9	• Robotic Vision		MATLAB Simulation	Robotic Vision Assignment
9	25/9	• Robotic Vision II		MATLAB Simulation	
10	2/10	• Trajectory Generation			
11	9/10	• Linear & Nonlinear Control		MATLAB Simulink Simulation	
12	16/10	• Introduction to I4.0 • Revision			Final Exam

Content

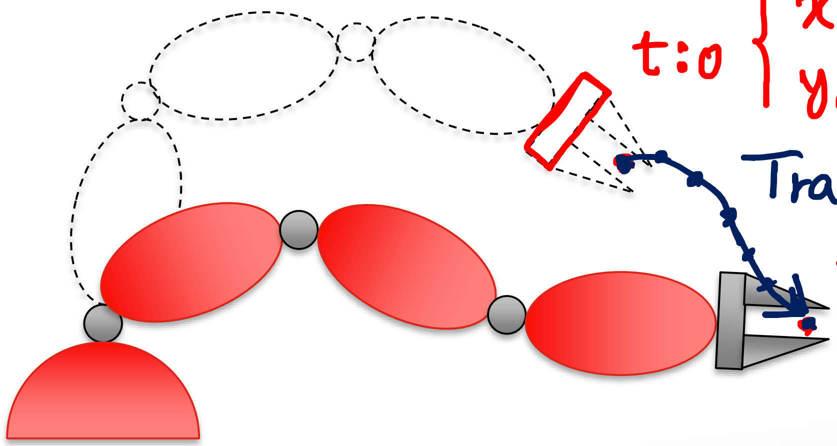
- Introduction
- Cartesian Space Schemes vs. Joint Space Schemes
- Cubic Polynomial
- Quintic Polynomial
- Linear Function with Parabolic Blends
- Matlab / Simulink Simulation

Content

- Introduction
- Cartesian Space Schemes vs. Joint Space Schemes
- Cubic Polynomial
- Quintic Polynomial
- Linear Function with Parabolic Blends
- Matlab / Simulink Simulation

Introduction

- In the past two weeks, you learnt how to find the target using vision. Today you will learn how to specify the path from current position to the target.
    - And next week, you will learn how to control the robot to follow this path.



$t:0 \begin{cases} x_0 = 1 \\ y_0 = 1 \end{cases}$

Trajectory

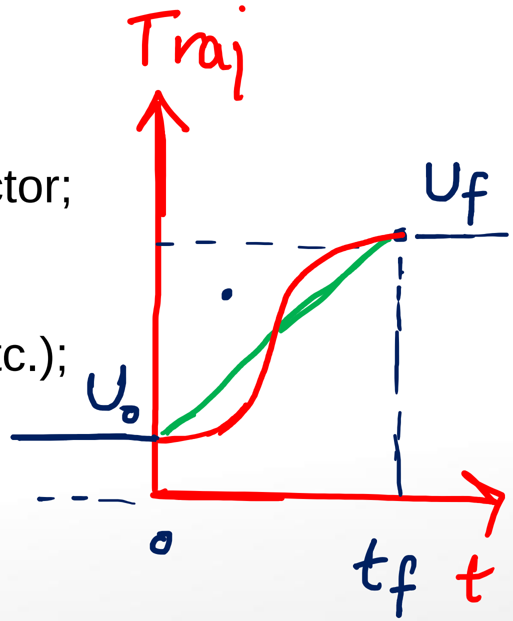
$t: 1\,sec \begin{cases} x_t = 2 \\ y_t = 0 \end{cases}$

t	:	x, y
0	:	1, 1
0.1	:	1.2, 0.8
0.2	:	1.3, 0.7
⋮	⋮	⋮
1	:	2, 0

- Here, we use the term "trajectory" in place of path.
    - Trajectory means a time history of position, velocity and acceleration for each degree of freedom.
    - For e.g. we may specify: at time 0, robot is at x=1,y=1; at time 0.1, robot should go to x=1.2, y=0.8; at time 0.2, x=1.3, y=0.7 etc.

Introduction

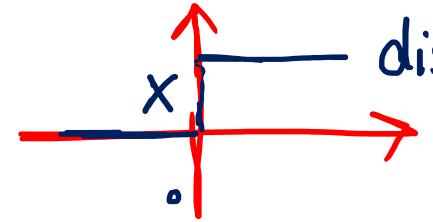
- However, the above example of specifying point by point at different times is not convenient.
- Can we just specify the:
  - Desired goal position and orientation for the end-effector;
  - The time to reach goal position;
  - General shape of the path (straight line, polynomial etc.);
  - (Optional) Some intermediate points / "via points"
- and let the system figure out the trajectory?



- NOTE: The term "points" in this lecture should be interpreted as frames containing both position and orientation!

Introduction

$u(t)$



discontinuous

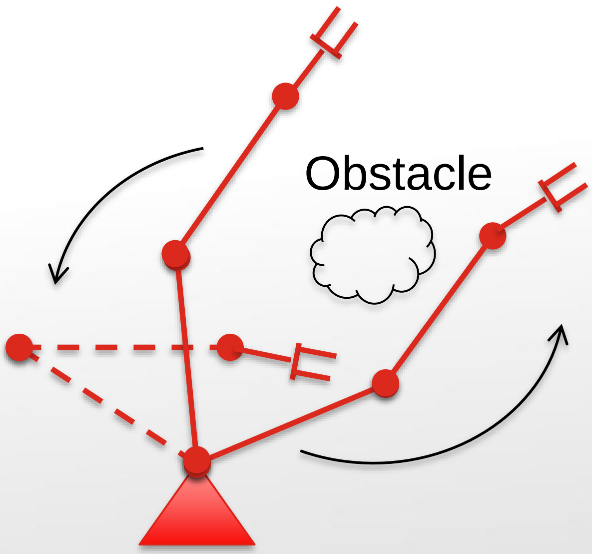
- It is often desirable that the motion of the manipulator to be smooth.
  - Continuous function, with continuous first derivative.

    $u(t)$ & $\dot{u}(t)$

  - It's even better if second derivative is continuous.

    $\ddot{u}(t)$

  - Reason: Rough and jerky motions causes vibrations due to resonance modes, as well as increases wear and tear.
- "Via points" are usually given for the purpose of collision avoidance.

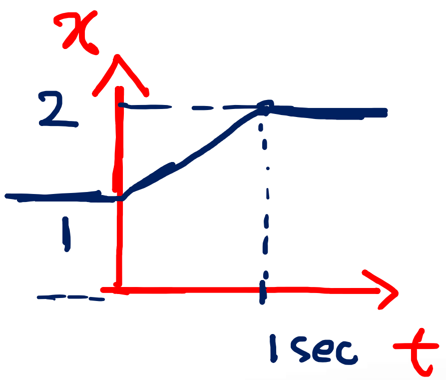


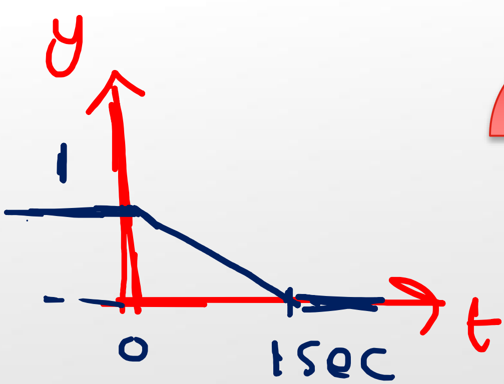
Content

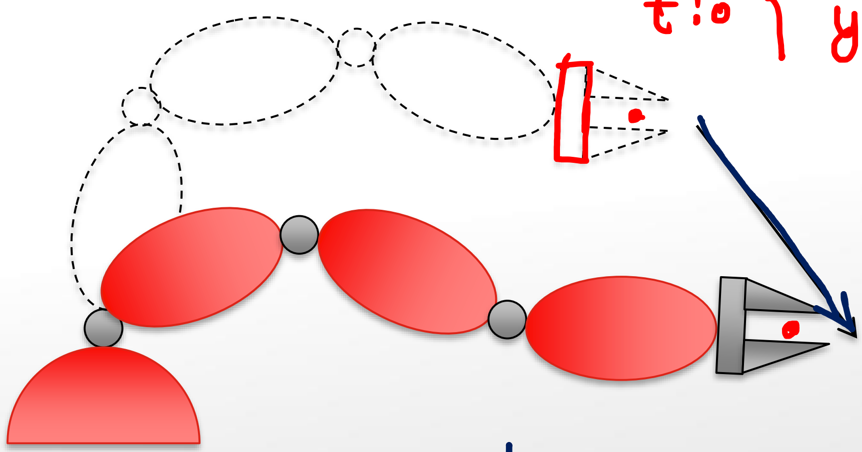
- Introduction
- Cartesian Space Schemes vs. Joint Space Schemes
- Cubic Polynomial
- Quintic Polynomial
- Linear Function with Parabolic Blends
- Matlab / Simulink Simulation

Cartesian Space Schemes

- Cartesian Space Schemes means specifying trajectory directly through position and orientation of the end-effector.
  - The advantage of Cartesian Space Schemes is that we can enforce certain shape of the trajectory (for e.g. straight line), or enforce orientation of the end-effector (for e.g. maintain same orientation throughout).





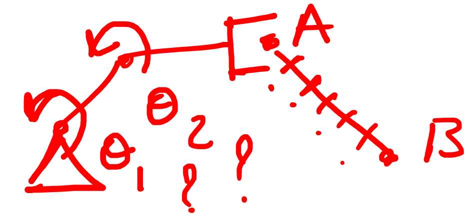


$t:0 \begin{cases} x=1 \\ y=1 \end{cases}$

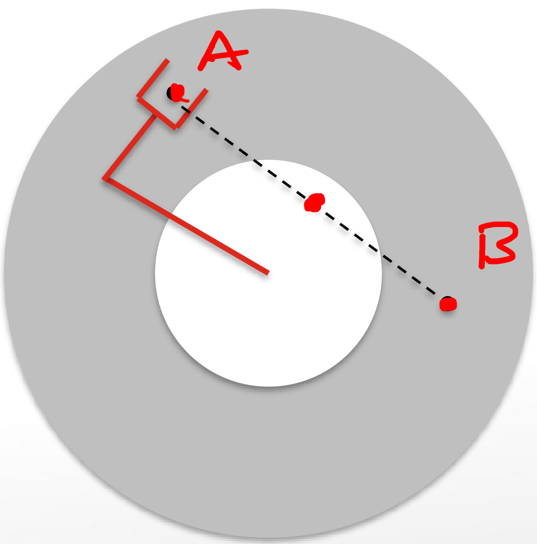
$t:1\,sec \begin{cases} x=2 \\ y=0 \end{cases}$

MoveL

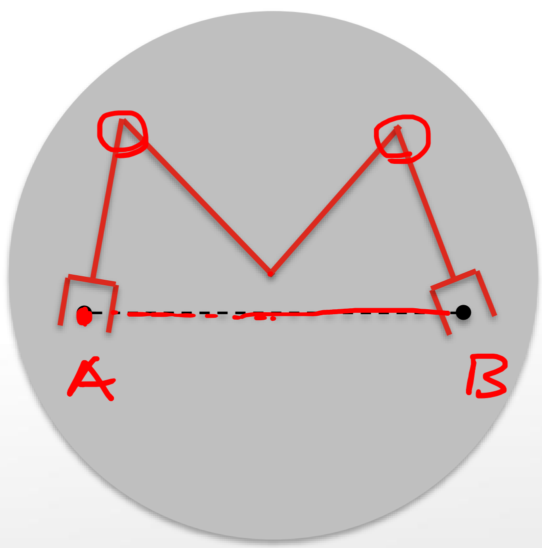
Cartesian Space Schemes



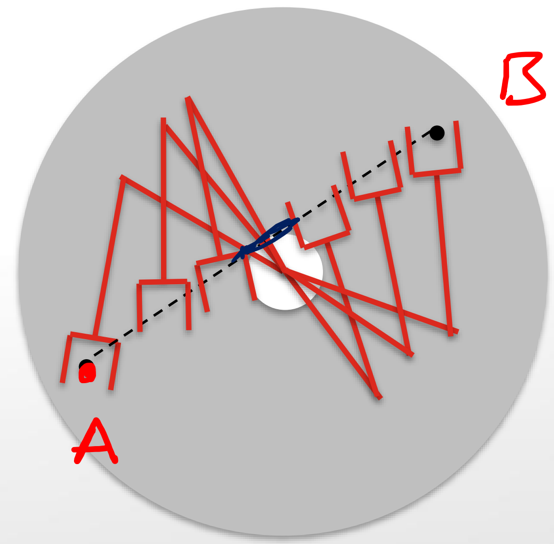
- However, Cartesian Space Schemes also have quite a few disadvantages:
  - Computationally expensive: After path is generated, inverse kinematics has to be solved at every time step (update rate) to calculate joint angles.
  - Prone to problems relating to workspace and singularities



Intermediate points not reachable



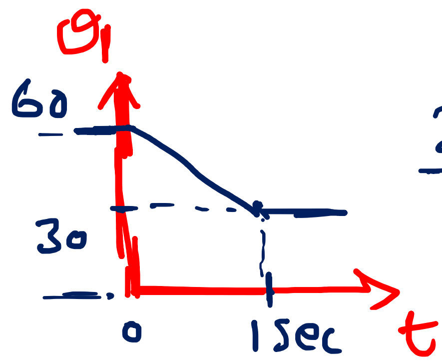
Start and end points reachable but in different configurations

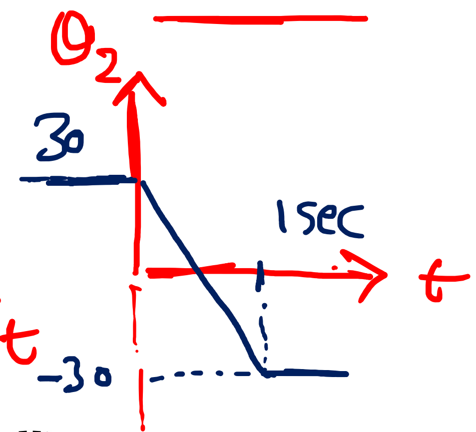


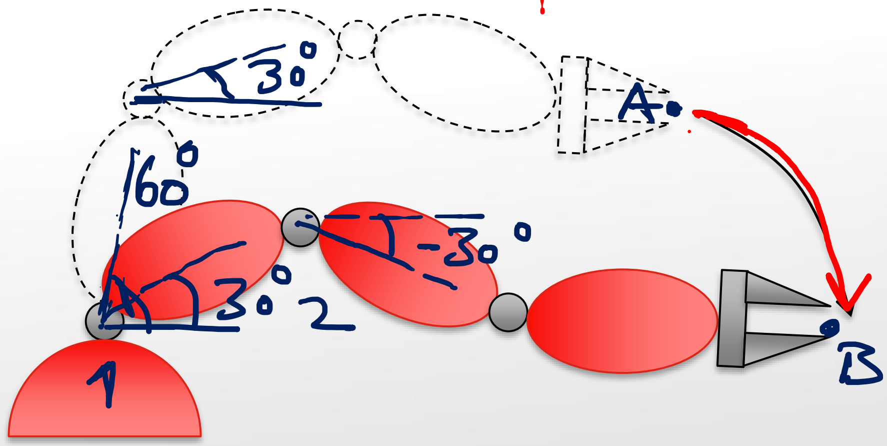
High joint rates near Singularities

Joint Space Schemes

- Joint Space Schemes means specifying trajectory directly through the joint angles.
  - Advantages:
    - Easy to compute.
    - No issue with singularities.
  - Disadvantage:
    - Path will not be linear.
    - This may be a problem if there are possible collisions.







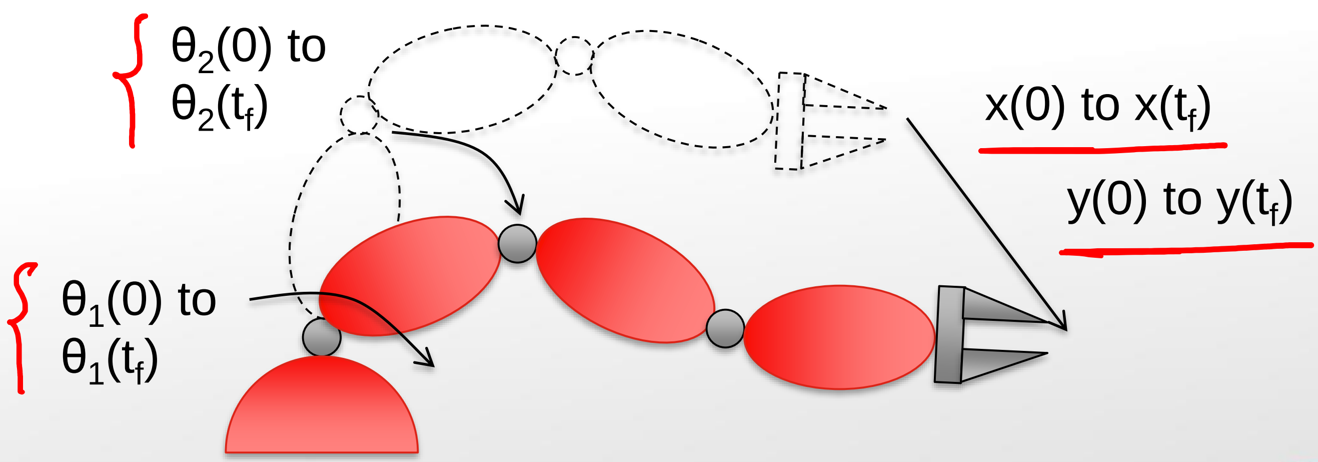
Move J

Content

- Introduction
- Cartesian Space Schemes vs. Joint Space Schemes
- Cubic Polynomial
- Quintic Polynomial
- Linear Function with Parabolic Blends
- Matlab / Simulink Simulation

General Solution

- In this section, we will use "u(t)" to represent any of the variables, be it the Cartesian terms (x, y, z, angles) or the joint variables (θ).
- A reminder of our question:
  - Given the start, end, and possibly some via points;
  - And given the time to reach goal position;
  - Generate a trajectory u(t) for the robot to follow.



Use "u(t) to represent any of the variables. Therefore: $u(0)$ to $u(t_f)$

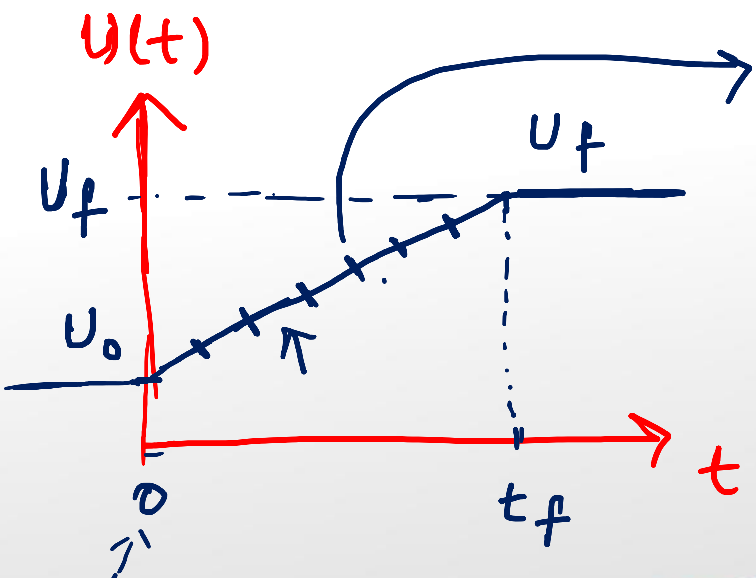
Straight Line

FwK → u_0 Vision → u_f

- We want to move from initial position to target position within time t_f.
- To do this, the general variable "u(t)" will change from its initial value to the target value within time t_f.

$$u(t): \begin{cases} u(0) = u_0 \\ u(t_f) = u_f \end{cases}$$

$$y = mx + c$$
$$u = mt + c$$



$$u(t) = \underbrace{\left(\frac{u_f - u_0}{t_f}\right)}_{\text{known}} t + \underbrace{u_0}_{\text{known}}$$

$t = 0.1$: $u(0.1) = m \times 0.1 + u_0$

$t = 0.2$: $u(0.2) = m \times 0.2 + u_0$

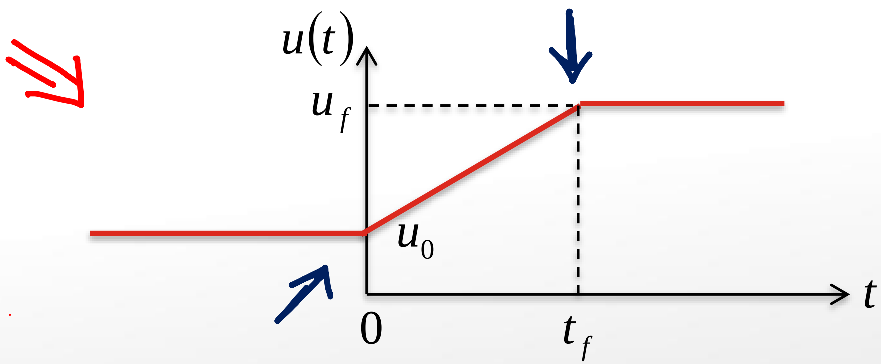
$t = t_f$

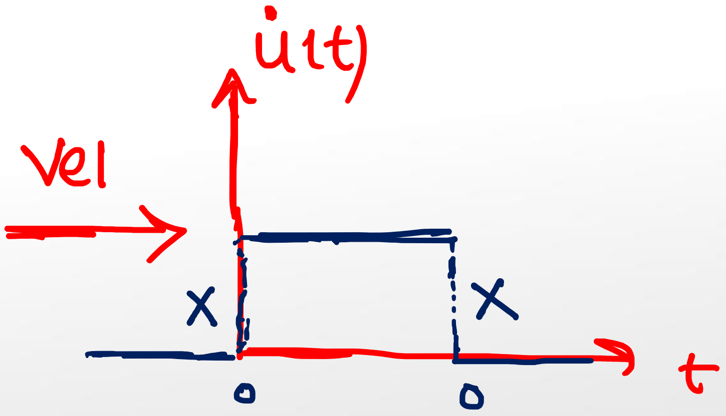
Straight Line

- We want to move from initial position to target position within time t_f.
- To do this, the general variable "u(t)" will change from its initial value to the target value within time t_f.

$$u(0) = u_0$$
$$u(t_f) = u_f$$

- The simplest path would be a straight line between the two values.





- Disadvantage: Discontinuous velocities at start and end points.

wear & Tear

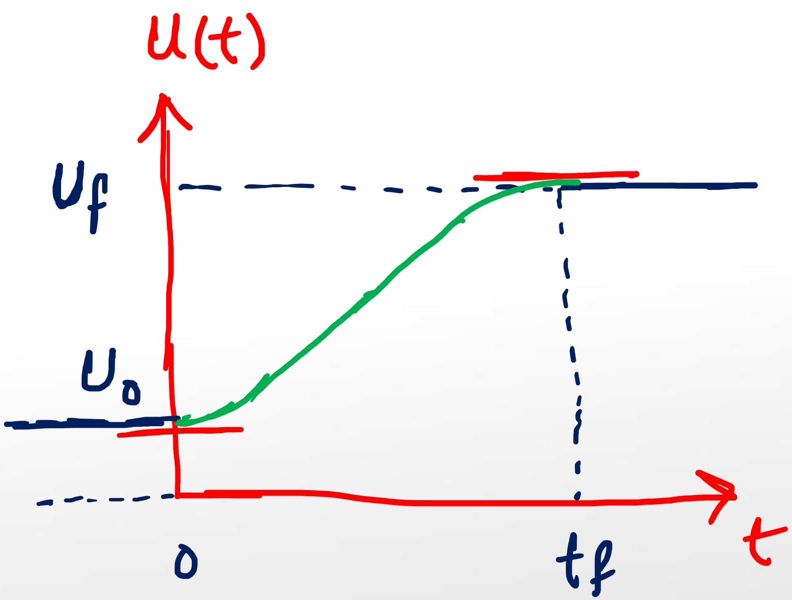
Cubic Polynomial

→ Smooth velocity

- To ensure that the velocities at the start and end points are zero, we can use a cubic polynomial:

$$u(t) = \underline{a_0} + \underline{a_1} t + \underline{a_2} t^2 + \underline{a_3} t^3$$

- There are four parameters which can satisfy four constraints:



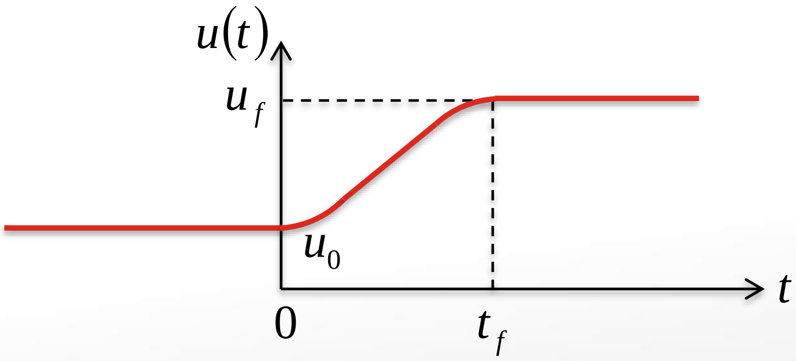
$$\begin{cases} u(0) = u_0 \\ \dot{u}(0) = 0 \\ u(t_f) = u_f \\ \dot{u}(t_f) = 0 \end{cases}$$

Cubic Polynomial

- To ensure that the velocities at the start and end points are zero, we can use a cubic polynomial: $u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$

- There are four parameters which can satisfy four constraints:

$$\begin{aligned} u(0) &= u_0 \\ u(t_f) &= u_f \\ \dot{u}(0) &= 0 \\ \dot{u}(t_f) &= 0 \end{aligned}$$



- The task is then to calculate the parameters a_0, a_1, a_2 and a_3.

Cubic Polynomial

- This can be done by solving the following simultaneous equations:

$$\rightarrow \quad u(0) = a_0 + a_1 0 + a_2 0^2 + a_3 0^3 = u_0 \quad \Longrightarrow \quad a_0 = u_0 \quad (1)$$

$$\rightarrow \quad \dot{u}(0) = a_1 + 2a_2 0 + 3a_3 0^2 = 0 \quad \Longrightarrow \quad a_1 = 0 \quad (2)$$

$$(1\text{-}2)\ \& \quad u(t_f) = \overset{u_0}{a_0} + \cancel{a_1}^{0} t_f + a_2 t_f^2 + a_3 t_f^3 = u_f$$

$$(2)\ \& \quad \dot{u}(t_f) = \cancel{a_1}^{0} + 2a_2 t_f + 3a_3 t_f^2 = 0$$

$$+3\times \quad a_2 t_f^2 + a_3 t_f^3 = U_f - U_0$$

$$-t_f\times \quad 2a_2 t_f + 3a_3 t_f^2 = 0$$

$$a_2 t_f^2 + 0 = 3(U_f - U_0)$$

$$(3) \quad a_2 = \frac{3}{t_f^2}(U_f - U_0)$$

- And the solution is:

$$(1)\ a_0 = u_0 \qquad (2)\ a_1 = 0 \qquad a_2 = \frac{3}{t_f^2}(u_f - u_0) \qquad a_3 = -\frac{2}{t_f^3}(u_f - u_0)$$

Cubic Polynomial

- Example:

$$t_f = 3\text{sec}$$
$$u(0) = u_0 = 15\deg$$
$$u(t_f) = u_f = 75\deg$$
$$\dot{u}(0) = 0$$
$$\dot{u}(t_f) = 0$$

$$u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

- The solution is:

$$a_0 = u_0 = 15 \qquad a_1 = 0 \qquad a_2 = \frac{3}{t_f^2}(u_f - u_0) = 20 \qquad a_3 = -\frac{2}{t_f^3}(u_f - u_0) = -4.44$$

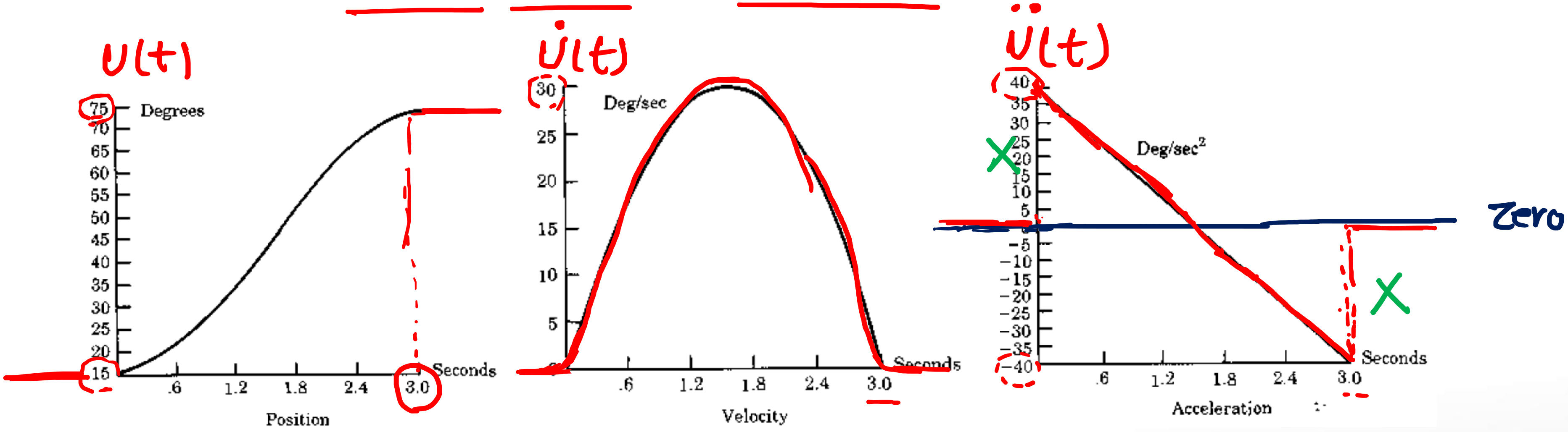
$$a_0 = 15 \qquad a_1 = 0 \qquad a_2 = \frac{3}{3^2}(75-15) \qquad a_3 = \frac{2}{3^3}(75-15)$$

- Thus the trajectory is:

$$u(t) = 15 + 20t^2 - 4.44t^3$$

Cubic Polynomial

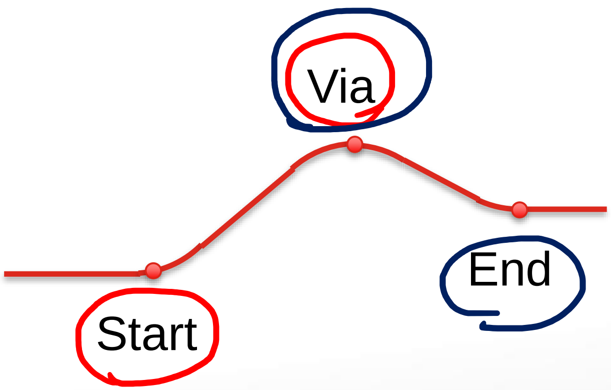
- The plots for position, velocity and acceleration are:

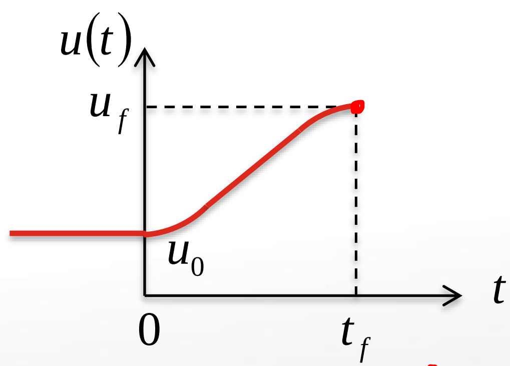


- Note that the accelerations at start and end positions are not zero.
  - This might create jerky motions.

Cubic Polynomial – Via Points

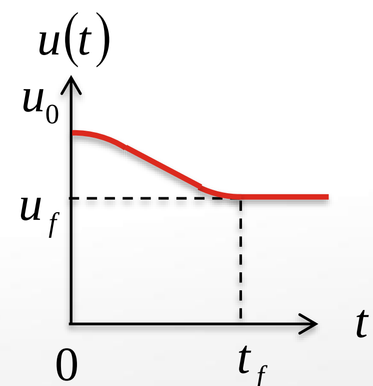
- We have looked at cubic polynomial connecting start and end point.
- What if we need the path to pass through a via point?
- Simple! Just split the path into segments (start to via point, via point to end) and derive cubic polynomial for each of them.





Segment 1

$\dot{u}(t_f)$?



Segment 2

$\dot{u}(0)$?

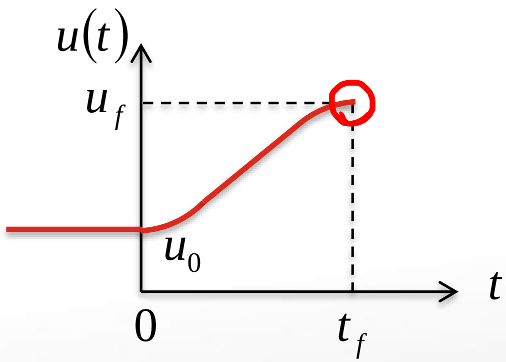
- However, the velocity at via point need not be zero:

Cubic Polynomial – Via Points

- For each segment, we will solve for a_0, a_1, a_2 and a_3 for the cubic polynomial

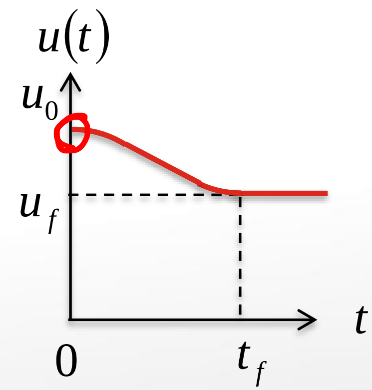
$$u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

- Of course, the constraints will be different:



Segment 1

$$\begin{aligned} u(0) &= u_0 \\ u(t_f) &= u_{\text{via}} \\ \dot{u}(0) &= 0 \\ \boxed{\dot{u}(t_f) = \dot{u}_{\text{via}}} \end{aligned}$$

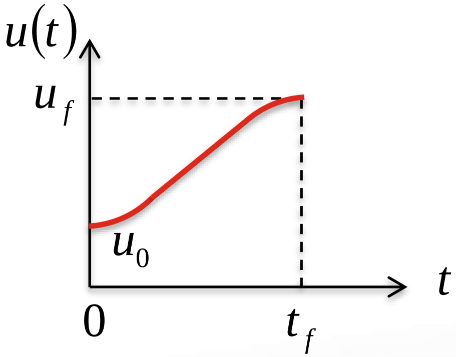


Segment 2

$$\begin{aligned} u(0) &= u_{\text{via}} \\ u(t_f) &= u_f \\ \boxed{\dot{u}(0) = \dot{u}_{\text{via}}} \\ \dot{u}(t_f) &= 0 \end{aligned}$$

Cubic Polynomial – Via Points

- For an even more general case where we have more than 1 via point, we will do the same: Just split the path into many segments and solve the simultaneous solutions for each of the segments.
- The start and end velocities of the i^{th} segment need not be zero. Therefore:



Segment i

$$u(0) = u_{\text{via}(i-1)}$$
$$u(t_f) = u_{\text{via}(i)}$$
$$\dot{u}(0) = \dot{u}_{\text{via}(i-1)}$$
$$\dot{u}(t_f) = \dot{u}_{\text{via}(i)}$$

Cubic Polynomial

- The simultaneous equations to be solved are thus:

$$u(0) = a_0 + a_1 0 + a_2 0^2 + a_3 0^3 = u_{\text{via}(i-1)}$$

$$u(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 = u_{\text{via}(i)}$$

$$\dot{u}(0) = a_1 + 2a_2 0 + 3a_3 0^2 = \dot{u}_{\text{via}(i-1)}$$

$$\dot{u}(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 = \dot{u}_{\text{via}(i)}$$

By differentiation of

$$u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

- And the solution is:

$$a_0 = u_{\text{via}(i-1)} \qquad a_1 = \dot{u}_{\text{via}(i-1)}$$

$$a_2 = \frac{3}{t_f^2}\left(u_{\text{via}(i)} - u_{\text{via}(i-1)}\right) - \frac{2}{t_f}\dot{u}_{\text{via}(i-1)} - \frac{1}{t_f}\dot{u}_{\text{via}(i)}$$

$$a_3 = -\frac{2}{t_f^3}\left(u_{\text{via}(i)} - u_{\text{via}(i-1)}\right) + \frac{1}{t_f^2}\left(\dot{u}_{\text{via}(i)} - \dot{u}_{\text{via}(i-1)}\right)$$

Content

- Introduction
- Cartesian Space Schemes vs. Joint Space Schemes
- Cubic Polynomial
- Quintic Polynomial
- Linear Function with Parabolic Blends
- Matlab / Simulink Simulation

Thank you!

Have a good evening.