Week 8 – Trajectory Planning

ELEC0129 Introduction to Robotics

Dr. Chow Yin Lai

Email: uceecyl@ucl.ac.uk

Schedule

Legend:
A: MPEB 6th Floor Lab
B: PC Lab

Legend:
C: 11am-1pm
D: 11.30am-1pm

Week	Recorded, uploaded by Friday of previous week	F2F Workshop, Mondays 11am-1pm and/or Wednesdays 9am-11am	Virtual Workshop, Mondays (C/D) and Wednesdays 9am-11am
1	(Scenario Week)		
2	Lec: Intro; Spatial description	A: Workshop: Offline programming	C: Workshop: Offline programming
3	Lec & Tut: Spatial description	A: Workshop: Build robot	D: Workshop: Build robot
4	Lec & Tut: Forward kinematics	A: Workshop: Forward kinematics	D: Workshop: Forward kinematics
5	Lec & Tut: Inverse kinematics	B: Workshop: Offline programming	C: Workshop: Offline programming
RW	(Reading Week)		
6	(Scenario Week)		
7	Lec & Tut: Jacobians	A: Workshop: Inverse kinematics	D: Workshop: Inverse kinematics
8	Lec: Trajectory Planning	A: Workshop: Trajectory planning	D: Workshop: Trajectory planning
9	Lec & Tut: Dynamics	A: Workshop: Trajectory planning	C: Workshop: Trajectory planning
10	Lec & Tut: Control	A: Workshop: Pick-and-place demo	C: Workshop: Pick-and-place demo

Content

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

Content

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

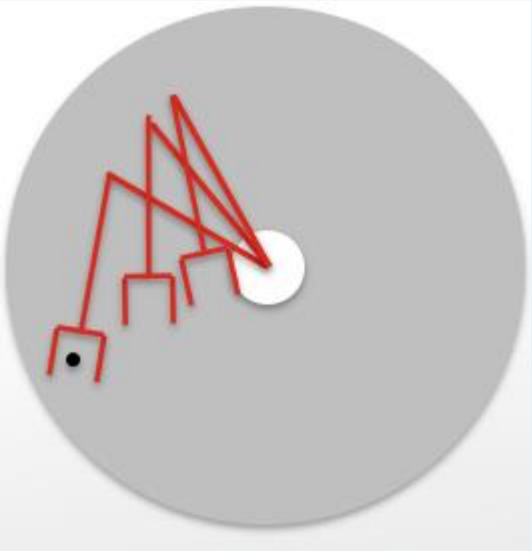
Introduction to Trajectory Planning (1)

- Trajectory planning means designing a time profile of position, velocity and acceleration of a movement.
- E.g. driving from A to B:

  - The velocity starts from 0, increases to V_{constant} and stays constant for some time.
  - As it approaches the target, the velocity decreases down to 0.

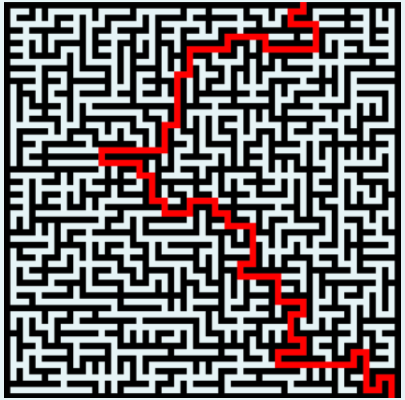
Introduction to Trajectory Planning (2)

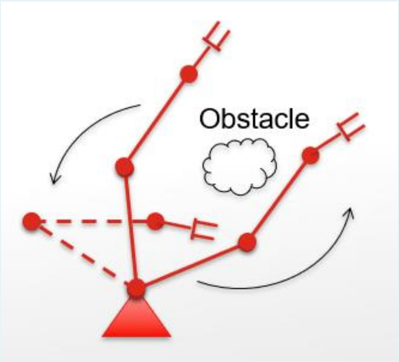
- E.g. robot arm moving from initial position towards object:
  - The arm also accelerates at the start and decelerates at the end.



Introduction to Trajectory Planning (3)

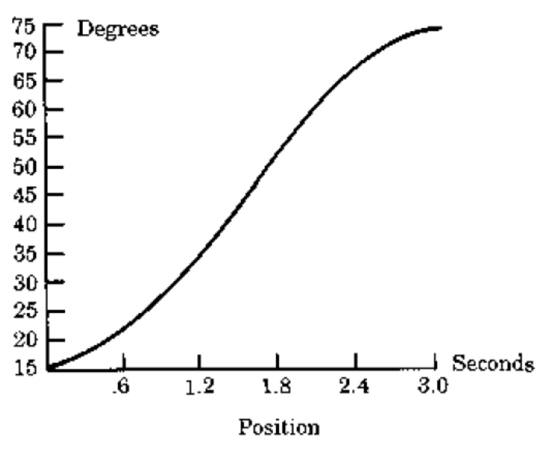
- Note: Trajectory planning is NOT designing the “geographical” path.
  - E.g. the discussion is NOT about how to generate a path for the car to move from A to B through the city.
  - Or how to generate a path for robot to avoid some obstacles.
- It is about the time profile of the motion.

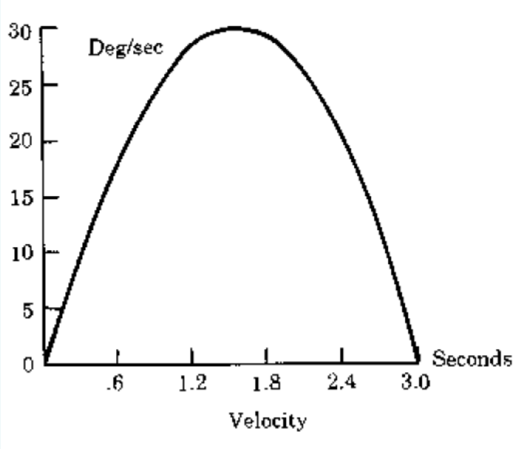


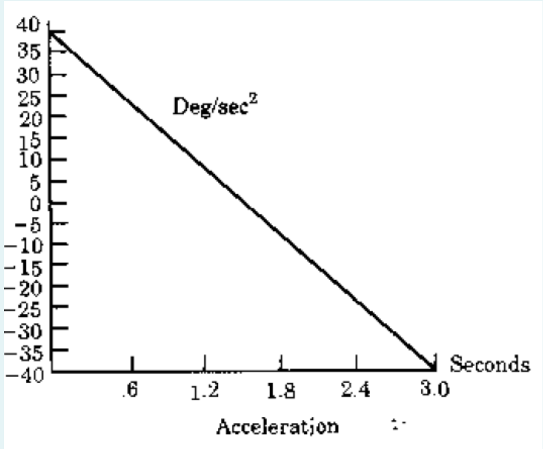


Introduction to Trajectory Planning (4)

- The designed time profile of position, velocity and acceleration can be visualized in graphs, for e.g.:







Introduction to Trajectory Planning (5)

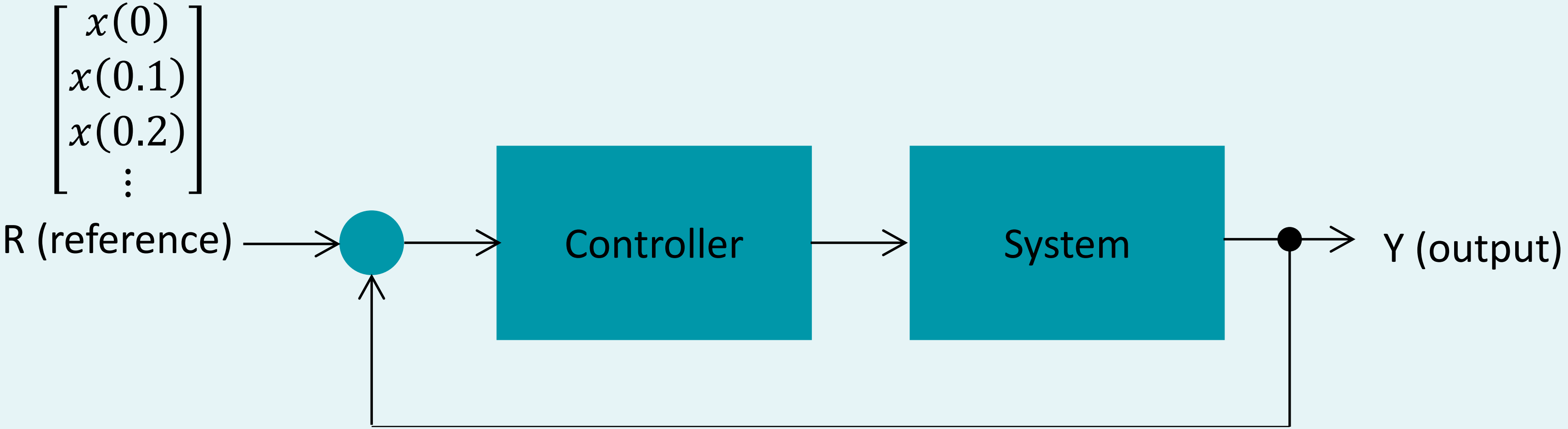
- However, we still need to express the time profile using mathematical functions.

$$\boxed{x = f(t), \dot{x} = f'(t), \ddot{x} = f''(t)}$$

- From the functions, the computer can then extract the numerical values of $x, \dot{x}, \ddot{x}$ at any given t.

Introduction to Trajectory Planning (6)

- The numerical values of $x, \dot{x}, \ddot{x}$ at different times t will be passed on to the control system as reference values for the system (car, robot) to follow.



Introduction to Trajectory Planning (7)

- In today's lecture, we will learn how to write the function $\boxed{x = f(t)}$ given:
  - The current position and orientation of robot / end-effector;
  - Desired goal position and orientation for the end-effector;
  - The time to reach goal position;
  - General shape of the path (straight line, polynomial etc.);

- $\boxed{\dot{x} = f'(t), \ddot{x} = f''(t)}$ then comes naturally through differentiation.

Content

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

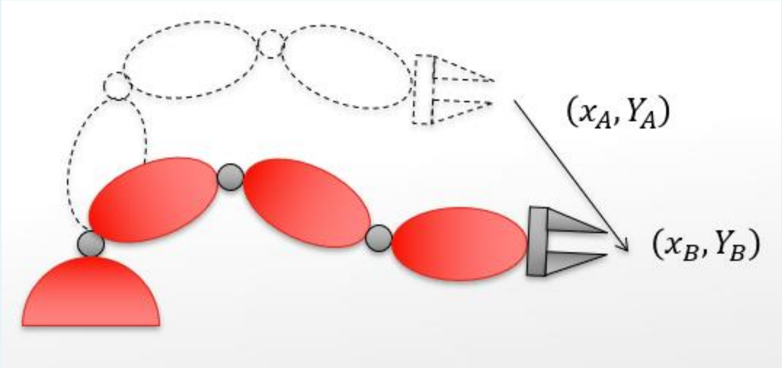
Cartesian Space Schemes (1)

- Cartesian Space Schemes means specifying trajectory directly through position and orientation of the end-effector.

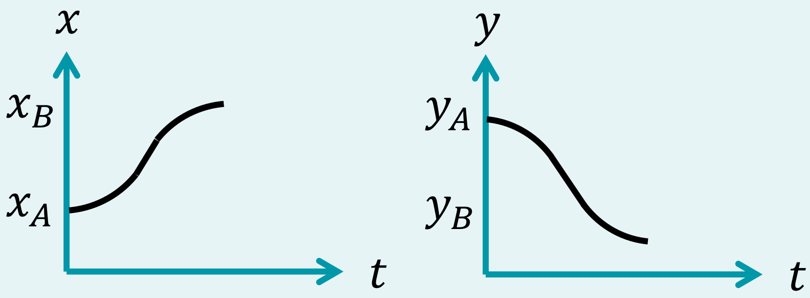
$$
\boxed{
\begin{aligned}
x &= f_x(t) \\
y &= f_y(t) \\
z &= f_z(t) \\
r_x &= f_{rx}(t) \\
r_y &= f_{ry}(t) \\
r_z &= f_{rz}(t)
\end{aligned}
}
$$

Cartesian Space Schemes (2)

- For e.g. the end-effector is required to move from (x_A, y_A) to (x_B, y_B).

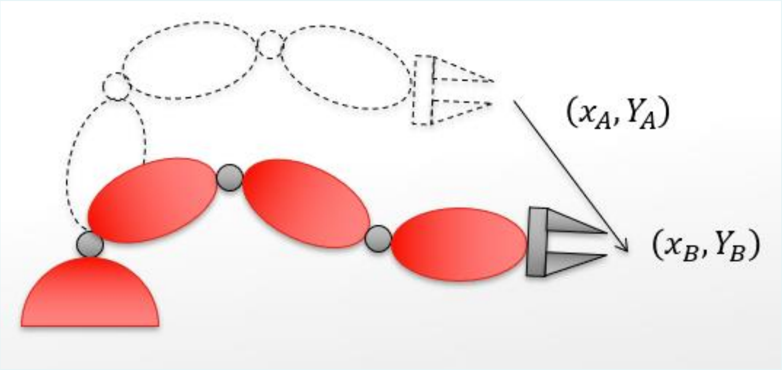


- The $x = f_x(t)$ and $y = f_y(t)$ graphs may look like this:



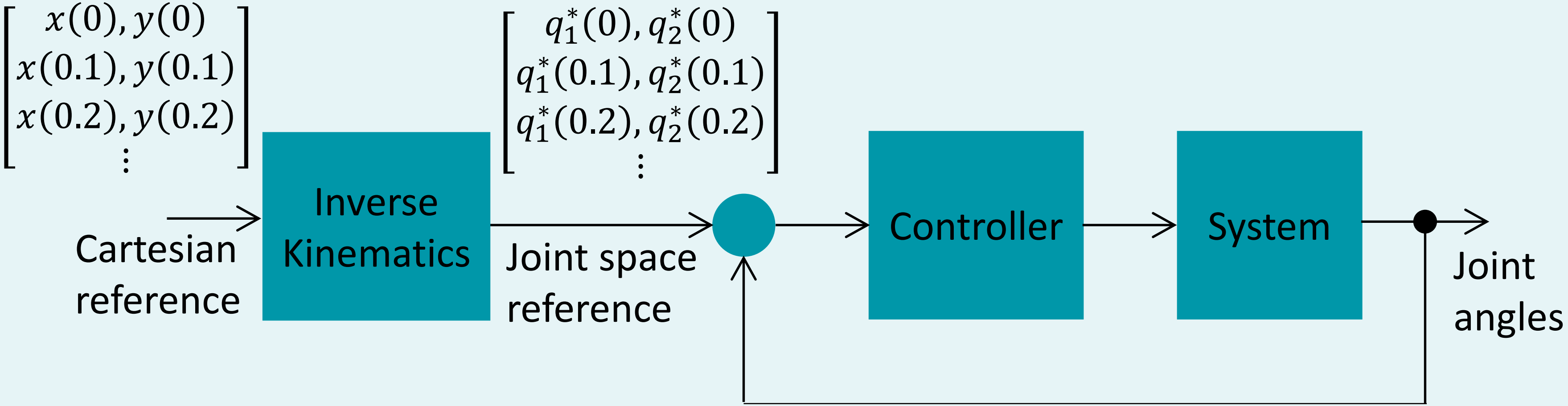
Cartesian Space Schemes (3)

- The advantage of Cartesian Space Schemes:
  - We can enforce certain shape of the geometrical path (for e.g. straight line),
  - Or enforce orientation of the end-effector (for e.g. maintain same orientation throughout).



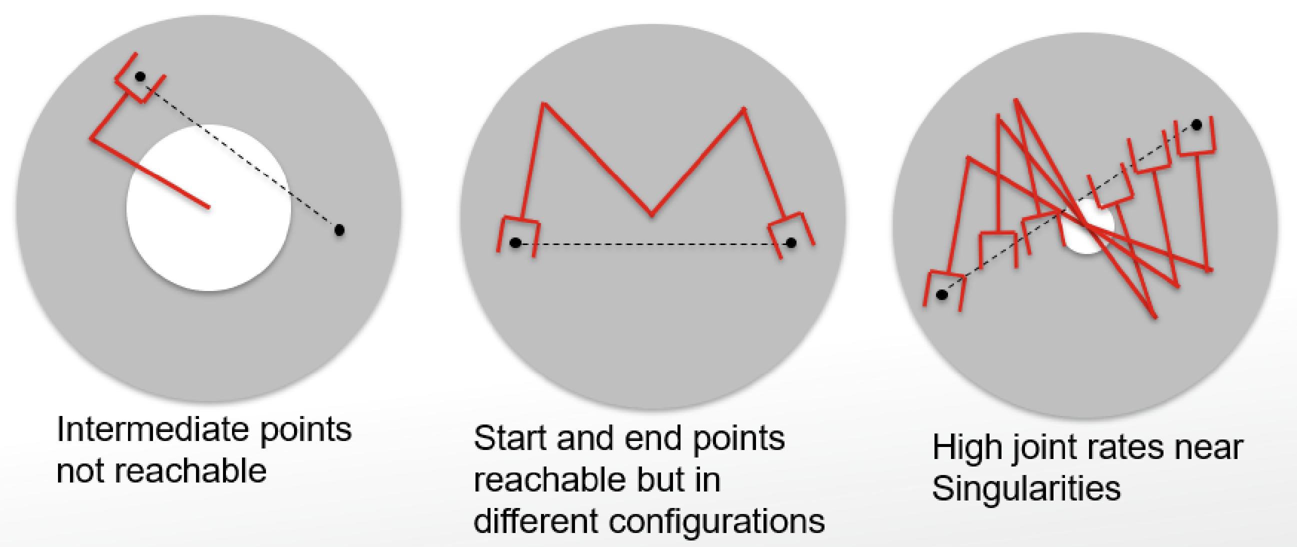
Cartesian Space Schemes (4)

- The disadvantages of Cartesian Space Schemes:
  1. Computationally expensive: Inverse kinematics has to be solved at every time step (update rate) to calculate joint angles.



Cartesian Space Schemes (5)

- The disadvantages of Cartesian Space Schemes:
  2. Prone to problems relating to workspace and singularities



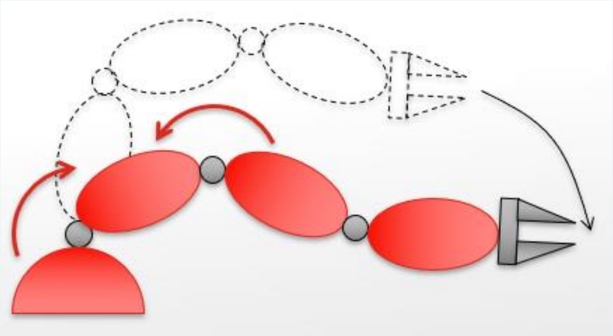
Joint Space Schemes (1)

- Joint Space Schemes means specifying trajectory directly through the joint angles.

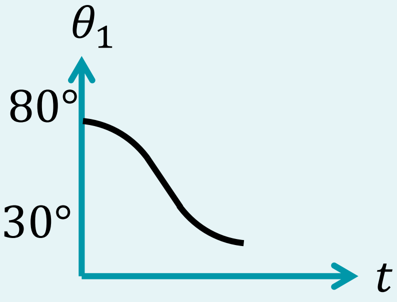
$$\boxed{\begin{aligned}
\theta_1 &= f_{\theta 1}(t)\\
\theta_2 &= f_{\theta 2}(t)\\
\theta_3 &= f_{\theta 3}(t)\\
\theta_4 &= f_{\theta 4}(t)\\
\theta_5 &= f_{\theta 5}(t)\\
\theta_6 &= f_{\theta 6}(t)
\end{aligned}}$$

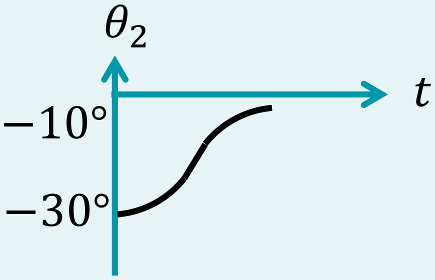
Joint Space Schemes (2)

- For e.g. θ_1 is required to change from 80° to 30°, while θ_2 is required to change from -30° to -10°.



- The $\theta_1 = f_{\theta 1}(t)$ and $\theta_2 = f_{\theta 2}(t)$graphs may look like this:

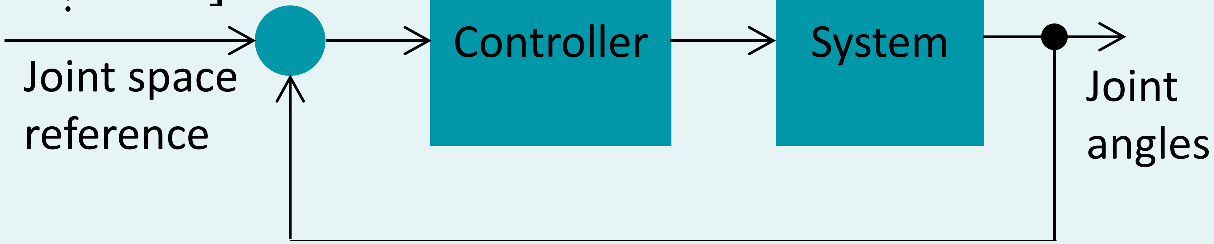




Joint Space Schemes (3)

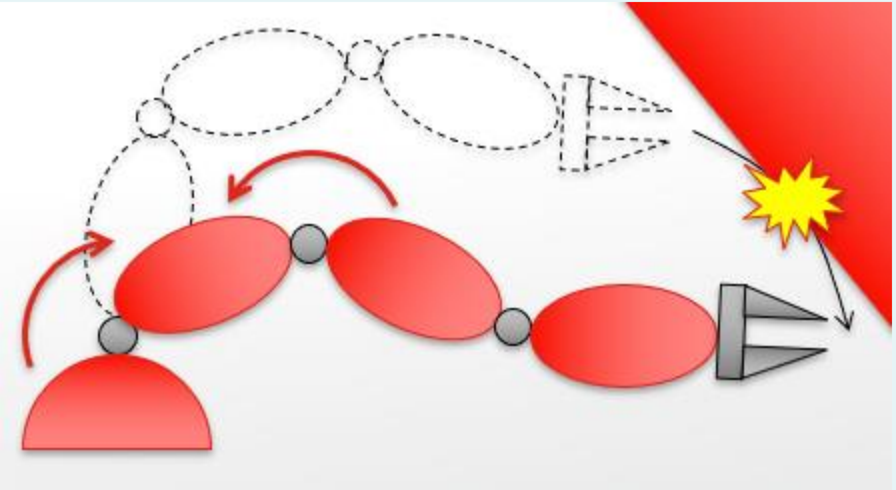
- Advantages of joint space schemes:
  - Easy to compute.
  - No issue with singularities.

$$\begin{bmatrix} q_1^*(0), q_2^*(0) \\ q_1^*(0.1), q_2^*(0.1) \\ q_1^*(0.2), q_2^*(0.2) \\ \vdots \end{bmatrix}$$



Joint Space Schemes (4)

- Disadvantages of joint space schemes:
  - Path will not be linear.
  - This may be a problem if there are possible collisions.



Notation

- Regardless of whether the trajectory is specified in Cartesian space of joint space,
- The design process (or the mathematical function) is the same.
- We will therefore use "$u(t)$" to represent any of the variables:

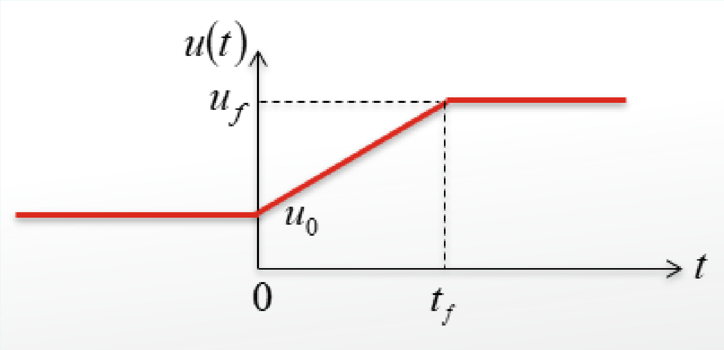
$$\boxed{u = f_u(t)}$$

Summary: What we want to do next

- Given:
  - The start position u_0,
  - The end position u_f,
  - The time to reach the final position t_f;
- Generate a trajectory $u = f_u(t)$ for the robot to follow.

Content

Straight Line



- The simplest trajectory would be a straight time profile between the start and end positions.
- The function is:

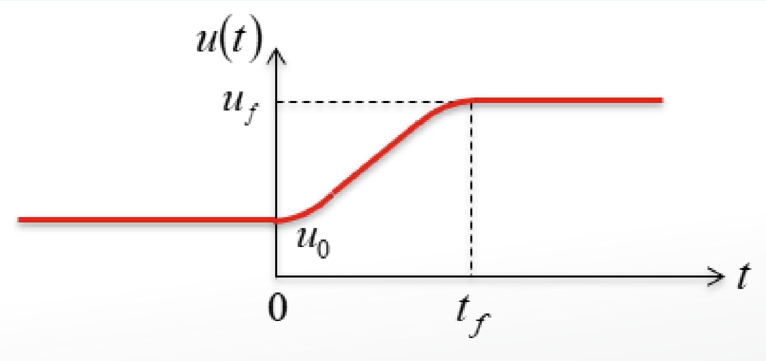
$$u(t) = \begin{cases} \dfrac{u_f - u_0}{t_f} t + u_0 & t < t_f \\ u_f & t \geq t_f \end{cases}$$

- Disadvantage: Discontinuous velocities at start and end points. Rough and jerky motions causes vibrations due to resonance modes, as well as increases wear and tear.

Cubic Polynomial (1)

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

$$\boxed{u(t) = \begin{cases} a_0 + a_1 t + a_2 t^2 + a_3 t^3 & t < t_f \\ u_f & t \geq t_f \end{cases}}$$



Cubic Polynomial (2)

- There are four parameters a_0, a_1, a_2, a_3 which can satisfy four constraints:

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

- The question is now: How do we calculate a_0, a_1, a_2, a_3?

Cubic Polynomial (3)

- This can be done by solving four simultaneous equations:

$$\boxed{\begin{aligned} u(0) &= a_0 + a_1 0 + a_2 0^2 + a_3 0^3 = a_0 = u_0 \\ u(t_f) &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 = u_f \\ \dot{u}(0) &= a_1 + 2a_2 0 + 3a_3 0^2 = a_1 = 0 \\ \dot{u}(t_f) &= a_1 + 2a_2 t_f + 3a_3 t_f^2 = 0 \end{aligned}}$$

  - where the last 2 equations (velocity) come from the differentiation of $u(t)$, i.e.

$$\boxed{\dot{u}(t) = \frac{d}{dt}(a_0 + a_1 t + a_2 t^2 + a_3 t^3) = a_1 + 2a_2 t + 3a_3 t^2}$$

Cubic Polynomial (4)

- The solutions to the simultaneous equations are:

$$\boxed{\begin{aligned} a_0 &= u_0 \\ a_1 &= 0 \\ a_2 &= \frac{3}{t_f^2}\left(u_f - u_0\right) \\ a_3 &= -\frac{2}{t_f^3}\left(u_f - u_0\right) \end{aligned}}$$

Cubic Polynomial (5)

- Example: $u_0 = 15°, u_f = 75°, t_f = 3\text{sec}$. Then:

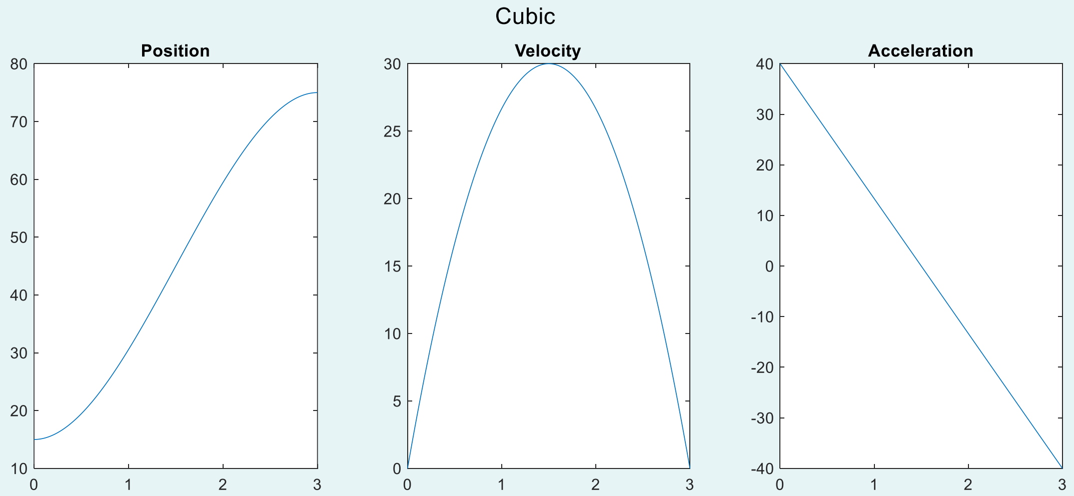
$$
\boxed{
\begin{aligned}
&a_0 = u_0 = 15 \\
&a_1 = 0 \\
&a_2 = \frac{3}{t_f^2}\left(u_f - u_0\right) = \frac{3}{3^2}(75 - 15) = 20 \\
&a_3 = -\frac{2}{t_f^3}\left(u_f - u_0\right) = -\frac{2}{3^3}(75 - 15) = -4.44
\end{aligned}
}
$$

- The trajectory is thus:

$$
\boxed{u(t) = \begin{cases} 15 + 20t^2 - 4.44t^3 & t < 3 \\ 75 & t \geq 3 \end{cases}}
$$

Cubic Polynomial (6)

- 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 (7)

- The Matlab code to generate the trajectory is as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%
% Boundary Conditions %
%%%%%%%%%%%%%%%%%%%%%%%

u0 = 15;
uf = 75;
tf = 3;

%%%%%%%%%%%%%%%%%%%%%
% Create Time Array %
%%%%%%%%%%%%%%%%%%%%%

t = 0:0.001:tf; % array of time from 0 to tf in steps of 0.001
t = t'; % make the time array into column vector
```

Cubic Polynomial (8)

- Continued:

```
%%%%%%%%%%%%%%%%%%%%
% Cubic Polynomial %
%%%%%%%%%%%%%%%%%%%%

a0 = u0;
a1 = 0;
a2 = 3/tf^2*(uf-u0);
a3 = -2/tf^3*(uf-u0);

uCubic = a0*ones(length(t),1)+a1*t+a2*t.^2+a3*t.^3;
uDotCubic = a1*ones(length(t),1)+2*a2*t+3*a3*t.^2;
uDoubleDotCubic = 2*a2*ones(length(t),1)+6*a3*t;

figure,sgtitle('Cubic')
subplot(1,3,1),plot(t,uCubic),title('Position')
subplot(1,3,2),plot(t,uDotCubic),title('Velocity')
subplot(1,3,3),plot(t,uDoubleDotCubic),title('Acceleration')
```

Content

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

Quintic Polynomial (1)

- As seen in the final slide on cubic polynomial, the acceleration at the start and end positions are not zero.

- This is because the cubic polynomial (with 4 parameters) could only satisfy 4 constraints: $u(0), u(t_f), \dot{u}(0), \dot{u}(t_f)$.

Quintic Polynomial (2)

- If we want to have control over accelerations, i.e. 2 additional constraints $\ddot{u}(0), \ddot{u}(t_f)$,
- then we will need a polynomial with 6 parameters – The Quintic Polynomial.

$$\boxed{u(t) = \begin{cases} a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 & t < t_f \\ u_f & t \geq t_f \end{cases}}$$

- To obtain the parameters of the Quintic polynomial, we use the same method of solving simultaneous equations.

Quintic Polynomial (3)

- From $u(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$ we have:

$$\boxed{\begin{aligned} u(0) &= a_0 + a_1 0 + a_2 0^2 + a_3 0^3 + a_4 0^4 + a_5 0^5 = a_0 = u_0 \\ u(t_f) &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 = u_f \end{aligned}}$$

- From $\dot{u}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$ we have:

$$\boxed{\begin{aligned} \dot{u}(0) &= a_1 + 2a_2 0 + 3a_3 0^2 + 4a_4 0^3 + 5a_5 0^4 = a_1 = 0 \\ \dot{u}(t_f) &= a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 = 0 \end{aligned}}$$

- From $\ddot{u}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$ we have:

$$\boxed{\begin{aligned} \ddot{u}(0) &= 2a_2 + 6a_3 0 + 12a_4 0^2 + 20a_5 0^3 = 2a_2 = 0 \\ \ddot{u}(t_f) &= 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 = 0 \end{aligned}}$$

Quintic Polynomial (4)

- Solving the simultaneous equations, the parameters are:

$$\boxed{\begin{aligned} a_0 &= u_0 \\ a_1 &= 0 \\ a_2 &= 0 \\ a_3 &= \frac{10}{t_f^3}\left(u_f - u_0\right) \\ a_4 &= -\frac{15}{t_f^4}\left(u_f - u_0\right) \\ a_5 &= \frac{6}{t_f^5}\left(u_f - u_0\right) \end{aligned}}$$

Quintic Polynomial (5)

- Example: $u_0 = 15°, u_f = 75°, t_f = 3\text{sec}$. Then:

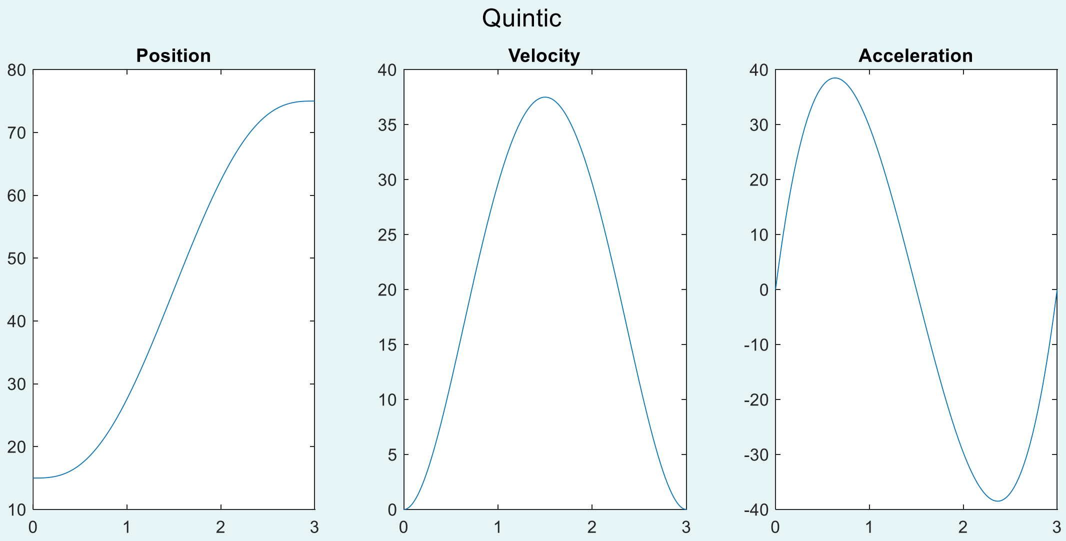
$$
\boxed{
\begin{aligned}
&a_0 = u_0 = 15, a_1 = 0, a_2 = 0 \\
&a_3 = \frac{10}{t_f^3}\left(u_f - u_0\right) = \frac{10}{3^3}(75 - 15) = 22.22 \\
&a_4 = -\frac{15}{t_f^4}\left(u_f - u_0\right) = -\frac{15}{3^4}(75 - 15) = -11.11 \\
&a_5 = \frac{6}{t_f^5}\left(u_f - u_0\right) = \frac{6}{3^5}(75 - 15) = 1.48148
\end{aligned}
}
$$

The trajectory is thus:

$$
\boxed{
u(t) = \begin{cases} 15 + 22.22t^3 - 11.11t^4 + 1.48148t^5 & t < 3 \\ 75 & t \geq 3 \end{cases}
}
$$

Quintic Polynomial (6)

- The position, velocity and acceleration profiles for Quintic polynomial are as follows:



- As can be seen, the acceleration is continuous at start and end.

Quintic Polynomial (7)

- The Matlab code to generate the trajectory is as follows:

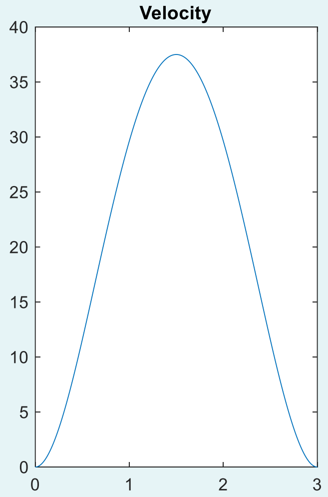
```
%%%%%%%%%%%%%%%%%%%%%%
% Quintic Polynomial %
%%%%%%%%%%%%%%%%%%%%%%

a0 = u0;
a1 = 0;
a2 = 0;
a3 = 10/tf^3*(uf-u0);
a4 = -15/tf^4*(uf-u0);
a5 = 6/tf^5*(uf-u0);

uQuintic = a0*ones(length(t),1)+a1*t+a2*t.^2+a3*t.^3+a4*t.^4+a5*t.^5;
uDotQuintic = a1*ones(length(t),1)+2*a2*t+3*a3*t.^2+4*a4*t.^3+5*a5*t.^4;
uDoubleDotQuintic = 2*a2*ones(length(t),1)+6*a3*t+12*a4*t.^2+20*a5*t.^3;

figure,sgtitle('Quintic')
subplot(1,3,1),plot(t,uQuintic),title('Position')
subplot(1,3,2),plot(t,uDotQuintic),title('Velocity')
subplot(1,3,3),plot(t,uDoubleDotQuintic),title('Acceleration')
```

Content

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

Linear Function w. Parabolic Blends (1)

- The cubic and quintic polynomials, while smooth, are not the most natural way when you think of a motion.
- For e.g. if you drive from A to B, you wouldn't
  - increase your speed slowy,
  - reaching the maximum velocity exactly halfway between A and B,
  - and then slowly decrease speed to zero.

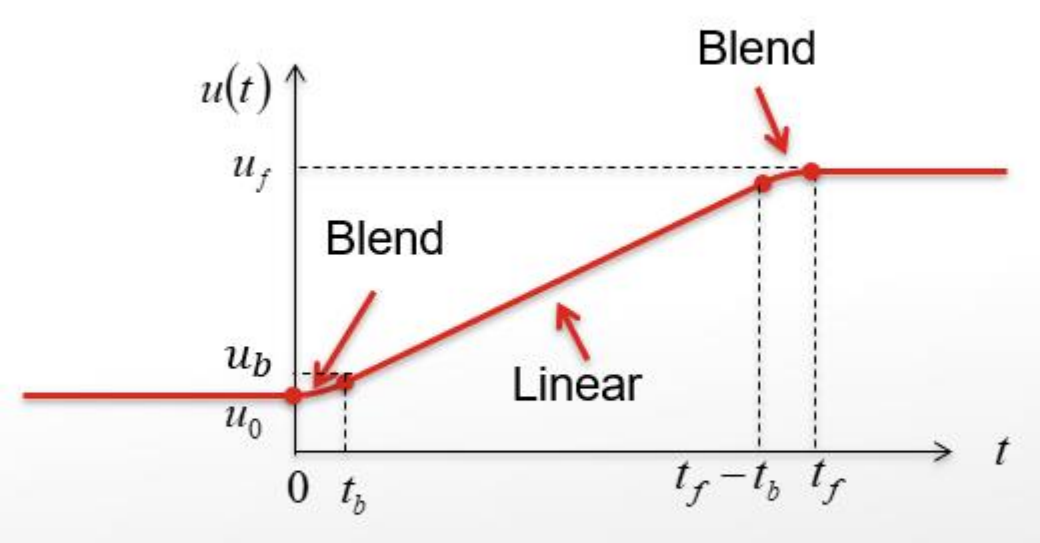


Linear Function w. Parabolic Blends (2)

- Instead, you would want to:
  - Increase your speed smoothly but rapidly up to a maximum velocity (e.g. 70 miles / hour),
  - Maintain the maximum velocity (constant) for a long time,
  - Decreases speed to zero smoothly but rapidly as you reach the destination.

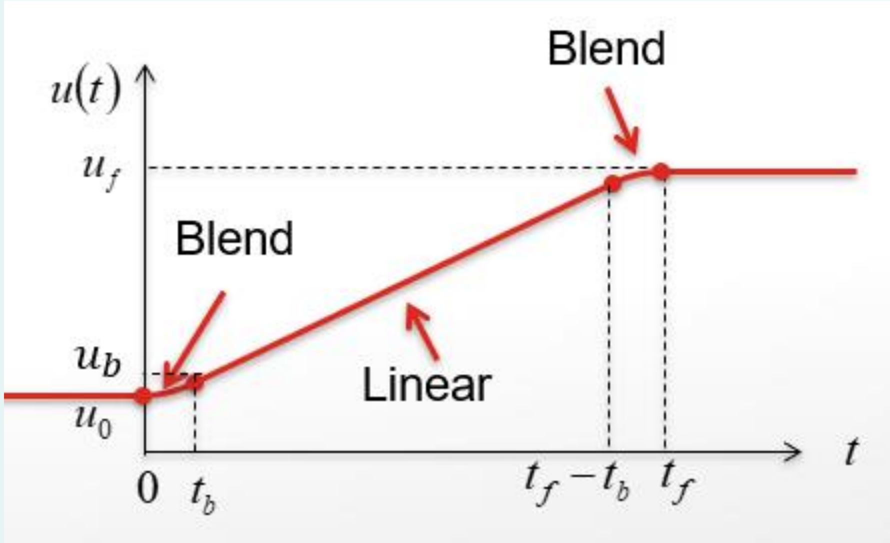
Linear Function w. Parabolic Blends (3)

- This can be achieved with the "linear function with parabolic blends" trajectory.



Derivation (1)

- To compute the trajectory, four pieces of information are needed:
  1. The start position u_0,
  2. The end position u_f,
  3. The time to reach the final position t_f;
  4. The acceleration of the first blend portion $\ddot{u}$.



Derivation (2)

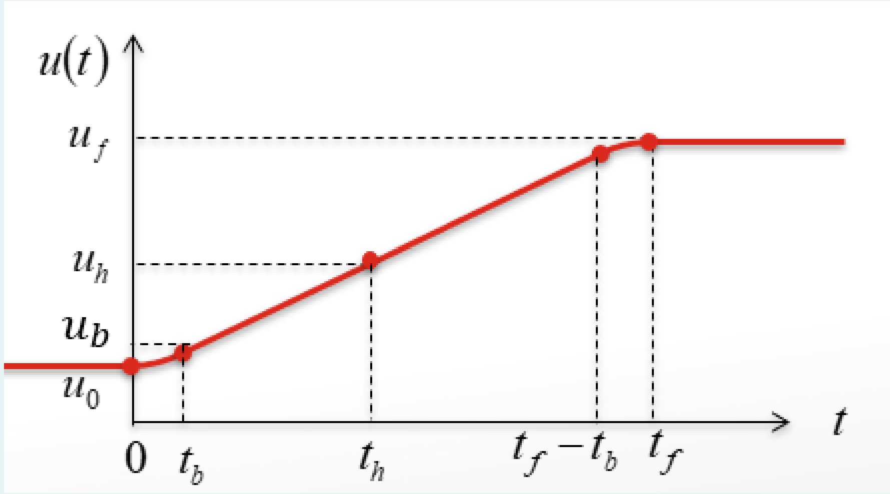
- Three assumptions are also required:

A1. Both the parabolic blends have the same time duration.

- Therefore the deceleration of the second blend portion $= -\ddot{u}$.

A2. The solution is symmetric about the halfway point in time (t_h) and position (u_h) .

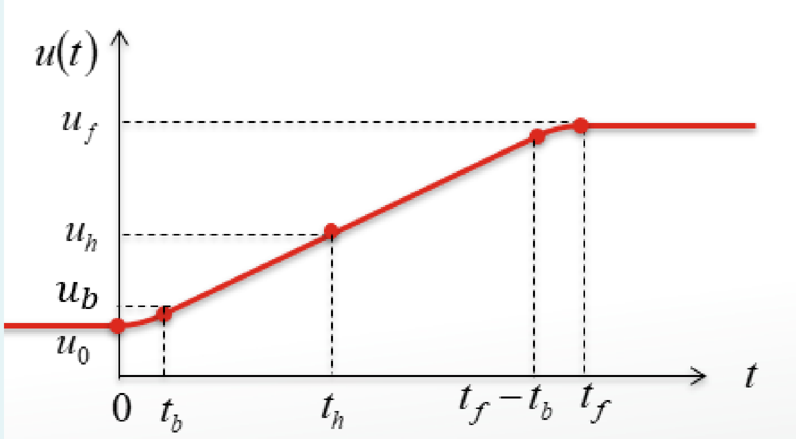
A3. The velocity at the end of blend region is same as that of linear region.



Derivation (3)

- Side note: How to calculate the velocity and position at any instant t in the blend region?
  - $\ddot{u}$ is a given constant, set by user.
  - The velocity is obtained by integration of acceleration:

$$\boxed{\dot{u} = \int \ddot{u}\, dt = \underbrace{\dot{u}_0}_{0} + \ddot{u}t = \ddot{u}t}$$



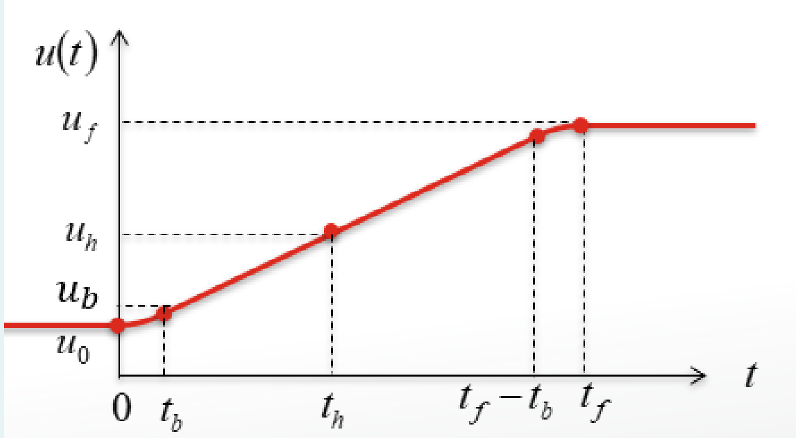
  - The position is obtained by integration of velocity:

$$\boxed{u = \int \dot{u}\, dt = u_0 + \frac{1}{2}\ddot{u}t^2}$$

Derivation (4)

- The trajectory can now be written as:

$$u(t) = \begin{cases} u_0 + \frac{1}{2}\ddot{u}t^2 & t < t_b \\ \frac{u_h - u_b}{t_h - t_b}(t - t_b) + u_b & t_b \le t < (t_f - t_b) \\ u_f - \frac{1}{2}\ddot{u}(t_f - t)^2 & t \ge (t_f - t_b) \end{cases}$$

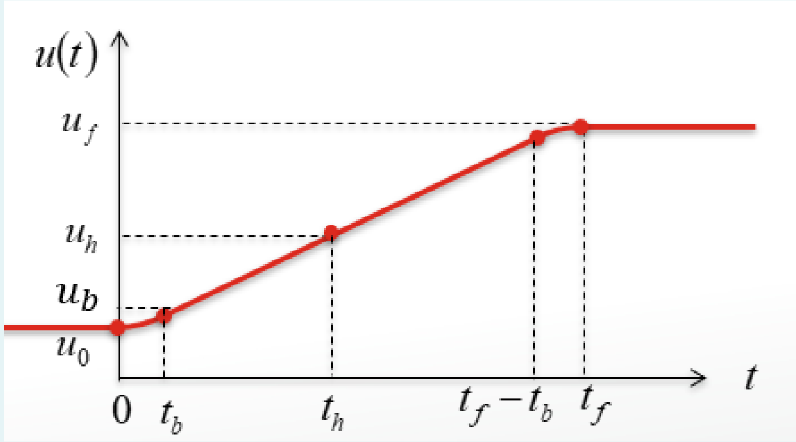


- Problem: u_b and t_b are still unknown.
- We need to calculate them to fully specify the trajectory.

Derivation (5)

- Firstly, we use assumption A3 that the velocity at the end of blend region is the same as that of linear region.

$$\boxed{\ddot{u}t_b = \frac{u_h - u_b}{t_h - t_b}} \qquad (1)$$

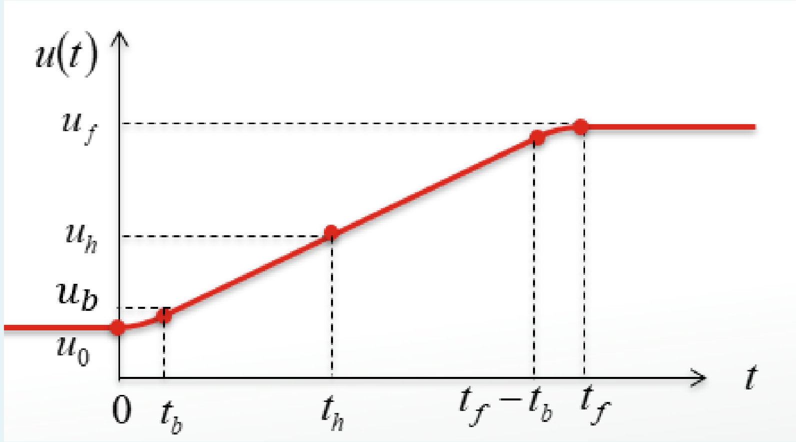


- Let's express u_b in terms of t_b, so that there is only one unknown.
- u_b is the position at time t_b, thus $\boxed{u_b = u_0 + \frac{1}{2}\ddot{u}t_b^2.} \qquad (2)$

Derivation (6)

- The assumption A2 about symmetry give us:

$$\boxed{\begin{aligned} u_h &= \frac{1}{2}\left(u_0 + u_f\right) & (3) \\ t_h &= \frac{1}{2} t_f & (4) \end{aligned}}$$



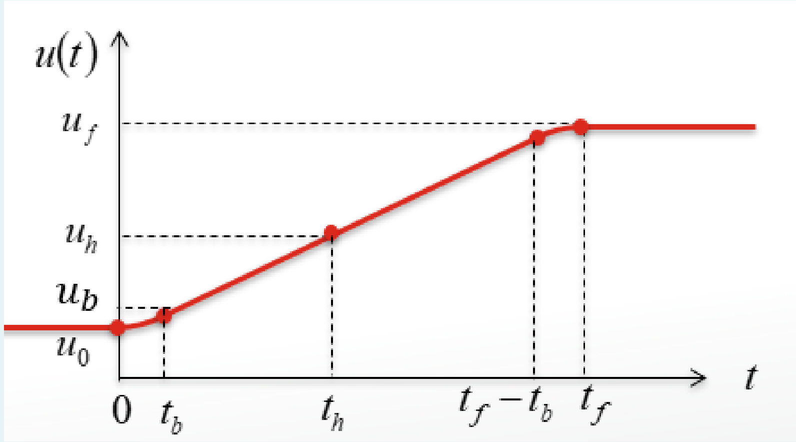
- Putting equations (2), (3), (4) into (1) gives:

$$\ddot{u} t_b = \frac{\frac{1}{2}\left(u_0 + u_f\right) - u_0 - \frac{1}{2}\ddot{u} t_b^2}{\frac{1}{2} t_f - t_b} \Rightarrow \boxed{\ddot{u} t_b^2 - \ddot{u} t_f t_b + \left(u_f - u_0\right) = 0}$$

Derivation (7)

- The final equation, $\ddot{u}t_b^2 - \ddot{u}t_f t_b + (u_f - u_0) = 0$, is a quadratic equation in t_b. The solution is:

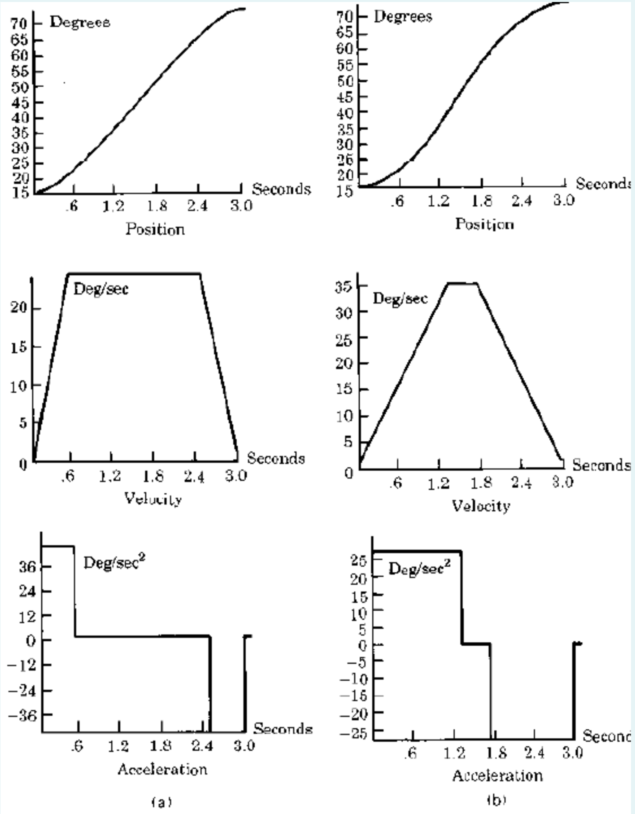
$$\boxed{\begin{aligned} t_b &= \frac{\ddot{u}t_f - \sqrt{\ddot{u}^2 t_f^2 - 4\ddot{u}(u_f - u_0)}}{2\ddot{u}} \\ &= \frac{t_f}{2} - \frac{\sqrt{\ddot{u}^2 t_f^2 - 4\ddot{u}(u_f - u_0)}}{2\ddot{u}} \end{aligned}}$$



  - Note: we choose only "minus" sign for square root because $t_b \le \frac{t_f}{2}$.
- Substitute t_b back into $\boxed{u_b = u_0 + \frac{1}{2}\ddot{u}t_b^2}$ to get u_b.

About the Desired Acceleration (1)

- The acceleration must be chosen to be high enough.
  - Otherwise solution to t_b will not exist.
- E.g. if acceleration is small, the linear region shrinks.
- Or, if acceleration is too small, there may be no more linear region.



About the Desired Acceleration (2)

- How to make sure the acceleration is high enough?
- It is such that $\boxed{t_b = \frac{t_f}{2} - \frac{\sqrt{\ddot{u}^2 t_f^2 - 4\ddot{u}(u_f - u_0)}}{2\ddot{u}}}$ exists.
- In other words:

$$\ddot{u}^2 t_f^2 - 4\ddot{u}(u_f - u_0) \geq 0$$

$$\ddot{u}^2 t_f^2 \geq 4\ddot{u}(u_f - u_0)$$

$$\boxed{\ddot{u} \geq 4\frac{(u_f - u_0)}{t_f^2}}$$

Summary of the Trajectory

- Given u_0, u_f, t_f.
- Choose desired acceleration $\ddot{u}$ which must satisfy $\boxed{\ddot{u} \geq 4\frac{(u_f - u_0)}{t_f^2}.}$
- Calculate $\boxed{t_b = \frac{t_f}{2} - \frac{\sqrt{\ddot{u}^2 t_f^2 - 4\ddot{u}(u_f - u_0)}}{2\ddot{u}}}$ and $\boxed{u_b = u_0 + \frac{1}{2}\ddot{u}t_b^2}$
- The trajectory is then: $\boxed{u(t) = \begin{cases} u_0 + \frac{1}{2}\ddot{u}t^2 & t < t_b \\ \frac{u_h - u_b}{t_h - t_b}(t - t_b) + u_b & t_b \leq t < (t_f - t_b) \\ u_f - \frac{1}{2}\ddot{u}(t_f - t)^2 & t \geq (t_f - t_b) \end{cases}}$

Example (1)

- $u_0 = 15°, u_f = 75°, t_f = 3$sec.
- The minimum acceleration is:

$$\boxed{\ddot{u} \ \geq 4\frac{\left(u_f - u_0\right)}{t_f^2} = 4\frac{(75 - 15)}{3^2} = 26.6666}$$

  - Here I have chosen $\ddot{u} = 100$.
- With this,

$$\boxed{t_b = \frac{t_f}{2} - \frac{\sqrt{\ddot{u}^2 t_f^2 - 4\ddot{u}\left(u_f - u_0\right)}}{2\ddot{u}} = \frac{3}{2} - \frac{\sqrt{100^2 \cdot 3^2 - 4 \cdot 100(75 - 15)}}{2 \cdot 100} = 0.2155}$$

Example (2)

- Also,

$$u_b = u_0 + \frac{1}{2}\ddot{u}t_b^2 = 15 + \frac{1}{2}100 \cdot 0.2155^2 = 17.3215$$

- The trajectory is then:

$$u(t) = \begin{cases} u_0 + \frac{1}{2}\ddot{u}t^2 & t < t_b \\ \frac{u_h - u_b}{t_h - t_b}(t - t_b) + u_b & t_b \leq t < (t_f - t_b) \\ u_f - \frac{1}{2}\ddot{u}(t_f - t)^2 & t \geq (t_f - t_b) \end{cases}$$

Example (3)

- The Matlab code is as follows:

```
%%%%%%%%%%%%%%%%%%%%%%%
% Boundary Conditions %
%%%%%%%%%%%%%%%%%%%%%%%

u0 = 15;
uf = 75;
tf = 3;
```

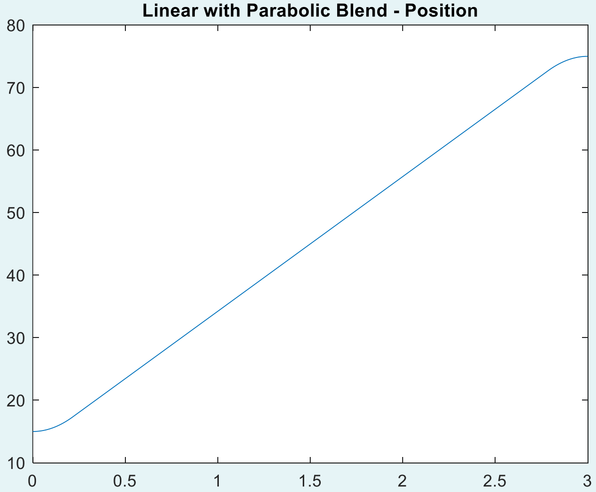
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear with Parabolic Blend %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

AccMin = 4*(uf-u0)/tf^2;
Acc = round(4*AccMin/100)*100; % Assume 4 x AccMin, round to closest 100
tb = tf/2 - sqrt(Acc^2*tf^2 - 4*Acc*(uf-u0))/2/Acc;
ub = u0 + 0.5*Acc*tb^2;
uh = 0.5*(u0+uf);
th = 0.5*tf;
```

Example (4)

```
% First blend

tBlend1 = 0:0.001:tb;
tBlend1 = tBlend1';
uBlend1 = u0*ones(length(tBlend1),1)+0.5*Acc*tBlend1.^2;

% Linear portion

tLinear = tb+0.001:0.001:(tf-tb);
tLinear = tLinear';
uLinear = (uh-ub)/(th-tb)*(tLinear-tb)+ub;

% Second blend

tBlend2 = (tf-tb+0.001):0.001:tf;
tBlend2 = tBlend2';
uBlend2 = uf*ones(length(tBlend2),1)-0.5*Acc*(tf-tBlend2).^2;
```

Example (5)

```
% Combine

tCombine = [tBlend1;tLinear;tBlend2];
uCombine = [uBlend1;uLinear;uBlend2];
figure,plot(tCombine,uCombine),title('Linear with Parabolic Blend - Position')
```



Content

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

Trajectory with Multiple Segments (1)

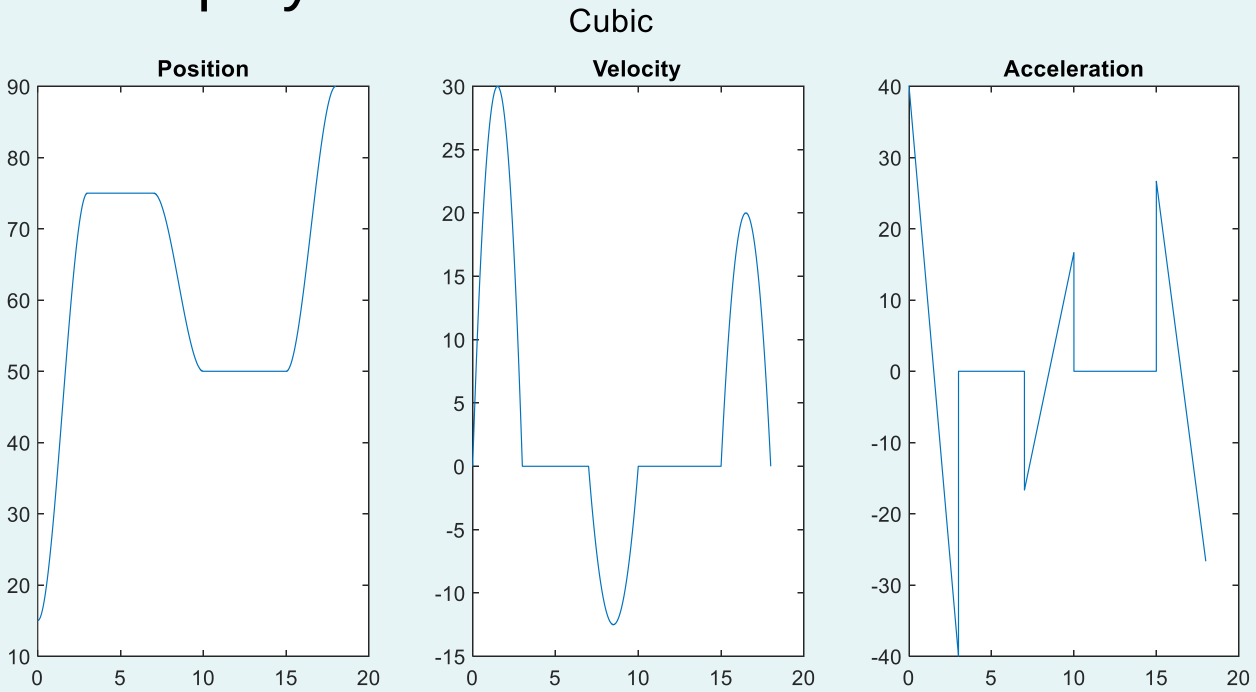
- So far, the provided Matlab codes have been generating trajectory for one single move only.
- What if several moves are required?
  - E.g. move from $15°$ to $75°$ at $t = 0$ with duration of 3 seconds;
  - Stay at $75°$ until next move;
  - Move from $75°$ to $50°$ at $t = 7$ with duration of 3 seconds;
  - Stay at $50°$ until next move;
  - Move from $50°$ to $90°$ at $t = 15$ with duration of 3 seconds;
  - Etc. Etc.

Trajectory with Multiple Segments (2)

- I will leave this as a coding exercise for you.
- Please try this out – You will need it for your robot workshop.

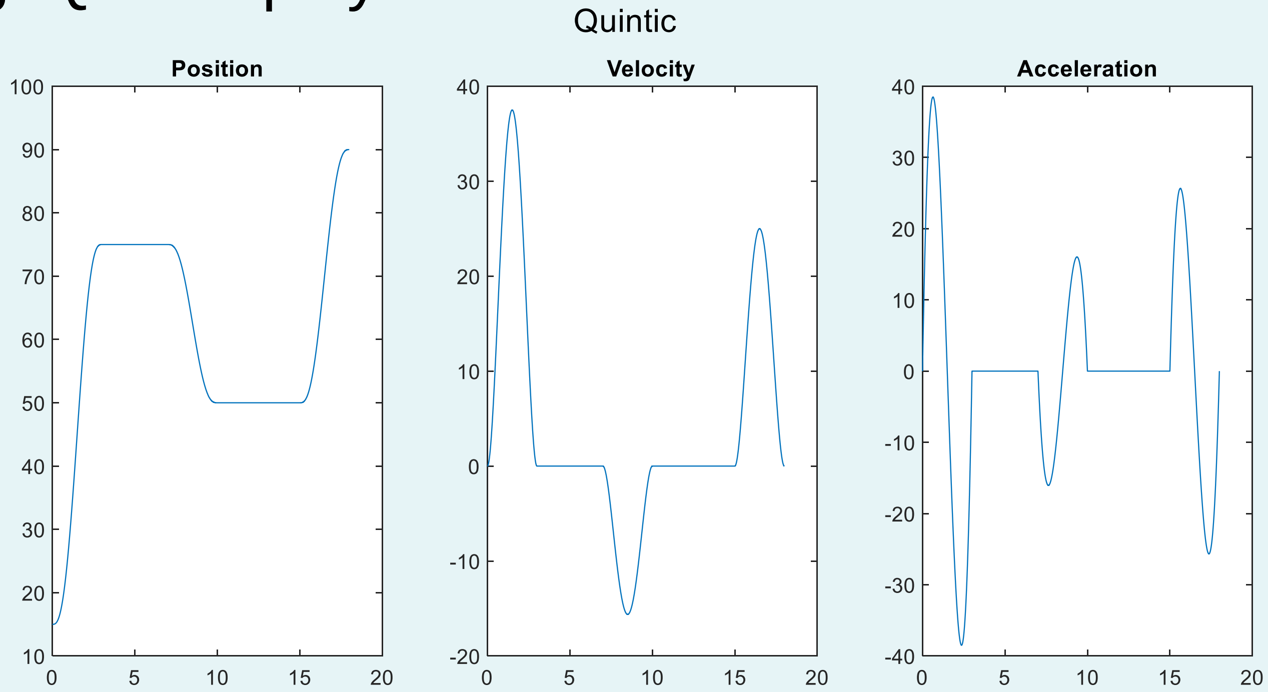
Trajectory with Multiple Segments (3)

- E.g. cubic polynomial:



Trajectory with Multiple Segments (4)

- E.g. Quintic polynomial:



Thank you for your attention!

Any questions?