Robot Perception and Learning

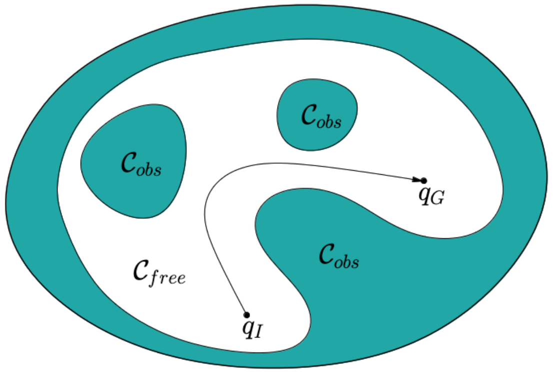
Kinodynamic Motion planning, Trajectory Generation, Feedback Control

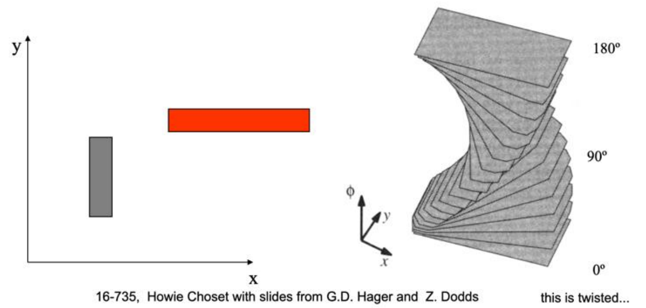
Tsung-Wei Ke

Fall 2025

Recap

Configuration Space (C-Space):

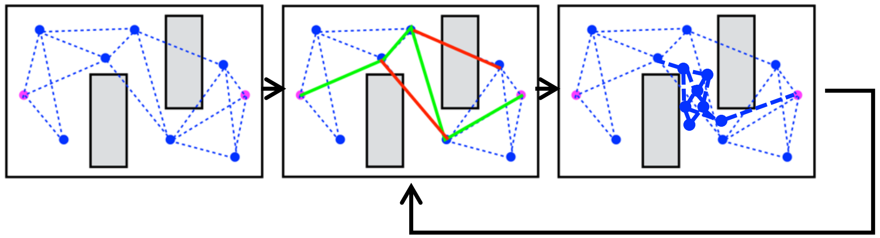




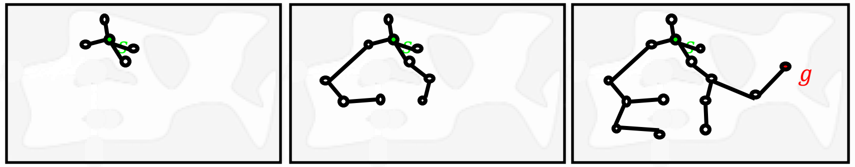
We can create and search the graph (e.g. grid search) in the C-space, however, the C-space is high-dimensional:

1. Computing the C-space obstacle is hard
2. Searching is computationally expensive

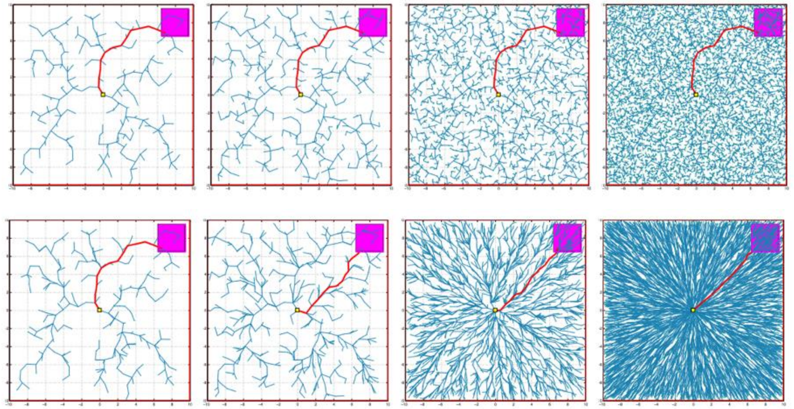
Motion Planning



Sampling-based Motion Planning



RRT vs. RRT*



Task and Planning

SubTask 1: Grasp the Cheezit

⋮

SubTask N: Grasp the mustard

- Subgoal configuration of the gripper pose
  - ➢ Motion planning for the trajectory

SubTask N+1: Lift up the mustard

- Subgoal configuration of the gripper pose
  - ➢ Motion planning for the trajectory

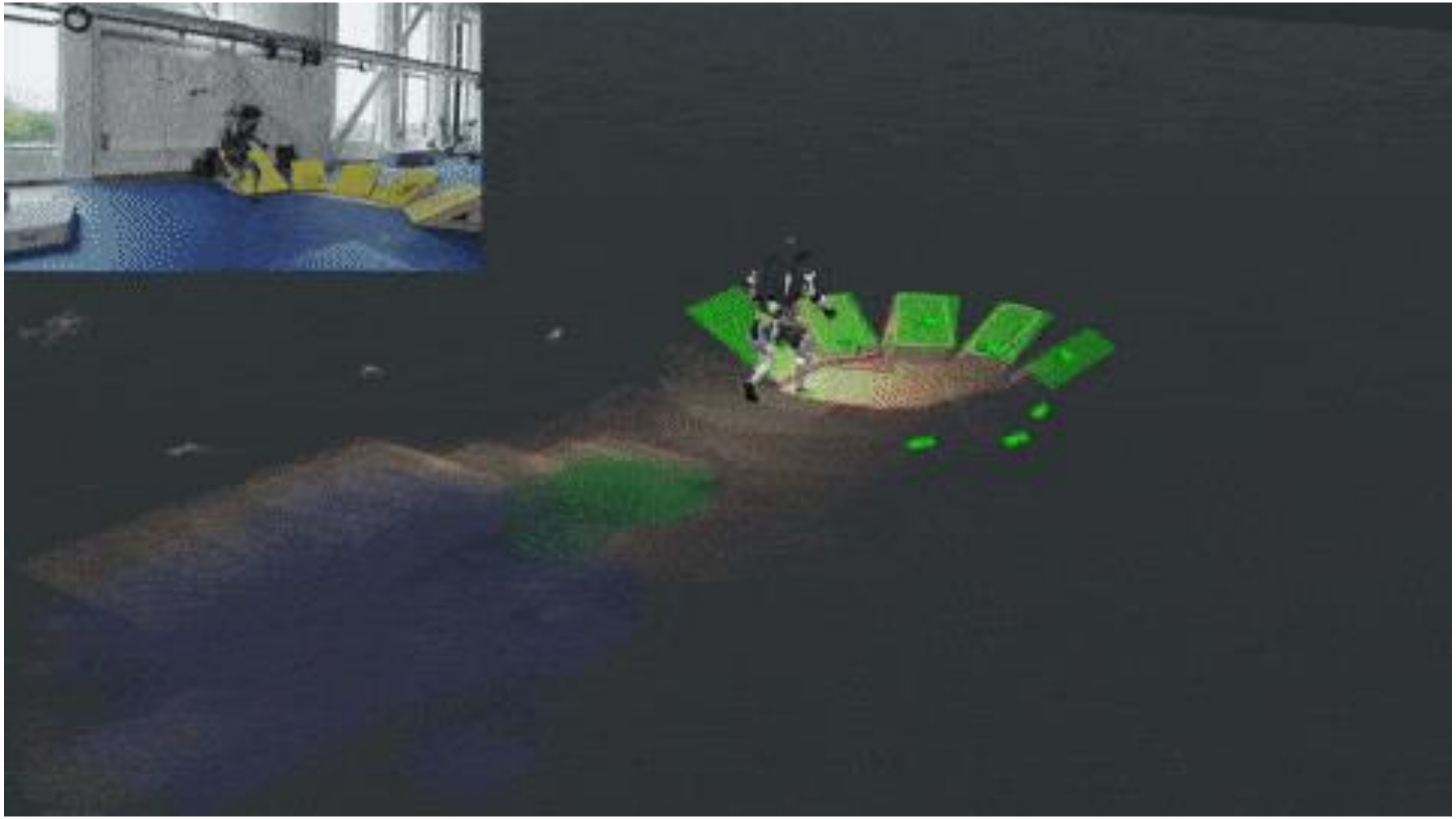
SubTask N+2: Carry the mustard above the blue region

- Subgoal configuration of the gripper pose
  - ➢ Motion planning for the trajectory

SubTask N+3: Put down the mustard

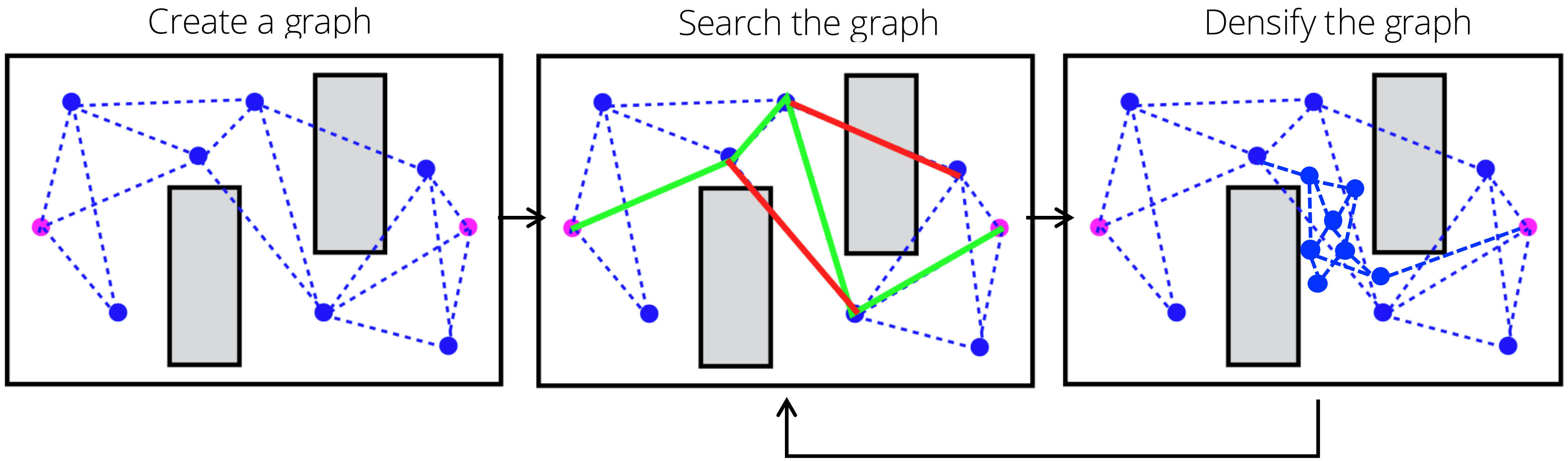
- Subgoal configuration of the gripper pose
  - ➢ Motion planning for the trajectory

How to Control Robots to Follow Plans?



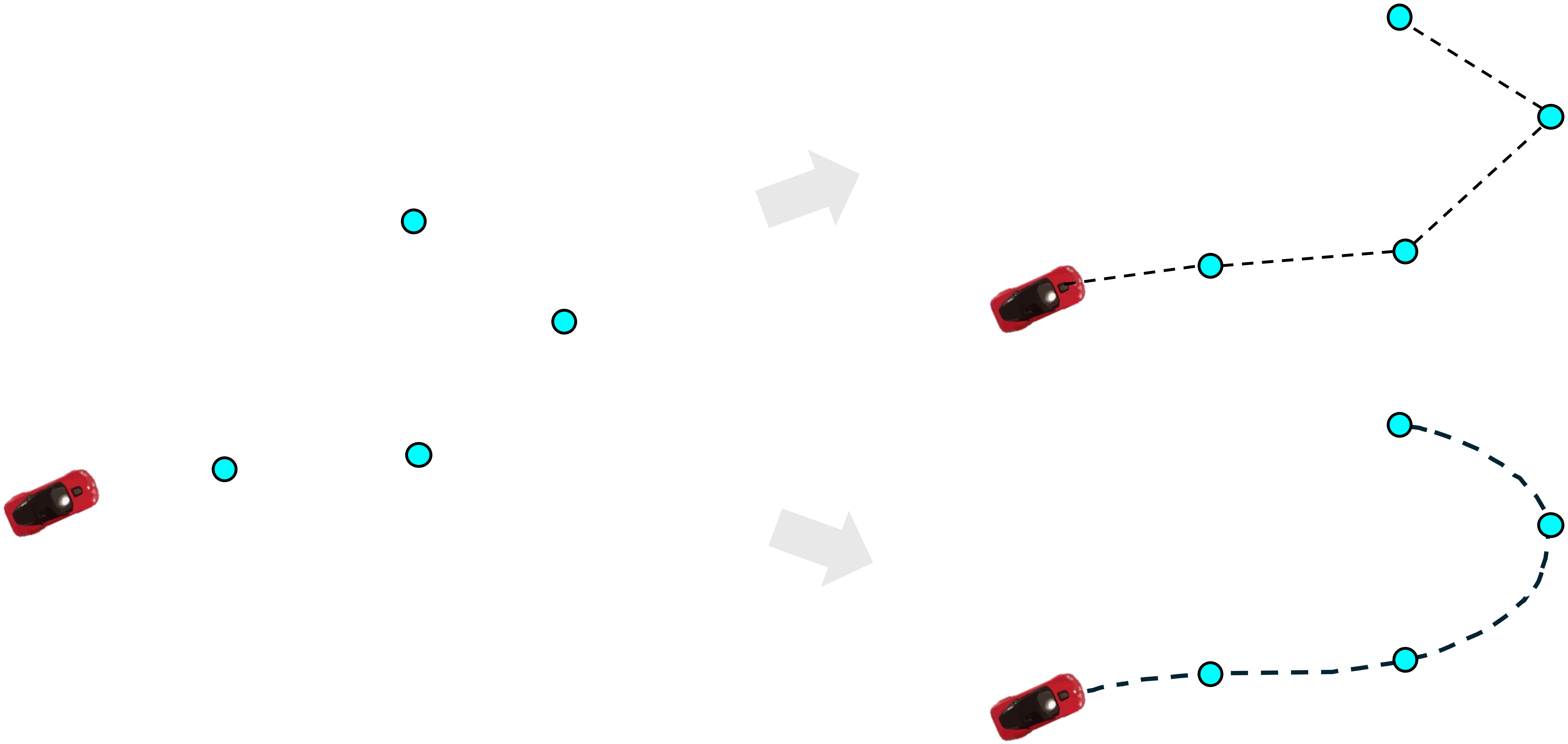
Video from Boston Dynamic
https://bostondynamics.com/blog/flipping-the-script-with-atlas/

What are Missing? We Ignore Motion Constraints



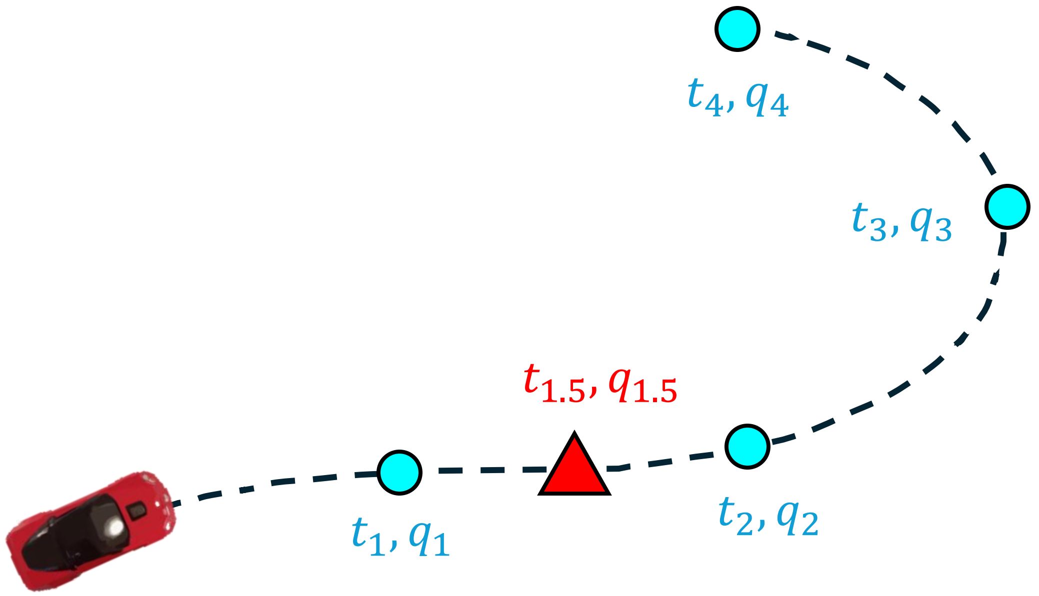
- We have only considered **path** and **obstacle**, but ignore:
  - ➢ How to traverse from one node to another

We Ignore How to Traverse the Planned Path



We Ignore the Dimension of Time for the Path

Trajectory: time-parameterized path $q(t)\ \forall t \in [0, T]$



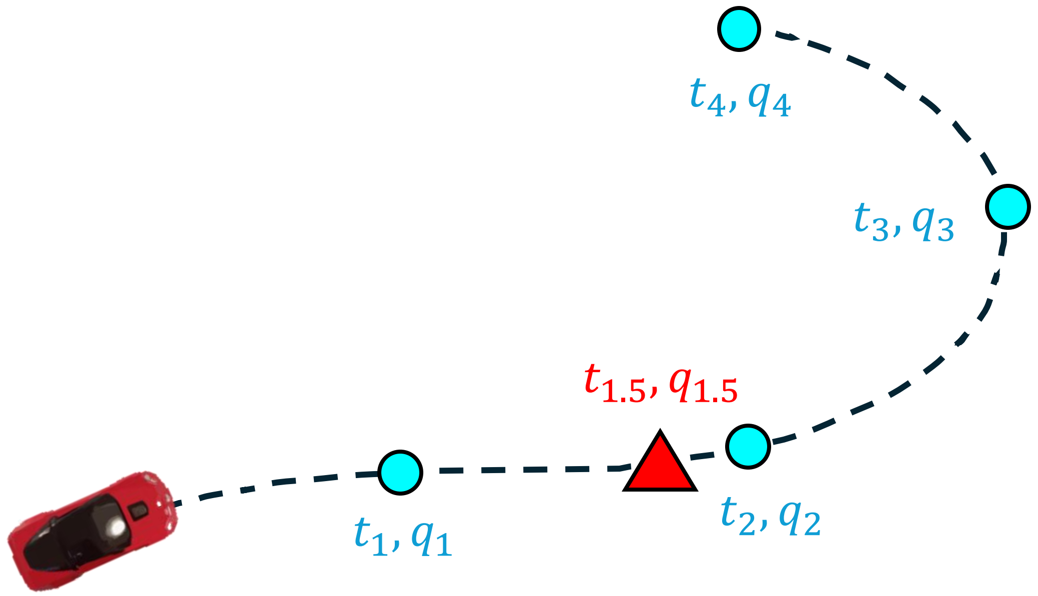


Image credit S. Choudhury

We Ignore How to Control to Follow the Trajectory

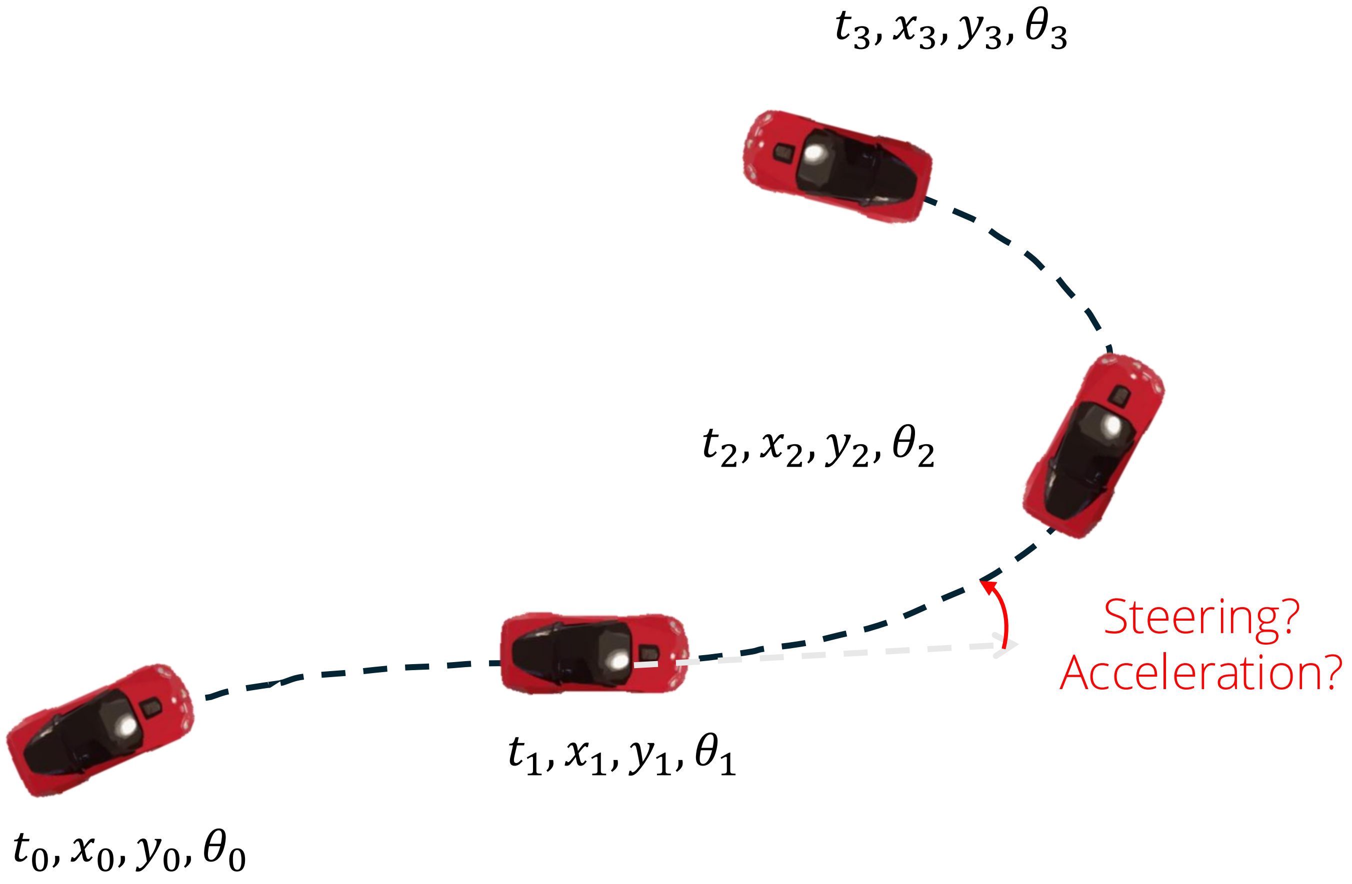
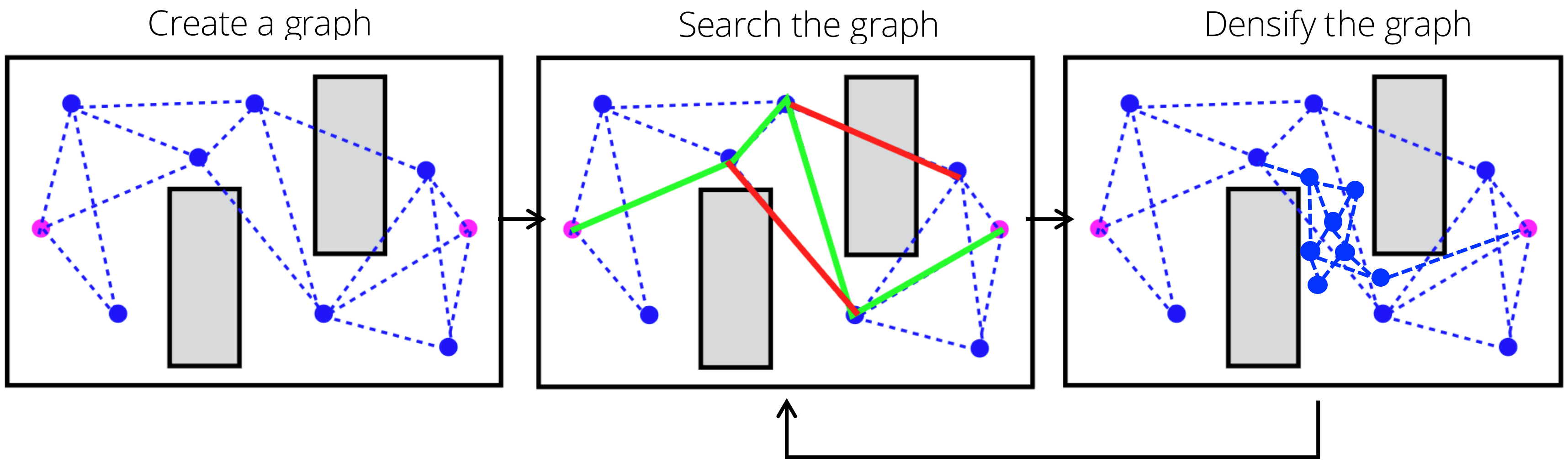


Image credit S. Choudhury

What are Missing? We Ignore Motion Constraints



- We have only considered **path** and **obstacle**, but ignore:
  - ➢ How to traverse from one node to another
  - ➢ Motion (velocity/acceleration) constraints
  - ➢ Safety constraints

Control is Hard, as Robots may be Underactuated…

Underactuated system: a mechanical system that cannot be commanded to follow arbitrary trajectories in configuration space. Obvious cases are systems that have less actuators than degrees of freedom.

Definition 1.1 (Underactuated Control Differential Equations) A second-order control differential equation described by the equations

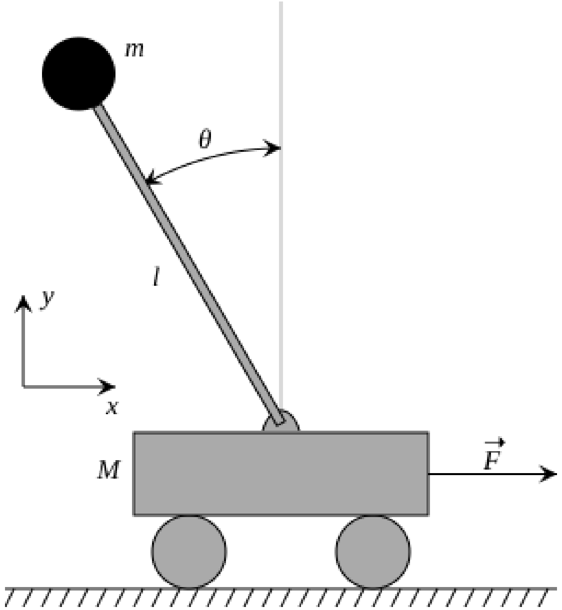
$$\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u}, t) \tag{1}$$

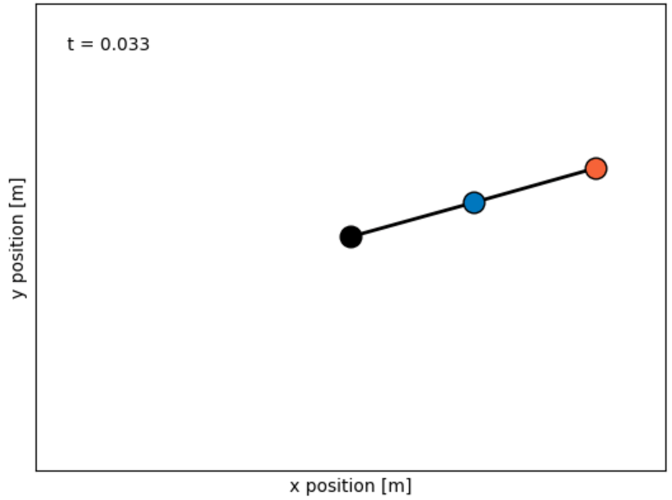
is *fully actuated* in state $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$ and time t if the resulting map $\mathbf{f}$ is surjective: for every $\ddot{\mathbf{q}}$ there exists a $\mathbf{u}$ which produces the desired response. Otherwise it is *underactuated* (at state $\mathbf{x}$ at time t).

$\boldsymbol{q}$: a vector of robot configurations; $\dot{\boldsymbol{q}}$: a vector of velocities

Control is Hard, as Robots may be Underactuated…

Underactuated system: a mechanical system that cannot be commanded to follow arbitrary trajectories in configuration space. Obvious cases are systems that have less actuators than degrees of freedom.

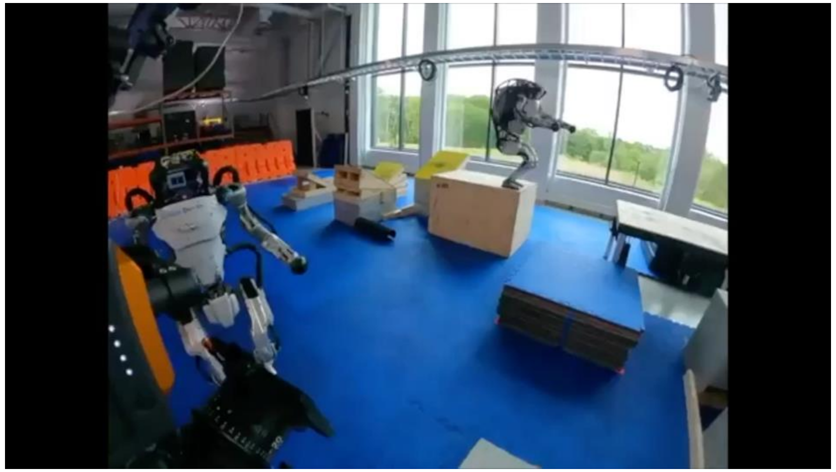




Video credit W. Felix, K. Shivesh, S. Lasse J., V. Shubam and J. Ma.

We Ignore the Dynamics…

https://www.youtube.com/watch?v=RqajKat0v-4

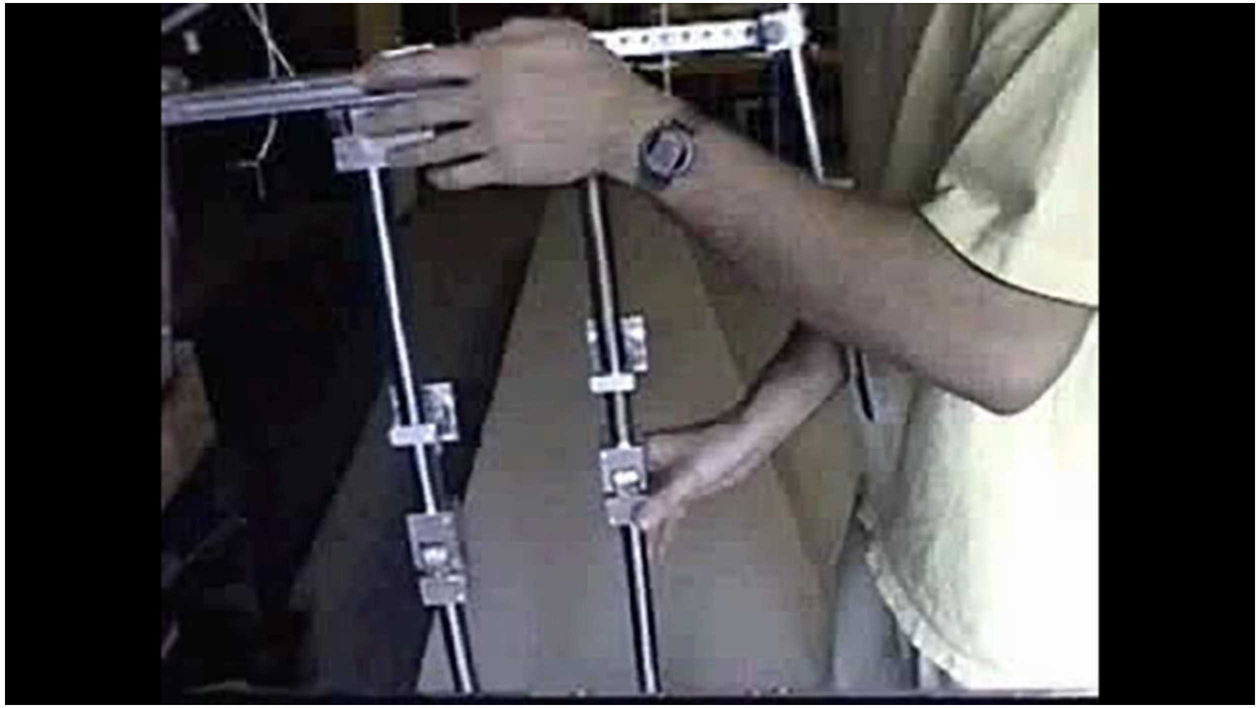


https://www.youtube.com/watch?v=-9EM5_VFlt8&t=2s

https://www.youtube.com/watch?v=eFsT3QgIvoI

Control Systems that Cancel vs. Involve Dynamic

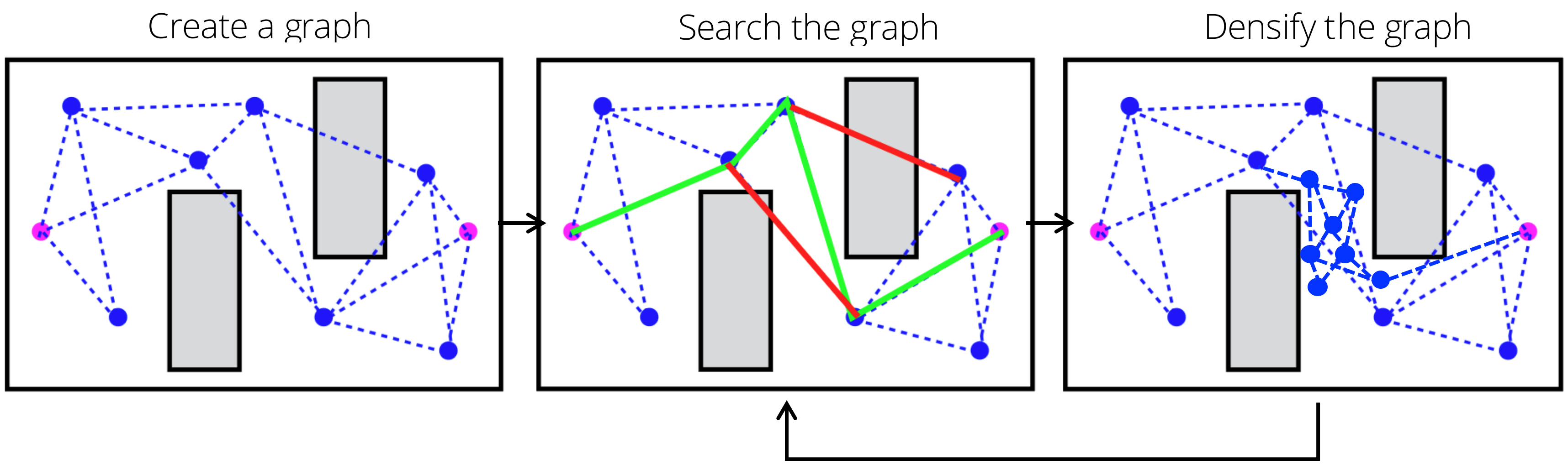
Honda Asimo: Video credit https://youtu.be/vA0xLVCb-OA?si=Z4MKkuM4Wr0H7NJw



Steven Collins. Video credit https://youtu.be/PK7cgLJD2nQ?si=5d-Ke74URVlejBpX

Which one looks more natural? Which one consumes less energy?

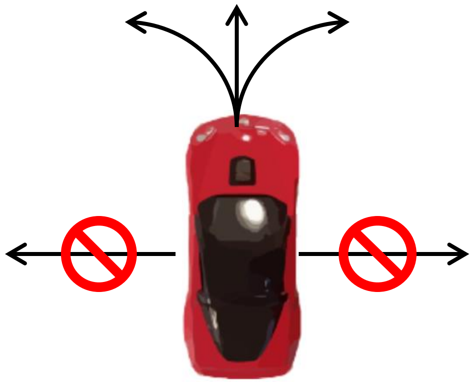
Can We Extend the Motion Planning Framework?

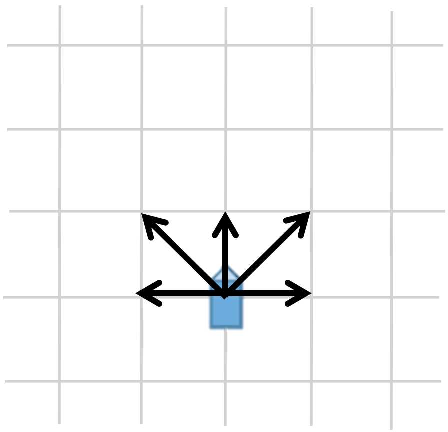


- Can we extend the motion planning framework to:
  - ➢ Consider configuration constraints
  - ➢ Consider motion (velocity/acceleration) constraints
  - ➢ Generate control trajectories
  - ➢ Involve dynamic

Can We Generate Paths that Respect Directional Constraints?

- Take *car* as an example. How can we reformulate the roadmap planners?





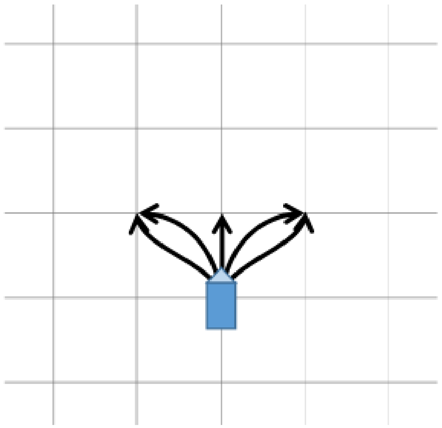
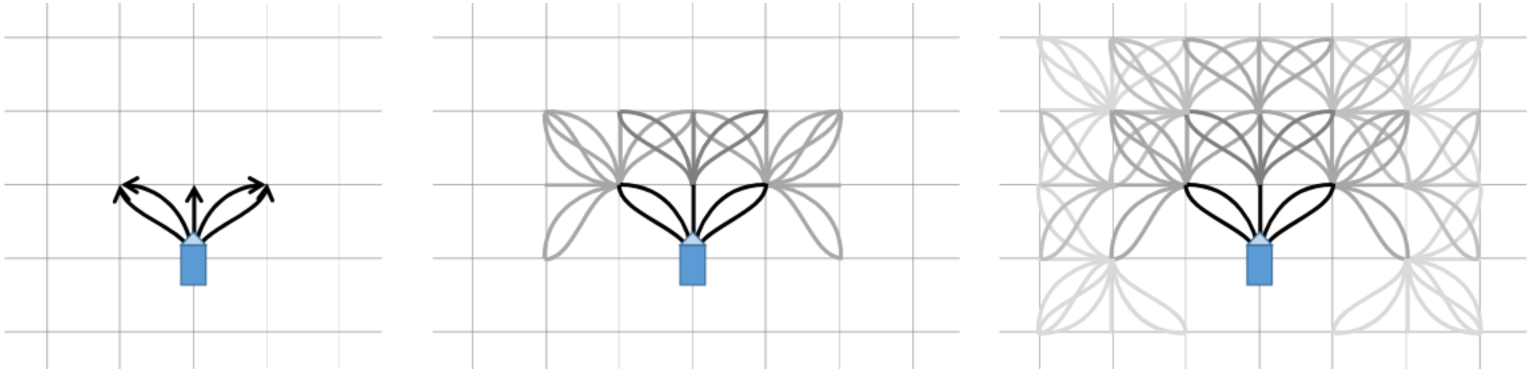


Image credit K. Hauser

Path Planning with Directional Constraints



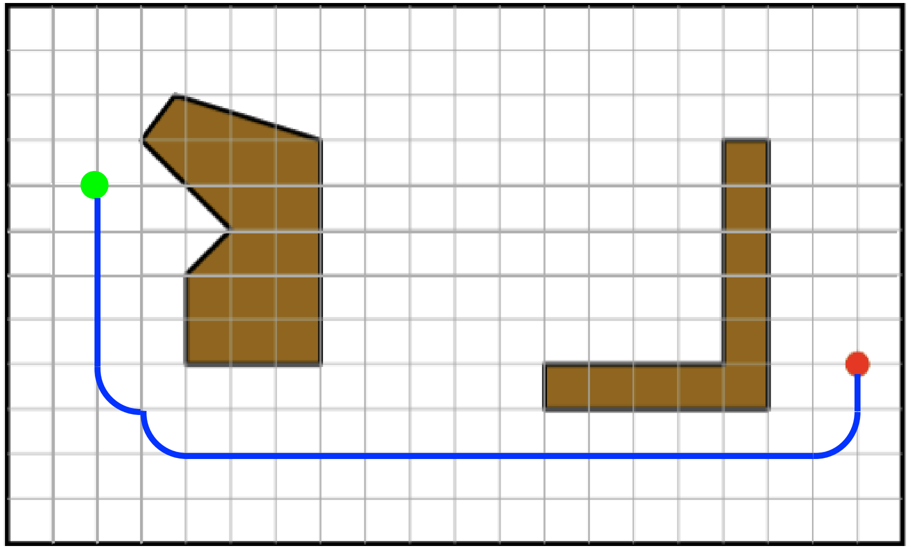
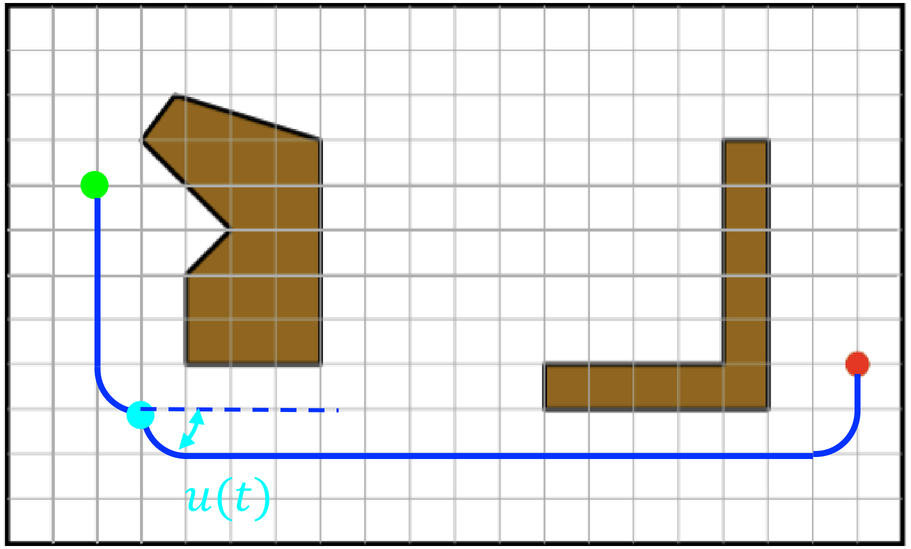


Image credit K. Hauser

Can We Reformulate Path Planning Problems to Generate Control Trajectory?

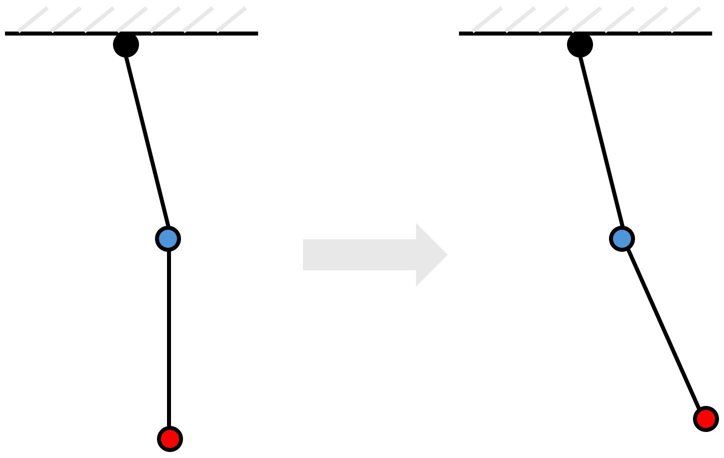
Control trajectory $u(t)$:

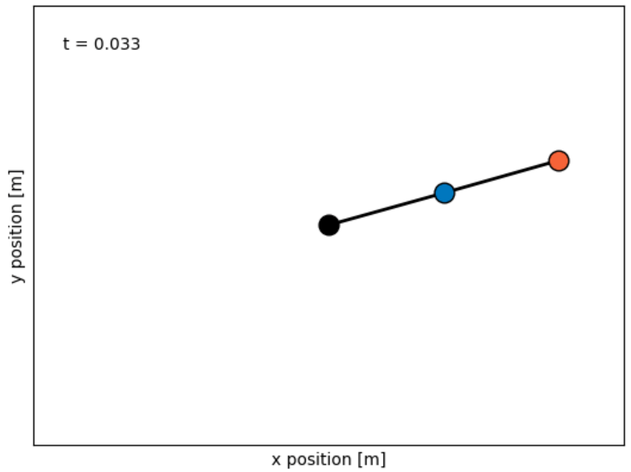
- Acceleration, determined by imposed force/torque, at time t
- Velocity at time t (e.g. stepper motors)



Control is Hard, as Robots may be Underactuated…

- **Underactuated system**: a mechanical system that cannot be commanded to follow arbitrary trajectories in configuration space. Obvious cases are systems that have less actuators than degrees of freedom.
- The double pendulum with one missing actuator is underactuated. The neighboring configurations take more motions to achieve

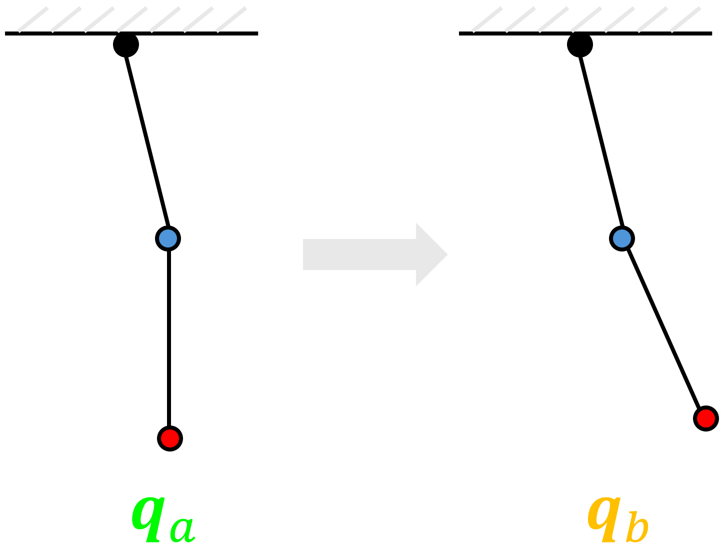


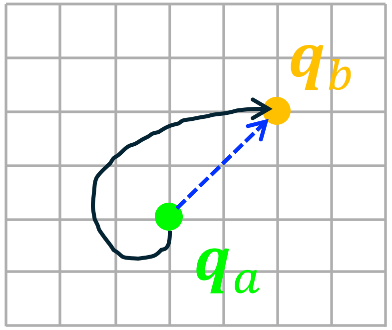


Video credit W. Felix, K. Shivesh, S. Lasse J., V. Shubam and J. Ma.

Control is Hard, as Robots may be Underactuated…

- **Underactuated system**: a mechanical system that cannot be commanded to follow arbitrary trajectories in configuration space. Obvious cases are systems that have less actuators than degrees of freedom.
- The double pendulum with one missing actuator is underactuated. The neighboring configurations take more motions to achieve
- Idea: Building graph by sampling real/simulated interactions

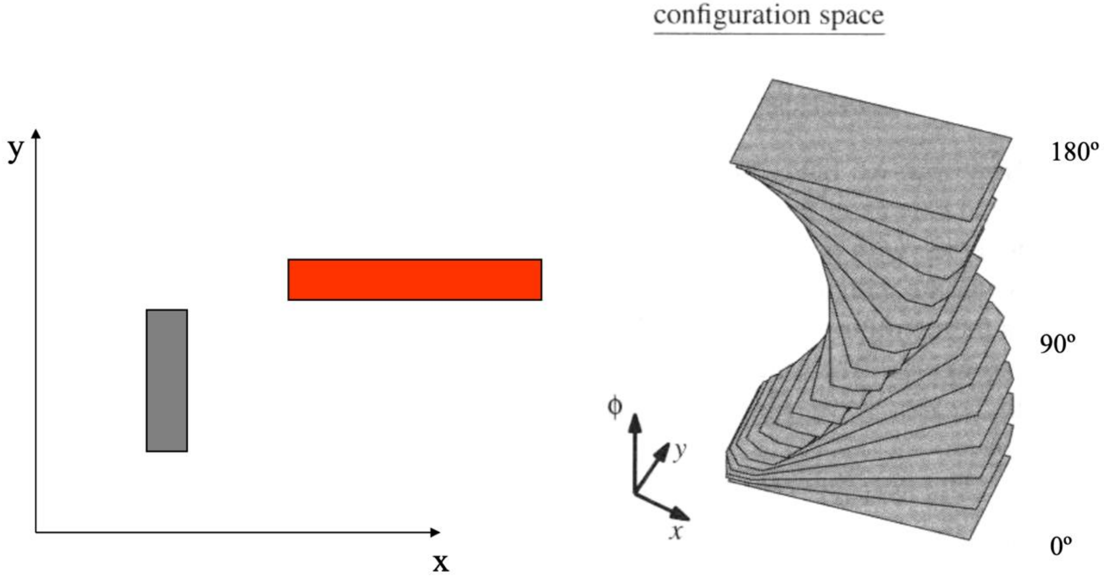




Actual trajectory depends on force constraints and velocity conditions

Can We Generate Control Trajectories with Motion Planning Methods?

- To generate control trajectories, we need to take dynamic into consideration
- Previously, we only consider "configurations" (joint angles) of a robot. Such representations ignore "dynamics" of a robot.
  - ➢ Is the $\boldsymbol{C}$-space obstacle fixed when the initial velocity $\boldsymbol{v}$ of an object varies?



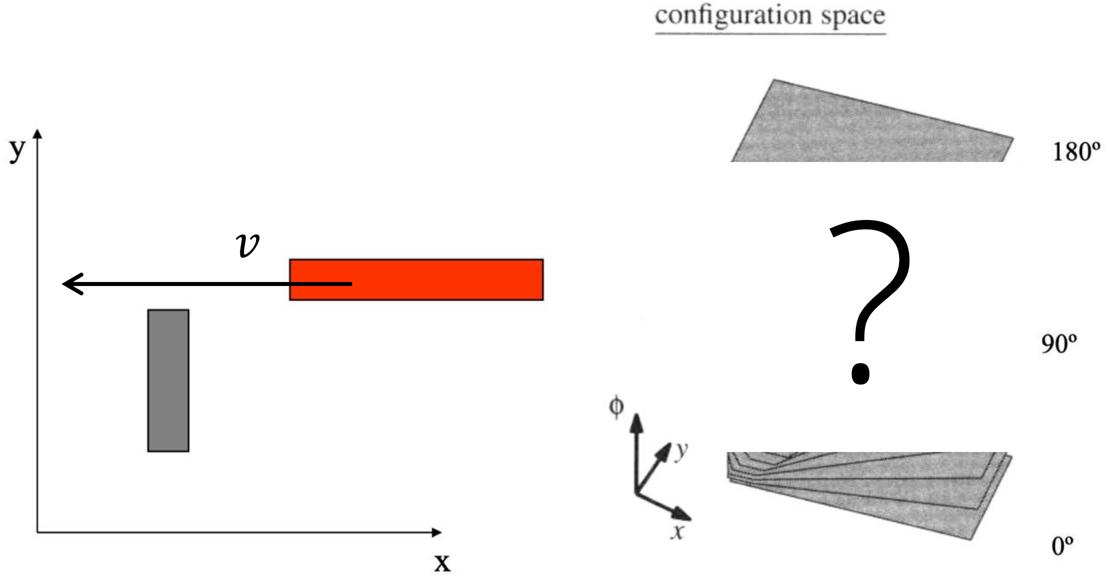


Image credit G.D. Hager and Z. Dodds

State-space Representations

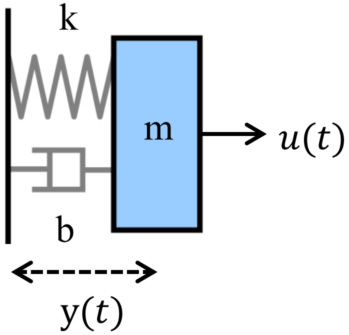
- Generalize the planning space from configurations (the configuration space $\mathcal{C}$) to dynamics (the state space $\mathcal{X}$)
- **Definition**: a state vector $x \in \mathcal{X}$ as $x = \begin{bmatrix} q \\ \dot{q} \\ \vdots \end{bmatrix}$, then we have $\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \\ \vdots \end{bmatrix} = f(x, u, t)$,

  where $q \in \mathcal{C}$, u denotes controls (e.g. torque) and function f describes how the motion changes conditioned on the state x, control u and time t

State-space Representations

- Generalize the planning space from configurations (the configuration space $\mathcal{C}$) to dynamics (the state space $\mathcal{X}$)
- **Definition**: a state vector $x \in \mathcal{X}$ as $x = \begin{bmatrix} q \\ \dot{q} \\ \vdots \end{bmatrix}$, then we have $\dot{x} = \begin{bmatrix} \dot{q} \\ \ddot{q} \\ \vdots \end{bmatrix} = f(x, u, t)$,

  where $q \in \mathcal{C}$, u denotes controls (e.g. torque) and function f describes how the motion changes conditioned on the state x, control u and time t



A 2nd-order equation of motion:

$$m\ddot{q}(t) = u(t) - b\dot{q}(t) - kq(t)$$

Rewrite the equation as:

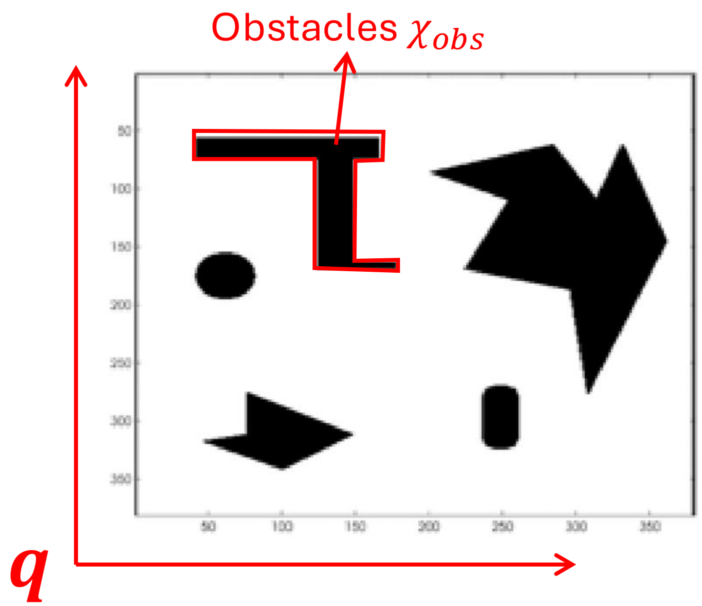
$$\dot{x} = \begin{bmatrix} \dot{q}(t) \\ \ddot{q}(t) \end{bmatrix} = \begin{bmatrix} \dot{q}(t) \\ \frac{1}{m}u(t) - \frac{b}{m}\dot{q}(t) - \frac{k}{m}q(t) \end{bmatrix} = \underbrace{[\quad]}_{A} x(t) + \underbrace{[\quad]}_{B} u(t)$$

Image adapted from https://en.wikipedia.org/wiki/Mass-spring-damper_model

Obstacles in the State Space

Let's assume we have a point mass robot

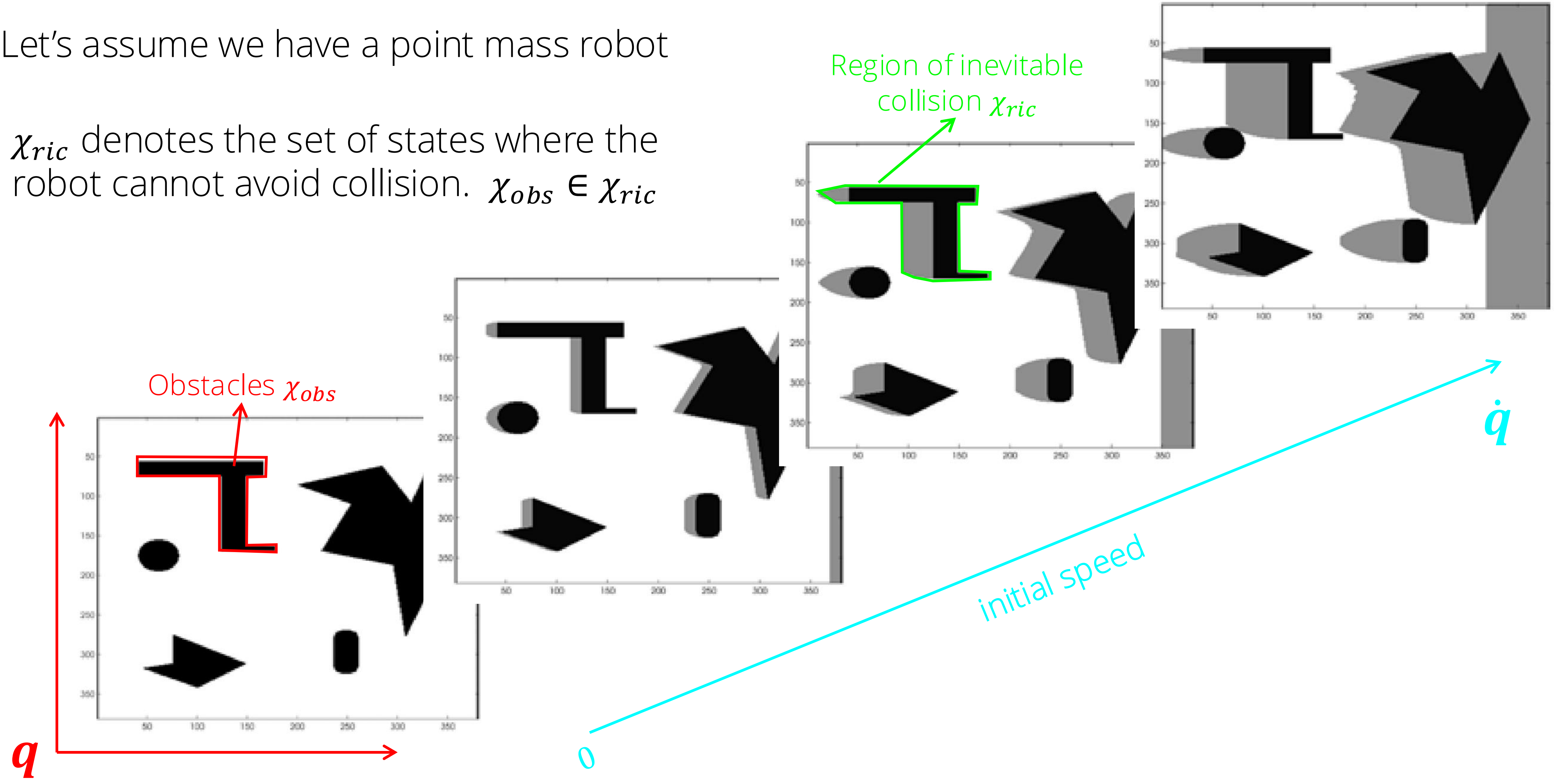
χ_{ric} denotes the set of states where the robot cannot avoid collision. $\chi_{obs} \in \chi_{ric}$



Obstacles in the State Space

Let's assume we have a point mass robot

χ_{ric} denotes the set of states where the robot cannot avoid collision. $\chi_{obs} \in \chi_{ric}$



Kinodynamic Planning with RRT

- Kinematic RRT: Grow a tree of **feasible configuration paths** from the start to the goal configuration
  - ➢ Select a random / the goal configuration $\boldsymbol{q}$, find the nearest configuration $\boldsymbol{q_{near}}$ to $\boldsymbol{q}$, and greedily move from $\boldsymbol{q_{near}}$ to $\boldsymbol{q}$.

- Kinodynamic RRT: Grow a tree of feasible **control trajectories** $\boldsymbol{u(t), t \in \{0, \ldots, T\}}$ from the start to the goal state
  - ➢ Select a random / the goal configuration $\boldsymbol{q}$, find the nearest configuration $\boldsymbol{q_{near}}$ to $\boldsymbol{q}$, randomly select a control $\boldsymbol{u}$, and execute $\boldsymbol{u}$ from $\boldsymbol{q_{near}}$.

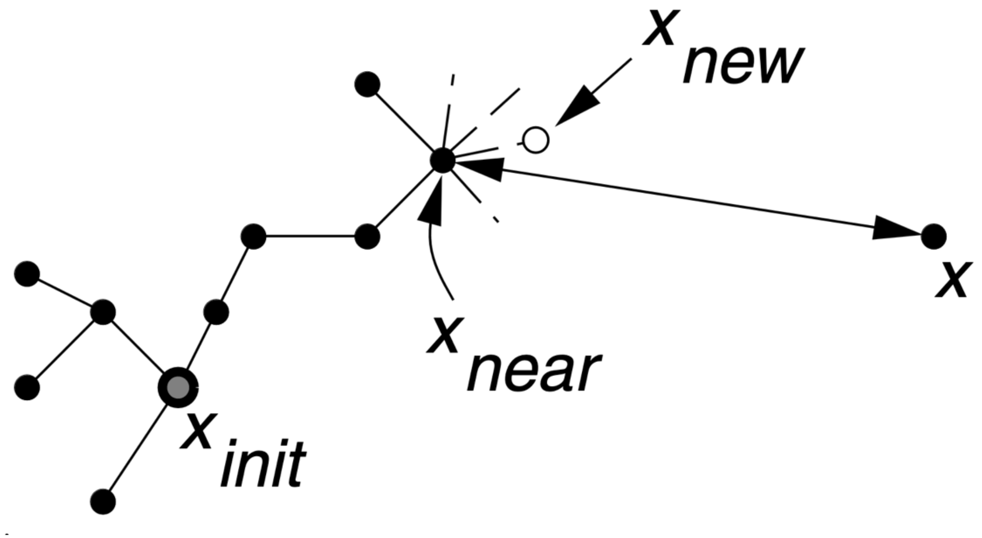
Randomized Kinodynamic Planning. S. LaVallel and J. J. Kuffner.

Kinodynamic Planning with RRT

```
BUILD_RRT(x_init)
    T.init(x_init);
    for k = 1 to K do
        x_rand ← RANDOM_STATE();
        EXTEND(T, x_rand);
    Return T
```

```
EXTEND(T, x)
    x_near ← NEAREST_NEIGHBOR(x, T);
    if NEW_STATE(x, x_near, x_new, [u_new]) then
        T.add_vertex(x_new);
        T.add_edge(x_near, x_new, [u_new]);
        if x_new = x then
            Return Reached;
        else
            Return Advanced;
    Return Trapped;
```

- Randomly sample u_{new}
- Find u_{new} that yields x_{new} as close as possible to x

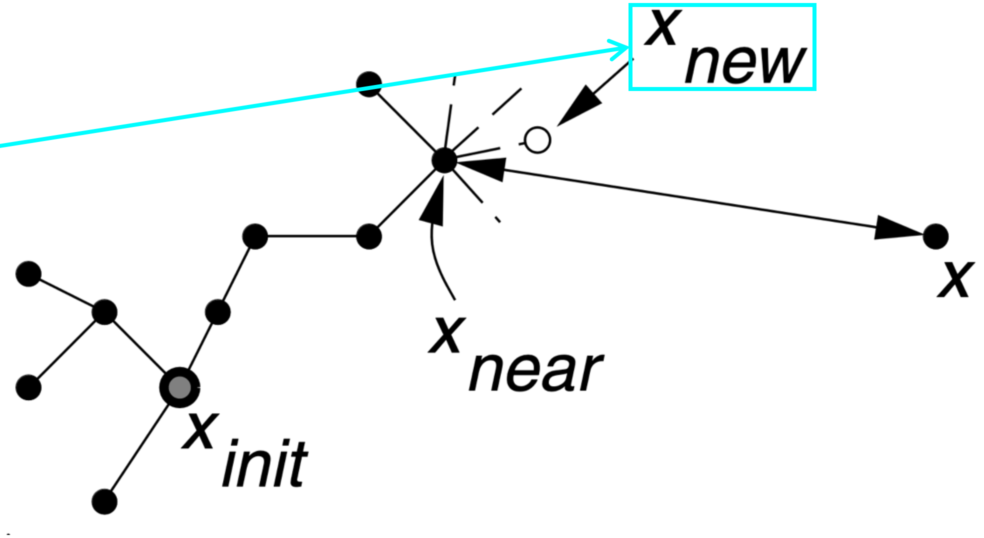


Kinodynamic Planning with RRT

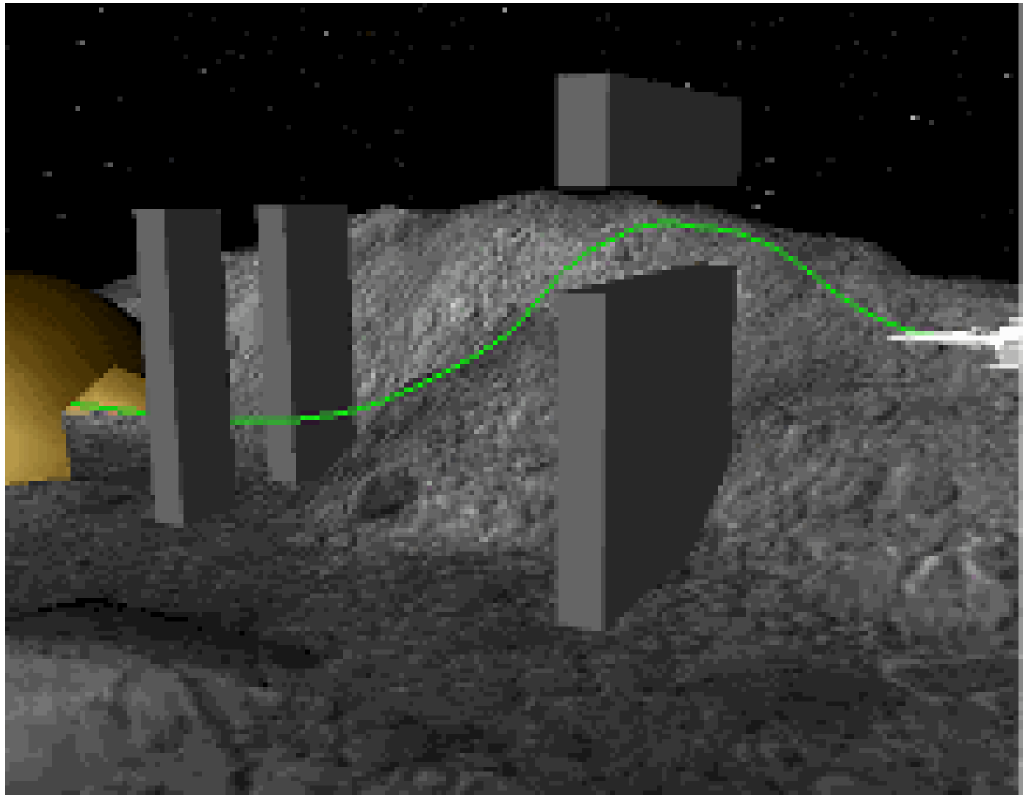
```
BUILD_RRT(x_init)
    T.init(x_init);
    for k = 1 to K do
        x_rand ← RANDOM_STATE();
        EXTEND(T, x_rand);
    Return T
```

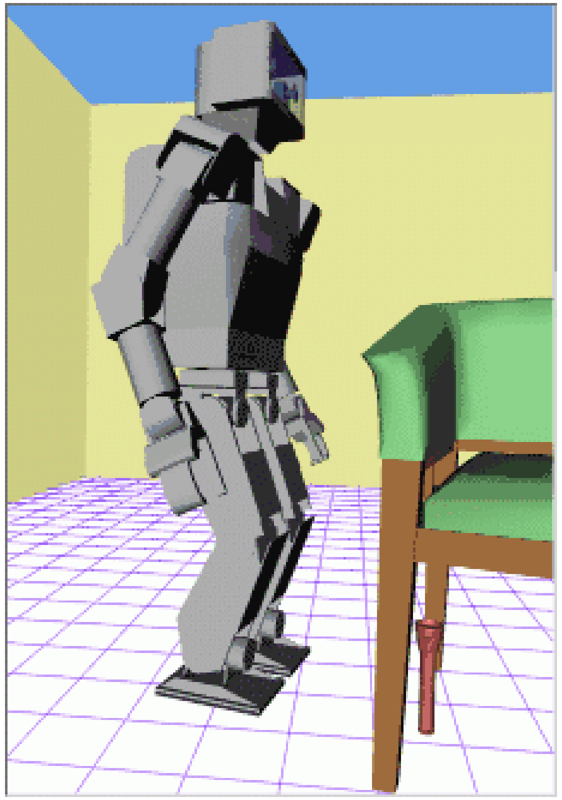
```
EXTEND(T, x)
    x_near ← NEAREST_NEIGHBOR(x, T);
    if NEW_STATE(x, x_near, x_new, u_new) then
        T.add_vertex(x_new);
        T.add_edge(x_near, x_new, u_new);
        if x_new = x then
            Return Reached;
        else
            Return Advanced;
    Return Trapped;
```

Execute u_{new} to obtain x_{new}. Also, do collision checking on x_{new}



Kinodynamic Planning with RRT





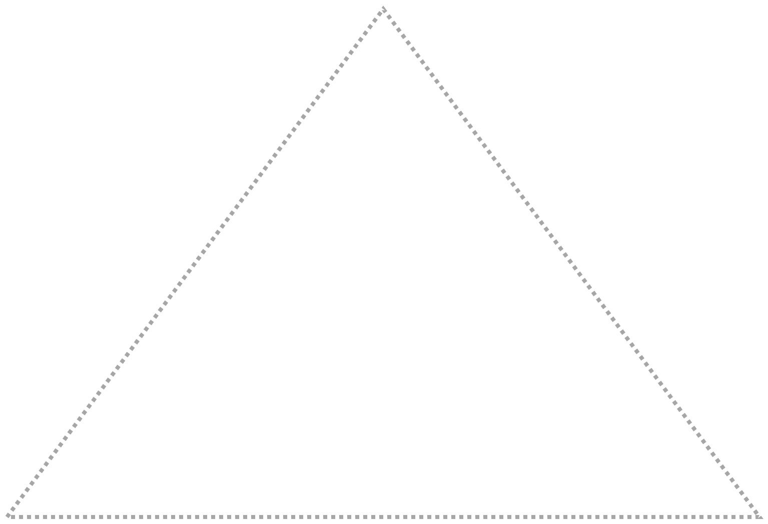
Vidoe credit S. LaVallel and J. J. Kuffner

Kinodynamic RRT Still Has Problems

- Metric choice: The Euclidean distance between two states often correspond poorly with the length of exact control trajectories
  - We need to decide x_{near}
  - We need to decide if x_{new} is close enough to x
- Control choice: we need to sample a lot of u_{new}, which is sample inefficient
- Open-loop control: we decide the whole control trajectory from the initial state. We need to replan again the whole trajectory if something goes wrong…

Randomized Kinodynamic Planning. S. LaVallel and J. J. Kuffner.

Open-loop Control: Plan and Execute the Whole Trajectory without Feedback

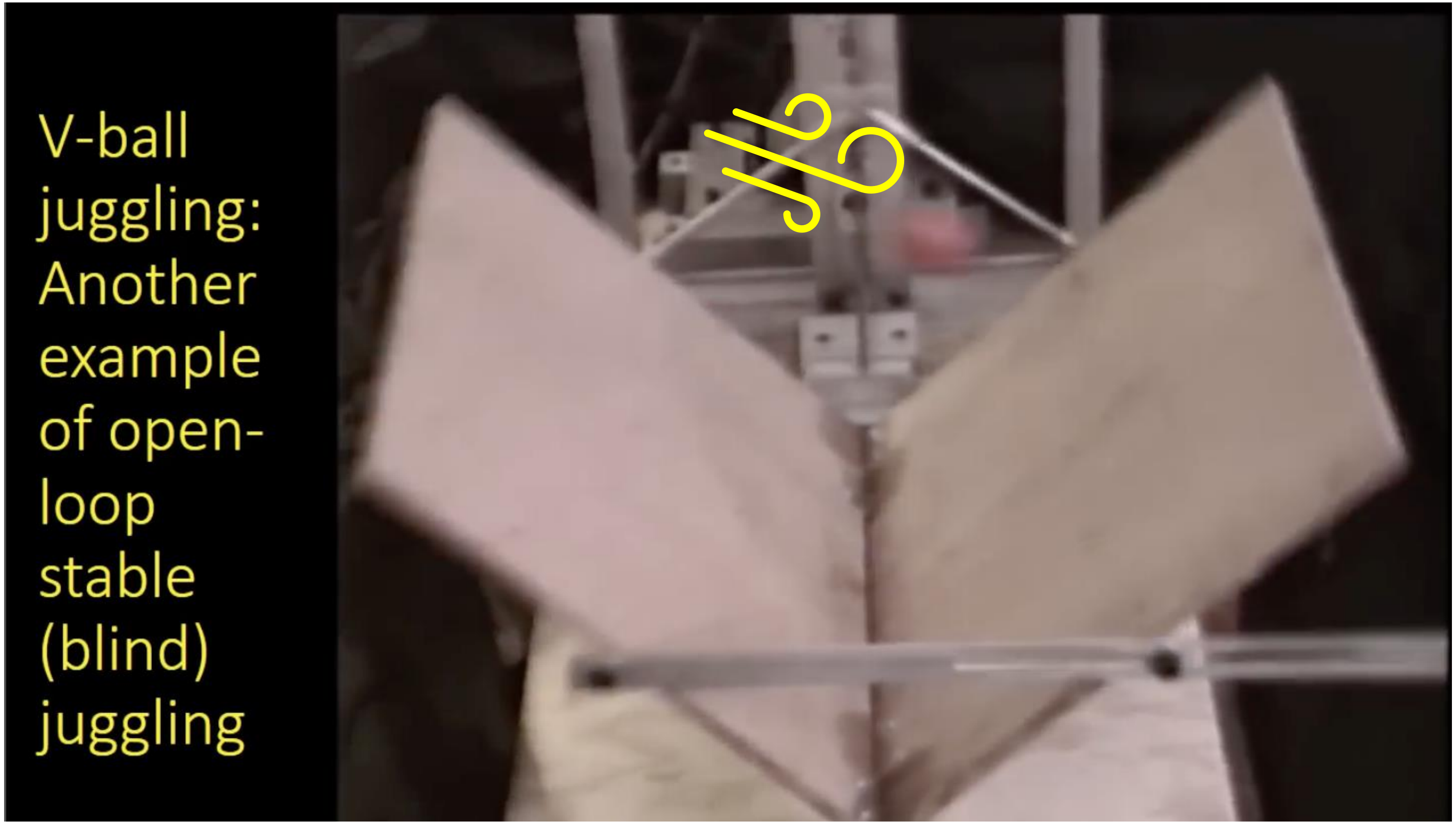


Open-loop Control: Plan and Execute the Whole Trajectory without Feedback

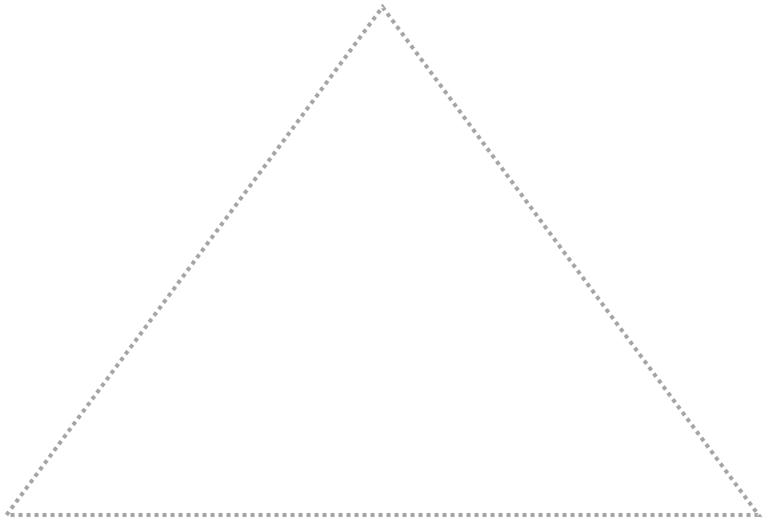
Open-loop Control for Robot Juggling

https://youtu.be/2ZfaADDlH4w?si=ilwJvLjgGYoZufP1&t=856

What if the Ball is Disturbed? We need to Re-decide the Control Inputs

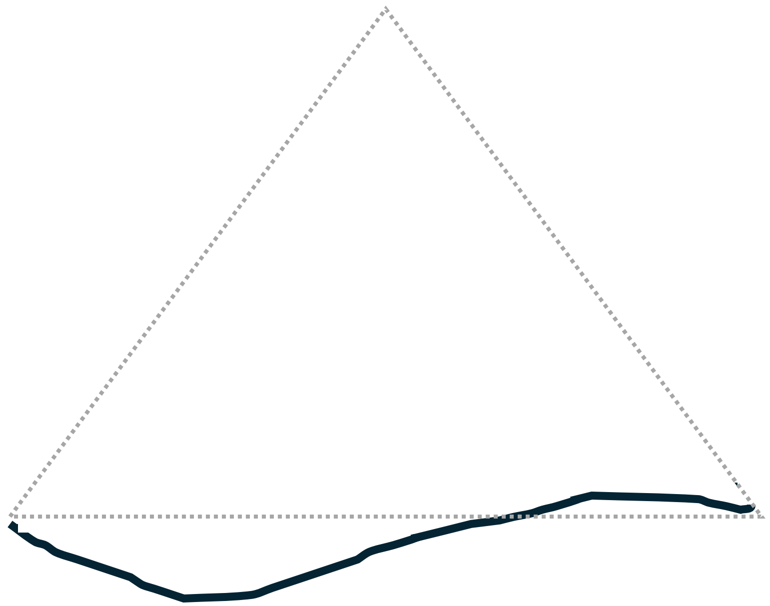


Close-loop Control: Repeat Planning and Executing the Trajectory with Feedback

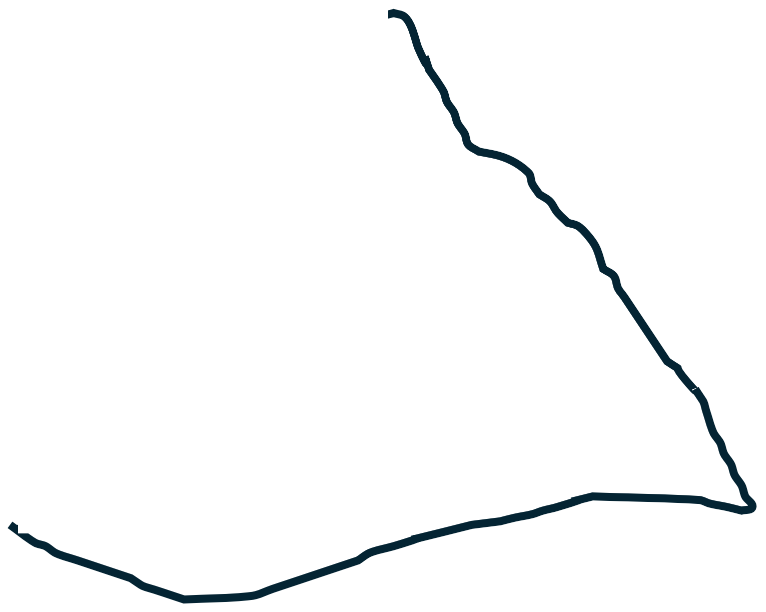


Close-loop Control: Repeat Planning and Executing the Trajectory with Feedback

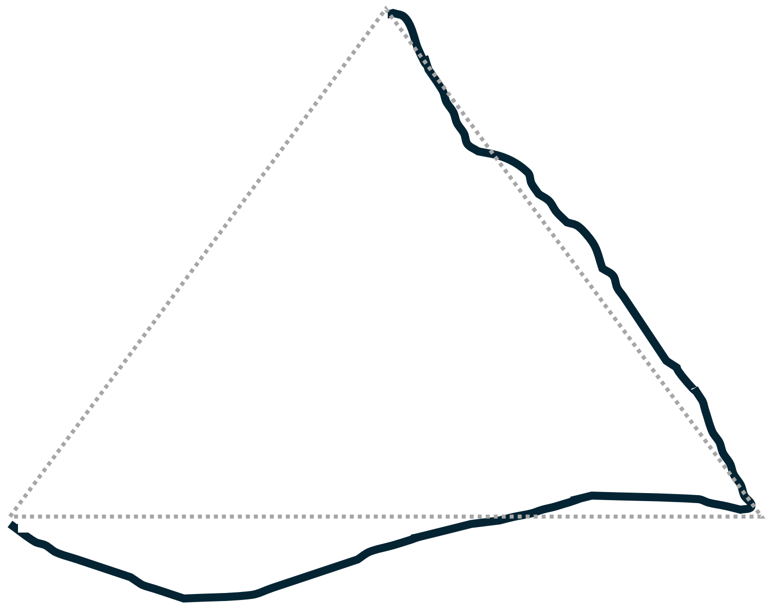
Close-loop Control: Repeat Planning and Executing the Trajectory with Feedback



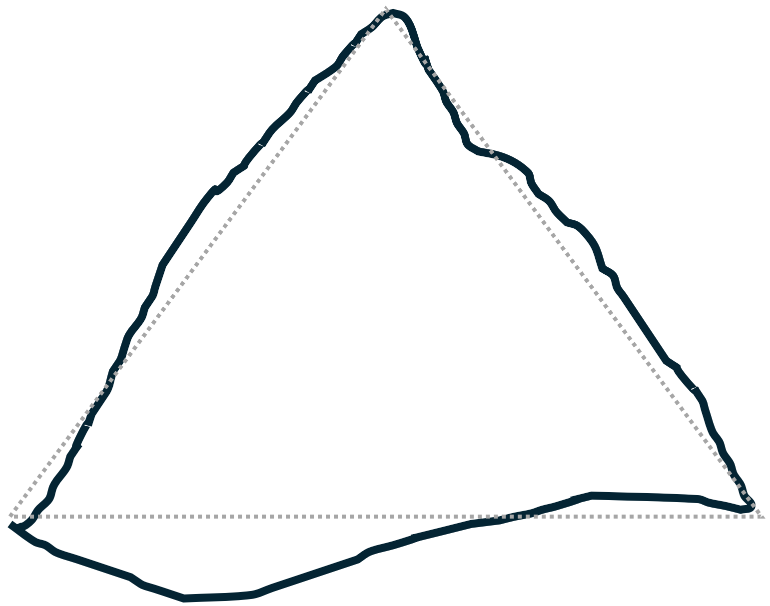
Close-loop Control: Repeat Planning and Executing the Trajectory with Feedback



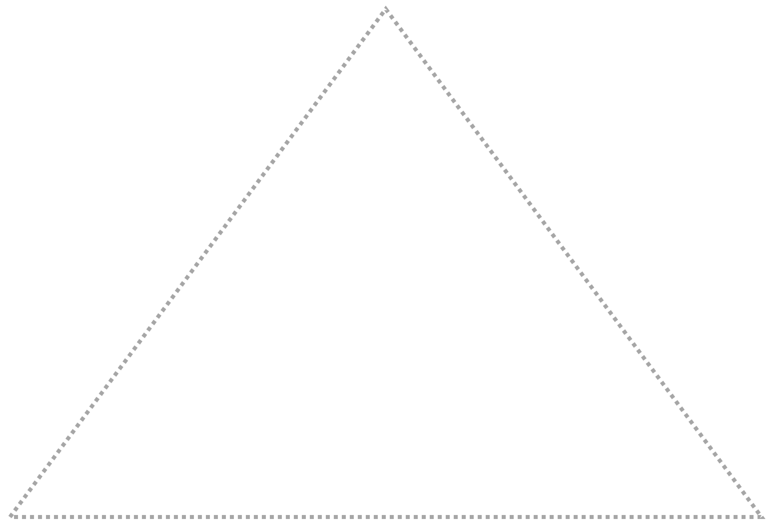
Close-loop Control: Repeat Planning and Executing the Trajectory with Feedback



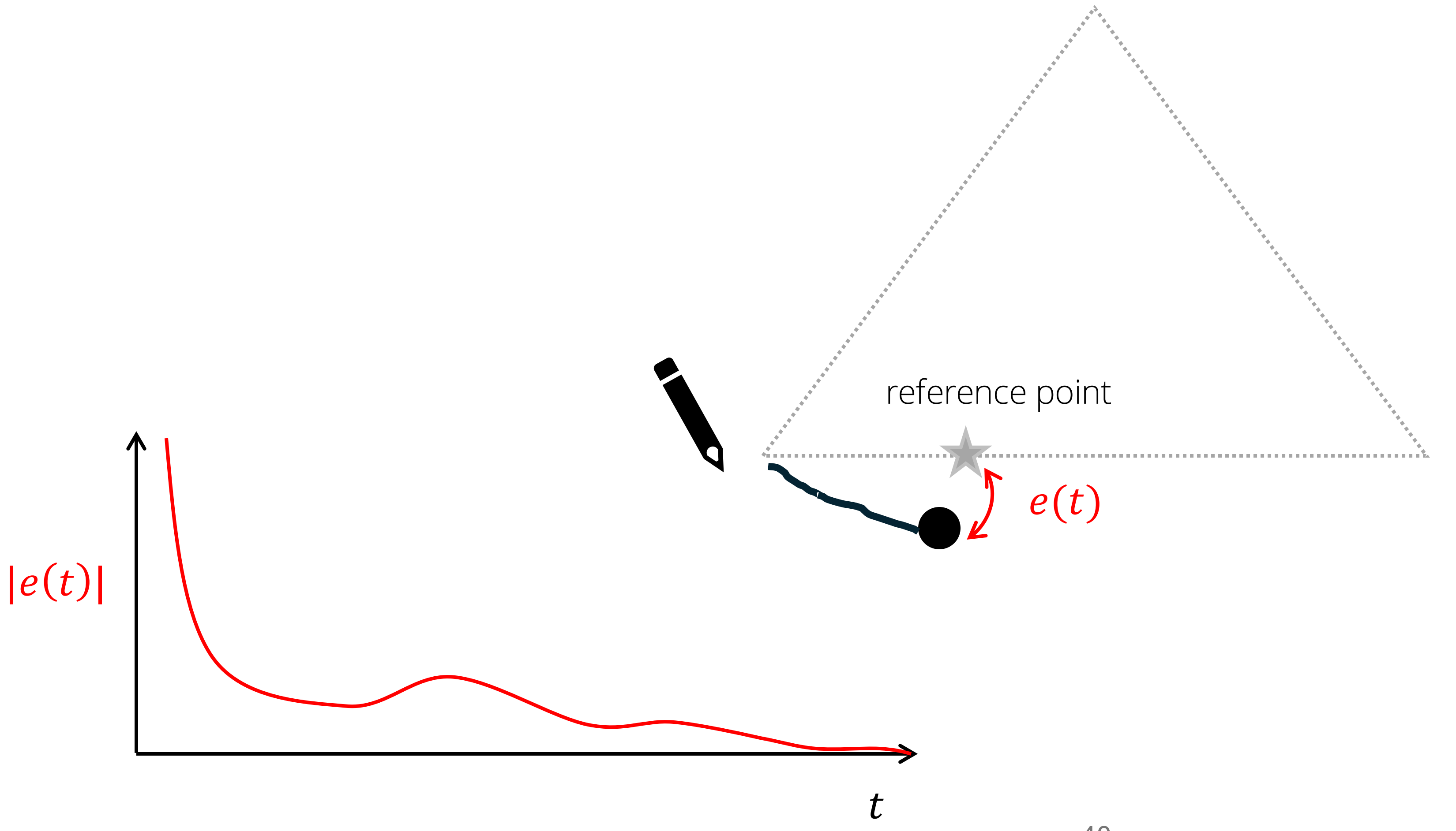
Close-loop Control: Repeat Planning and Executing the Trajectory with Feedback



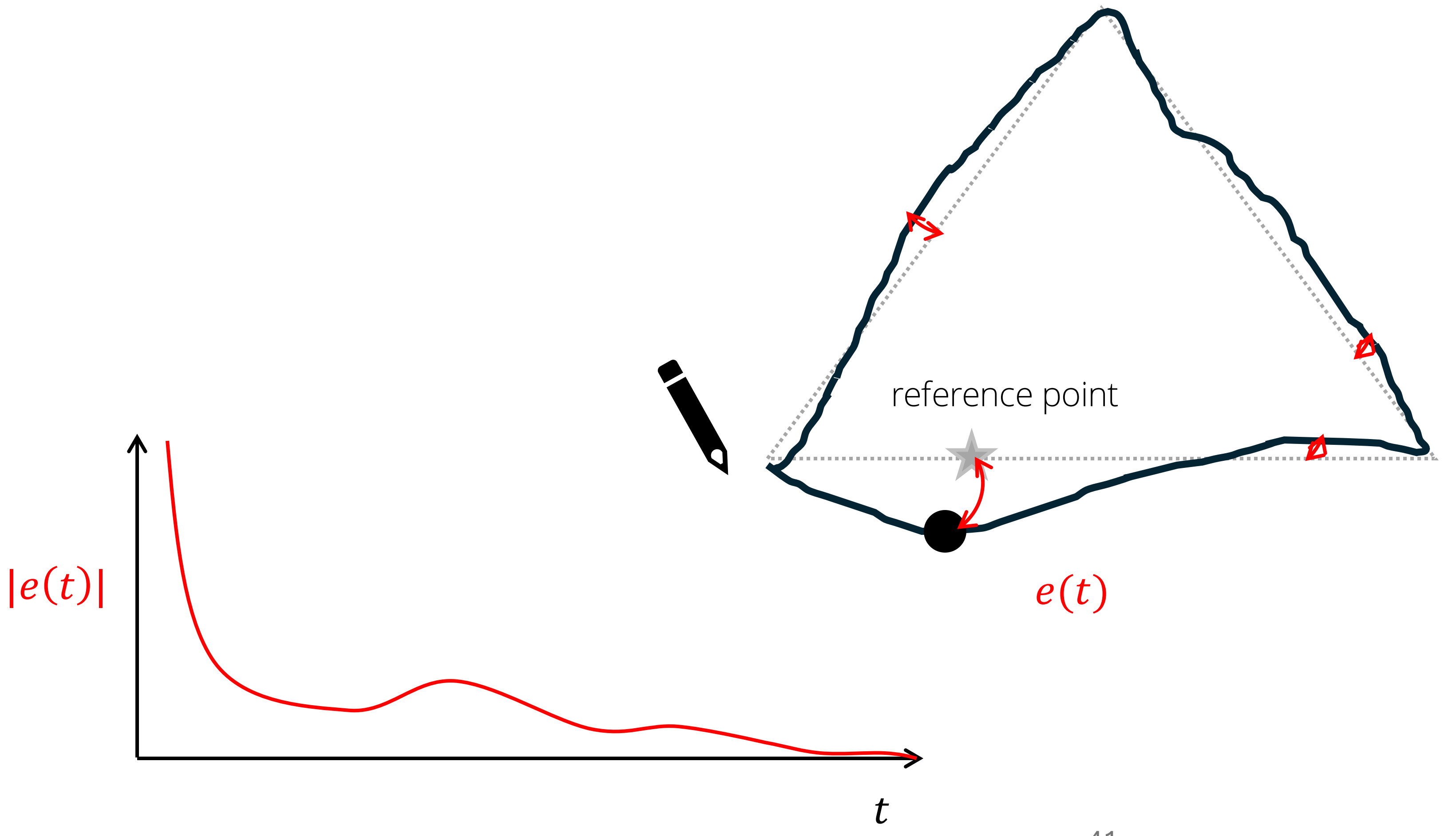
Feedback Control: Track a reference trajectory



Measure and Minimize Error to Reference Points

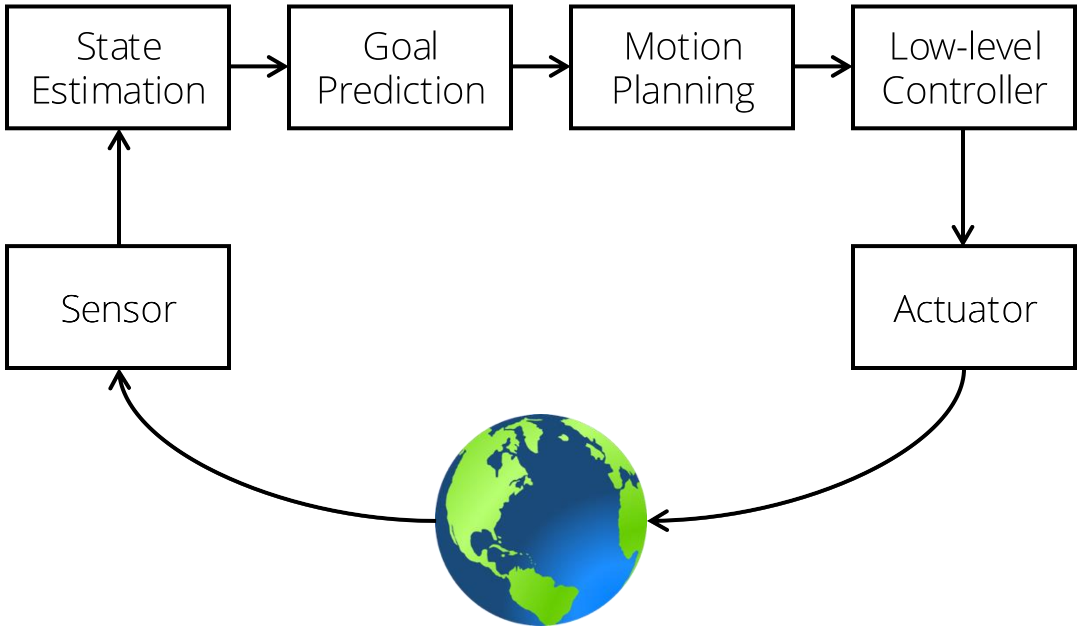


Measure and Minimize Error to Reference Points



Basic Recipe of Planning and Control

1. Motion planning: compute a feasible path
   - grid search / PRM / RRT ...
2. Controller: predict a control to follow the path or minimize the "tracking" error.

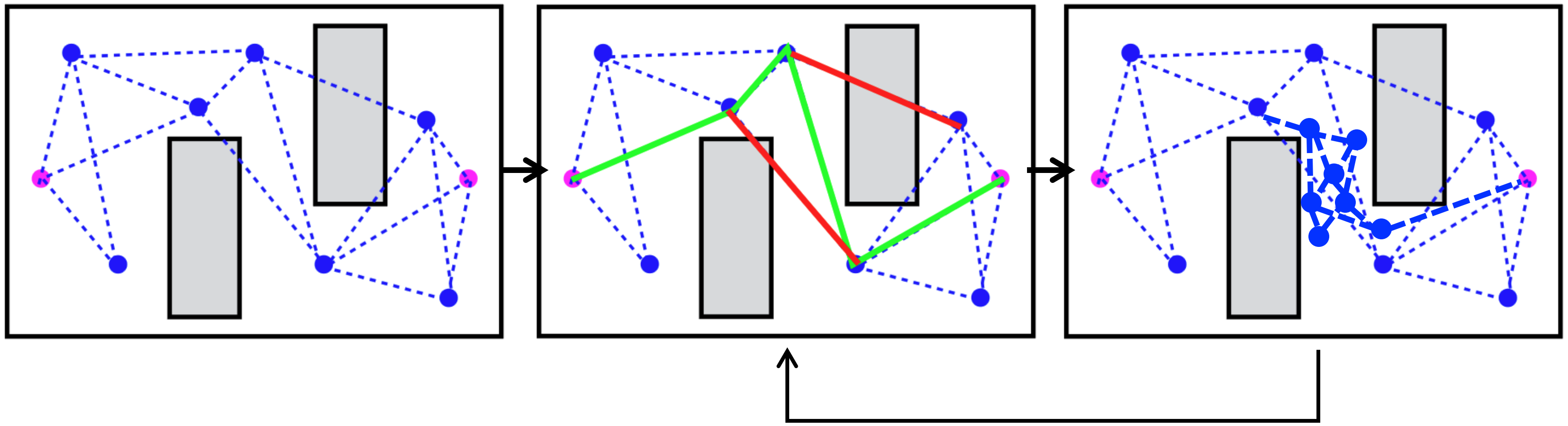


Motion Planning Computes a Sparse Path

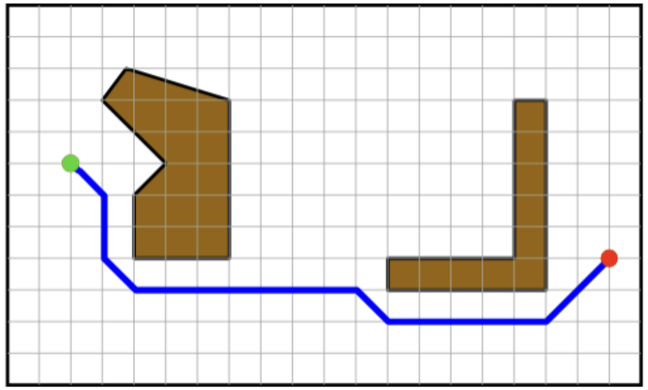
Create a graph

Search the graph

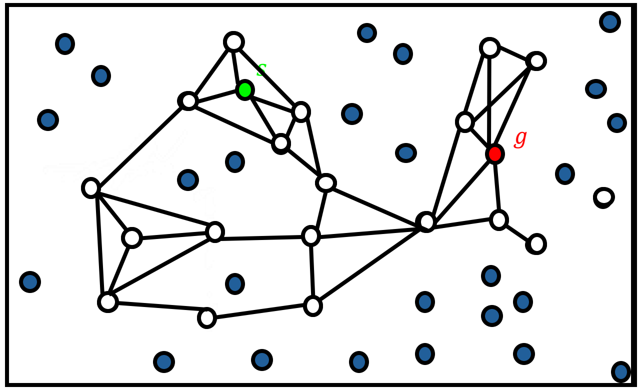
Densify the graph



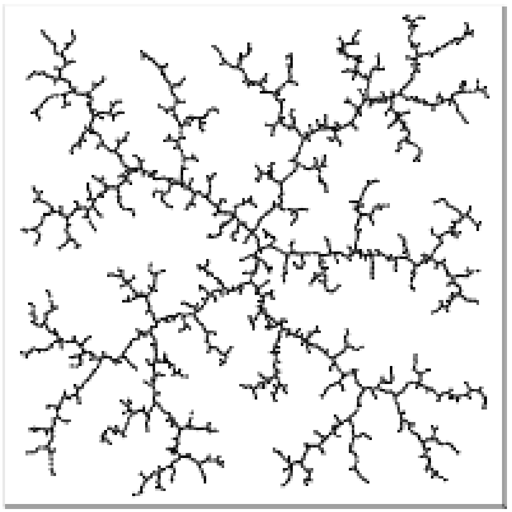
Roadmap planner



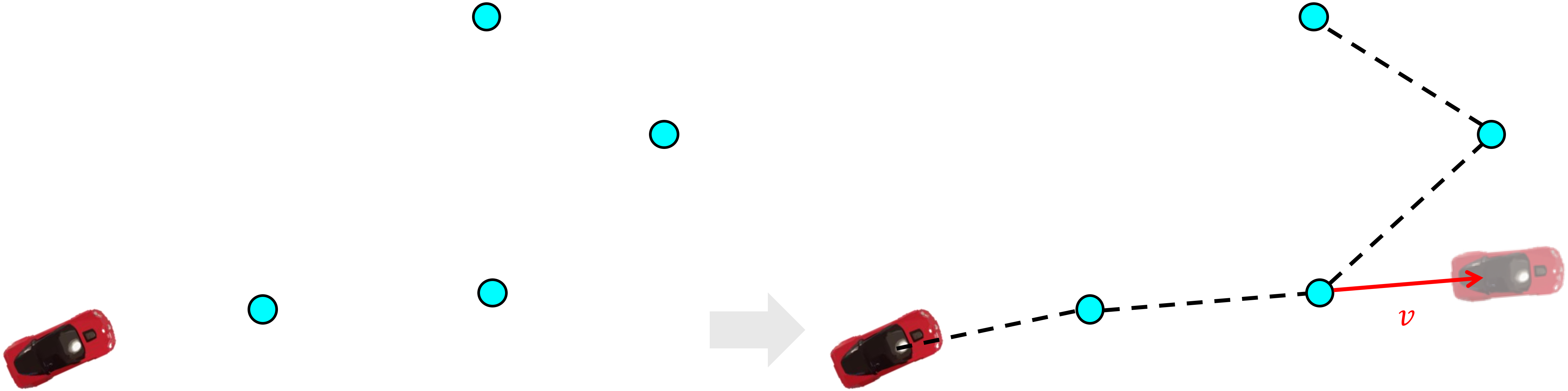
Probabilistic Roadmaps

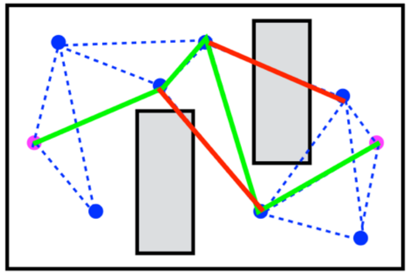


RRT



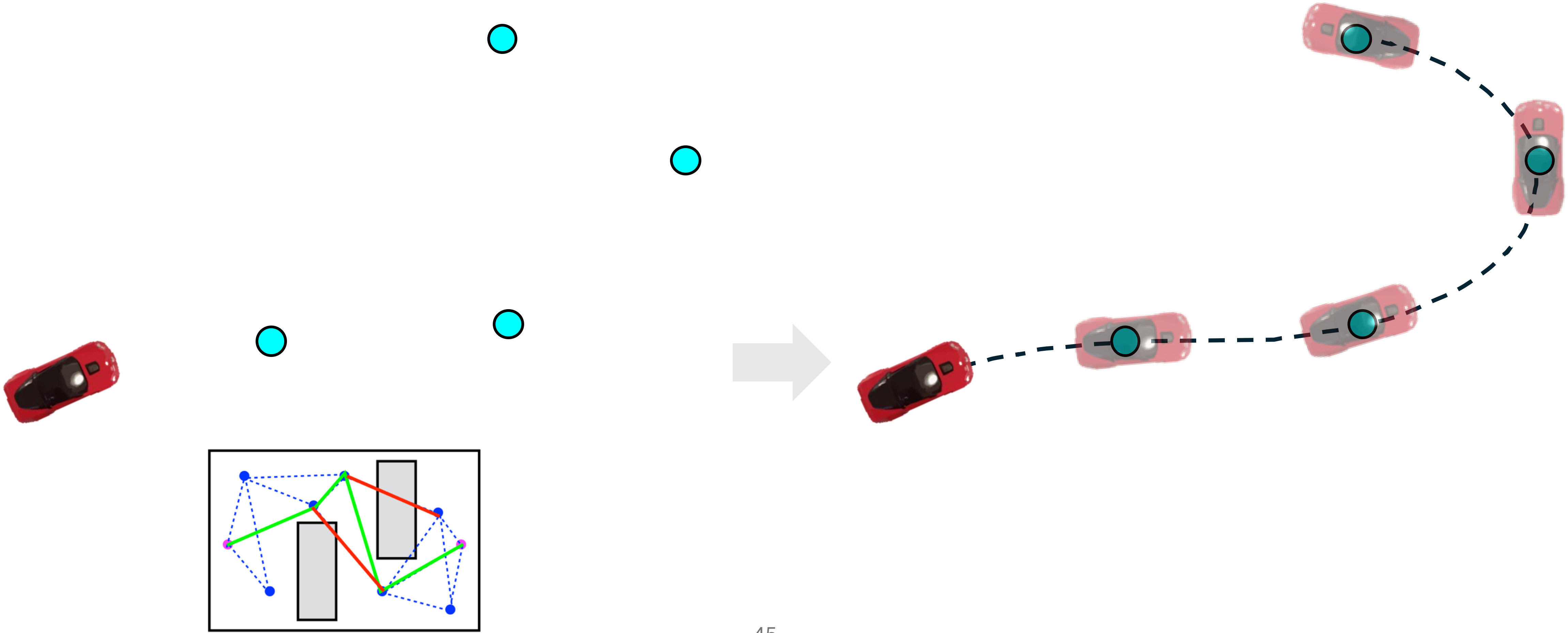
Should We Traverse the Path in Straight Lines?



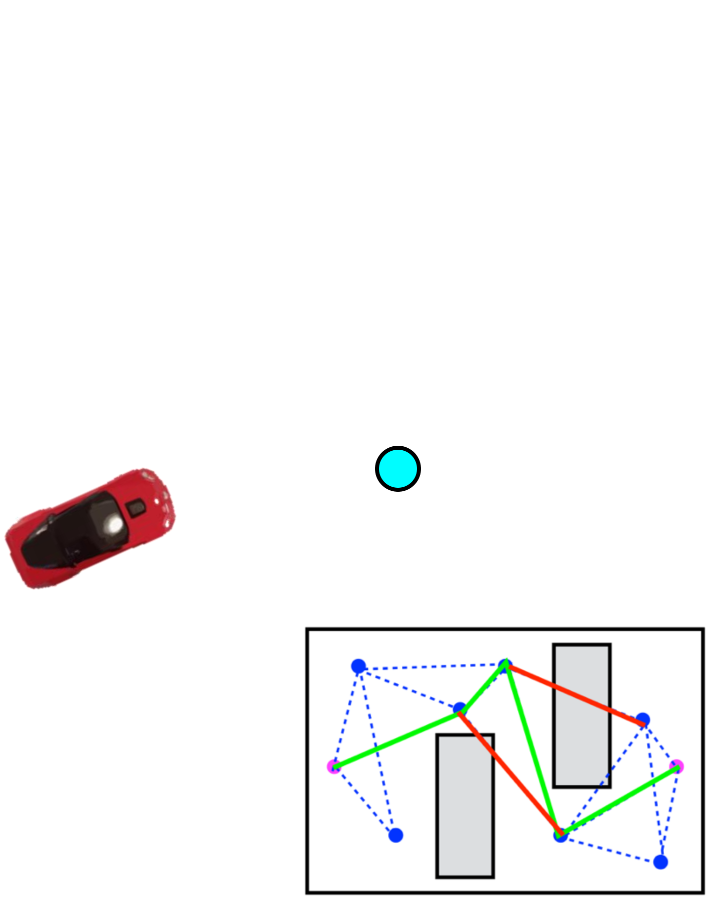


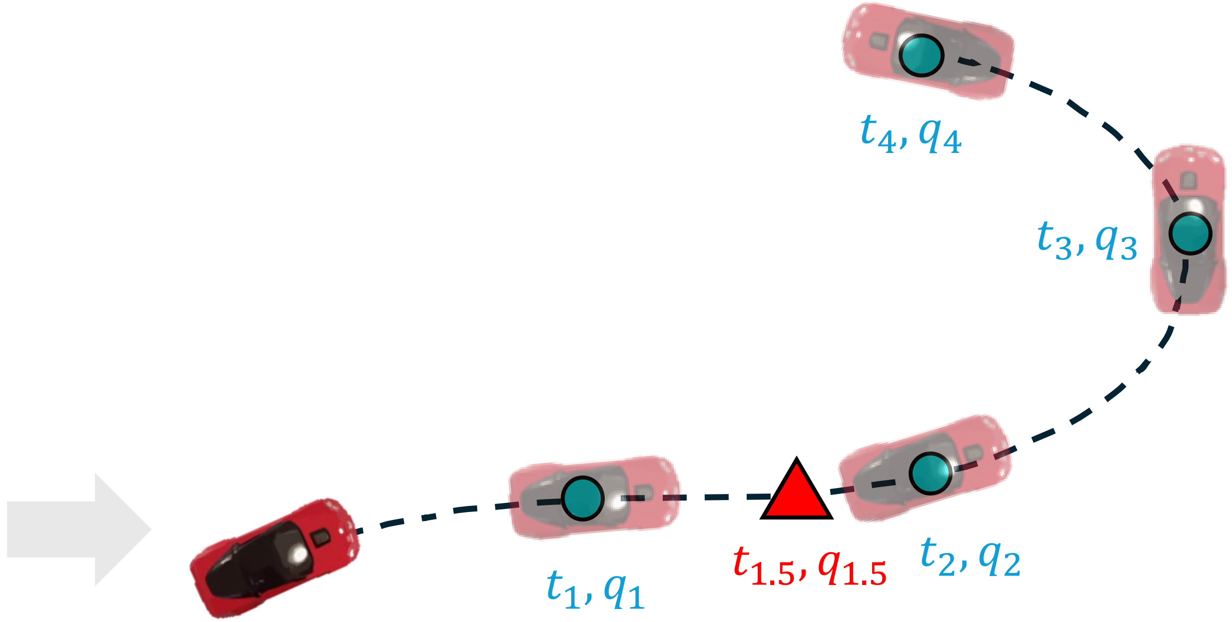
Straight lines are discontinuous,
robots could overshoot…

Traverse in Smooth Curves



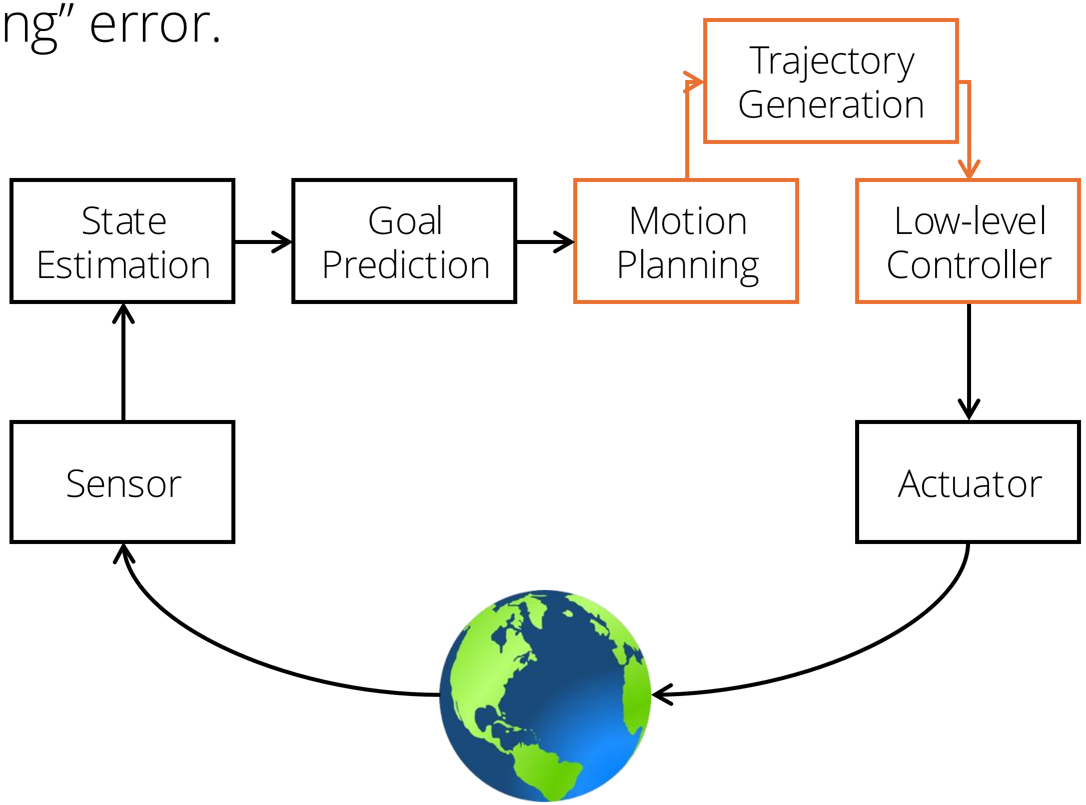
Traverse in Smooth Trajectories





Basic Recipe of Planning and Control

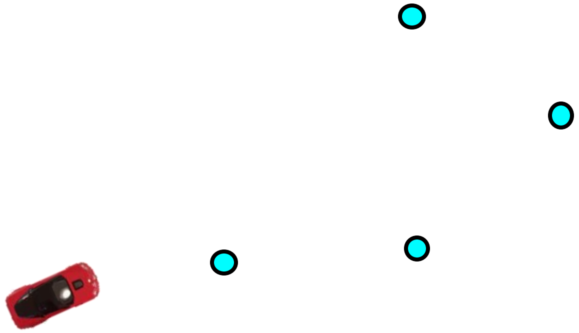
1. Motion planning: compute a feasible path
   - grid search / PRM / RRT …
2. **Trajectory generalization: time-scale the path into a trajectory**
3. Controller: predict a control to follow the path or minimize the "tracking" error.

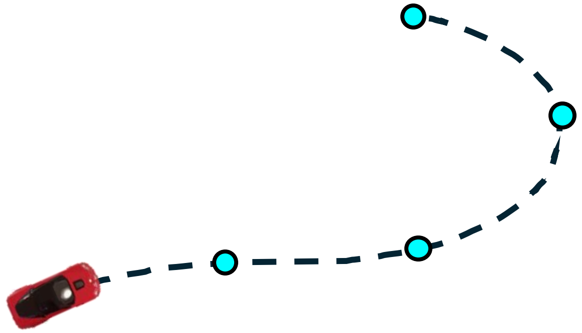


Trajectory Generation

- Constraints as continuity, smoothness, velocity/acceleration limit, kinematic,
- optimality criteria (min transfer time / energy...)

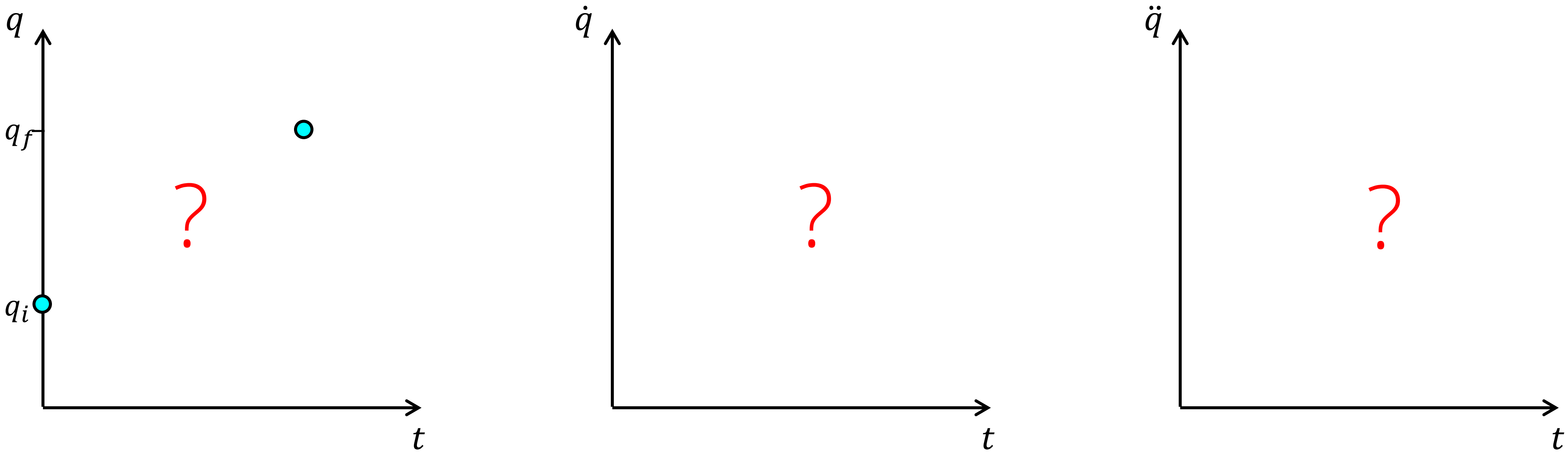
Path $q_0, q_1, \dots, q_N$ → **Trajectory Generator** → Trajectory $q(t)$





Point-to-Point Trajectory Generation

- Let's start with an easy example: point-to-point trajectory generation

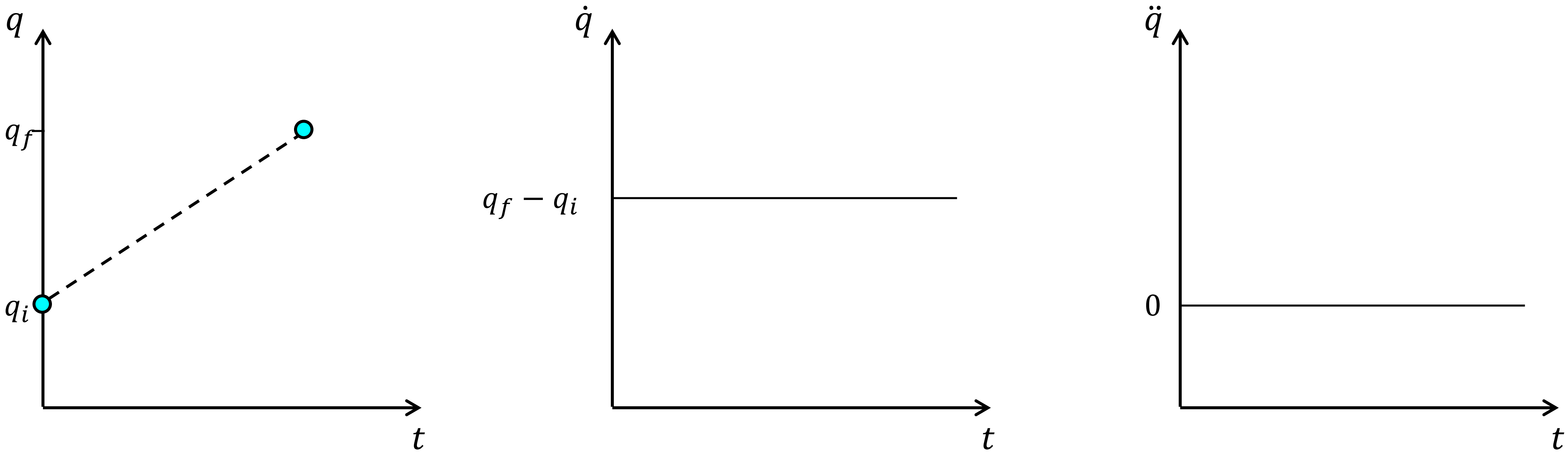


Point-to-Point Trajectory Generation with Straight Lines

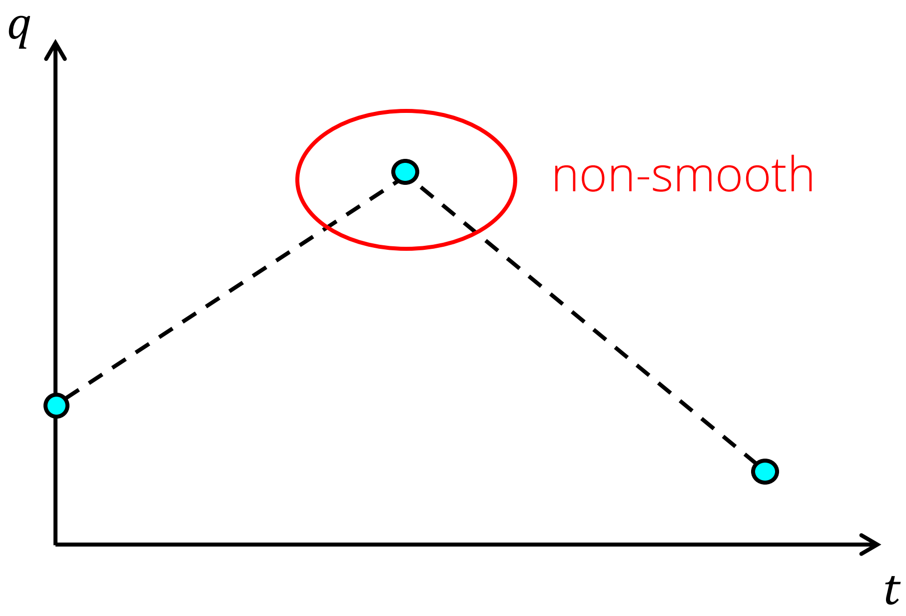
- Straight-line paths:

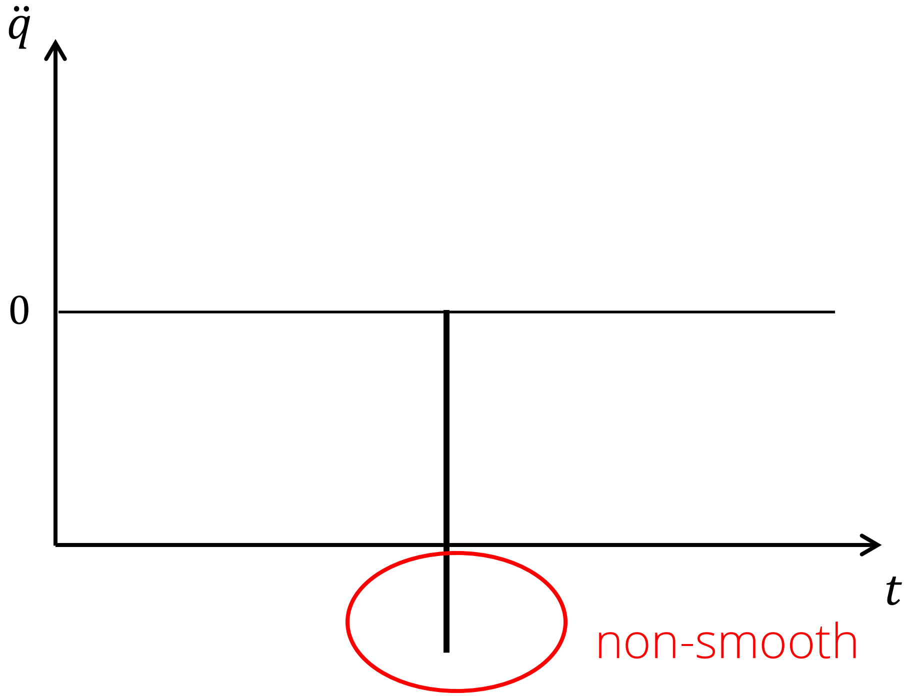
$$q(t) = q_i + t\left(q_f - q_i\right)$$

$$\dot{q}(t) = \frac{q_f - q_i}{t} \text{ and } \ddot{q}(t) = 0$$



Point-to-Point Trajectory Generation with Straight Lines





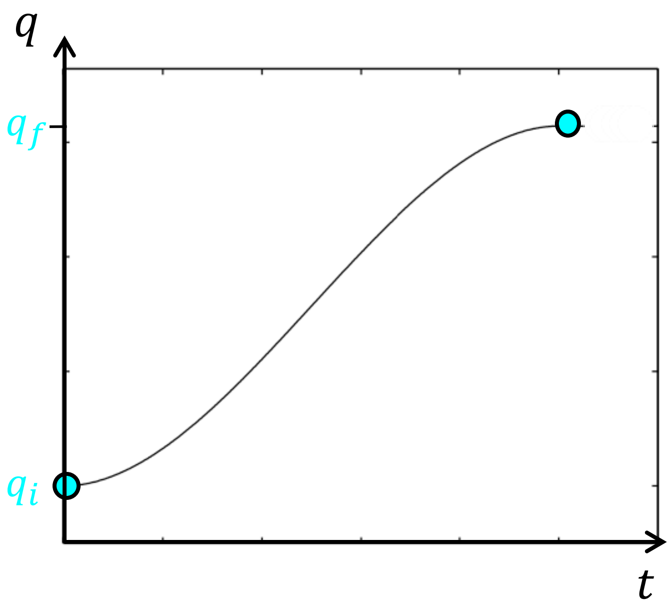
Point-to-Point Trajectory Generation with Cubic Polynomials

- Cubic polynomial paths:

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

$$\dot{q}(t) = 3a_3 t^2 + 2a_2 t + a_1$$

$$\ddot{q}(t) = 6a_3 t + 2a_2$$



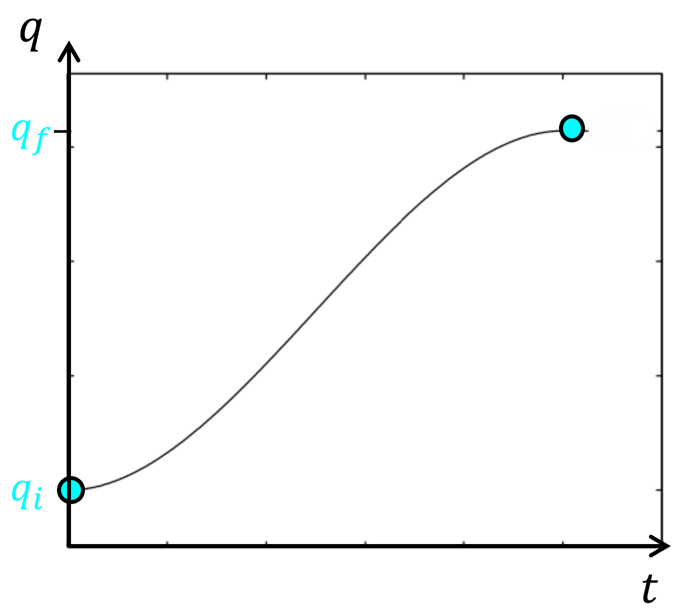
Cubic Polynomial Paths

- Equations:

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

$$\dot{q}(t) = 3a_3 t^2 + 2a_2 t + a_1$$

$$\ddot{q}(t) = 6a_3 t + 2a_2$$



Solve the equation

- when $t = 0$

$$a_3 0^3 + a_2 0^2 + a_1 0 + a_0 = q_i \Rightarrow a_0 = q_i$$

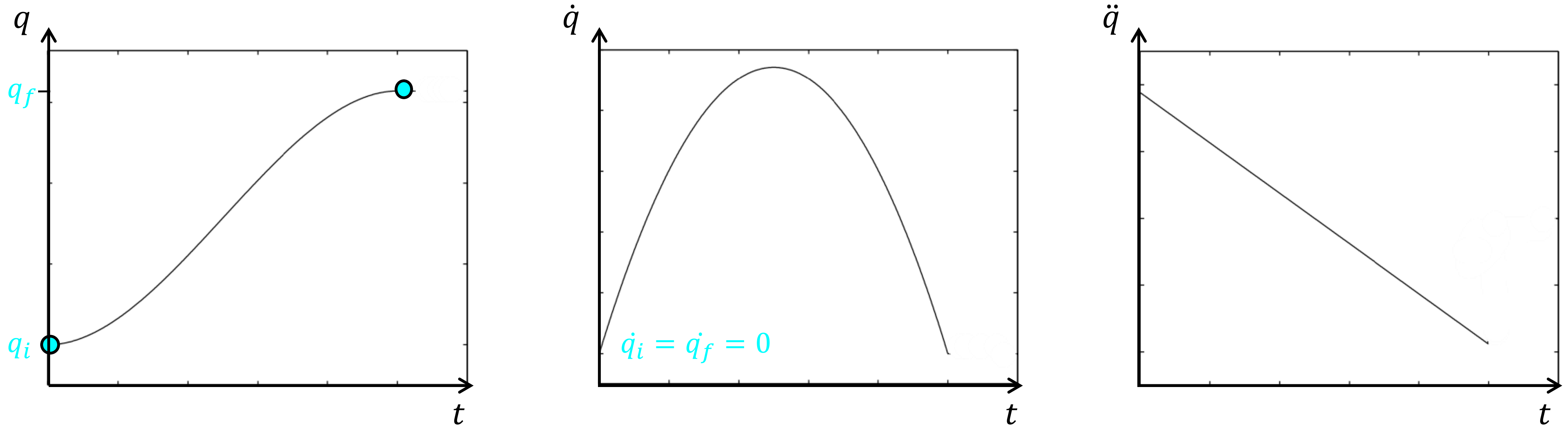
$$3a_3 0^2 + 2a_2 0 + a_1 = \dot{q}_i \Rightarrow a_1 = \dot{q}_i$$

- when $t = t_f$

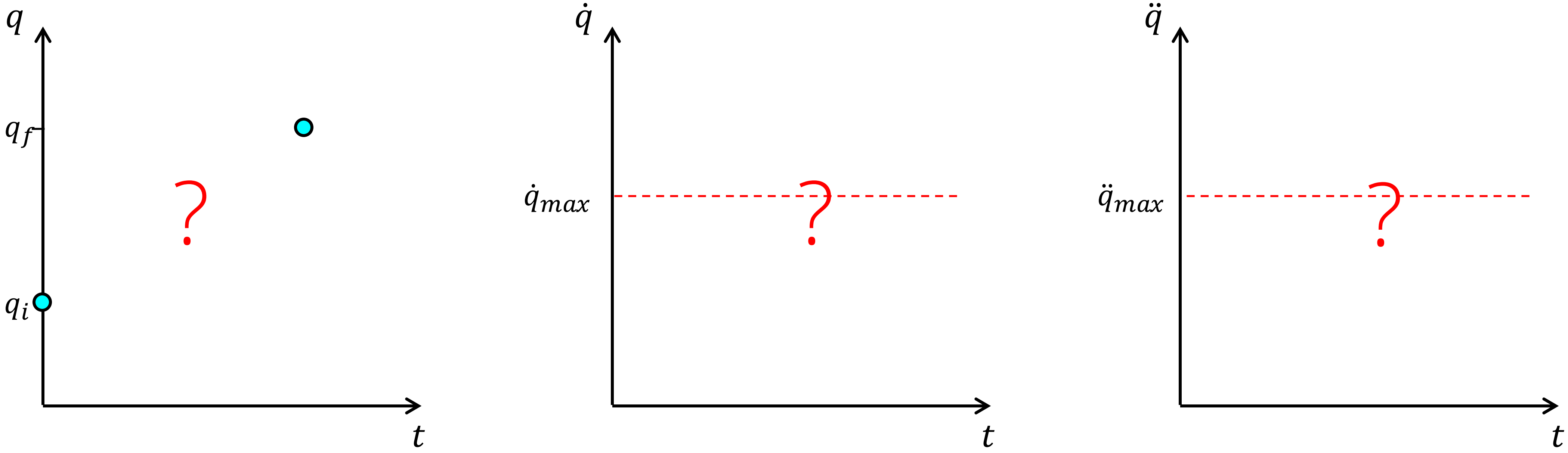
$$a_3 {t_f}^3 + a_2 {t_f}^2 + a_1 t_f + a_0 = q_f$$

$$3a_3 {t_f}^2 + 2a_2 t_f + a_1 = \dot{q}_f$$

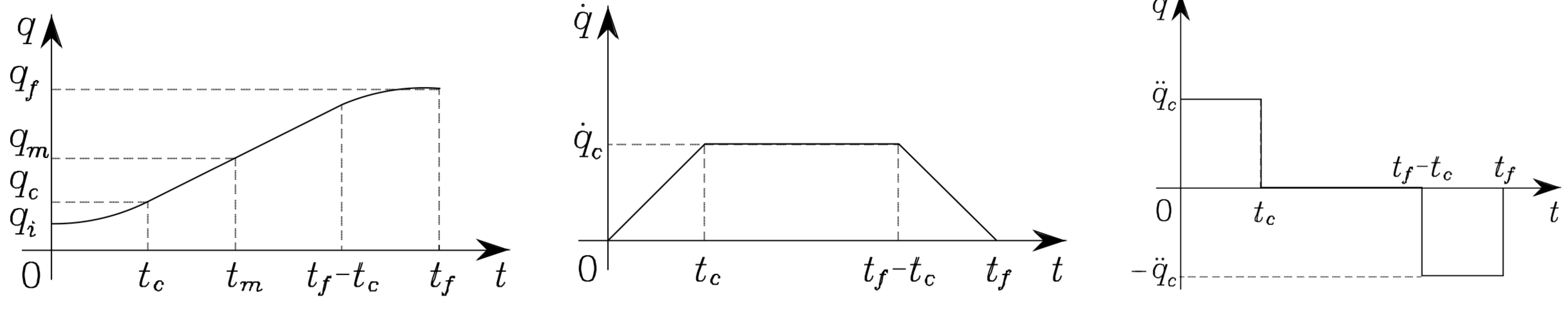
Point-to-Point Trajectory Generation with Cubic Polynomials



What if We Want to Impose Velocity/Acceleration Constraints?



Point-to-Point Trajectory Generation with Trapezoidal Motion Profiles

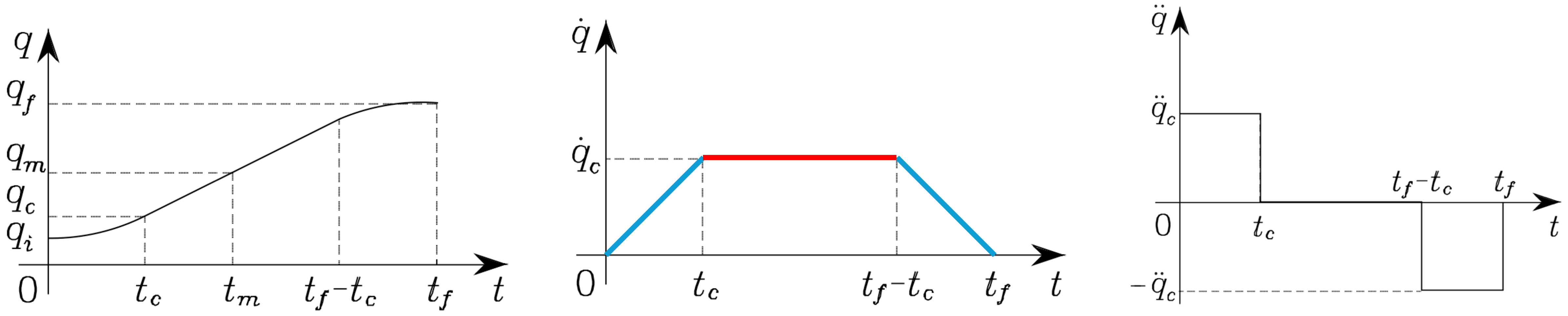


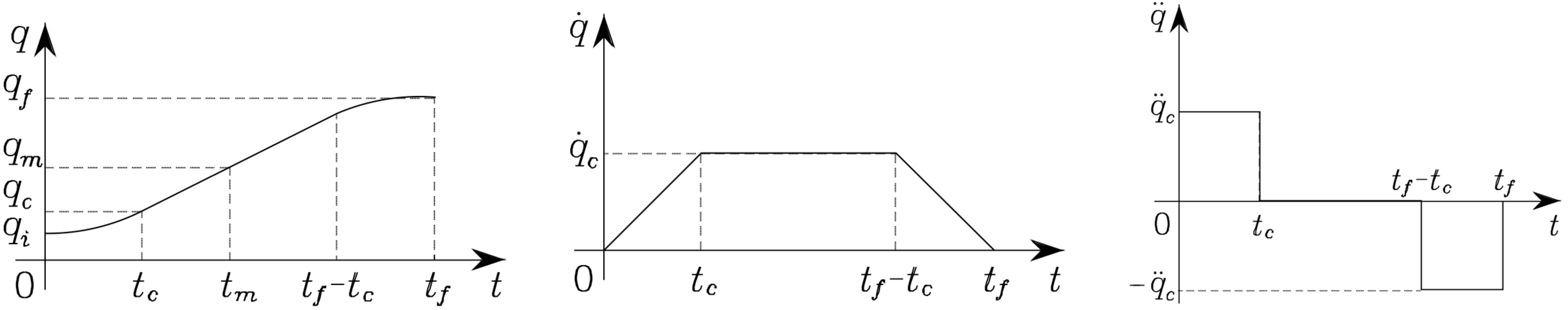
Robotics: Modeling, Planning and Control. Siciliano et al.

Point-to-Point Trajectory Generation with Trapezoidal Motion Profiles

Also known as Linear Segments with Parabolic Blends:

1. Fit with quadratic polynomials (linear ramp velocity)
2. At the blend time, switch to a linear function (constant velocity)





- Average point: $q_m = (q_f + q_i)/2$ at $t_m = t_f/2$

- Velocity:

$$\ddot{q}_c t_c = \frac{q_m - q_c}{t_m - t_c}$$

$$q_c = q_i + \frac{1}{2}\ddot{q}_c {t_c}^2$$

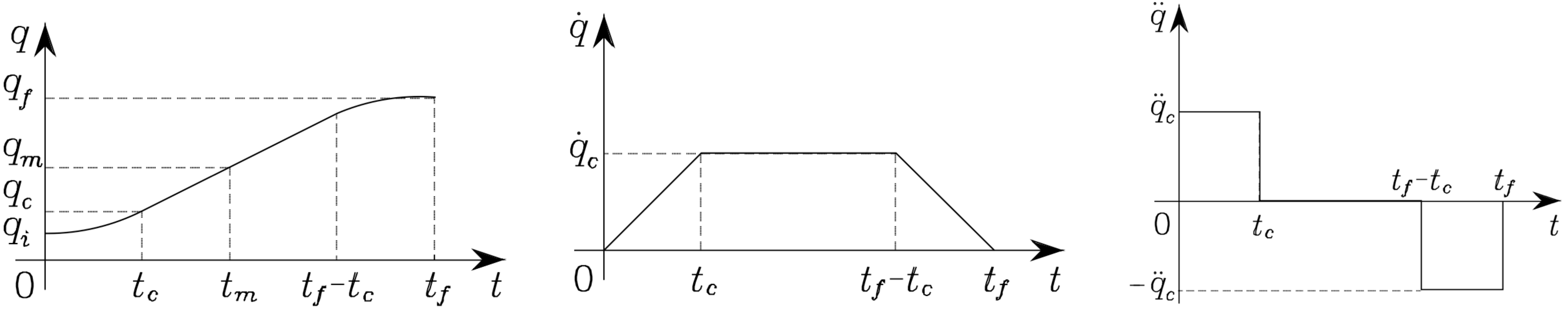
$$\ddot{q}_c {t_c}^2 - \ddot{q}_c t_f t_c + q_f - q_i = 0$$

We can specify $\ddot{q}_c$

$$t_c = \frac{t_f}{2} - \frac{1}{2}\sqrt{\frac{t_f^2 \ddot{q}_c - 4(q_f - q_i)}{\ddot{q}_c}}.$$

Constraints

$$\boxed{|\ddot{q}_c| \geq \frac{4|q_f - q_i|}{t_f^2}}$$



- Average point: $q_m = (q_f + q_i)/2$ at $t_m = t_f/2$

- Velocity:

$$\ddot{q}_c t_c = \frac{q_m - q_c}{t_m - t_c}$$

$$q_c = q_i + \frac{1}{2}\ddot{q}_c {t_c}^2$$

$$\ddot{q}_c {t_c}^2 - \ddot{q}_c t_f t_c + q_f - q_i = 0$$

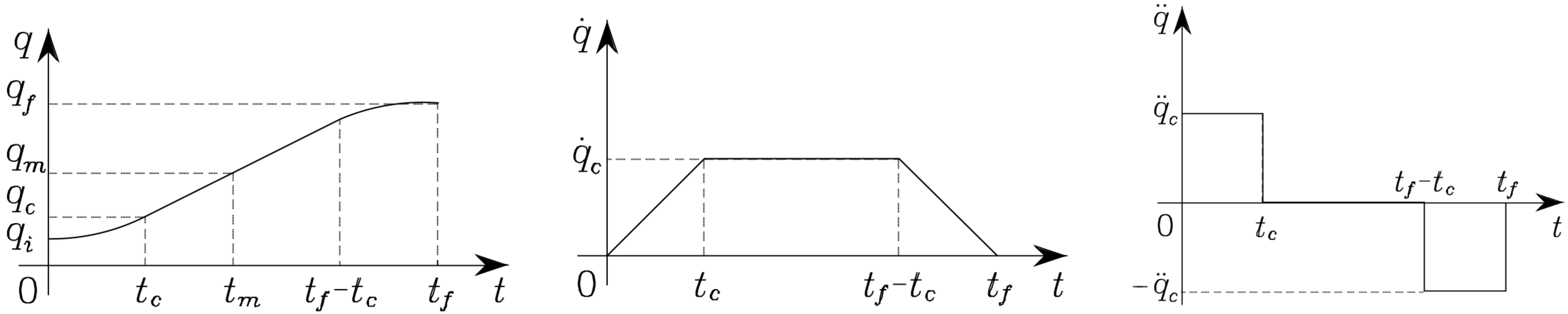
We can specify $\dot{q}_c$

$$t_c = \frac{q_i - q_f + \dot{q}_c t_f}{\dot{q}_c}$$

Constraints

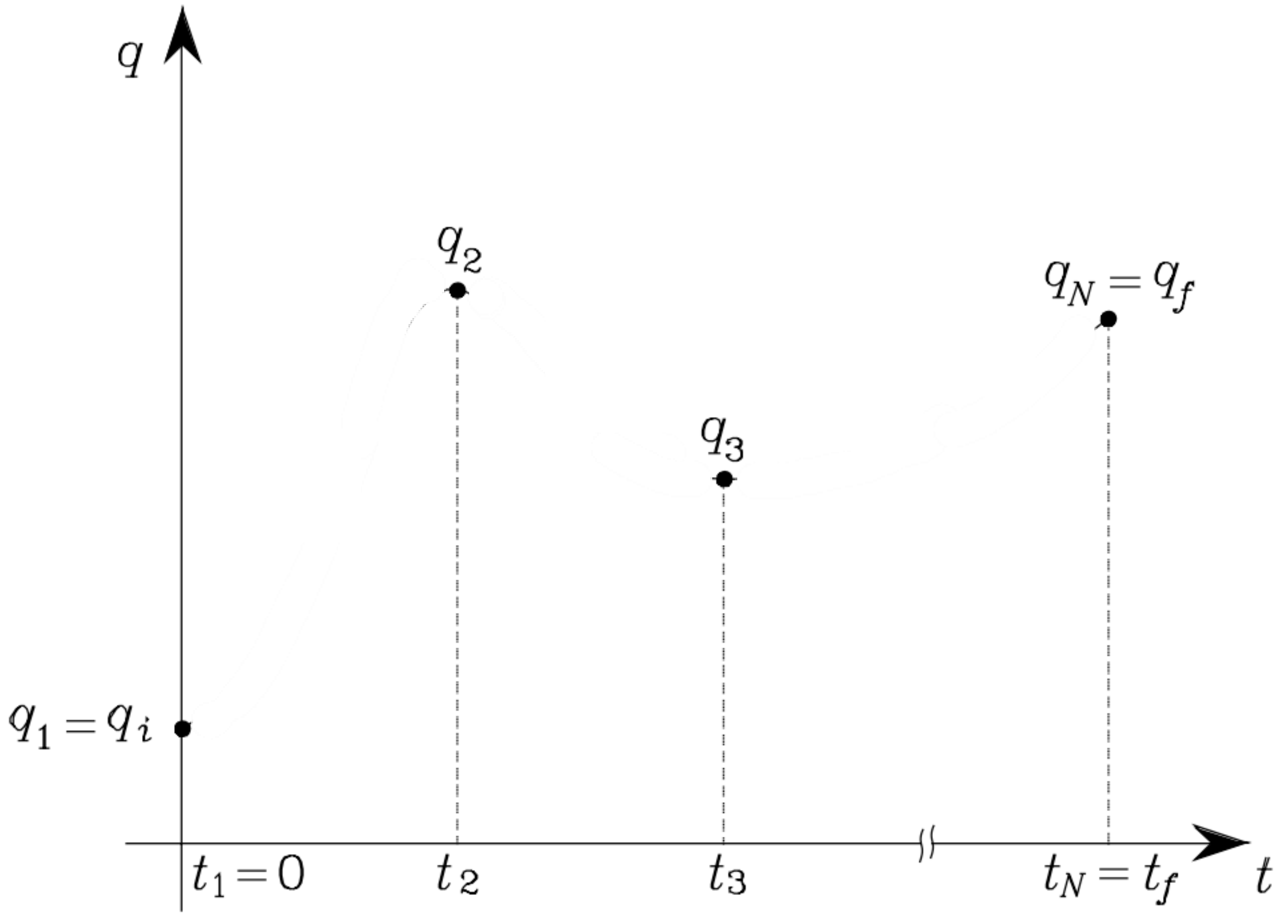
$$\boxed{\frac{|q_f - q_i|}{t_f} < |\dot{q}_c| \leq \frac{2|q_f - q_i|}{t_f}}$$

Point-to-Point Trajectory Generation with Trapezoidal Motion Profiles



$$q(t) = \begin{cases} q_i + \frac{1}{2}\ddot{q}_c t^2 & 0 \le t \le t_c \\ q_i + \ddot{q}_c t_c (t - t_c/2) & t_c < t \le t_f - t_c \\ q_f - \frac{1}{2}\ddot{q}_c (t_f - t)^2 & t_f - t_c < t \le t_f . \end{cases}$$

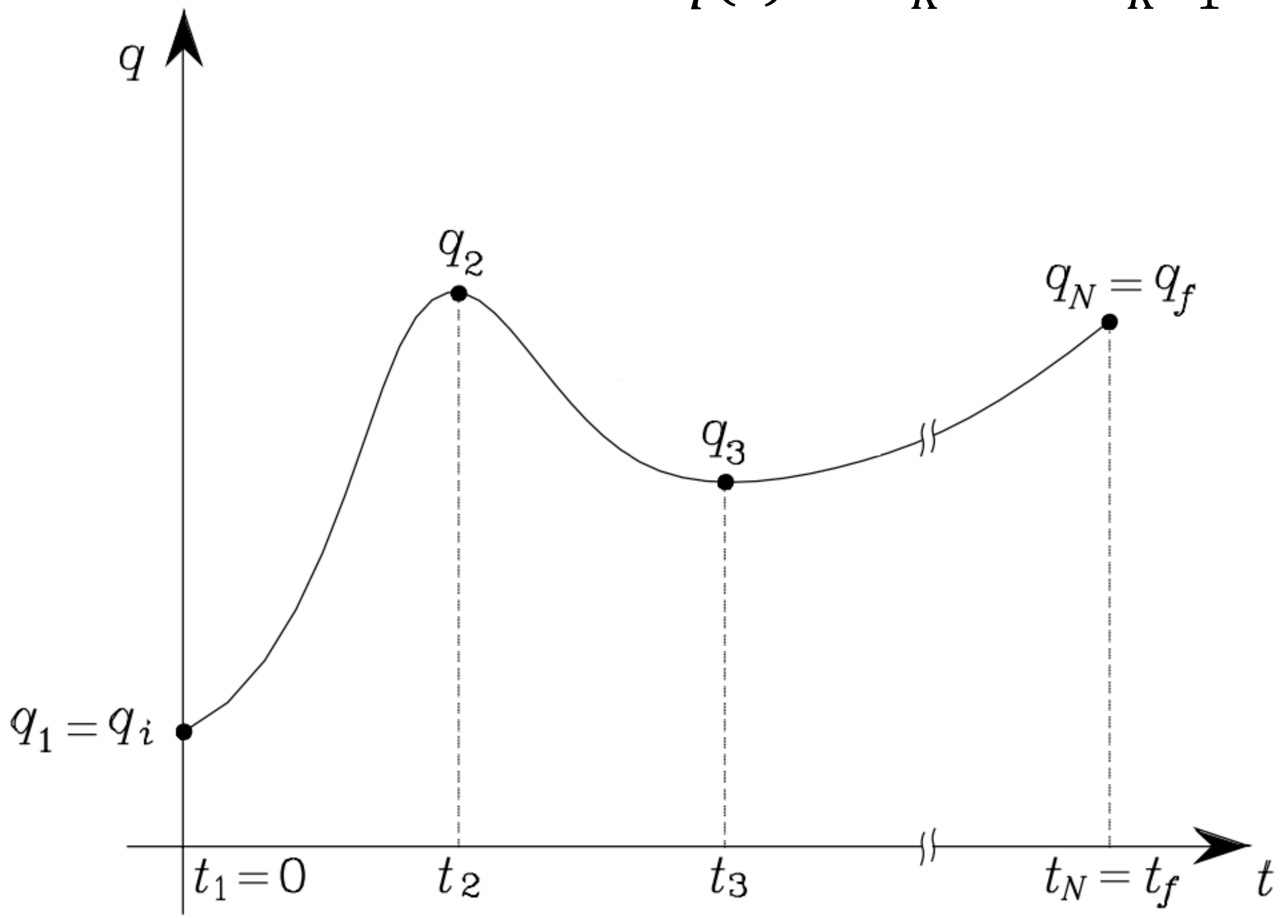
What if We Want to Generate Trajectory from Multiple Waypoints?



Robotics: Modeling, Planning and Control. Siciliano et al.

Can We Fit a High-Order Polynomials?

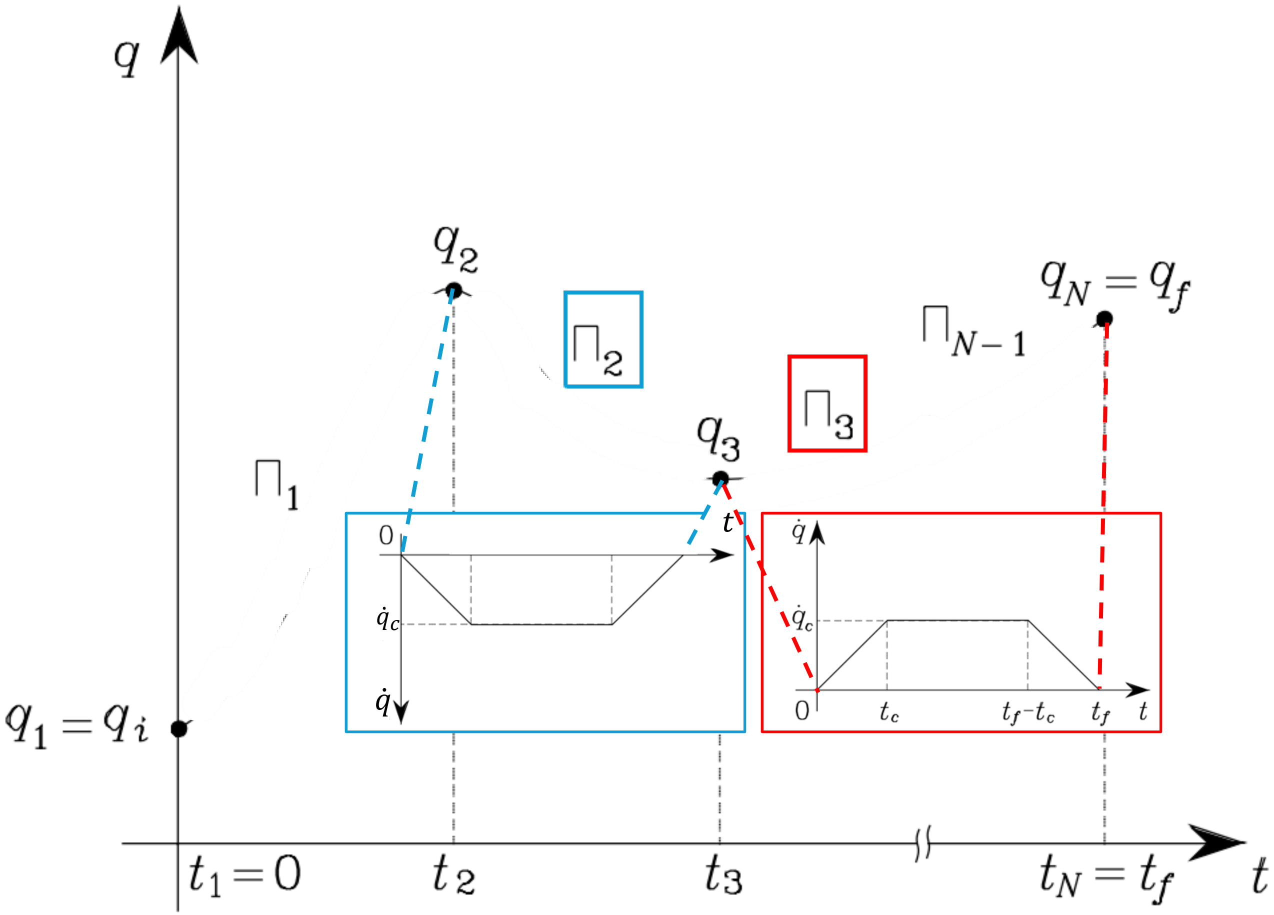
$$q(t) = a_K t^K + a_{K-1} t^{K-1} + \cdots + a_1 t + a_0$$



Drawbacks:

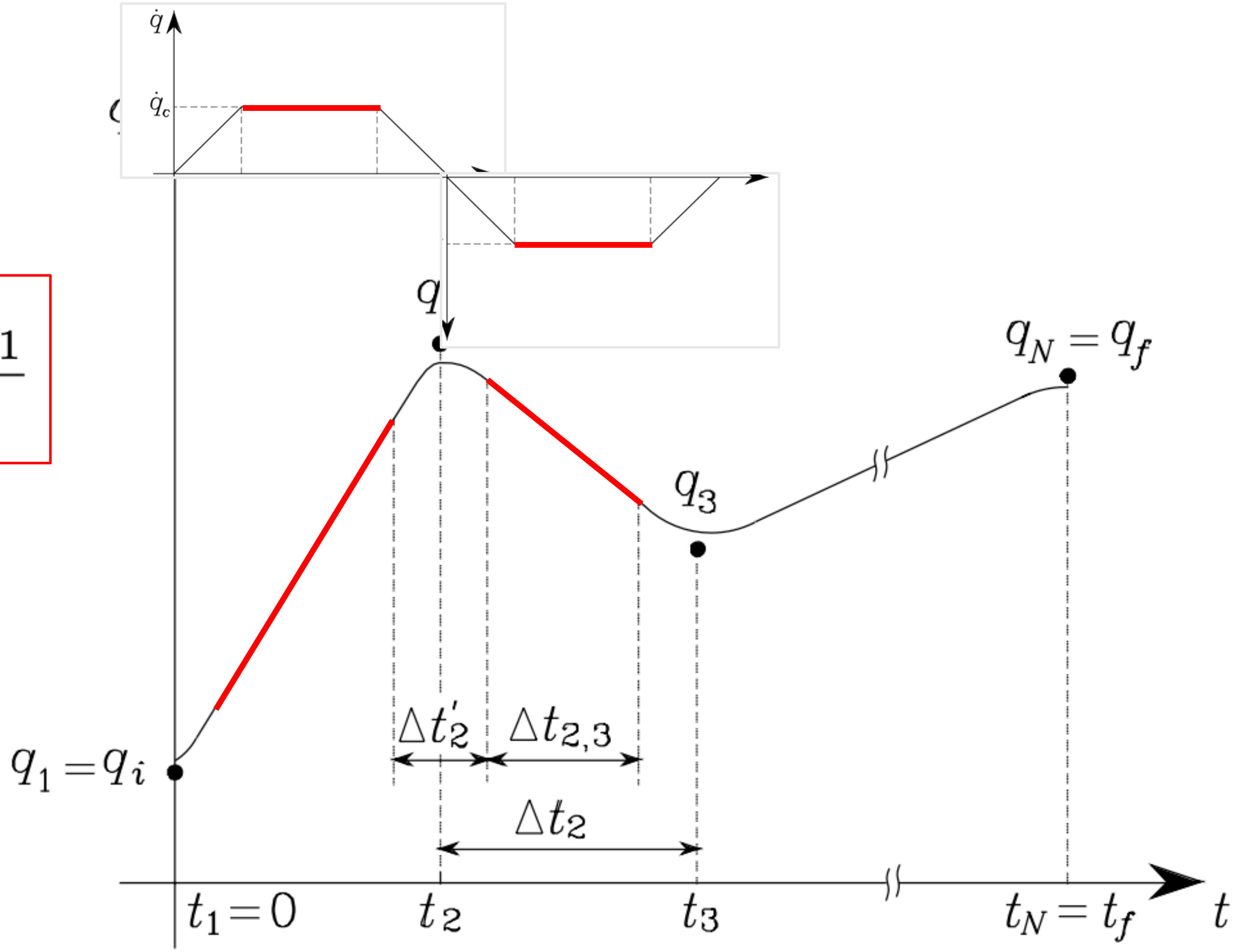
- The polynomial function overfits, resulting in oscillatory curves
- The system of constraint equations is heavy to solve

Can We Fit with Multiple Trapezoidal Motion Profiles?



Can We Fit with Multiple Trapezoidal Motion Profiles?

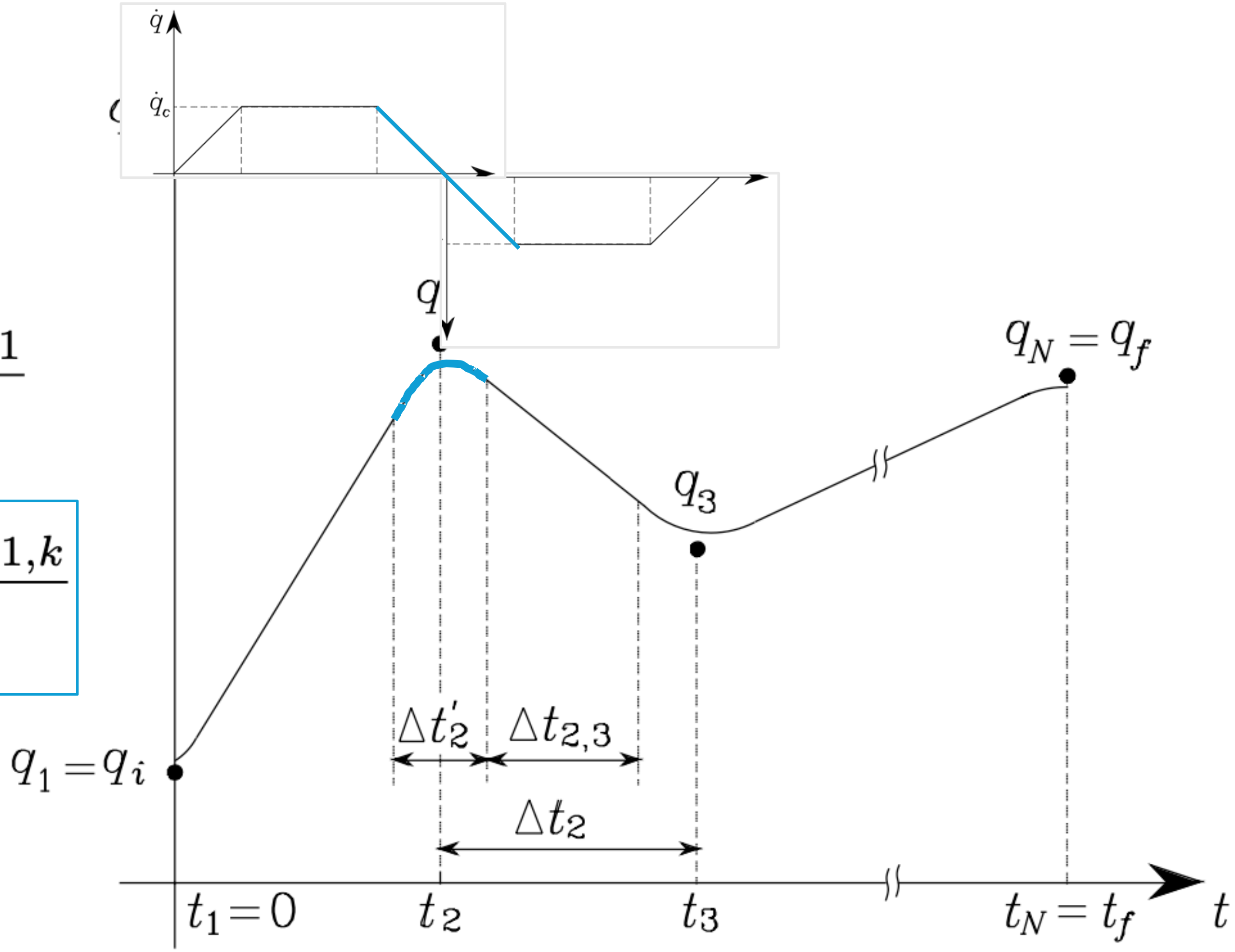
$$\dot{q}_{k-1,k} = \frac{q_k - q_{k-1}}{\Delta t_{k-1}}$$



Can We Fit with Multiple Trapezoidal Motion Profiles?

$$\dot{q}_{k-1,k} = \frac{q_k - q_{k-1}}{\Delta t_{k-1}}$$

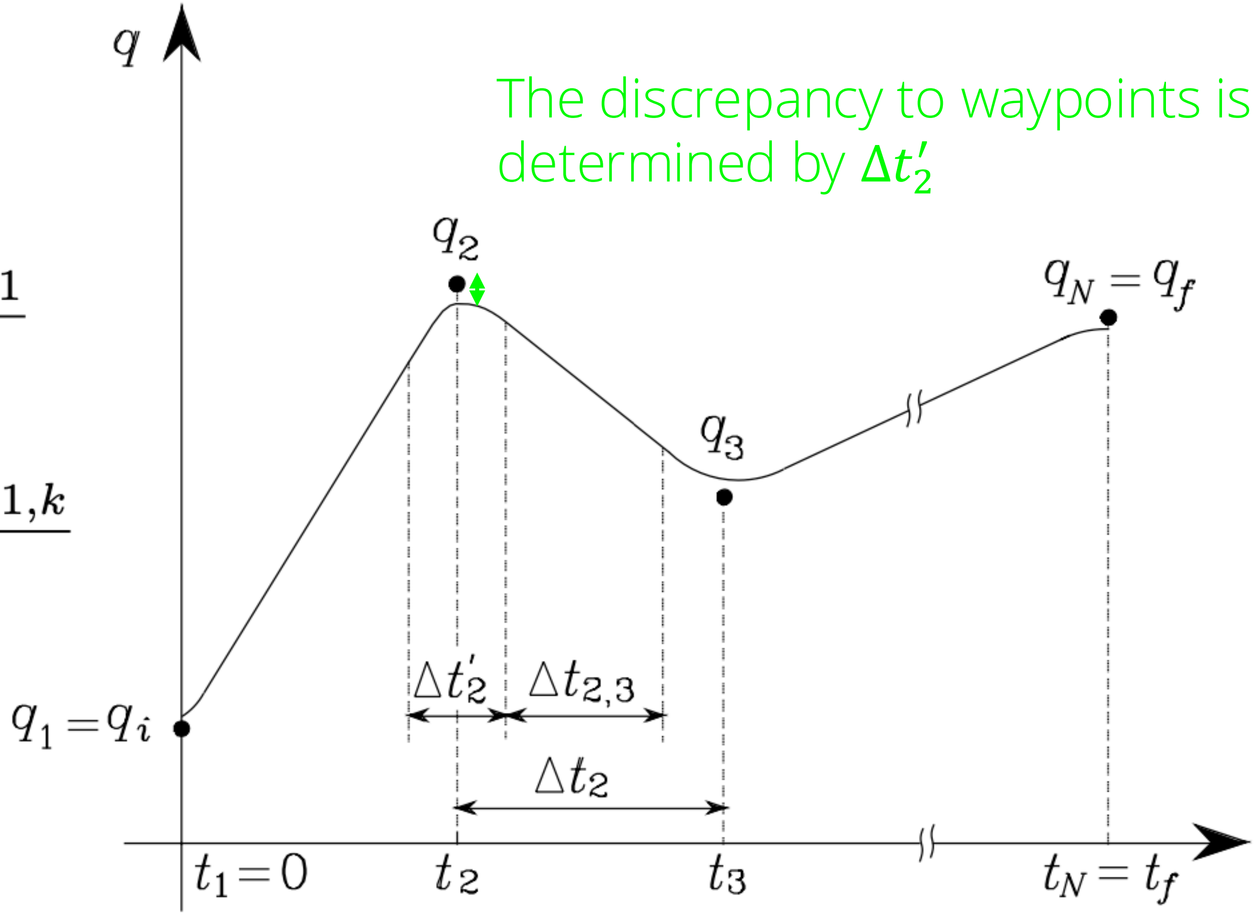
$$\boxed{\ddot{q}_k = \frac{\dot{q}_{k,k+1} - \dot{q}_{k-1,k}}{\Delta t'_k}}$$



Can We Fit with Multiple Trapezoidal Motion Profiles?

$$\dot{q}_{k-1,k} = \frac{q_k - q_{k-1}}{\Delta t_{k-1}}$$

$$\ddot{q}_k = \frac{\dot{q}_{k,k+1} - \dot{q}_{k-1,k}}{\Delta t'_k}$$

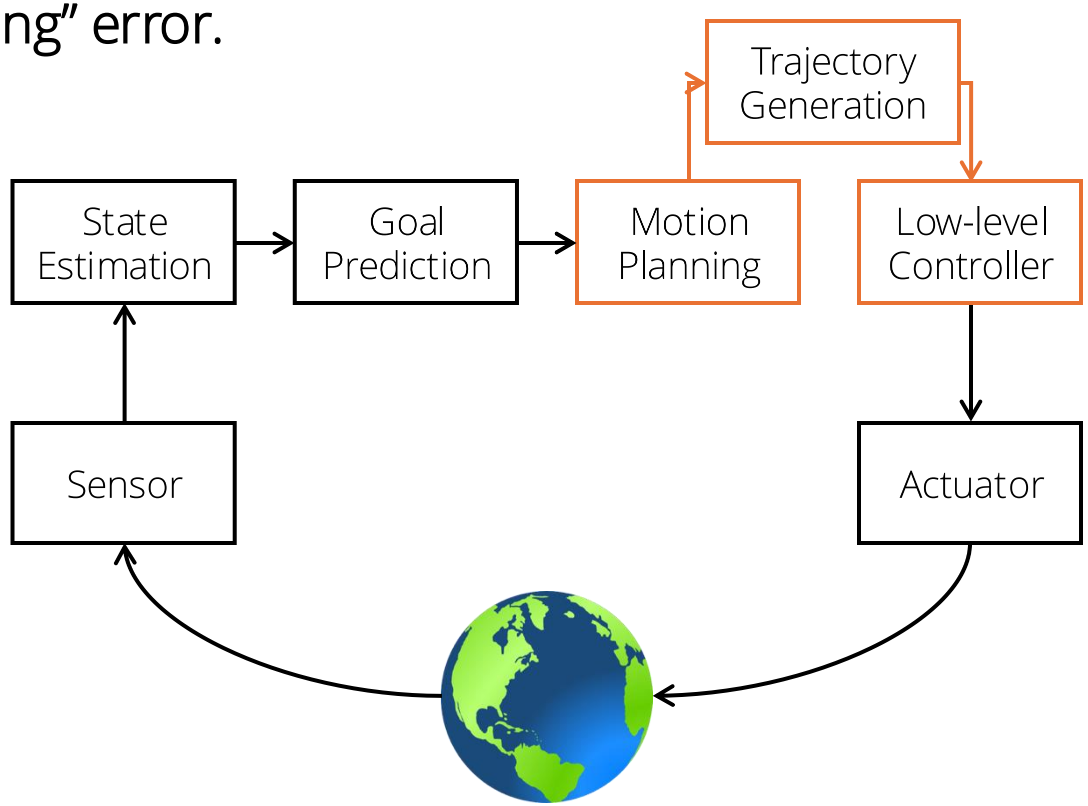


The discrepancy to waypoints is determined by $\Delta t'_2$

See "Turning paths into Trajectories Using Parabolic Blends" by T. Kunz and M. Stilman for more details

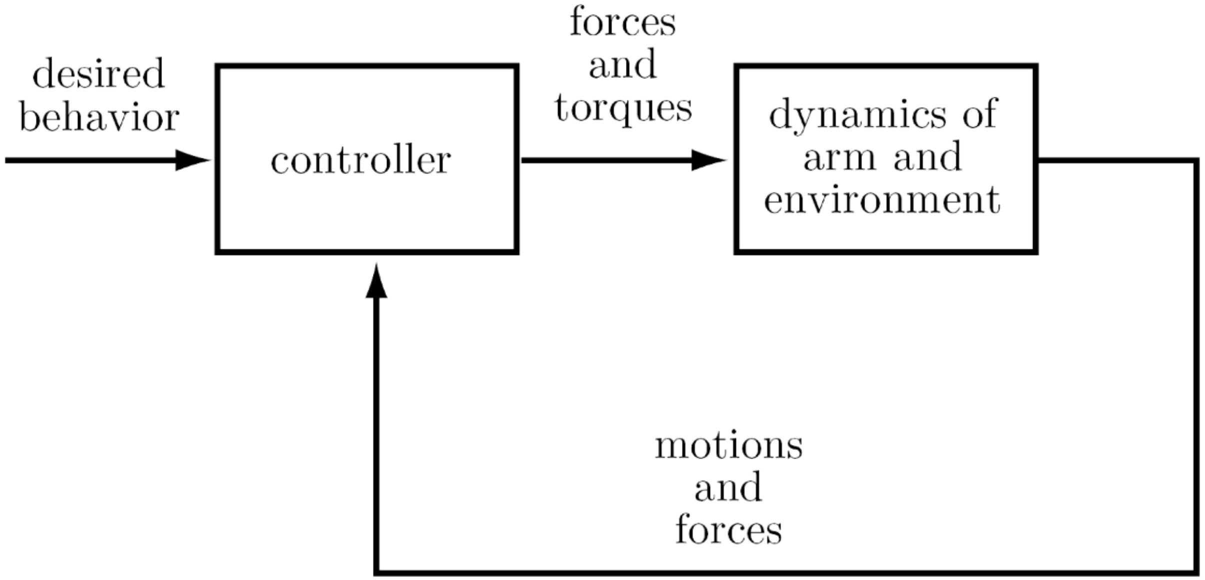
Basic Recipe of Planning and Control

1. Motion planning: compute a feasible path
   - grid search / PRM / RRT …
2. Trajectory generalization: time-scale the path into a trajectory
3. Controller: predict a control to follow the path or minimize the ”tracking” error.



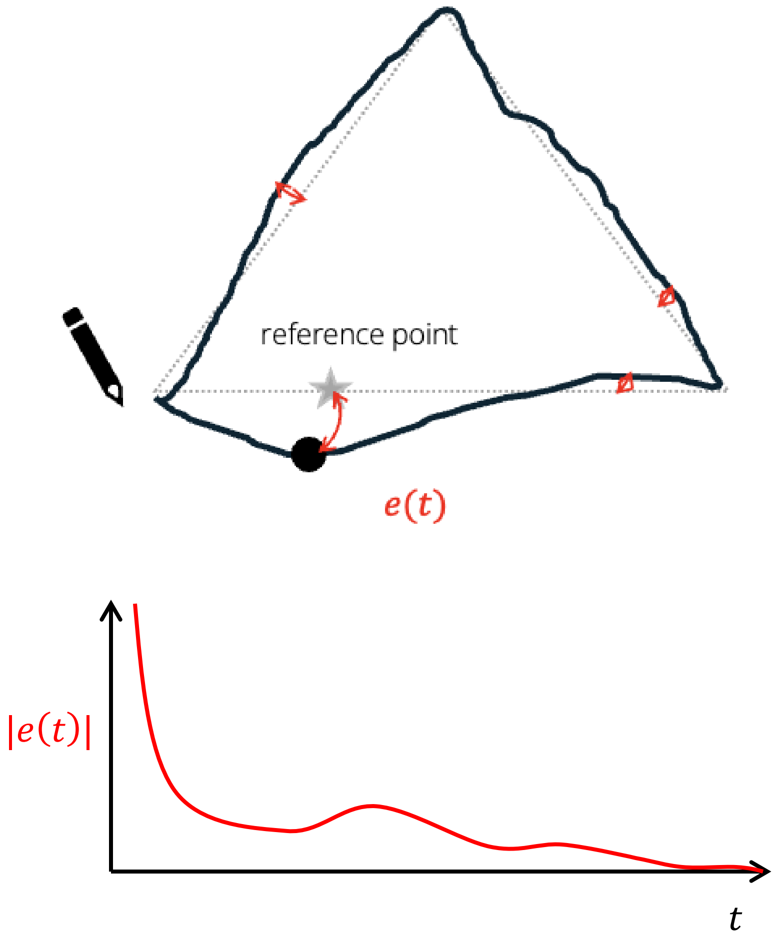
Control Systems

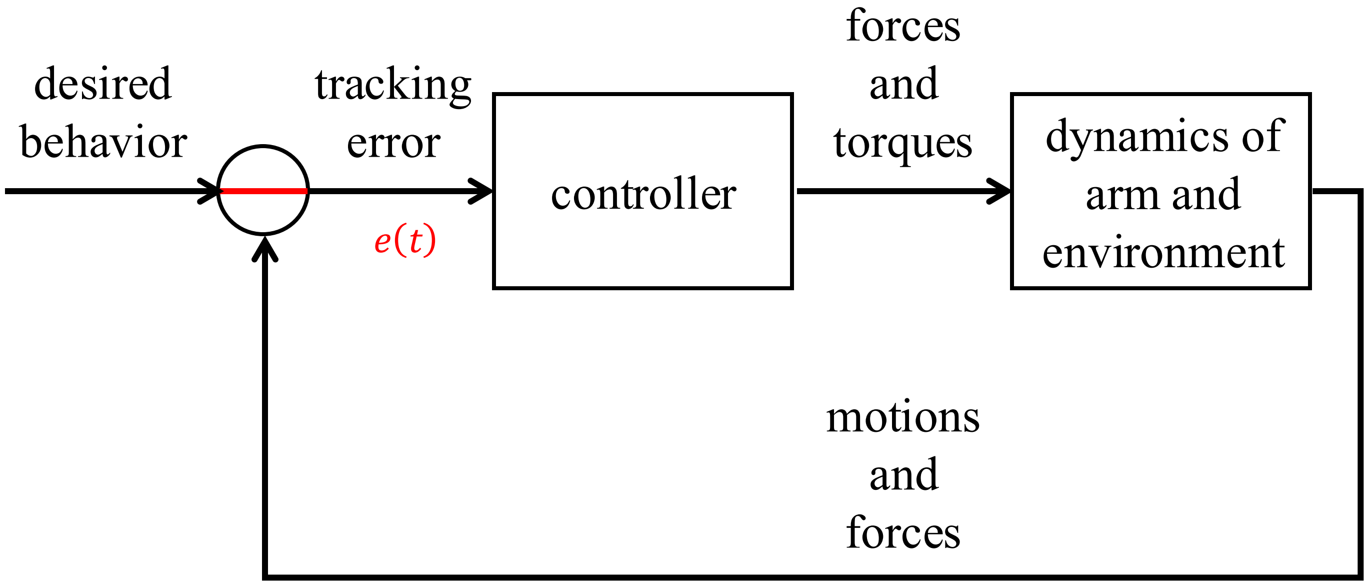
1. Controller: predict a control to follow the path / desired behavior
    - Feedback control
    - Optimal control
    - …
2. Control $u(t)$ can be velocity / torque inputs



Modern Robotics. K. Lynch and F. Park

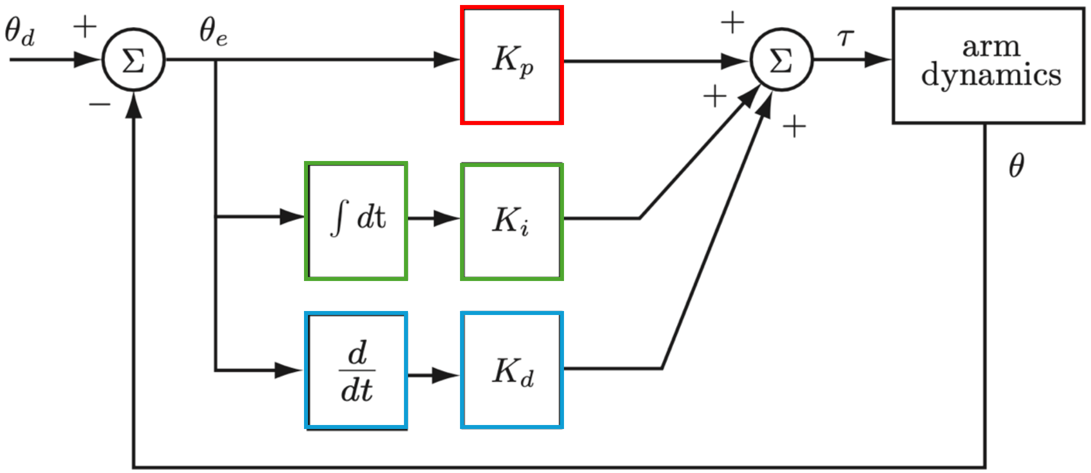
Feedback Controller: Continuously Adjust the Control Input to Minimize the Tracking Error





An Overview of PID Controller

Image generated by Gemini



- Current temperature difference (immediate error)
- Future temperature difference (derivatives of errors)
- Accumulated temperature difference (accumulated error)

Linear Error Dynamics

- Let's define $q_e(t) = q_d(t) - q(t)$ as the tracking error, where $q_d(t)$ is the desired behavior and $q(t)$ is the current behavior

- The purpose of the feedback controller is to create an error dynamics such that $q_e(t)$ tends to a small value, as t increases

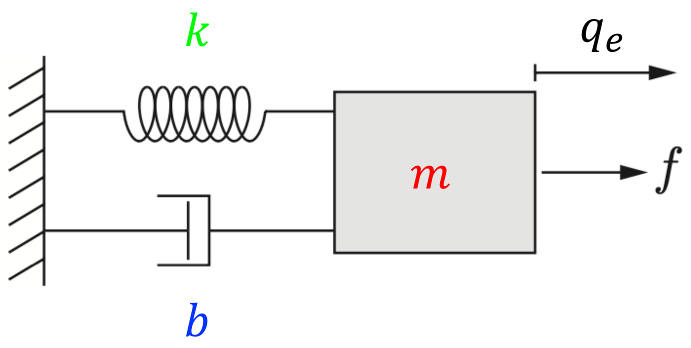
Linear Error Dynamics

- Linear error dynamics:

Control signal

$$a_p {q_e}^{(p)}(t) + a_{p-1} {q_e}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t) = \boxed{c}$$

- An example of a 2nd-order error dynamics: the linear mass-spring-damper



$$m\ddot{q}_e(t) + b\dot{q}_e(t) + kq_e(t) = f$$

- If $c = 0$:

$$a_p q_e{}^{(p)}(t) + a_{p-1} q_e{}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t) = 0$$
$$\Rightarrow q_e{}^{(p)}(t) = -\frac{1}{a_p}\left[a_{p-1} q_e{}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t)\right]$$
$$\Rightarrow q_e{}^{(p)}(t) = -a_{p-1}{}' q_e{}^{(p-1)}(t) - \cdots - a_2{}' \ddot{q}_e(t) - a_1{}' \dot{q}_e(t) - a_0{}' q_e(t)$$

- Express a pth-order differential equation as p coupled 1st-order differential equations:

$$x_1 = q_e$$
$$x_2 = \dot{x}_1 = \dot{q}_e$$
$$\vdots$$
$$x_p = \dot{x}_{p-1} = q_e{}^{(p-1)}$$

- If $c = 0$:

$$a_p {q_e}^{(p)}(t) + a_{p-1} {q_e}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t) = 0$$

$$\Rightarrow {q_e}^{(p)}(t) = -\frac{1}{a_p}\left[a_{p-1} {q_e}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t)\right]$$

$$\Rightarrow \boxed{{q_e}^{(p)}(t) = -{a_{p-1}}' {q_e}^{(p-1)}(t) - \cdots - {a_2}' \ddot{q}_e(t) - {a_1}' \dot{q}_e(t) - {a_0}' q_e(t)}$$

- Express a p^th^-order differential equation as p coupled 1^st^-order differential equations:

$$x_1 = q_e$$
$$x_2 = \dot{x}_1 = \dot{q}_e$$
$$\vdots$$
$$x_p = \dot{x}_{p-1} = {q_e}^{(p-1)}$$

$$\boxed{\dot{x}_p = -{a_{p-1}}' x_p - \cdots - {a_2}' x_3 - {a_1}' x_2 - {a_0}' x_1}$$

- If $c = 0$:

$$a_p {q_e}^{(p)}(t) + a_{p-1} {q_e}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t) = 0$$

$$\Rightarrow {q_e}^{(p)}(t) = -\frac{1}{a_p}\left[a_{p-1} {q_e}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t)\right]$$

$$\Rightarrow \boxed{{q_e}^{(p)}(t) = -{a_{p-1}}' {q_e}^{(p-1)}(t) - \cdots - {a_2}' \ddot{q}_e(t) - {a_1}' \dot{q}_e(t) - {a_0}' q_e(t)}$$

- Express a p^th^-order differential equation as p coupled 1^st^-order differential equations:

$$\begin{array}{c} x_1 = q_e \\ x_2 = \dot{x}_1 = \dot{q}_e \\ \vdots \\ x_p = \dot{x}_{p-1} = {q_e}^{(p-1)} \end{array} \rightarrow \begin{array}{c} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \vdots \\ \dot{x}_{p-1} = x_p \end{array}$$

$$\boxed{\dot{x}_p = -{a_{p-1}}' x_p - \cdots - {a_2}' x_3 - {a_1}' x_2 - {a_0}' x_1}$$

- If $c = 0$:

$$a_p {q_e}^{(p)}(t) + a_{p-1} {q_e}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t) = 0$$

$$\Rightarrow {q_e}^{(p)}(t) = -\frac{1}{a_p}\left[a_{p-1} {q_e}^{(p-1)}(t) + \cdots + a_2 \ddot{q}_e(t) + a_1 \dot{q}_e(t) + a_0 q_e(t)\right]$$

$$\Rightarrow {q_e}^{(p)}(t) = -a_{p-1}' {q_e}^{(p-1)}(t) - \cdots - a_2' \ddot{q}_e(t) - a_1' \dot{q}_e(t) - a_0' q_e(t)$$

- Express a p^th-order differential equation as p coupled 1^st-order differential equations:

$$\boxed{\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_{p-1} &= x_p \end{aligned}}$$

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_p \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ -a_0' & -a_1' & -a_2' & \cdots & -a_{p-2}' & -a_{p-1}' \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}}_{x}$$

$$\boxed{\dot{x}_p = -a_{p-1}' x_p - \cdots - a_2' x_3 - a_1' x_2 - a_0' x_1}$$

Linear Error Dynamics

- $\dot{\boldsymbol{x}}(t) = \boldsymbol{Ax}(t)$ has solution $\boldsymbol{x}(t) = e^{\boldsymbol{A}t}\boldsymbol{x}(0)$
- Just like $x(t) = e^{at}x(0)$ converge if $a < 0$, $\boldsymbol{x}(t)$ converge if $\boldsymbol{A}$ is negative definite
- Matrix $\boldsymbol{A}$ is negative definite iff all eigenvalues of $\boldsymbol{A}$ (which maybe complex) have negative real components. The eigenvalues s of A must satisfy:

$$\det(sI - A) = s^p + a_{p-1}{}'s^{p-1} + \cdots + a_2{}'s^2 - a_1{}'s^1 - a_0{}' = 0$$

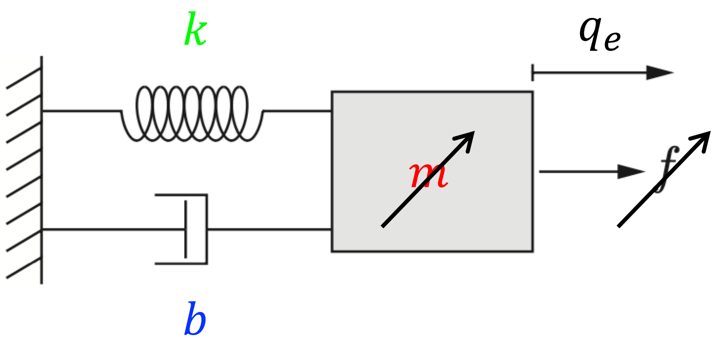
with necessary conditions:

1. $a_i{}' > 0 \ \forall i$
2. ...

1st-order Error Dynamics

- Take linear mass-spring-damper for example, let $m = 0$ and $f = 0$:

$$b\dot{q}_e(t) + kq_e(t) = 0 \Rightarrow \dot{q}_e(t) + \frac{k}{b}q_e(t) = 0$$

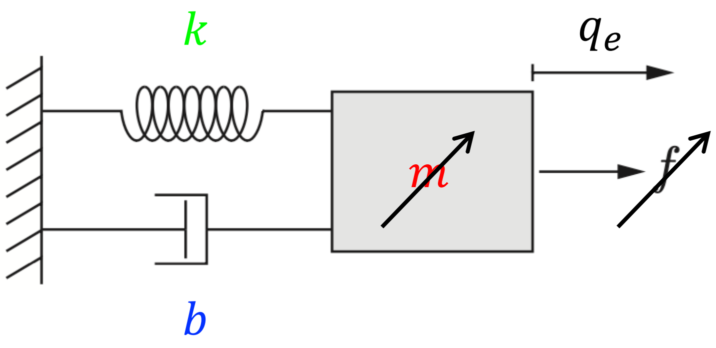


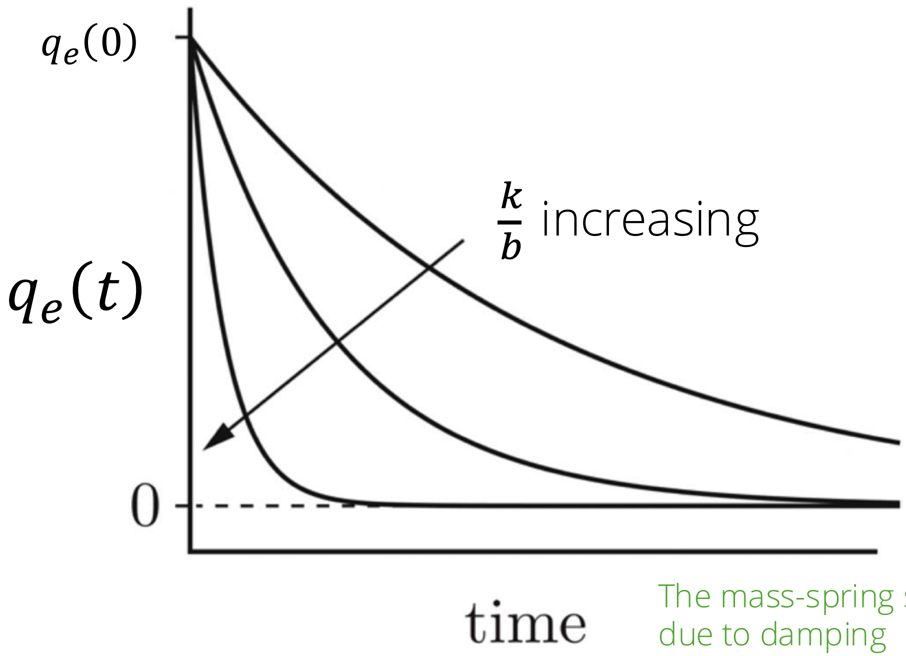
1^{st}-order Error Dynamics

- Take linear mass-spring-damper for example, let $m = 0$ and $f = 0$:

$$b\dot{q}_e(t) + kq_e(t) = 0 \Rightarrow \dot{q}_e(t) + \frac{k}{b}q_e(t) = 0$$

- We have solution $q_e(t) = e^{-\frac{k}{b}t}q_e(0)$





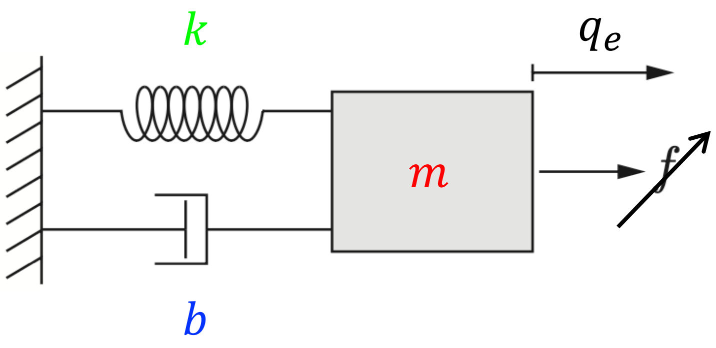
The mass-spring stops due to damping

2^{nd}-order Error Dynamics

- Take linear mass-spring-damper for example, let $f = 0$:

$$m\ddot{q}_e(t) + b\dot{q}_e(t) + kq_e(t) = 0 \Rightarrow \ddot{q}_e(t) + \frac{b}{m}\dot{q}_e(t) + \frac{k}{m}q_e(t) = 0$$

$$\Rightarrow \ddot{q}_e(t) + 2\xi\omega_n\dot{q}_e(t) + {\omega_n}^2 q_e(t) = 0$$



$\omega_n = \sqrt{\frac{k}{m}}$ is known as the natural frequency

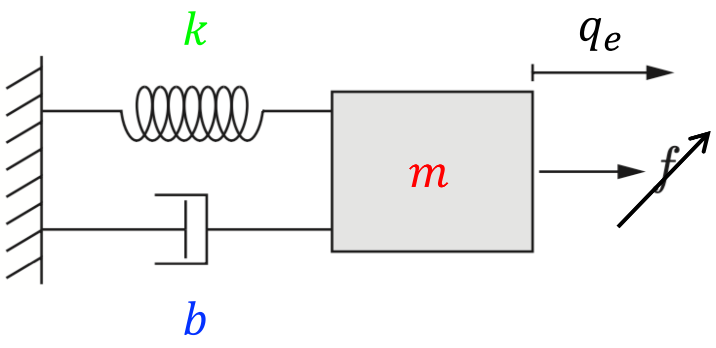
$\xi = \frac{b}{2\sqrt{km}}$ is known as the damping ratio

2^{nd}-order Error Dynamics

- Take linear mass-spring-damper for example, let $f = 0$:

$$m\ddot{q}_e(t) + b\dot{q}_e(t) + kq_e(t) = 0 \Rightarrow \ddot{q}_e(t) + \frac{b}{m}\dot{q}_e(t) + \frac{k}{m}q_e(t) = 0$$

$$\Rightarrow \ddot{q}_e(t) + 2\xi\omega_n\dot{q}_e(t) + {\omega_n}^2 q_e(t) = 0$$



The characteristic polynomial:

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0$$

Two roots:

$$s_1 = -\zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1}$$

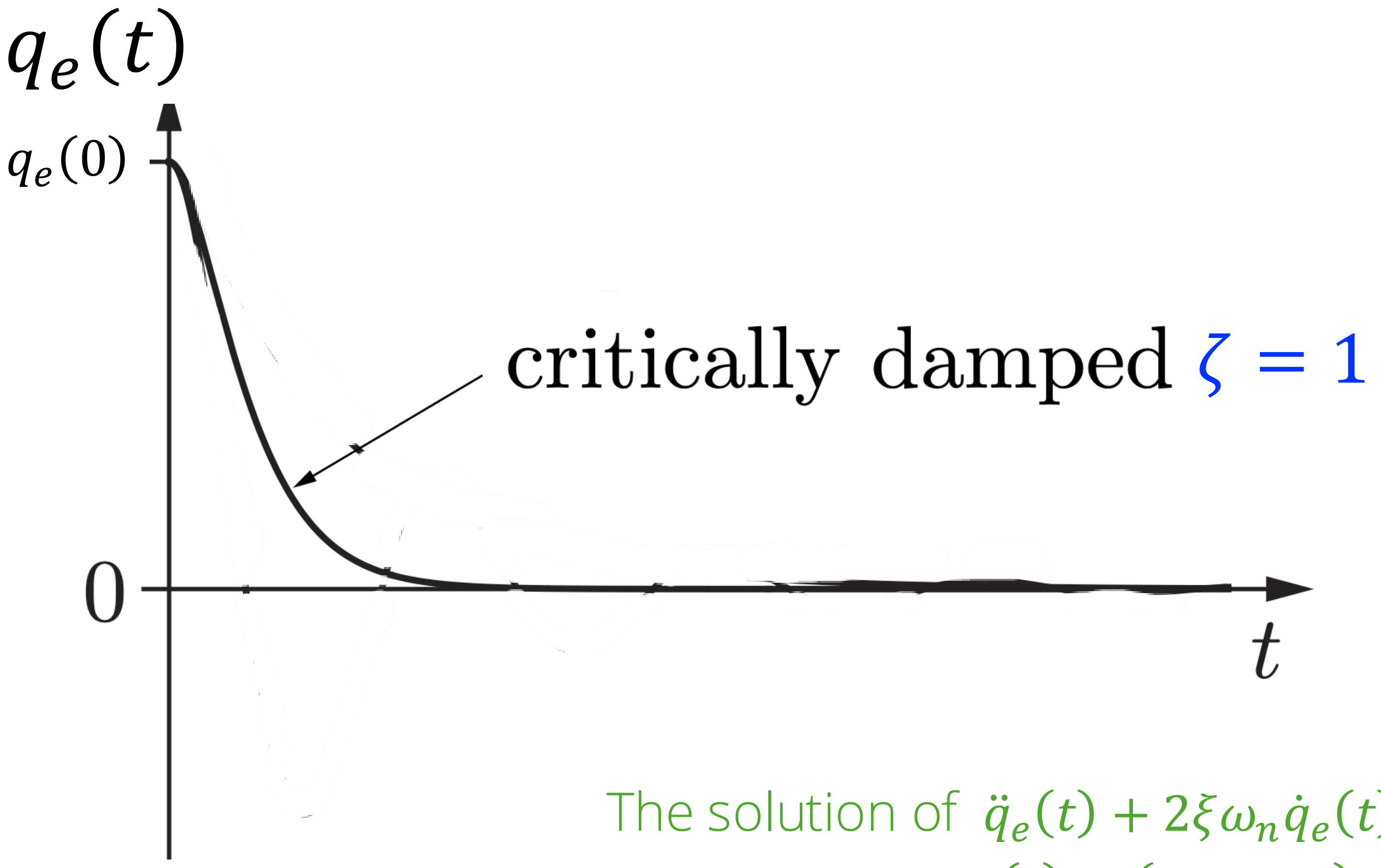
$$s_2 = -\zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1}.$$

Stable conditions:

$$\zeta\omega_n > 0$$

$$\omega_n^2 > 0.$$

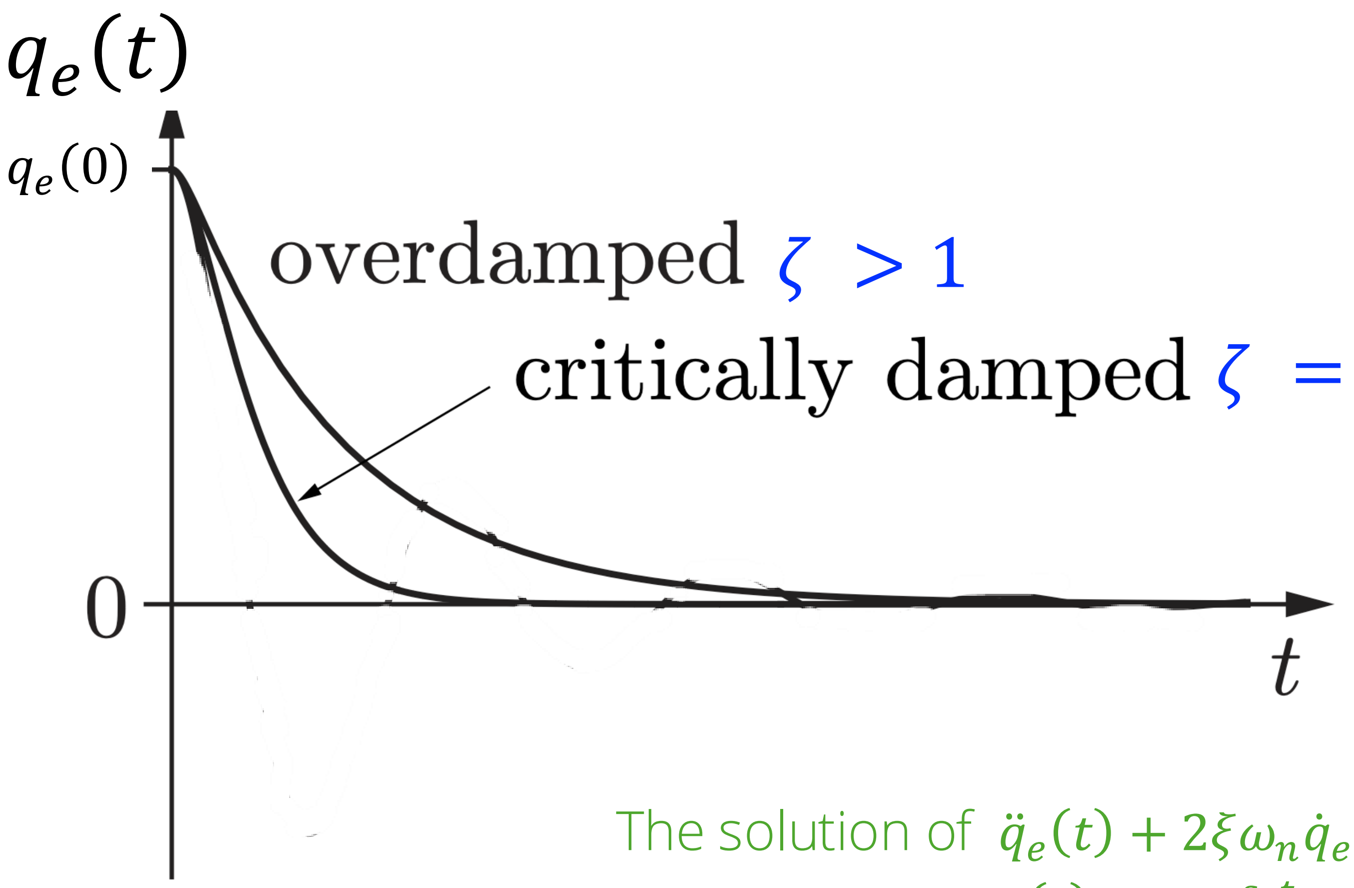
2nd-order Error Dynamics



The solution of $\ddot{q}_e(t) + 2\xi\omega_n\dot{q}_e(t) + {\omega_n}^2 q_e(t) = 0$:

$$q_e(t) = (c_1 + c_2 t)e^{-\omega_n t}$$

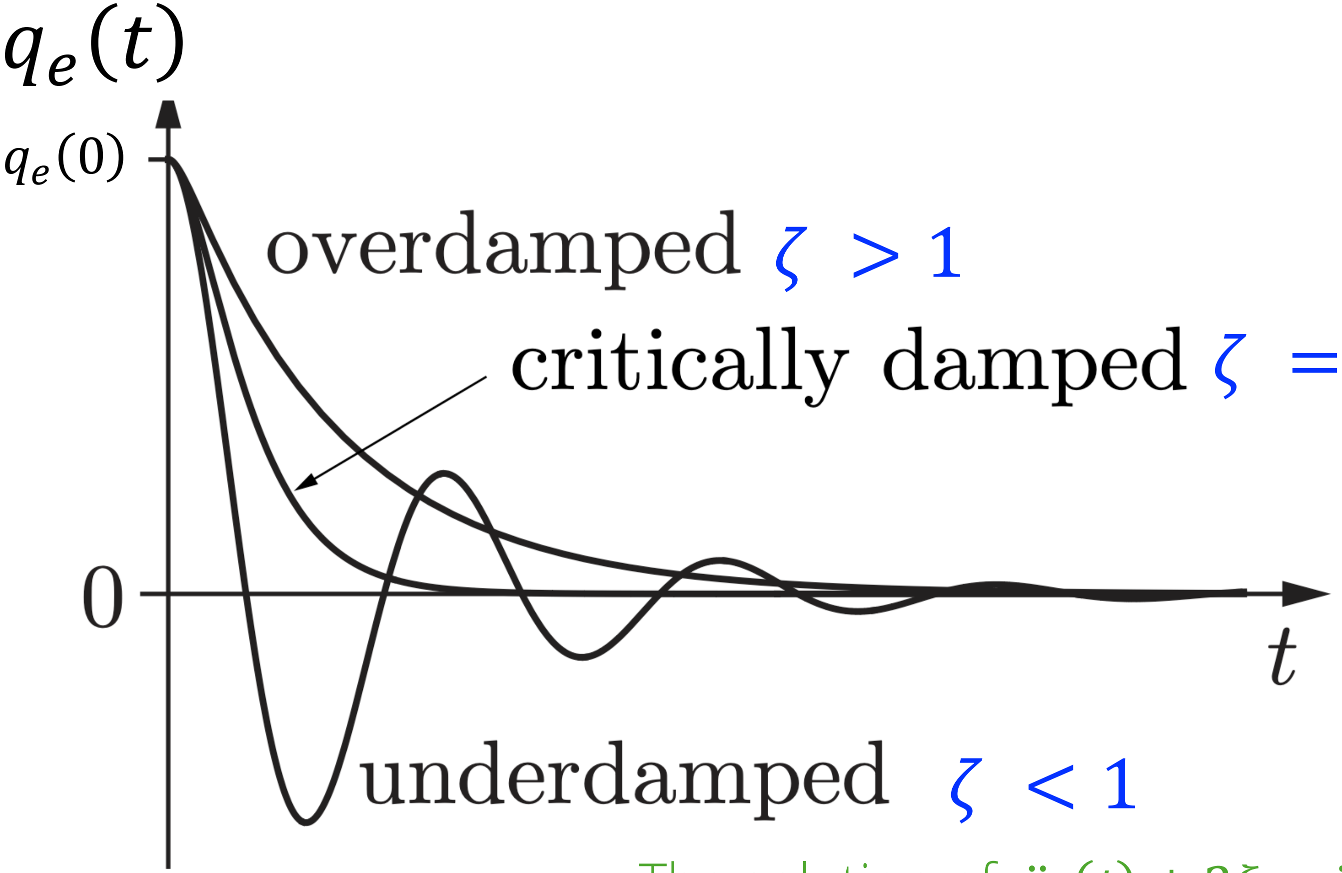
2nd-order Error Dynamics



The solution of $\ddot{q}_e(t) + 2\xi\omega_n\dot{q}_e(t) + \omega_n{}^2 q_e(t) = 0$:

$$q_e(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}$$

2^{nd}-order Error Dynamics



The solution of $\ddot{q}_e(t) + 2\xi\omega_n\dot{q}_e(t) + {\omega_n}^2 q_e(t) = 0$:

$$q_e(t) = \left(c_1 \cos \omega_n\sqrt{1-\xi^2}\, t + c_2 \sin \omega_n\sqrt{1-\xi^2}\, t\right) e^{-\zeta\omega_n t}$$

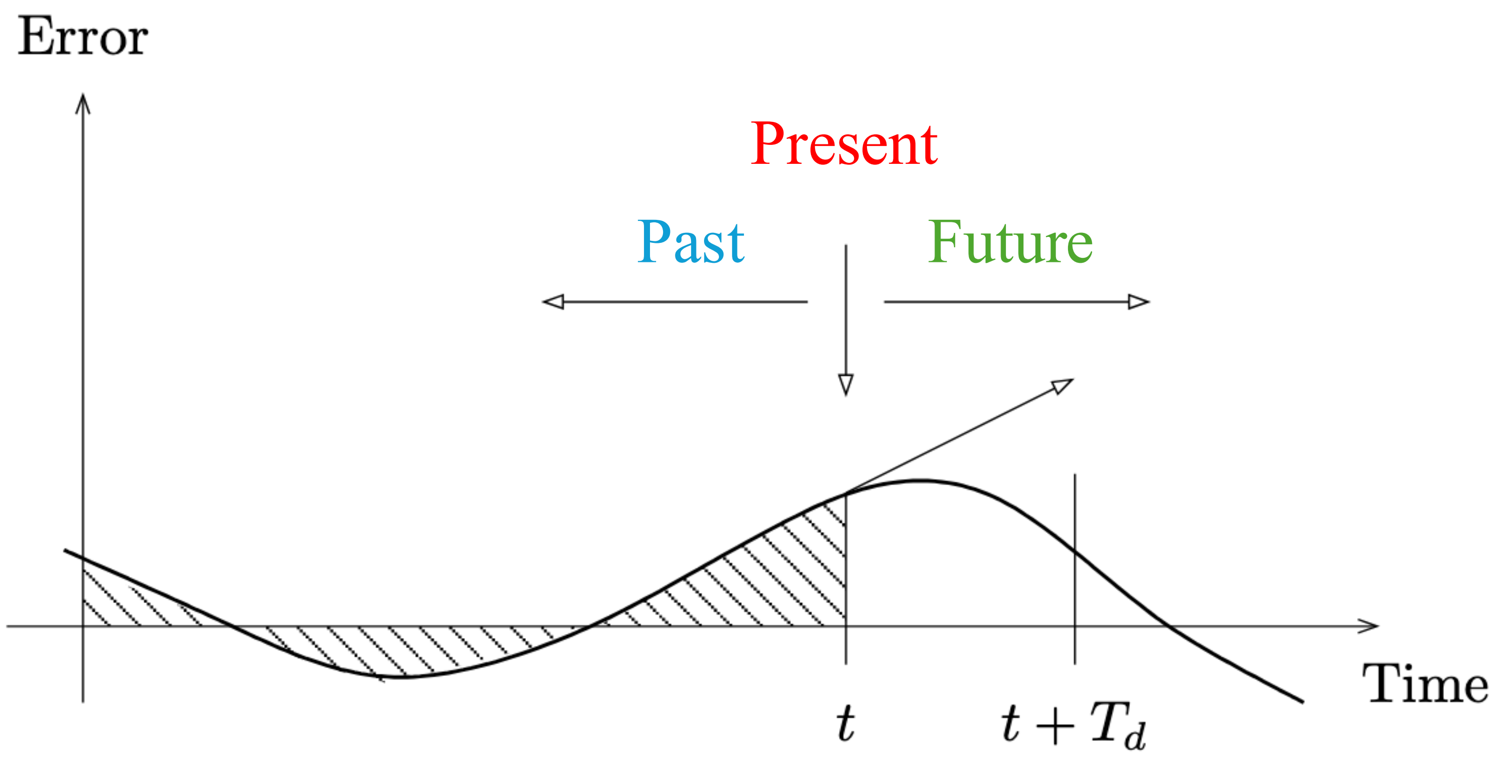
PID Feedback Controller

- PID feedback controller:
  - 1^{st}-order error (immediate error)
  - Higher-order error (derivatives of errors)
  - Historical error (accumulated error)

$$u(t) = K_p q_e(t) + K_d \dot{q}_e(t) + K_i \int_0^t q_e(t)\, dt$$

- PID feedback controller:
  - 1st-order error (immediate error)
  - Higher-order error (derivatives of errors)
  - Historical error (accumulated error)

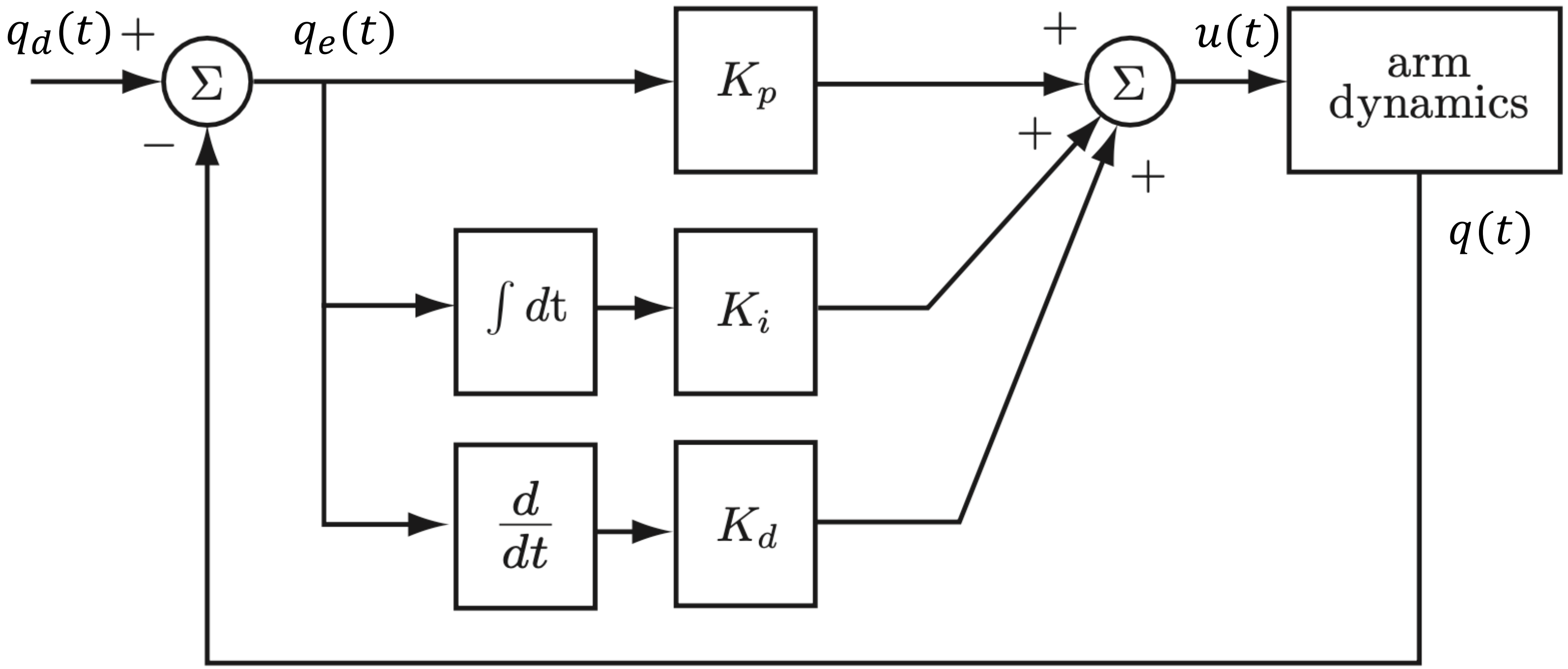
$$u(t) = K_p q_e(t) + K_d \dot{q}_e(t) + K_i \int_0^t q_e(t)\, dt$$



Feedback Systems: An Introduction for Scientists and Engineers. K. J. Åström and R. M. Murray.

A Simplified Block Diagram of PID Controller

- Control $u(t)$ can be velocity / torque inputs



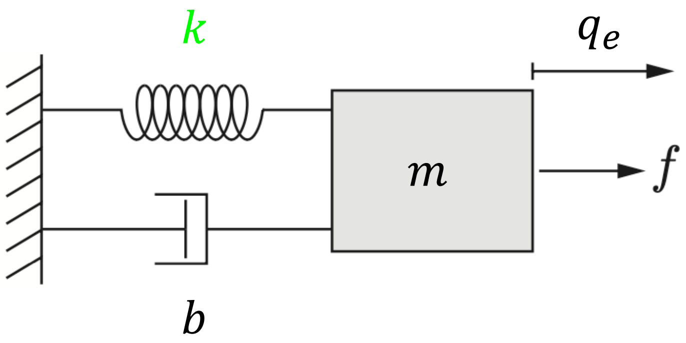
Proportional Controller: Minimize Immediate Error

- Control with velocity inputs:

$$u(t) = \dot{q}(t) = K_p\left(\boxed{q_d(t)} - q(t)\right) = K_p q_e(t)$$

desired $q(t)$

- The constant control gain K_p acts somewhat like a virtual spring



Proportional Controller: Minimize Immediate Error

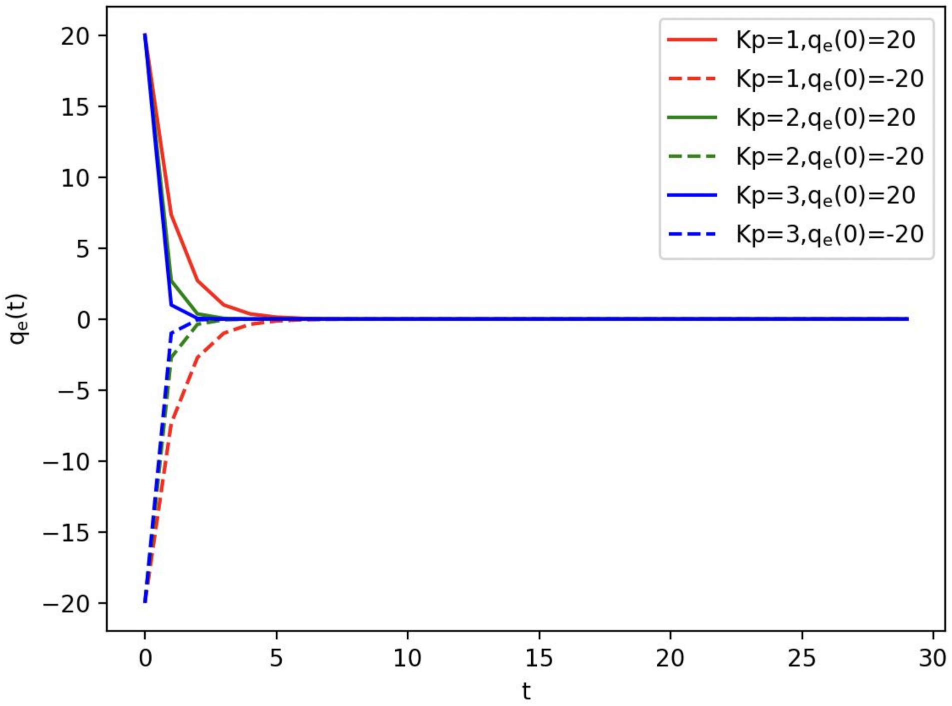
- Control with <u>velocity inputs</u>:

$$u(t) = \dot{q}(t) = K_p\big(q_d(t) - q(t)\big) = K_p q_e(t)$$

- Case 1: : if $q_d(t)$ is constant

$$\dot{q}_e(t) = \cancel{\dot{q}_d(t)} - \dot{q}(t) \Rightarrow \dot{q}(t) = -\dot{q}_e(t)$$

$$-\dot{q}_e(t) = K_p q_e(t) \Rightarrow q_e(t) = e^{-K_p t} q_e(0)$$



Proportional Controller: Minimize Immediate Error

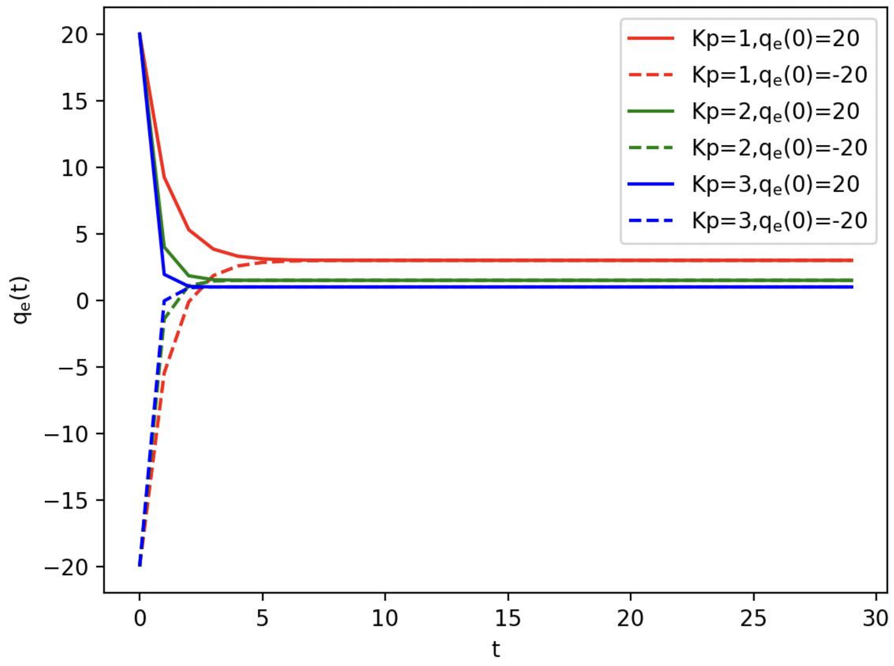
- Control with velocity inputs:

$$u(t) = \dot{q}(t) = K_p\big(q_d(t) - q(t)\big) = K_p q_e(t)$$

- Case 2: if $q_d(t)$ is not constant, but $\dot{q_d}(t)$ is constant

$$\dot{q}_d(t) = c$$

$$\dot{q}_e(t) = c - \dot{q}(t) = c - K_p q_e(t) \Rightarrow q_e(t) = \frac{c}{K_p} + (q_e(0) - \frac{c}{K_p})e^{-K_p t}$$



Proportional Controller: Minimize Immediate Error

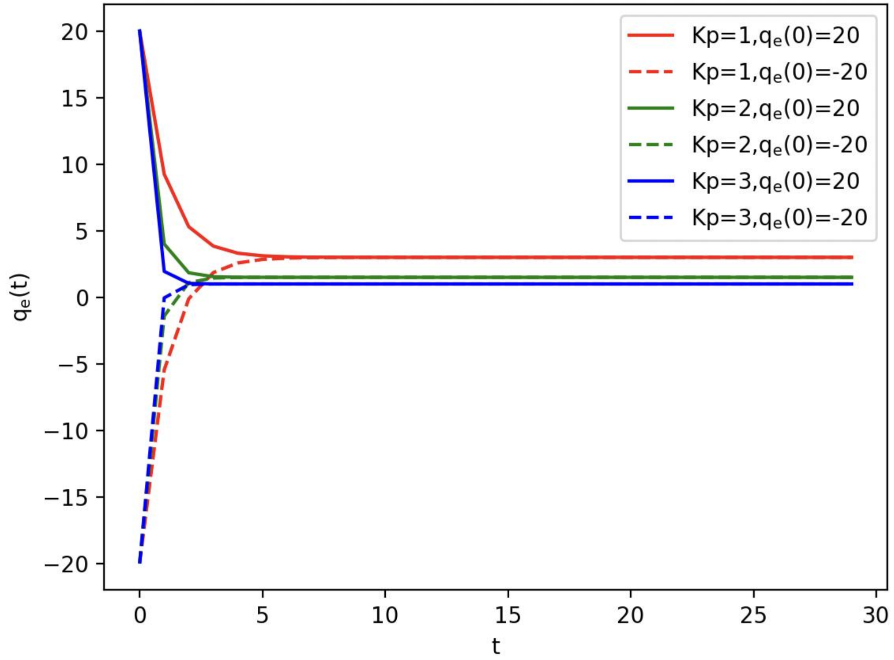
- Control with velocity inputs:

$$u(t) = \dot{q}(t) = K_p\big(q_d(t) - q(t)\big) = K_p q_e(t)$$

- Case 2: if $q_d(t)$ is not constant, but $\dot{q_d}(t)$ is constant

$$\dot{q}_d(t) = c$$

$$\dot{q}_e(t) = c - \dot{q}(t) = c - K_p q_e(t) \Rightarrow q_e(t) = \frac{c}{K_p} + (q_e(0) - \frac{c}{K_p})e^{-K_p t}$$



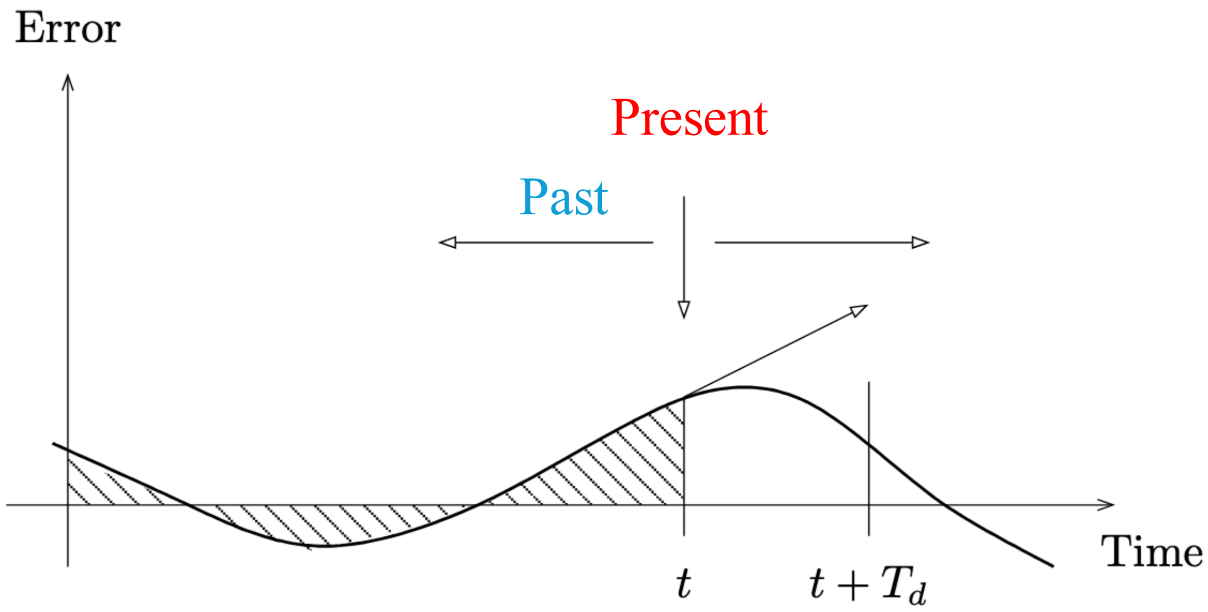
Solution?

- Increase K_p. What if K_p is bounded?
- Minimize accumulated error

Proportional Integral Controller: Minimize Historical Error

- Control with velocity inputs:

$$\dot{q}(t) = K_p q_e(t) + K_i \int_0^t q_e(t)\, dt$$



Proportional Integral Controller: Minimize Historical Error

- Control with velocity inputs:

$$u(t) = \dot{q}(t) = K_p q_e(t) + K_i \int_0^t q_e(t)\, dt$$

- If $\dot{q_d}(t)$ is constant

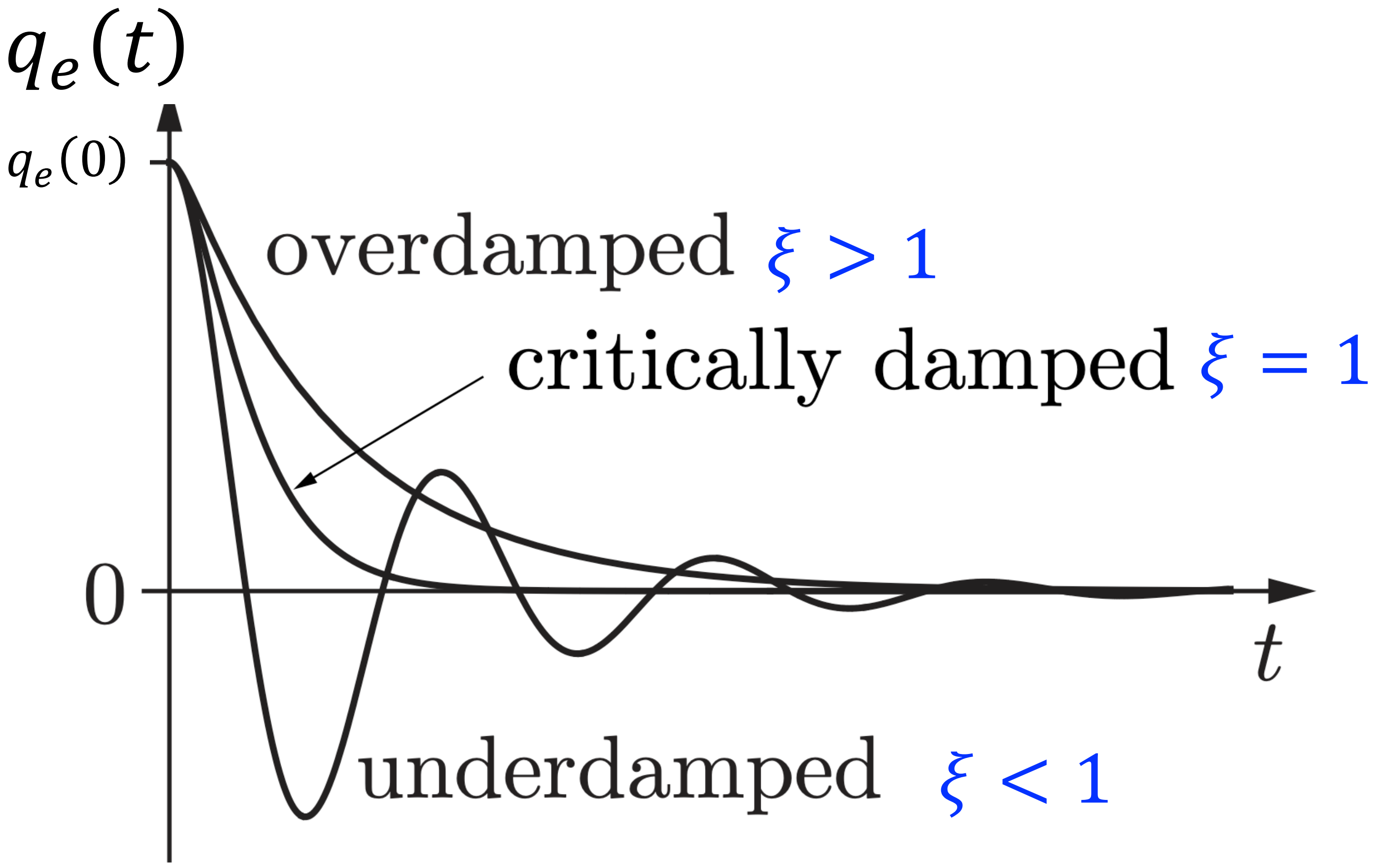
$$\dot{q}(t) = \dot{q}_d(t) - \dot{q}_e(t) = K_p q_e(t) + K_i \int_0^t q_e(t)\, dt$$

$$\Longrightarrow c = \dot{q}_e(t) + K_p q_e(t) + K_i \int_0^t q_e(t)\, dt$$

(Take derivatives)

$$\Longrightarrow \underbrace{\ddot{q}_e(t) + K_p \dot{q}_e(t) + K_i q_e(t)}_{\text{second-order error dynamics}} = 0$$

2nd-order Error Dynamics



2^{nd}-order Error Dynamics

- Given the standard second-order form:

$$\ddot{q}_e(t) + 2\zeta\omega_n\dot{q}_e(t) + {\omega_n}^2 q_e(t) = 0$$

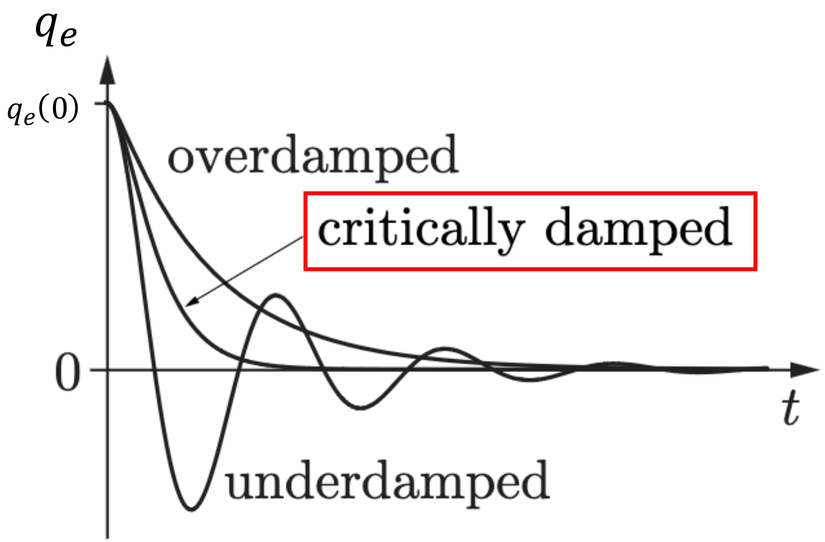
- Our PI controller:

$$\ddot{q}_e(t) + K_p\dot{q}_e(t) + K_i q_e(t) = 0$$

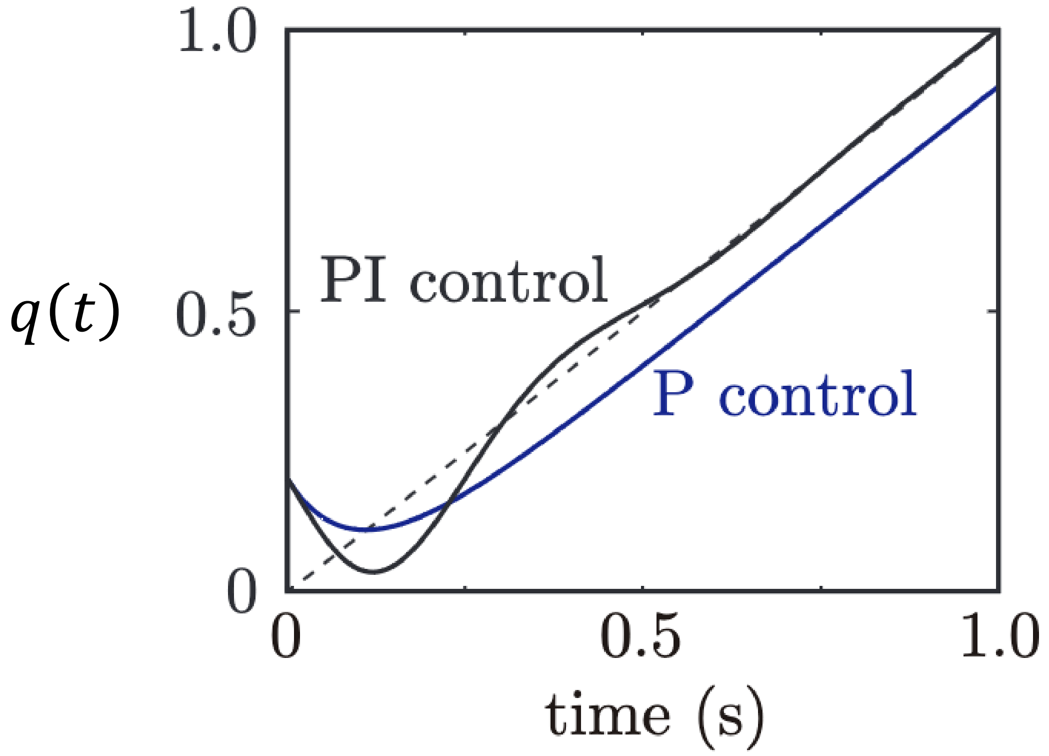
- We have $\omega_n = \sqrt{K_i}$ and $\zeta = \frac{K_p}{2\sqrt{K_i}}$

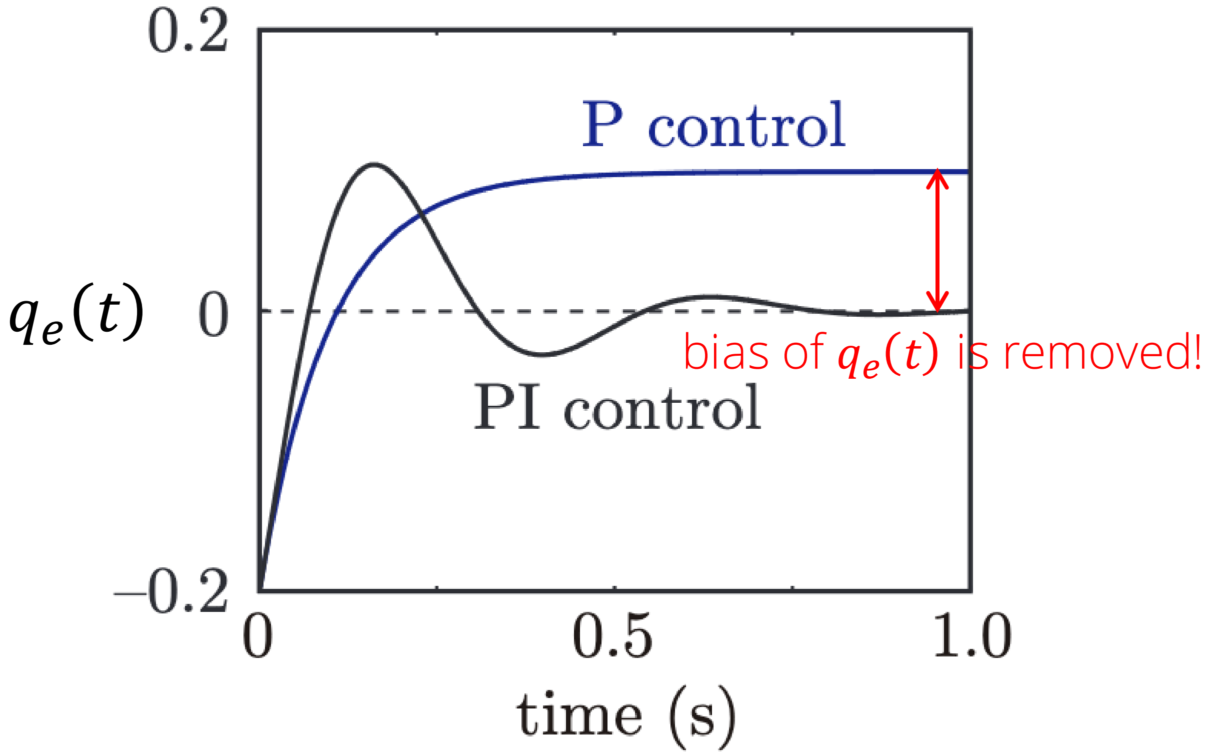
- To achieve critically damped state, we need $\zeta = 1$, $\zeta\omega_n > 0$ and ${\omega_n}^2 > 0$

Need to tune carefully!



PI Controller vs. P Controller

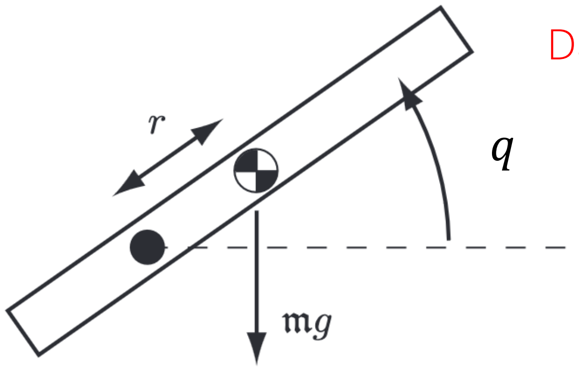




How About Force Inputs?

- Velocity inputs are limited to applications with low/predictable force-torque requirements.   Here, we consider control force inputs.
- Let's take a single-joint robot for example

$$\tau = M\ddot{q}(t) + mgr\cos q(t) + \underbrace{b\dot{q}(t)}_{\text{Damping (e.g friction)}}$$



- PID controller: $\tau = K_p q_e(t) + K_d \dot{q}_e(t) + K_i \int_0^t q_e(t)\,dt$

Image credit K. Lynch and F. Park

Proportional Derivatives Controller

- PD controller: $\tau = K_p q_e(t) + K_d \dot{q}_e(t)$
- Let's consider the robot is placed on a plane ($g = 0$)

$$M\ddot{q}(t) + \cancel{mgr\cos q(t)} + b\dot{q}(t) = K_p q_e(t) + K_d \dot{q}_e(t)$$

$$\Rightarrow M\ddot{q}(t) + b\dot{q}(t) = K_p(q_d(t) - q(t)) + K_d(\dot{q}_d(t) - \dot{q}(t))$$

Proportional Derivatives Controller

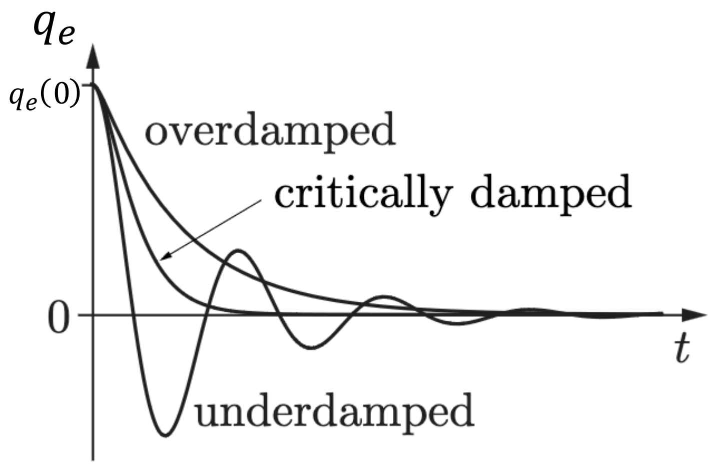
- PD controller: $\tau = K_p q_e(t) + K_d \dot{q}_e(t)$
- Let's consider the robot is placed on a plane ($g = 0$)

$$M\ddot{q}(t) + b\dot{q}(t) = K_p(q_d(t) - q(t)) + K_d(\dot{q}_d(t) - \dot{q}(t))$$

- Case 1: if $q_d(t) = c$ is constant
  - ➢ We have $\dot{q}_d = \ddot{q}_d = 0, q_e = c - q, \dot{q} = -\dot{q}_e, \ddot{q} = -\ddot{q}_e$
  - ➢ Substituting $q, \dot{q}, \ddot{q}$, we have

$$\boxed{M\ddot{q}_e(t) + (b + K_d)\dot{q}_e(t) + K_p q_e(t) = 0}$$

Doesn't this look familiar?



Proportional Derivatives Controller

- Let's consider the robot is placed vertically

$$M\ddot{q}(t) + mgr\cos q(t) + b\dot{q}(t) = K_p(q_d(t) - q(t)) + K_d(\dot{q}_d(t) - \dot{q}(t))$$

- Case 1: if $q_d(t) = c$ is constant

$$\underbrace{M\ddot{q}_e(t) + (b + K_d)\dot{q}_e(t) + K_p q_e(t) = mgr\cos q(t)}$$

Very complex to get an explicit solution. But let's observe when the system is stable

Proportional Derivatives Controller

- Let's consider the robot is placed vertically

$$M\ddot{q}(t) + mgr\cos q(t) + b\dot{q}(t) = K_p(q_d(t) - q(t)) + K_d(\dot{q}_d(t) - \dot{q}(t))$$

- Case 1: if $q_d(t) = c$ is constant

$$\underbrace{M\ddot{q}_e(t) + (b + K_d)\dot{q}_e(t) + K_p q_e(t) = mgr\cos q(t)}$$

when the joint comes to rest ($\ddot{q}_e(t) = 0$ and $\dot{q}_e(t) = 0$),
the final error $q_e(t) \neq 0$:

$$K_p q_e(t) = mgr\cos q(t)$$

In other words, the robot has to provide a torque to hold the link at rest $\theta \neq \pm\frac{\pi}{2}$, which happens when $q_e \neq 0$

Add Integral Controller to Reduce the Bias

- Let's consider the robot is placed vertically and a PID controller is used:

$$M\ddot{q}_e(t) + (b + K_d)\dot{q}_e(t) + K_p q_e(t) + K_i \int_0^t q_e(t)\, dt = mgr\cos q(t)$$

- Still a complex equation…. Let's envision what happens when the system is close to equilibrium $q_e = 0$ (and thus $\dot{q}$ is also close to 0), since the desired $q_d(t)$ is constant :

Explain this better!

(Take derivatives)

$$M\dddot{q}_e(t) + (b + K_d)\ddot{q}_e(t) + K_p \dot{q}_e(t) + K_i q_e(t) = 0$$

PID Controller

$$M\dddot{q}_e(t) + (b + K_d)\ddot{q}_e(t) + K_p\dot{q}_e(t) + K_i q_e(t) = 0$$

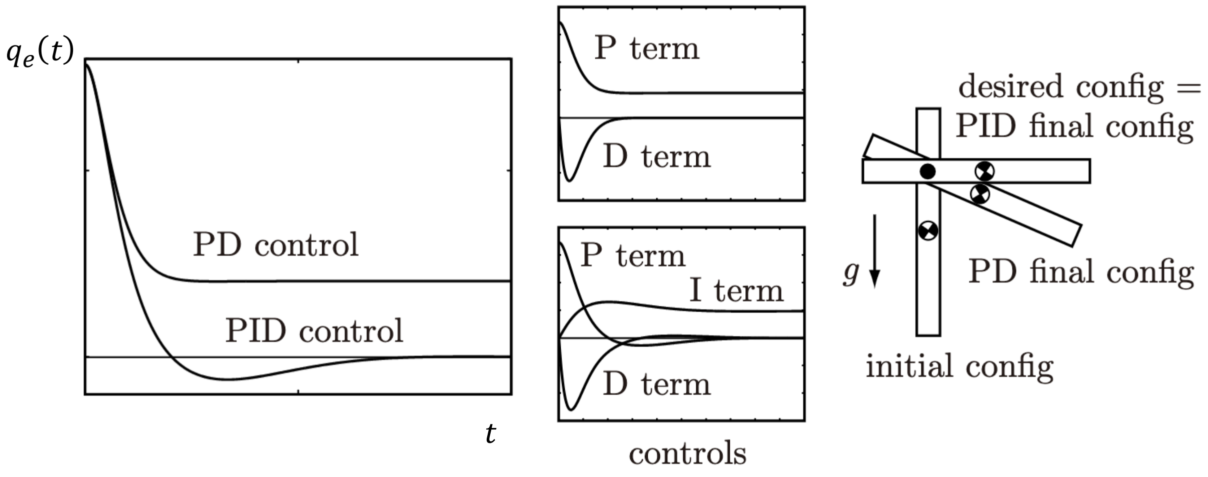
- The characteristic equation is

$$s^3 + \frac{b + K_d}{M}s^2 + \frac{K_p}{M}s + \frac{K_i}{M} = 0$$

- The conditions for q_e to converge to zero

$$\boxed{\begin{array}{rcl} & & K_d > -b \\ & & K_p > 0 \\ \frac{(b+K_d)K_p}{M} & > & K_i > 0. \end{array}}$$

Require parameter tuning!

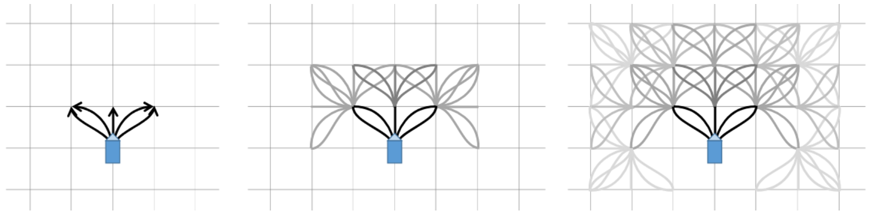


There are more optimal controllers than PID

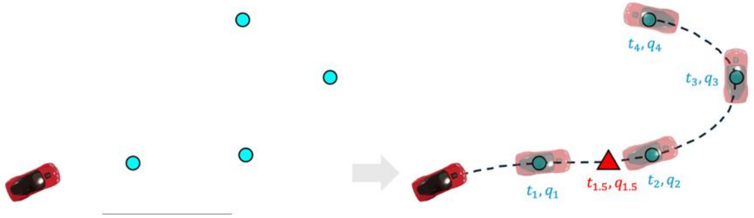
https://youtu.be/onOd7xWbGAk?si=azyRe0Gz_yOe29nw

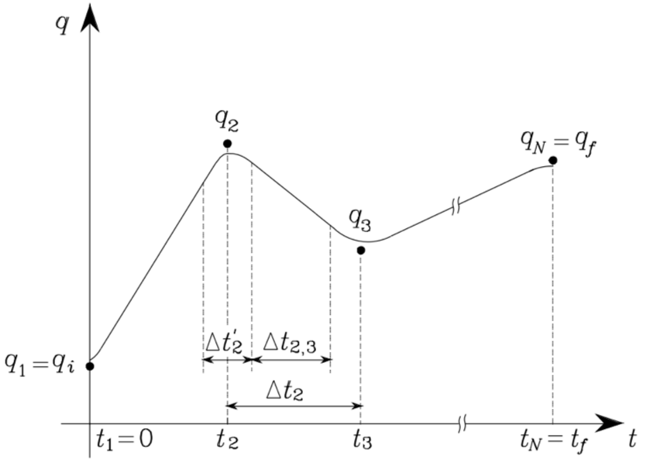
Summary

Motion Planning with Directional Constraints

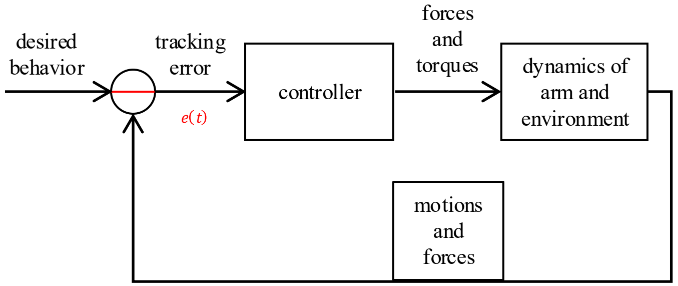


Trajectory Generation

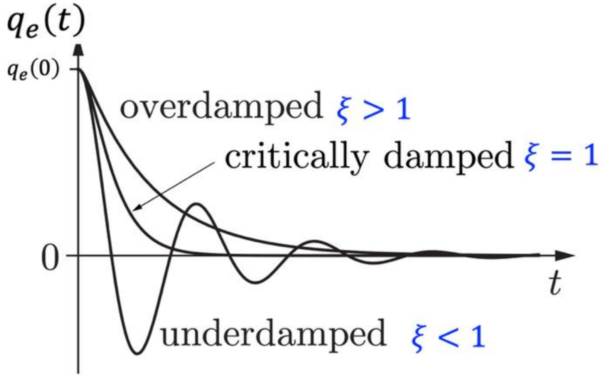




Feedback Controller



$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_p \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ -a_0' & -a_1' & -a_2' & \cdots & -a_{p-2}' & -a_{p-1}' \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}}_{x}$$



Kinodynamic RRT