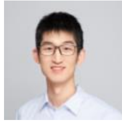
深蓝学院
shenlanxueyuan.com

# Foundations of Robotics

## Lec 7: Trajectory Generation




主讲人 滕瀚哲

美国加州大学河滨分校
ARCS实验室博士

# Outline

- 1. Trajectory Generation
- 2. Time Scaling
- 3. Waypoint Navigation
- 4. Homework

# Outline

1. Trajectory Generation
2. Time Scaling
3. Waypoint Navigation
4. Homework

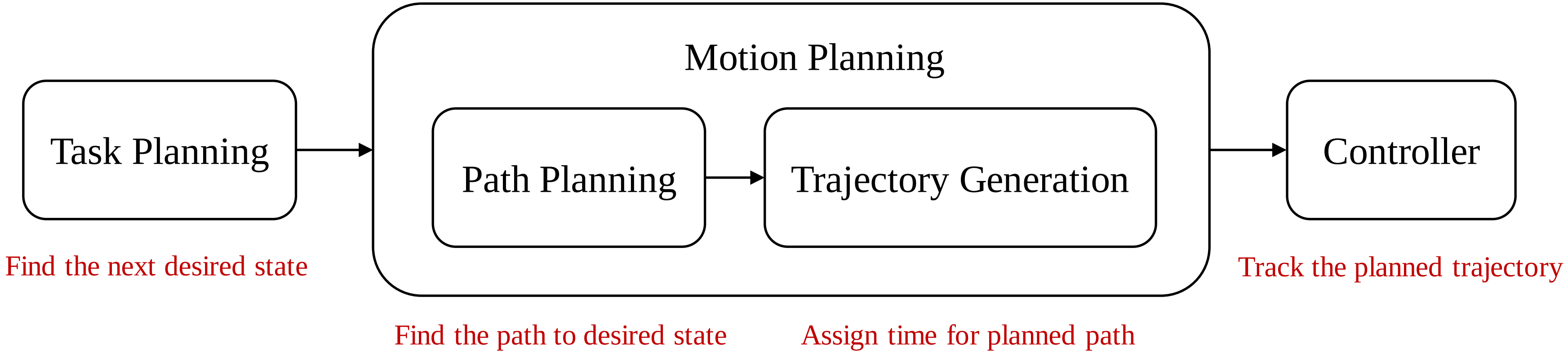
# Trajectory Generation

Recall the two kinematics problems in an open-chain manipulator.

- Forward Kinematics: $\theta \in \mathbb{R}^n \quad \rightarrow \quad X \in SE(3)$
- Inverse Kinematics: $X \in SE(3) \quad \rightarrow \quad \theta \in \mathbb{R}^n$

Once the desired state has been identified, how to achieve it?

- There has to be a continuous <u>trajectory</u> to follow.



# Trajectory Generation

What is a trajectory?

- A trajectory consists of two components: a path and a time scaling.

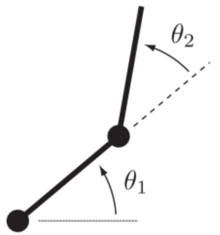
| | | |
|---|---|---|
| Path | $\theta(s), s \in [0, 1]$ | Maps a scalar path parameter $s$ to the robot's configuration space $\theta$. |
| Time scaling | $s(t), t \in [0, T]$ | Assigns a value $s \in [0,1]$ to each time $t \in [0, T]$. |
| Trajectory | $\theta\big(s(t)\big)$ or<br>$\theta(t), t \in [0, T]$ | A path $\theta(s)$ with an associated time scaling $s(t)$. |

Using the chain rule, the velocity and acceleration along the trajectory can be written as

- $\dot{\theta} = \frac{d\theta}{ds}\dot{s}$
- $\ddot{\theta} = \frac{d\theta}{ds}\ddot{s} + \frac{d^2\theta}{ds^2}\dot{s}^2$

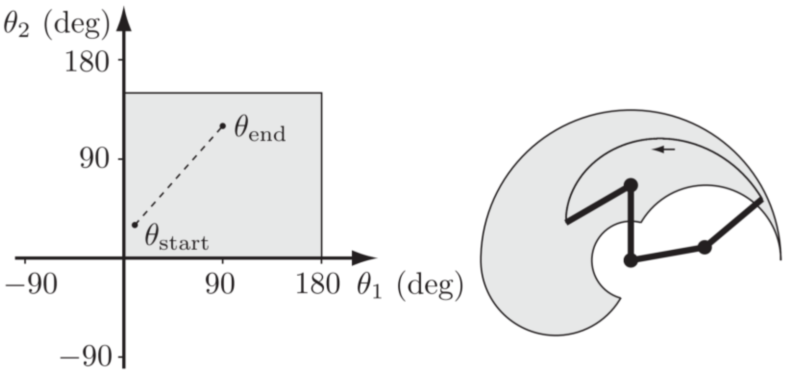
where $\theta(s)$ and $s(t)$ must be twice differentiable.

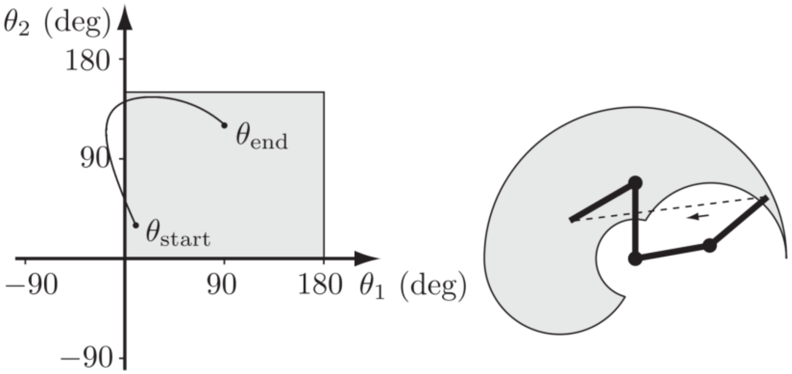
# Trajectory Generation




Example: straight-line paths in joint space and task space

- A 2R robot arm with joint limits $0° \leq \theta_1 \leq 180°$ and $0° \leq \theta_2 \leq 150°$.




Straight-line path in joint space

$$\theta(s) = \theta_{\text{start}} + s(\theta_{\text{end}} - \theta_{\text{start}}), s \in [0,1]$$
$$\dot{\theta} = \dot{s}(\theta_{\text{end}} - \theta_{\text{start}})$$
$$\ddot{\theta} = \ddot{s}(\theta_{\text{end}} - \theta_{\text{start}})$$

Straight-line path in task space

$$X(s) = X_{\text{start}} + s(X_{\text{end}} - X_{\text{start}}), s \in [0,1]$$
$$\dot{X} = \dot{s}(X_{\text{end}} - X_{\text{start}})$$
$$\ddot{X} = \ddot{s}(X_{\text{end}} - X_{\text{start}})$$

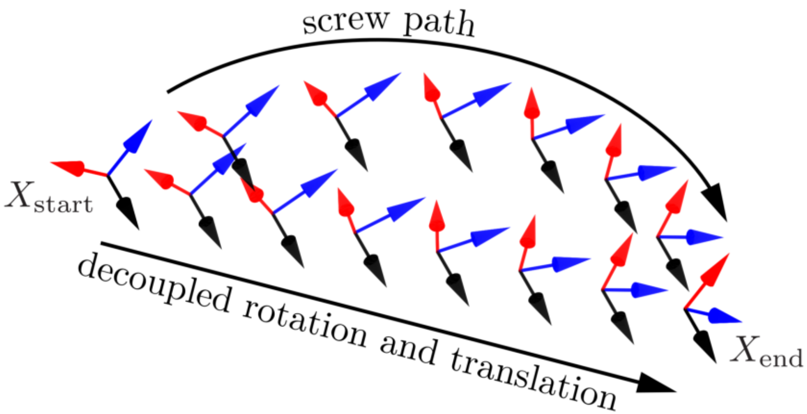
# Trajectory Generation

Example: straight-line paths in $SE(3)$

- How to define a "straight" line?
- $T(s) = T_{\text{start}} + s(T_{\text{end}} - T_{\text{start}})$ does not generally lie in $SE(3)$

Option one: constant screw axis

- Start configuration: $T_{\text{start}}$ or $T_{s,\text{start}}$
- End configuration: $T_{\text{end}}$ or $T_{s,\text{end}}$
- Matrix representation of the twist: $\log\left(T_{s,\text{start}}^{-1} T_{s,\text{end}}\right)$
- The "straight-line" path: $T(s) = T_{\text{start}} \exp\left(\log\left(T_{\text{start}}^{-1} T_{\text{end}}\right) s\right)$




Option two: decouple rotation and translation

- $p(s) = p_{\text{start}} + s(p_{\text{end}} - p_{\text{start}})$
- $R(s) = R_{\text{start}} \exp\left(\log\left(R_{\text{start}}^{\top} R_{\text{end}}\right) s\right)$

# Outline

# Time Scaling

Time scaling $s(t)$ of a path should ensure that
- the motion is appropriately smooth, and that
- any constraints on robot velocity and acceleration are satisfied.

The two most frequently used time scaling methods: polynomial and trapezoidal.

Polynomial Time Scaling
- Key idea: identify constraints and solve for coefficients of a polynomial function.
- Smooth motion when constraints are provided in the form of waypoints (pos, vel, acc).
- Make it smoother: use higher orders of polynomials.

Trapezoidal Time Scaling
- Key idea: accelerate to maximum speed, maintain the speed, decelerate to zero speed.
- Fastest straight-line motion when there are known limits on velocities and accelerations.
- Make it smoother: S-curve time scaling.

# Polynomial Time Scaling

Example: a robot moves from point $A$ to point $B$ using 3rd-order polynomial time scaling.

- The time scaling function to be fitted: $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, t \in [0, T]$
- Initial constraints at the start point: $s(0) = 0, \dot{s}(0) = 0$
- Terminal constraints at the end point: $s(T) = 1, \dot{s}(T) = 0$

Let's solve for coefficients using the provided constraints.

- $s(0) = a_0 = 0$
- $\dot{s}(0) = a_1 = 0$
- $s(T) = a_0 + a_1 T + a_2 T^2 + a_3 T^3 = 1$
- $\dot{s}(T) = a_1 + 2a_2 T + 3a_3 T^2 = 0$

Take the derivative:
$\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2, t \in [0, T]$

This leads to four coefficients: $a_0 = 0$, $a_1 = 0$, $a_2 = \frac{3}{T^2}$, $a_3 = -\frac{2}{T^3}$.

The final time scaling function: $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 = \frac{3t^2}{T^2} - \frac{2t^3}{T^3}$.

# Polynomial Time Scaling

In fact, we can solve for coefficients more efficiently by organizing constraints into matrices.

Constraints

- $s(0) = a_0 = 0$
- $\dot{s}(0) = a_1 = 0$
- $s(T) = a_0 + a_1 T + a_2 T^2 + a_3 T^3 = 1$
- $\dot{s}(T) = a_1 + 2a_2 T + 3a_3 T^2 = 0$

In a matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Alternatively,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 0 & 0 & 0 & 1 \\ T^3 & T^2 & T & 1 \\ 0 & 0 & 1 & 0 \\ 3T^2 & 2T & 1 & 0 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

They are all correct, as long as each row can represent a constraint!

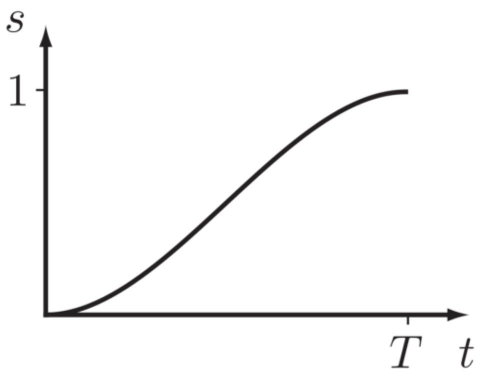
# Polynomial Time Scaling

3rd-order Polynomial Time Scaling

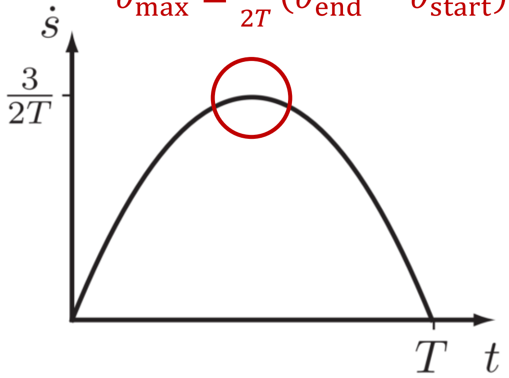
- $s(t) = \frac{3t^2}{T^2} - \frac{2t^3}{T^3}$
- $\dot{s}(t) = \frac{6t}{T^2} - \frac{6t^2}{T^3}$
- $\ddot{s}(t) = \frac{6}{T^2} - \frac{12t}{T^3}$

Putting $s(t)$ back into path $\theta(s)$ leads to

- $\theta(t) = \theta_{\text{start}} + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}})$
- $\dot{\theta}(t) = \left(\frac{6t}{T^2} - \frac{6t^2}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}})$
- $\ddot{\theta}(t) = \left(\frac{6}{T^2} - \frac{12t}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}})$

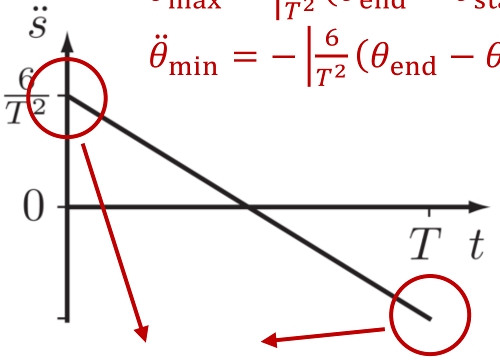



$\dot{\theta}_{\max} = \frac{3}{2T}(\theta_{\text{end}} - \theta_{\text{start}})$




$\ddot{\theta}_{\max} = \left|\frac{6}{T^2}(\theta_{\text{end}} - \theta_{\text{start}})\right|$

$\ddot{\theta}_{\min} = -\left|\frac{6}{T^2}(\theta_{\text{end}} - \theta_{\text{start}})\right|$




Discontinuities!

# Polynomial Time Scaling

5th-order Polynomial Time Scaling

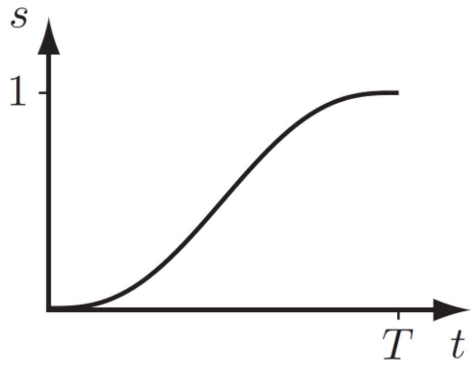
- $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5, t \in [0, T]$
- $\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$
- $\ddot{s}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$
- Initial constraints:

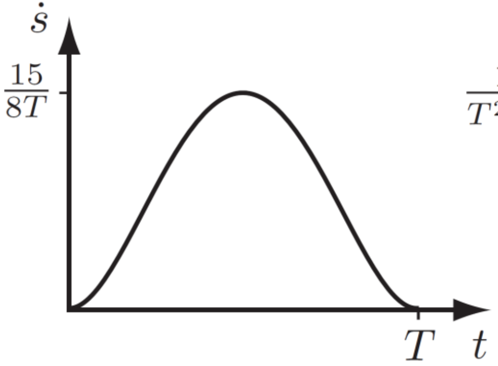
$$s(0) = 0, \dot{s}(0) = 0, \ddot{s}(0) = 0$$

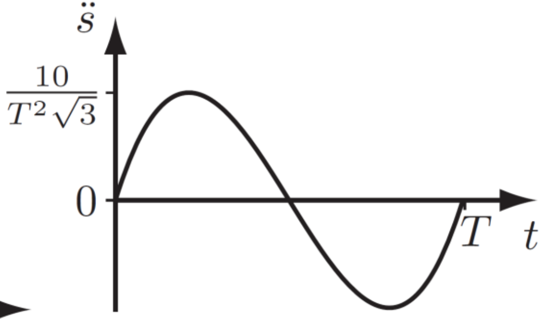
- Terminal constraints:

$$s(T) = 1, \dot{s}(T) = 0, \ddot{s}(T) = 0$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 & T^5 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 \\ 0 & 0 & 2 & 6T & 12T^2 & 20T^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$



# Polynomial Time Scaling

7th-order Polynomial Time Scaling

- $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6 + a_7 t^7, t \in [0, T]$
- $\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 + 6a_6 t^5 + 7a_7 t^6$
- $\ddot{s}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 + 30a_6 t^4 + 42a_7 t^5$
- $\dddot{s}(t) = 6a_3 + 24a_4 t + 60a_5 t^2 + 120a_6 t^3 + 210a_7 t^4$
- Initial constraints: $s(0) = 0, \dot{s}(0) = 0, \ddot{s}(0) = 0, \dddot{s}(0) = 0$
- Terminal constraints: $s(T) = 1, \dot{s}(T) = 0, \ddot{s}(T) = 0, \dddot{s}(T) = 0$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 & T^5 & T^6 & T^7 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 & 6T^5 & 7T^6 \\ 0 & 0 & 2 & 6T & 12T^2 & 20T^3 & 30T^4 & 42T^5 \\ 0 & 0 & 0 & 6 & 24T & 60T^2 & 120T^3 & 210T^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- Jerk: third time derivative of position
- Snap: fourth time derivative of position

- Pay attention to numerical stability when using high-order polynomials

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors", ICRA 2011

# Trapezoidal Time Scaling

Trapezoidal Time Scaling

- Fastest straight-line motion under known constant velocity and acceleration constraints
- Three phases: acceleration, coast, deceleration

Phase one: acceleration when $0 \leq t \leq \frac{v}{a}$

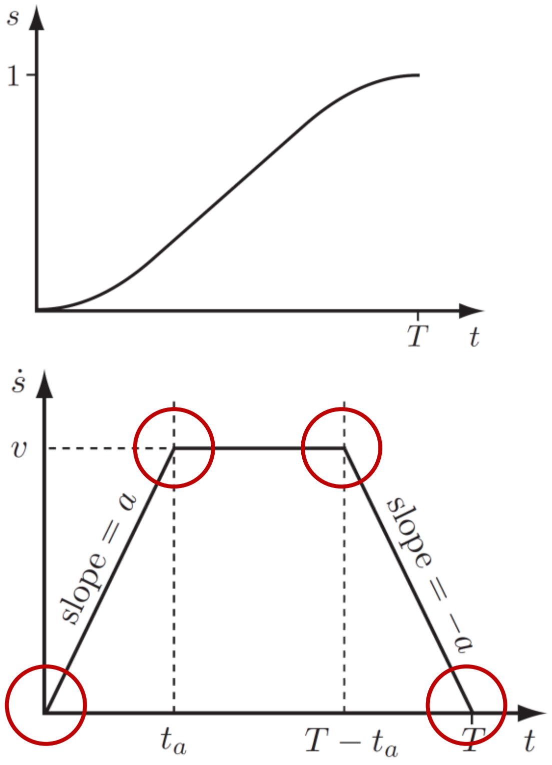
- $\ddot{s}(t) = a, \dot{s}(t) = at, s(t) = \frac{1}{2}at^2$

Phase two: coast when $\frac{v}{a} < t \leq T - \frac{v}{a}$

- $\ddot{s}(t) = 0, \dot{s}(t) = v, s(t) = vt - \frac{v^2}{2a}$

Phase three: deceleration when $T - \frac{v}{a} < t \leq T$

- $\ddot{s}(t) = -a, \dot{s}(t) = a(T-t), s(t) = \frac{2avT - 2v^2 - a^2(t-T)^2}{2a}$

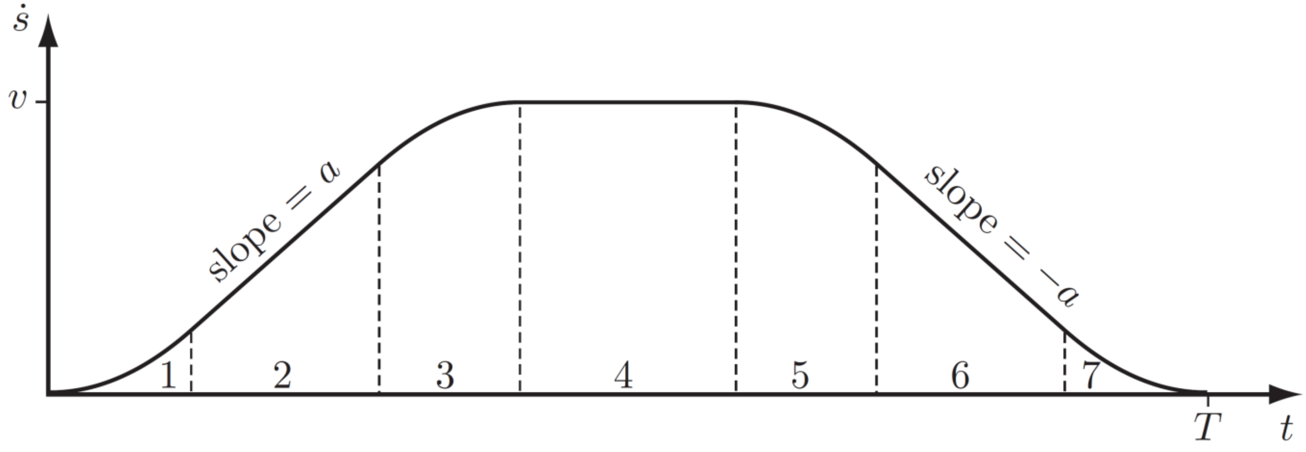



Discontinuities!

# S-Curve Time Scaling

S-Curve Time Scaling

- A popular motion profile in motor control, because it avoids vibrations or oscillations induced by step changes in acceleration
- Seven phases: 1, 3, 5, 7: constant (positive/negative) jerk
  2, 6: constant (positive/negative) acceleration
  4: coasting at constant velocity



# Outline

# Waypoint Navigation

Now let's have the robot pass through a sequence of waypoints in 2D space.

Polynomial Via Point Trajectories (Waypoint Navigation)

- Use 3rd-order polynomial time scaling for each segment and each dimension.
- Use actual waypoints and directly solve $\theta(t)$ instead of $s(t)$.
  - ➢ In this way, the shape of the trajectory is also encoded in polynomial functions, rather than just straight-line paths for $\theta(s), s \in [0,1]$.

Two ways to assign time

- One absolute time history: $[T_1, T_2], [T_2, T_3], [T_3, T_4]$
  - ➢ Better visualization, clear math, adopted by the textbook.
- Multiple relative time durations: $[0, T_1], [0, T_2], [0, T_3]$
  - ➢ Better numerical stability, commonly used in practice.

# Waypoint Navigation

Example: solve a trajectory from four waypoints

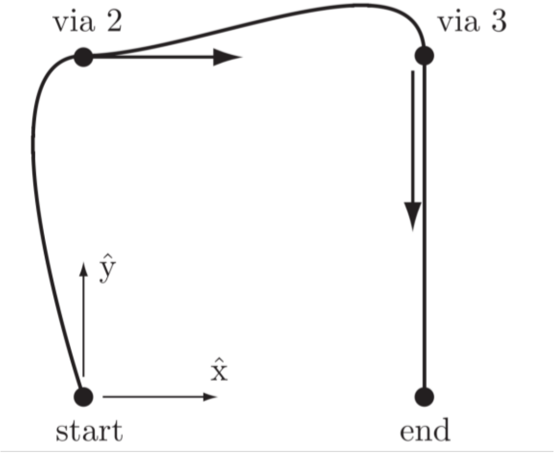
| Waypoint | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Position | $(0,0)$ | $(0,2)$ | $(2,2)$ | $(2,0)$ |
| Velocity | $(0,0)$ | $(3,0)$ | $(0,-3)$ | $(0,0)$ |

First segment (waypoint 1 to waypoint 2)

- $x_1(t) = a_{x,0} + a_{x,1}t + a_{x,2}t^2 + a_{x,3}t^3, t \in [0, T_1]$
- $y_1(t) = a_{y,0} + a_{y,1}t + a_{y,2}t^2 + a_{y,3}t^3, t \in [0, T_1]$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T_1 & T_1^2 & T_1^3 \\ 0 & 1 & 2T_1 & 3T_1^2 \end{bmatrix} \begin{bmatrix} a_{x,0} \\ a_{x,1} \\ a_{x,2} \\ a_{x,3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 3 \end{bmatrix} \quad \Longrightarrow \quad x_1(t)$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T_1 & T_1^2 & T_1^3 \\ 0 & 1 & 2T_1 & 3T_1^2 \end{bmatrix} \begin{bmatrix} a_{y,0} \\ a_{y,1} \\ a_{y,2} \\ a_{y,3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix} \quad \Longrightarrow \quad y_1(t)$$



# Waypoint Navigation

Example: solve a trajectory from four waypoints

| Waypoint | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Position | $(0,0)$ | $(0,2)$ | $(2,2)$ | $(2,0)$ |
| Velocity | $(0,0)$ | $(3,0)$ | $(0,-3)$ | $(0,0)$ |

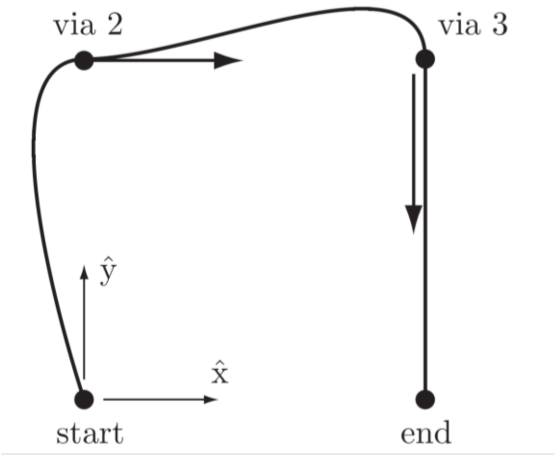
Second segment (waypoint 2 to waypoint 3)

- $x_2(t) = a_{x,0} + a_{x,1}t + a_{x,2}t^2 + a_{x,3}t^3, t \in [0, T_2]$
- $y_2(t) = a_{y,0} + a_{y,1}t + a_{y,2}t^2 + a_{y,3}t^3, t \in [0, T_2]$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T_2 & T_2^2 & T_2^3 \\ 0 & 1 & 2T_2 & 3T_2^2 \end{bmatrix} \begin{bmatrix} a_{x,0} \\ a_{x,1} \\ a_{x,2} \\ a_{x,3} \end{bmatrix} = \begin{bmatrix} \boxed{\begin{matrix} 0 \\ 3 \end{matrix}} \\ 2 \\ 0 \end{bmatrix} \Longrightarrow \quad x_2(t)$$

Continuity constraint:
start point in segment 2
= end point in segment 1

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T_2 & T_2^2 & T_2^3 \\ 0 & 1 & 2T_2 & 3T_2^2 \end{bmatrix} \begin{bmatrix} a_{y,0} \\ a_{y,1} \\ a_{y,2} \\ a_{y,3} \end{bmatrix} = \begin{bmatrix} \boxed{\begin{matrix} 2 \\ 0 \end{matrix}} \\ 2 \\ -3 \end{bmatrix} \Longrightarrow \quad y_2(t)$$



# Waypoint Navigation

Example: solve a trajectory from four waypoints

| Waypoint | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Position | $(0,0)$ | $(0,2)$ | $(2,2)$ | $(2,0)$ |
| Velocity | $(0,0)$ | $(3,0)$ | $(0,-3)$ | $(0,0)$ |

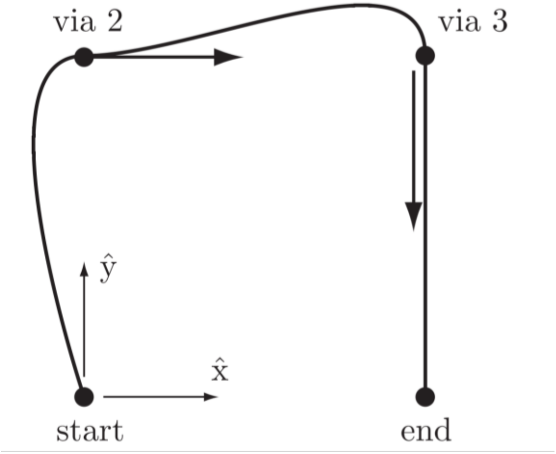
Third segment (waypoint 3 to waypoint 4)

- $x_3(t) = a_{x,0} + a_{x,1}t + a_{x,2}t^2 + a_{x,3}t^3, t \in [0, T_3]$
- $y_3(t) = a_{y,0} + a_{y,1}t + a_{y,2}t^2 + a_{y,3}t^3, t \in [0, T_3]$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T_3 & T_3^2 & T_3^3 \\ 0 & 1 & 2T_3 & 3T_3^2 \end{bmatrix} \begin{bmatrix} a_{x,0} \\ a_{x,1} \\ a_{x,2} \\ a_{x,3} \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 2 \\ 0 \end{bmatrix} \Longrightarrow x_3(t)$$

Continuity constraint:
start point in segment 3
= end point in segment 2

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T_3 & T_3^2 & T_3^3 \\ 0 & 1 & 2T_3 & 3T_3^2 \end{bmatrix} \begin{bmatrix} a_{y,0} \\ a_{y,1} \\ a_{y,2} \\ a_{y,3} \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ 0 \\ 0 \end{bmatrix} \Longrightarrow y_3(t)$$



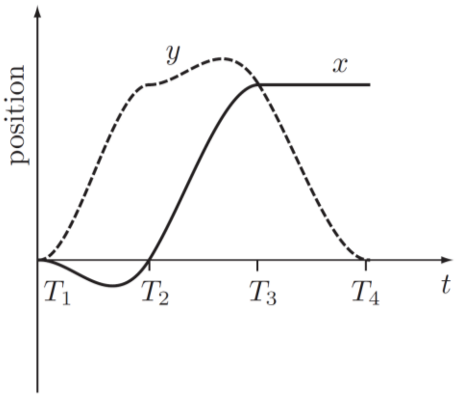
# Waypoint Navigation

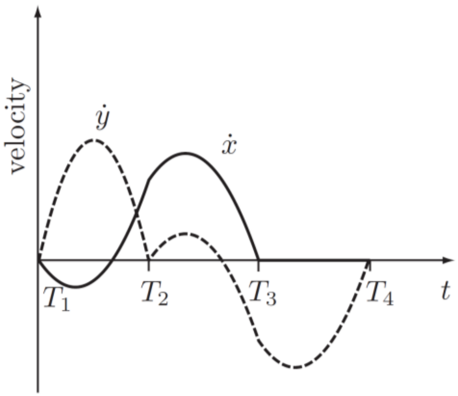
Example: solve a trajectory from four waypoints

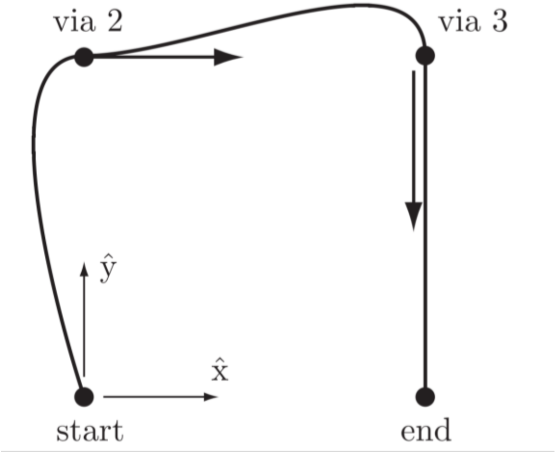
Now we have obtained six polynomial functions using multiple relative time durations.

To have a better visualization, we combine them into a common time history shown as below.

| Waypoint | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Position | $(0,0)$ | $(0,2)$ | $(2,2)$ | $(2,0)$ |
| Velocity | $(0,0)$ | $(3,0)$ | $(0,-3)$ | $(0,0)$ |



# Waypoint Navigation

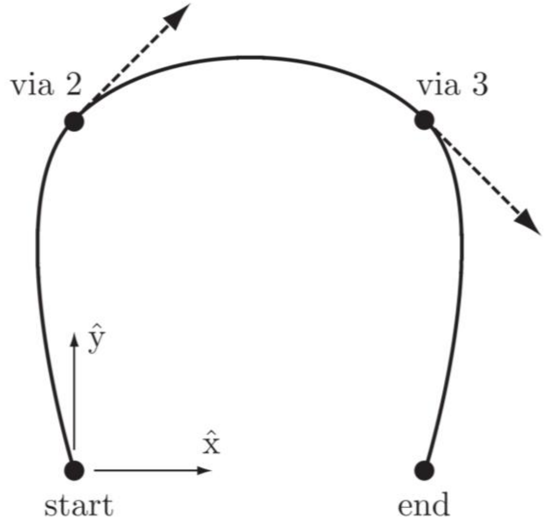
Can we do better?

- Use "reasonable" velocities for via points
    - Estimate magnitude from time duration
    - Pick direction from previous/next waypoints
- Do not specify velocities for via points
    - Save them as free variables in optimization
    - May add constraints on accelerations instead
- Use higher order polynomials

Constrained optimization for trajectory generation

- Specify an objective function (e.g., minimize energy) and formulate equality or inequality constraints (e.g., continuity constraints)
- Linear programming is the simplest form, where both objective function and constraints are linear

| Waypoint | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Position | $(0,0)$ | $(0,2)$ | $(2,2)$ | $(2,0)$ |
| Velocity | $(0,0)$ | $(1,1)$ | $(1,-1)$ | $(0,0)$ |



# Outline

# Homework

(1) Textbook Exercises: 9.2, 9.5, 9.7

(2) Lab Assignments: Trajectory generation using $3^{rd}$-order polynomials

- Provided the position of a sequence of waypoints, write scripts in C++ or Python to generate a trajectory passing through all waypoints using $3^{rd}$-order polynomials
- Make the robot follow it, save the trajectory and plot figures.

(3) (Optional) One absolute time history vs. multiple relative time durations

- Replicate our example discussed in the lecture using 3rd-order polynomials and both timing methods (in any programming languages, such as MATLAB, Python).
- Plot their position and velocity profiles in x and y dimensions, respectively, and compare the difference. Pay attention to how the polynomial functions are shifted.

深蓝学院
shenlanxueyuan.com

# Thanks for Listening!