Content from Drew Bagnell, Pieter Abeel

Iterative LQR & Model Predictive Control

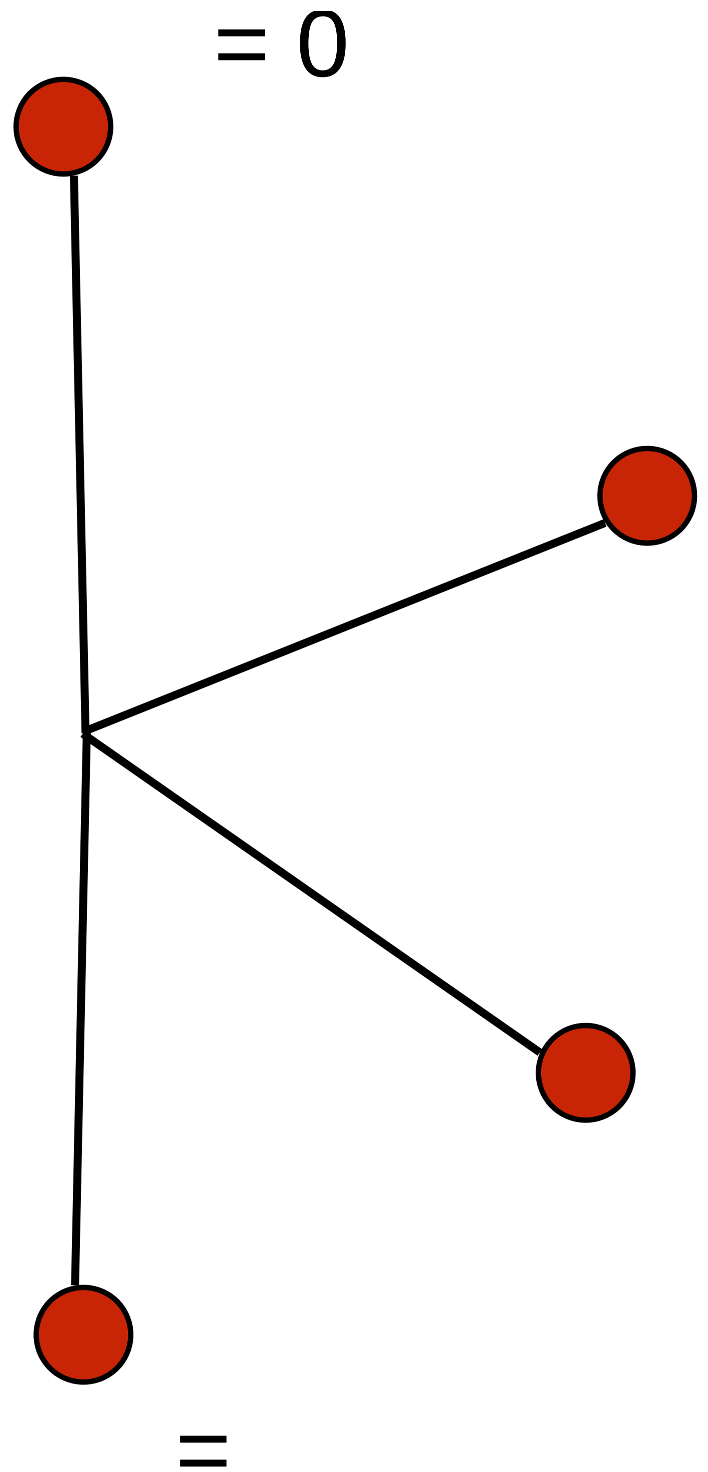
Sanjiban Choudhury

TAs: Matthew Rockett, Gilwoo Lee, Matt Schmittle

Table of Controllers

	Control Law	Uses model	Stability Guarantee	Minimize Cost
PID		No	No	No
Pure Pursuit		Circular arcs	Yes - with assumptions	No
Lyapunov		Non-linear	Yes	No
LQR		Linear	Yes	Quadratic

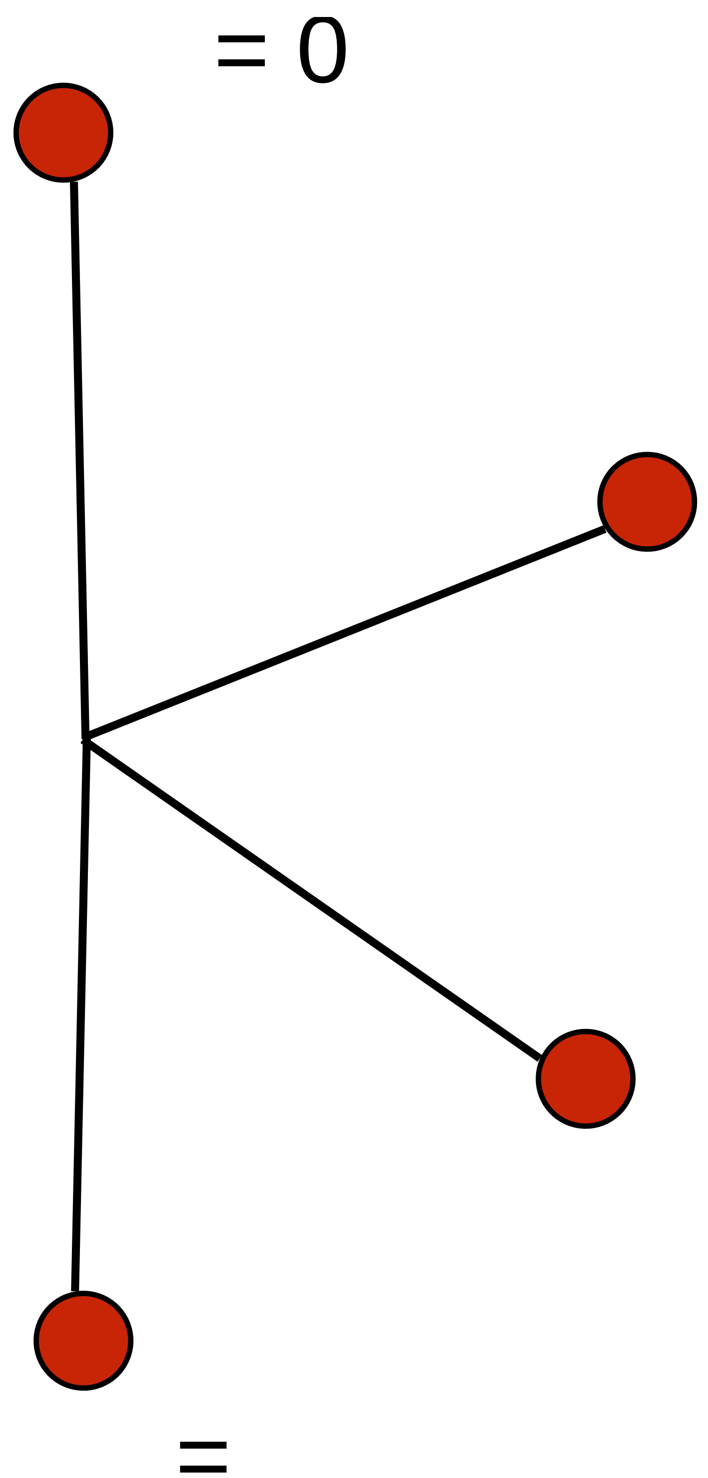
Can we use LQR to swing up a pendulum



Can we use LQR to swing up a pendulum

No!

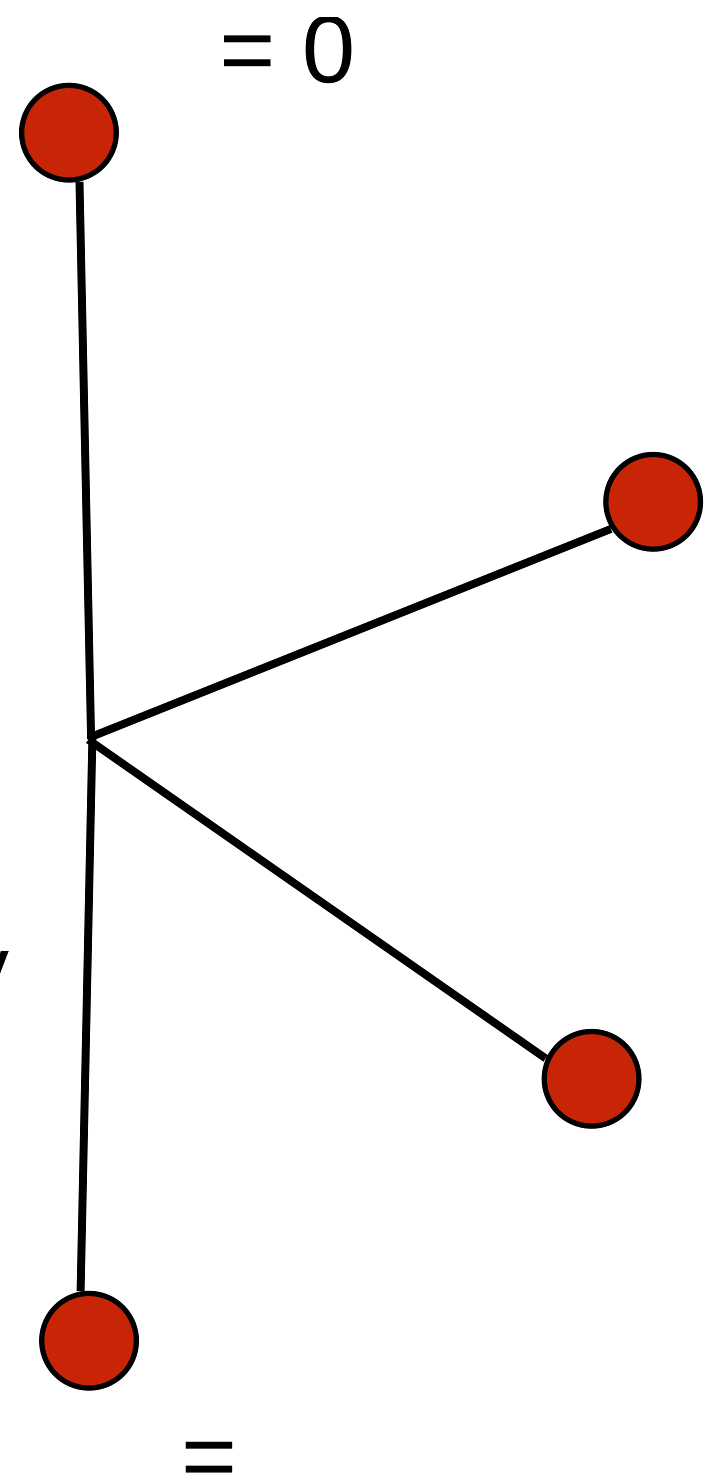
(Large angles imply large linearization error)



Can we use LQR to swing up a pendulum

No!

(Large angles imply
large linearization error)



But we can track
a reference swing up trajectory
(small linearization error)

But, rst we need to talk about time-varying systems

Today’s objectives

1. LQR for time-varying systems
2. Trajectory following with iLQR
3. General nonlinear trajectory optimization with iLQR
4. Model predictive control (MPC)

LQR for Time-Varying Dynamical Systems

$$x_{t+1} = A_t x_t + B_t u_t$$

LQR for Time-Varying Dynamical Systems

$$x_{t+1} = A_t x_t + B_t u_t$$

$$c(x_t, u_t) = x_t^T Q_t x_t + u_t^T R_t u_t$$

LQR for Time-Varying Dynamical Systems

$$x_{t+1} = A_t x_t + B_t u_t$$

$$c(x_t, u_t) = x_t^T Q_t x_t + u_t^T R_t u_t$$

Straight forward to get LQR equations

$$\checkmark \quad K_t = (R_t + B_t^T V_{t+1} B_t)^{1} B_t^T V_{t+1} A_t$$

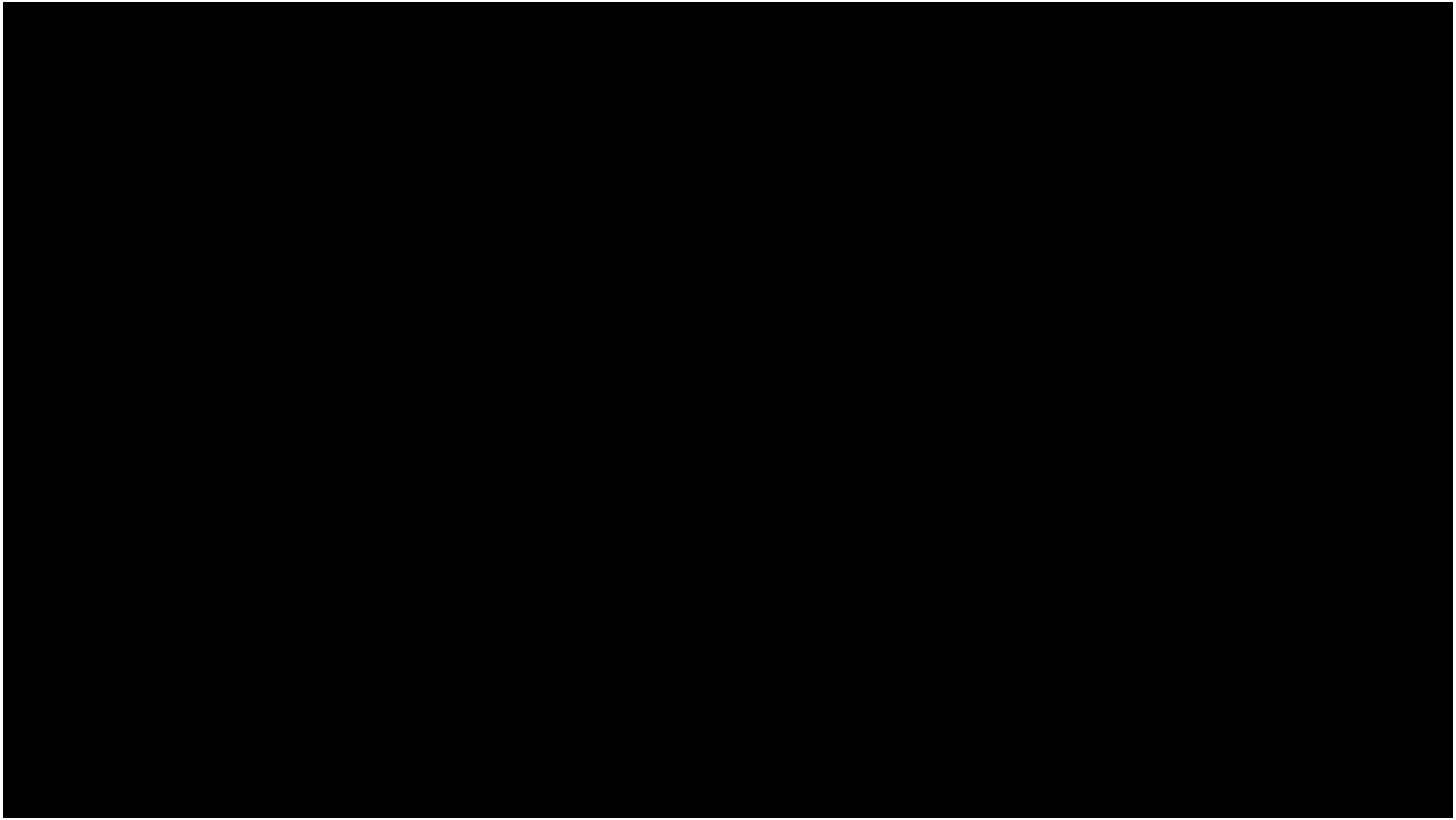
$$\overset{P_t}{V_t} = Q_t + K_t^T R_t K_t + (A_t + B_t K_t)^T \overset{P_{t+1}}{V_{t+1}} (A_t + B_t K_t)$$

Discrete Algebraic Riccati equation

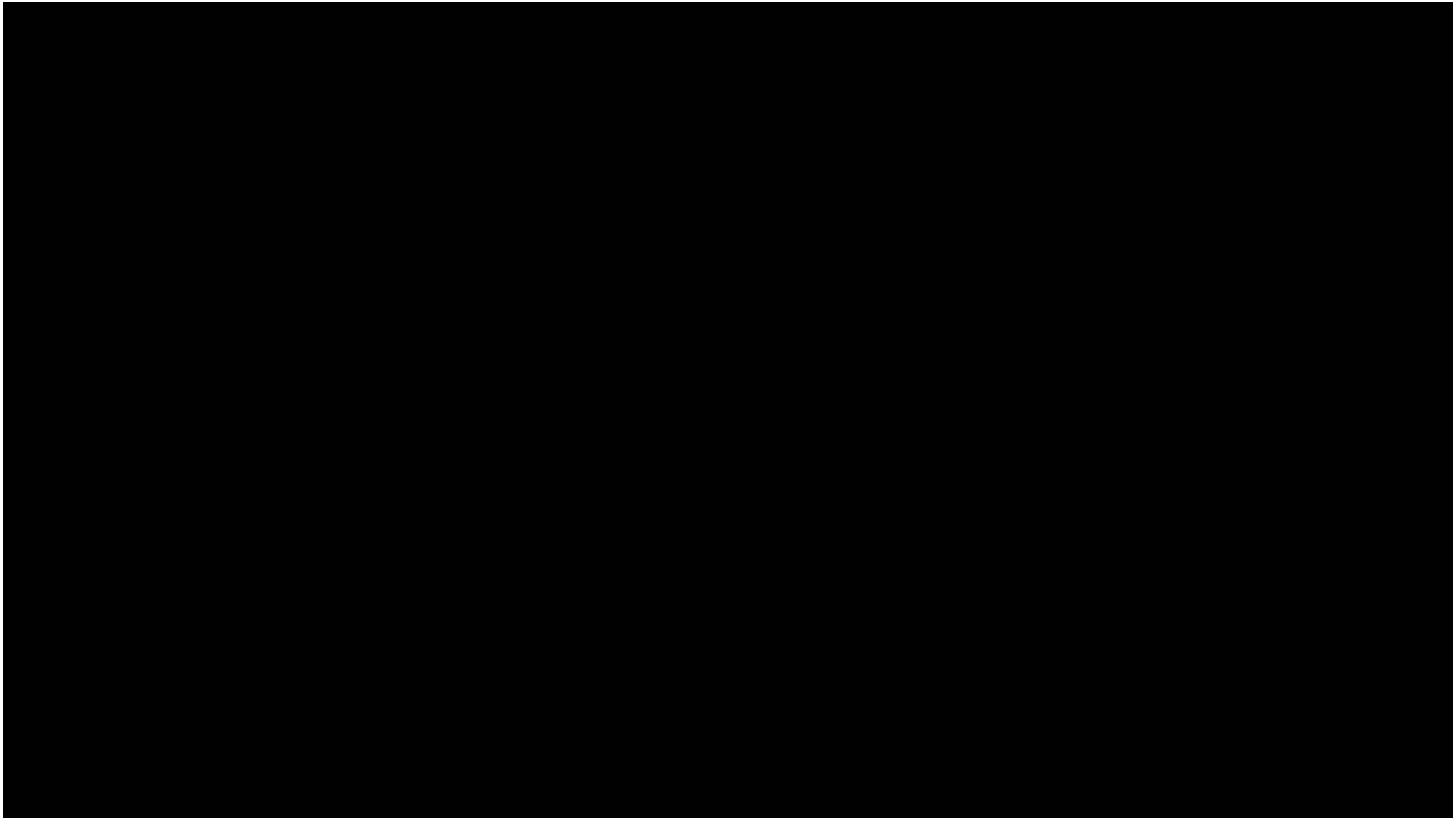
Why do we care about time-varying?

Ans: Linearization about a trajectory

Trajectory tracking for stationary rolls?

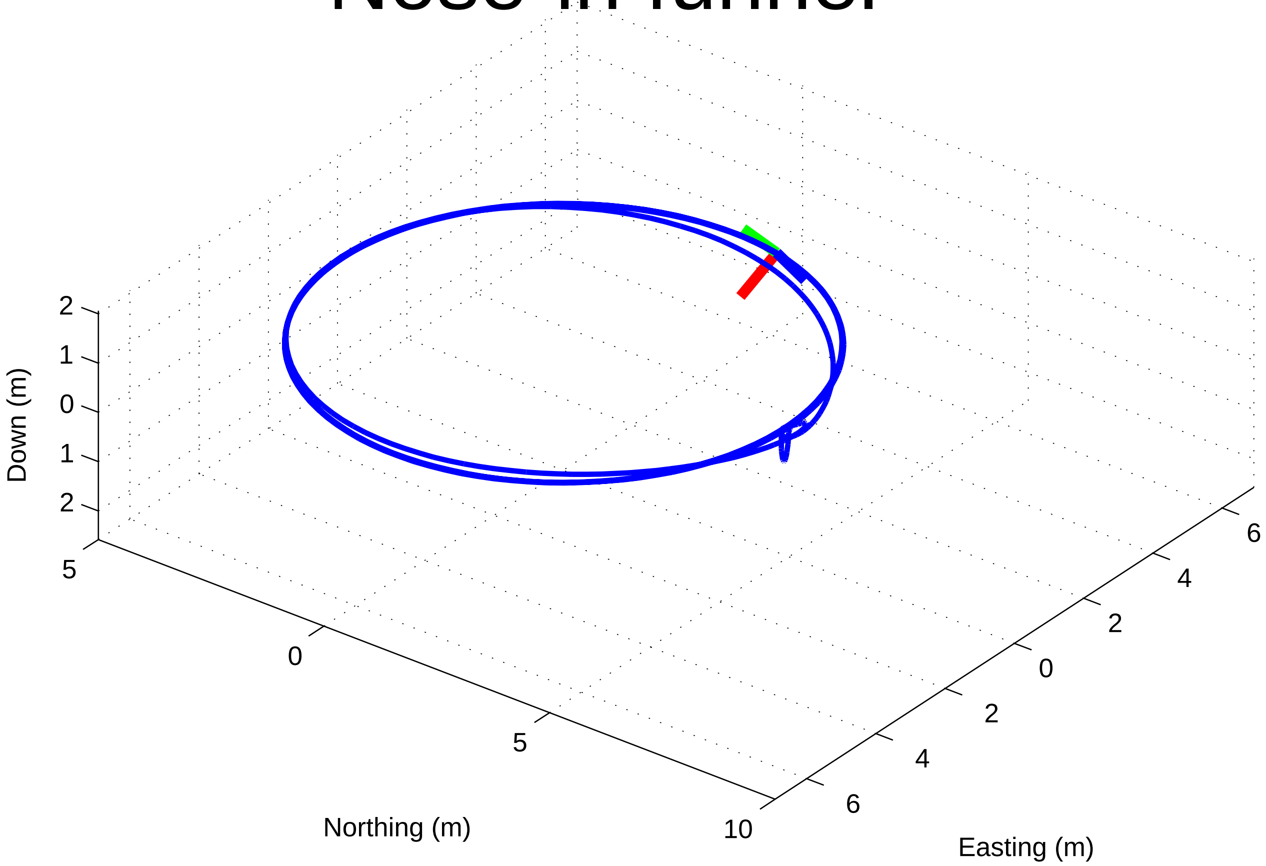


Trajectory tracking for stationary rolls?

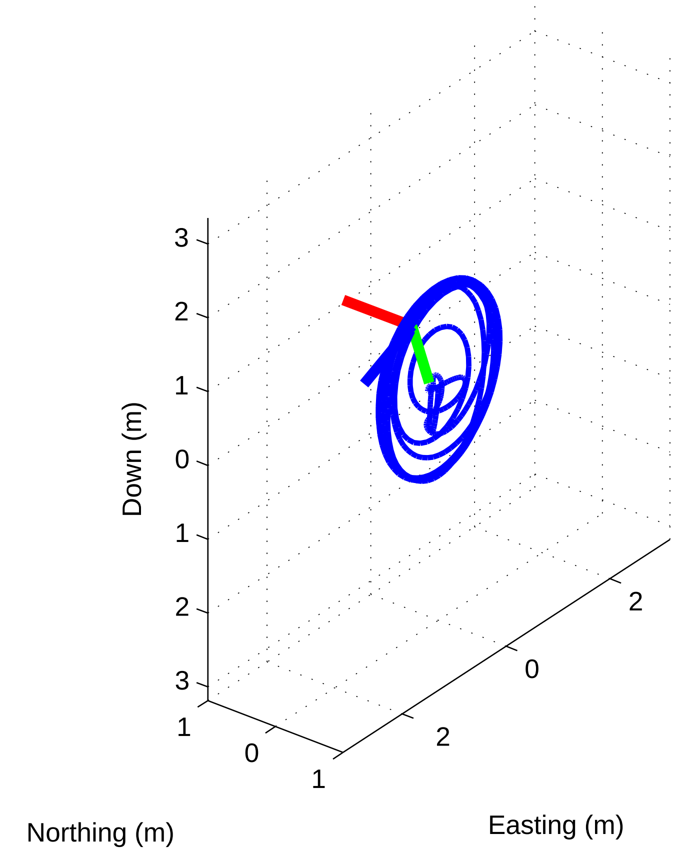


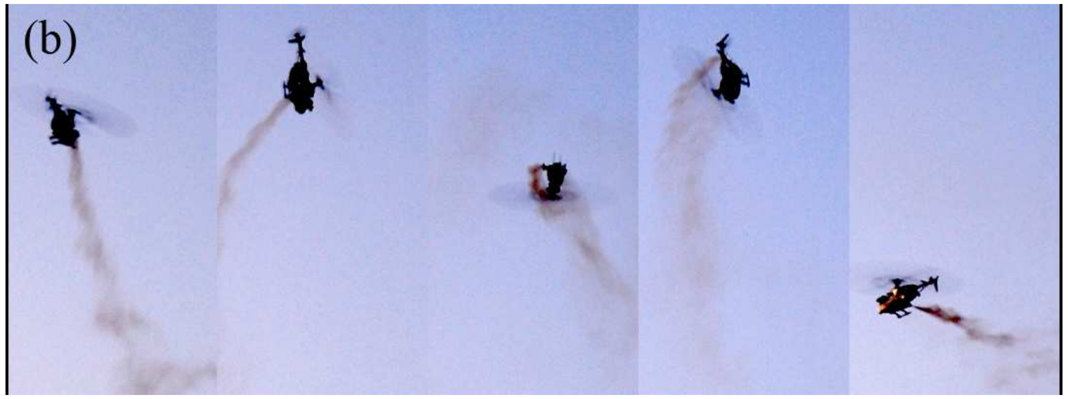
How do we get such behaviors?

Nose-in funnel



Stationary rolls





Task: Minimize tracking error

$$\min_{u_0, u_1, \ldots, u_{T-1}} \sum_{t=0}^{T-1} c(x_t, u_t)$$

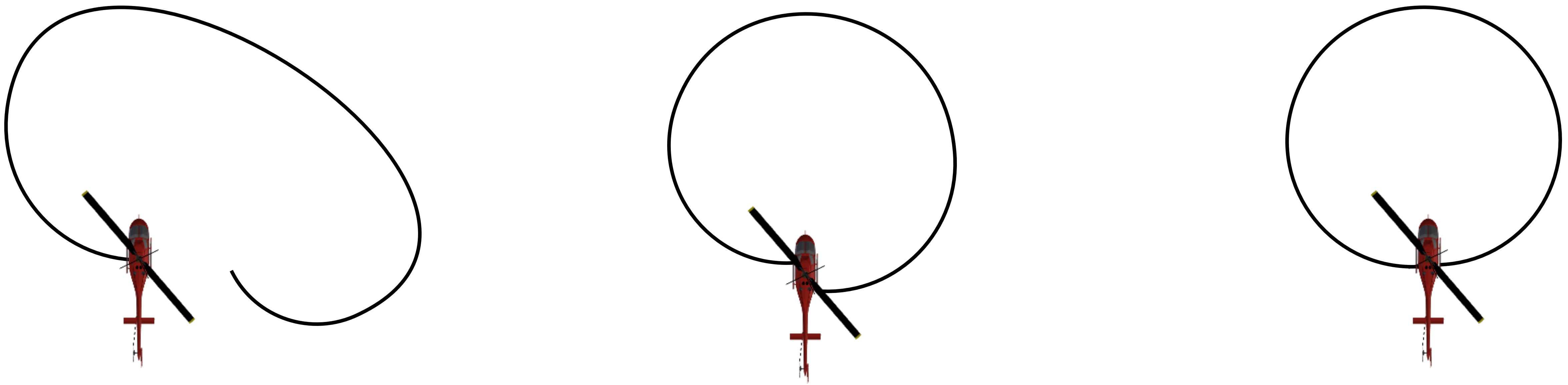
$$\text{subject to } x_{t+1} = f(x_t, u_t) \; \forall t$$

In this scenario, cost is simply a quadratic tracking cost

Why is this a hard optimization problem?

Iterative LQR (iLQR)

Start by guessing a control sequence, Forward simulate dynamics, Linearize about trajectory, Solve for new control sequence and repeat!



i=0 i=10 i=100

Unicycle

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + v_t \cos(\theta_t)\, dt \\ y_t + v_t \sin(\theta_t)\, dt \\ \theta_t + \omega_t\, dt \end{bmatrix}$$

$$\underbrace{\qquad}_{\underline{s}_{t+1}} = \underbrace{\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix}}_{\underline{s}_t} + \underbrace{\begin{bmatrix} \boxed{\cos(\theta_t)}\, dt & 0 \\ \boxed{\sin(\theta_t)}\, dt & 0 \\ 0 & dt \end{bmatrix}}_{B(\underline{s}_t)} \underbrace{\begin{bmatrix} v_t \\ \omega_t \end{bmatrix}}_{\underline{u}_t}$$

$$s_{t+1} = A\,\underline{s}_t + B\,\underline{u}_t$$

System is Linear

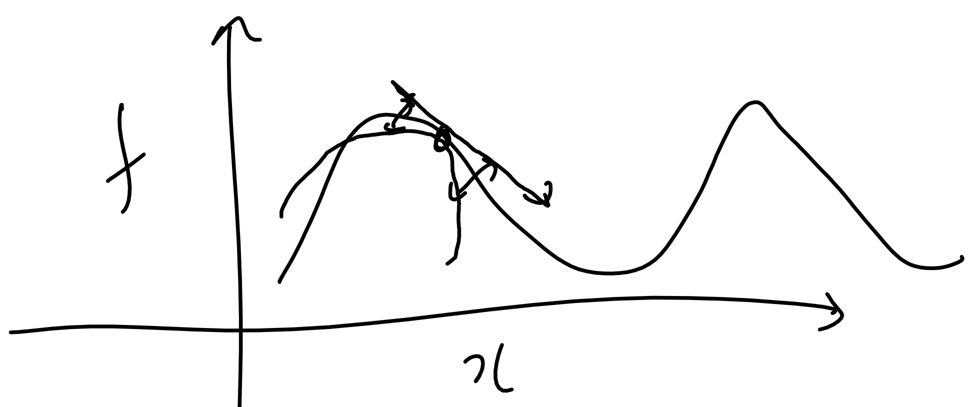
A and B must be constants or independent of $\underline{s}_t$ and $\underline{u}_t$

Non-linear system dynamics

$$\vec{s}_{t+1} = f(\underline{s}_t, \underline{u}_t)$$

If you know how to solve a linear problem, then to convert a non-linear problem in to the linear one, take TAYLOR SERIES approximation

Iterative → iLQR solves the optimal control problem by approximating cost function as Quadratic
system dynamics as Linear or affine

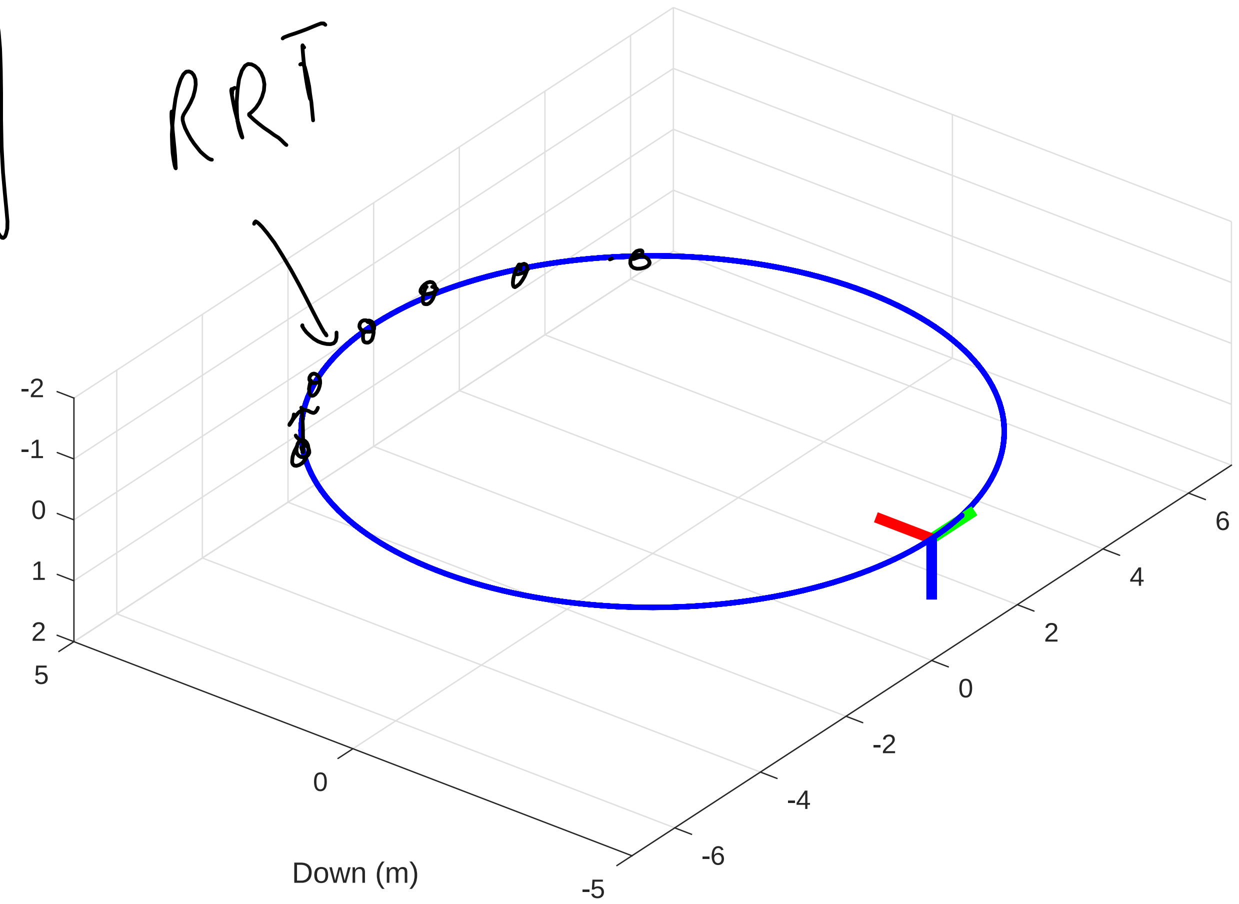


Step 1: Get a reference trajectory

$$x_0^{ref}, u_0^{ref}, x_1^{ref}, u_1^{ref}, \ldots, x_{T\;1}^{ref}, u_{T\;1}^{ref}$$

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

RRT



Note: Simply executing open loop trajectory wont work!

Step 2: Initialize your algorithm

get from some other controller like PID

Choose initial trajectory at iteration 0 to linearize about

TAYLOR series

$$x^0(t), u^0(t) = \{x^0_0, u^0_0, x^0_1, u^0_1, \ldots, x^0_{T-1}, u^0_{T-1}\}$$

Step 2: Initialize your algorithm

Choose initial trajectory at iteration 0 to linearize about

$$x^0(t), u^0(t) = \{x_0^0, u_0^0, x_1^0, u_1^0, \ldots, x_{T\ 1}^0, u_{T\ 1}^0\}$$

It’s a good idea to choose the reference trajectory

Step 2: Initialize your algorithm

Choose initial trajectory at iteration 0 to linearize about

$$x^0(t), u^0(t) = \{x_0^0, u_0^0, x_1^0, u_1^0, \ldots, x_{T-1}^0, u_{T-1}^0\}$$

It's a good idea to choose the reference trajectory

Initialization is very important!

We will be perturbing this initial trajectory

Step 3: Linearize your dynamics!

At a given iteration i, we are going to linearize about

$$x_0^i, u_0^i, x_1^i, \ldots$$

Step 3: Linearize your dynamics!

At a given iteration i, we are going to linearize about

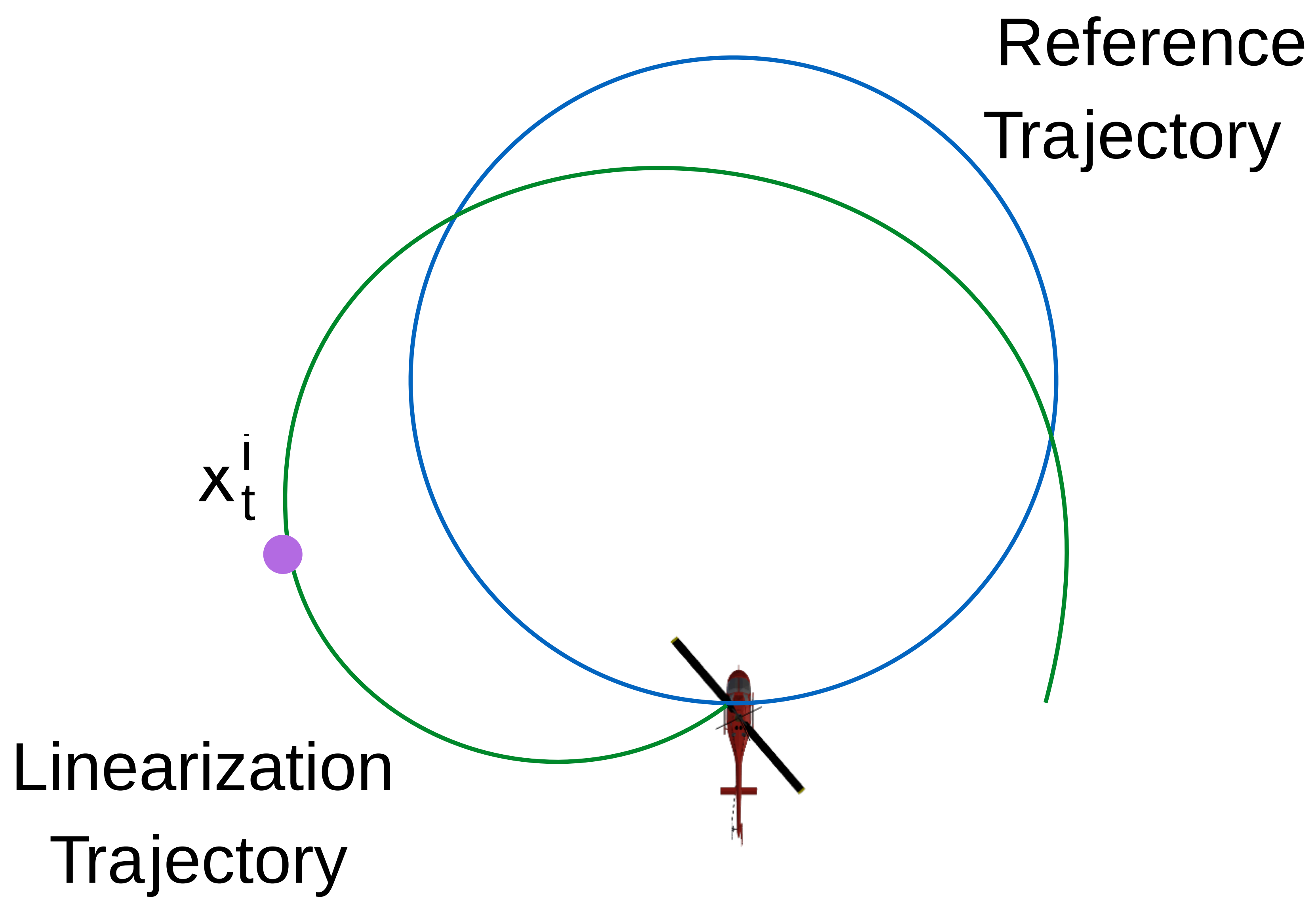
$$x_0^i, u_0^i, x_1^i, \ldots$$

Change of variable - we will track the delta perturbations

$$x_t = x_t \quad x_t^i$$

$$u_t = u_t \quad u_t^i$$

Step 3: Linearize your dynamics!



Step 3: Linearize your dynamics!

$x_t = x_t \quad x^i_t$ $\qquad$ $u_t = u_t \quad u^i_t$

Step 3: Linearize your dynamics!

$i=0$ $\qquad\qquad\qquad\qquad$ $i=0$

$$x_t = x_t \quad x_t^i \qquad\qquad u_t = u_t \quad u_t^i$$

TAYLOR series for vector-valued vector function

$$x^i_{t+1} + \delta x_{t+1} = f(x^i_t + \delta x_t,\ u^i_t + \delta u_t) \approx f(x^i_t, u^i_t) + \underbrace{\boxed{\frac{\partial f}{\partial x}}\Bigg|_{x^i_t, u^i_t}}_{A_t} \delta x_t + \underbrace{\boxed{\frac{\partial f}{\partial u}}\Bigg|_{x^i_t, u^i_t}}_{B_t} \delta u_t$$

$$x_{t+1} = A_t\ x_t + B_t\ u_t + (f(x_t^i, u_t^i) \quad x_{t+1}^i),$$

$$A_t = \frac{@f}{@x}\bigg|_{x_t^i} \qquad\qquad B_t = \frac{@f}{@u}\bigg|_{u_t^i}$$

Jacobian $\qquad\qquad$ Jacobian

Step 3: Linearize your dynamics!

$$x_t = x_t \quad x_t^i \qquad\qquad u_t = u_t \quad u_t^i$$

Affine function

Linear

This is an a ne system, not linear

$$x_{t+1} = A_t x_t + B_t u_t + (f(x_t^i, u_t^i) \quad x_{t+1}^i)$$

constant

$$A_t = \frac{@f}{@x}_{x_t^i} \qquad\qquad B_t = \frac{@f}{@u}_{u_t^i}$$

$$\underbrace{\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix}}_{\underline{s}_{t+1}} = \underbrace{\begin{bmatrix} x_t + v_t \cos(\theta_t)\,dt \\ y_t + v_t \sin(\theta_t)\,dt \\ \theta_t + \omega_t\,dt \end{bmatrix}}_{\underline{f}(\underline{s}_t,\, u_t) \in \mathbb{R}^{3\times 1}}$$

Jacobian

$$\frac{\partial f}{\partial s} = \begin{bmatrix} \frac{\partial f}{\partial s_1} & \frac{\partial f}{\partial s_2} & \cdots & \frac{\partial f}{\partial s_3} \\ \frac{\partial f_2}{\partial s_1} & & & \\ \vdots & & & \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & -v_t \sin(\theta_t)\,dt \\ 0 & 1 & v_t \cos(\theta_t)\,dt \\ 0 & 0 & dt \end{bmatrix}}_{A_t}$$

$$\frac{\partial f}{\partial u_t} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}_{3\times 2}$$

Step 3: Linearize your dynamics!

$$\boxed{x_w = R x_o + t}$$

Linear

Affine

$$\begin{bmatrix} x_w \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ 1 \end{bmatrix}$$

Homogenous coordinate system $\begin{bmatrix} x_t \\ 1 \end{bmatrix}$

A ne dynamics is now linear!

$$\delta x_{t+1} = A_{t+1} \delta x_t + B_{t+1} \delta u_t + f(x_t^i, u_t) - x_{t+1}^i$$

$$\begin{bmatrix} x_{t+1} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} A_{t+1} & f(x_t^i, u_t^i) - x_{t+1}^i \\ 0 & 1 \end{bmatrix}}_{\tilde{A}_t} \begin{bmatrix} x_t \\ 1 \end{bmatrix} + \underbrace{\begin{bmatrix} B_{t+1} \\ 0 \end{bmatrix}}_{\tilde{B}_t} u_t$$

Step 4: Quadricize cost about trajector

Our cost function is already quadratic, otherwise we would apply Taylor expansion

$$c(x_t, u_t) = (\,x_t \quad \underbrace{x_t^{ref}}_{x_g}\,)^T \underbrace{Q}_{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}(x_t \quad \underbrace{x_t^{ref}}_{x_g}\,) + (\,u_t \quad u_t^{ref}\,)^T R(u_t \quad u_t^{ref}\,)$$

$$\|x_t - x_g\|^2$$

Step 4: Quadricize cost about trajectory

Our cost function is already quadratic, otherwise we would apply Taylor expansion

$x_t^i + \delta x_t$

$$c(x_t, u_t) = (x_t \quad x_t^{ref})^T Q (x_t \quad x_t^{ref}) + (u_t \quad u_t^{ref})^T R (u_t \quad u_t^{ref})$$

$$- \begin{matrix} x_t \\ 1 \end{matrix}^T \underbrace{\begin{matrix} Q & Q(x_t^i \quad x_t^{ref}) \\ (x_t^i \quad x_t^{ref})^T Q & (x_t^i \quad x_t^{ref})^T (x_t^i \quad x_t^{ref}) \end{matrix}}_{\breve{Q}_t} \begin{matrix} x_t \\ 1 \end{matrix}$$

$$+$$

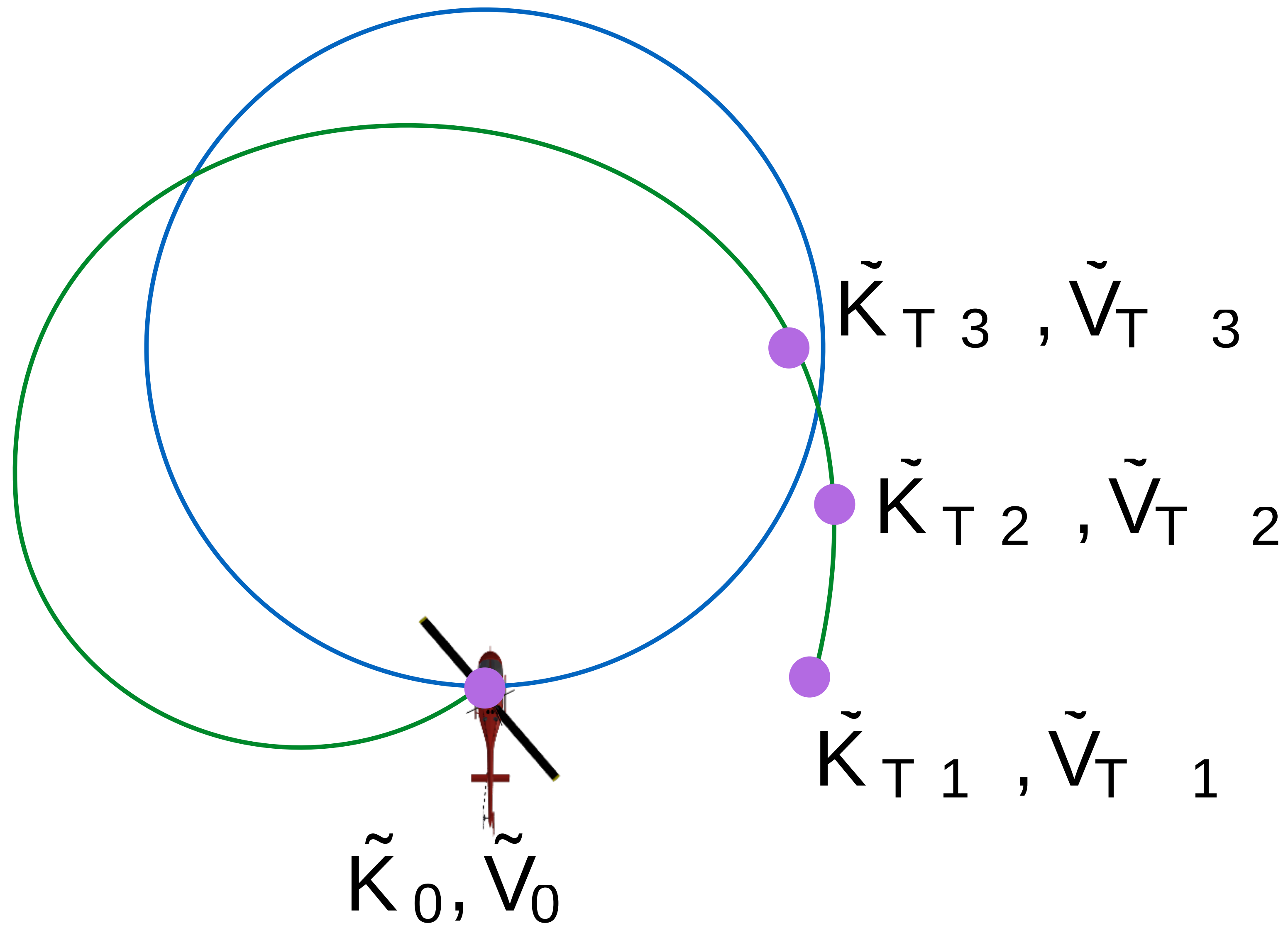
$$\begin{matrix} u_t \\ 1 \end{matrix}^T \underbrace{\begin{matrix} R & R(u_t^i \quad u_t^{ref}) \\ (u_t^i \quad u_t^{ref})^T R & (u_t^i \quad u_t^{ref})^T (u_t^i \quad u_t^{ref}) \end{matrix}}_{\breve{R}_t} \begin{matrix} u_t \\ 1 \end{matrix}$$

We have all the ingredients to call LQR

$$\tilde{K}_t = (\tilde{R}_t + \tilde{B}_t^T \tilde{V}_{t+1} \tilde{B}_t)^{1} \tilde{B}_t^T \tilde{V}_{t+1} \tilde{A}_t$$

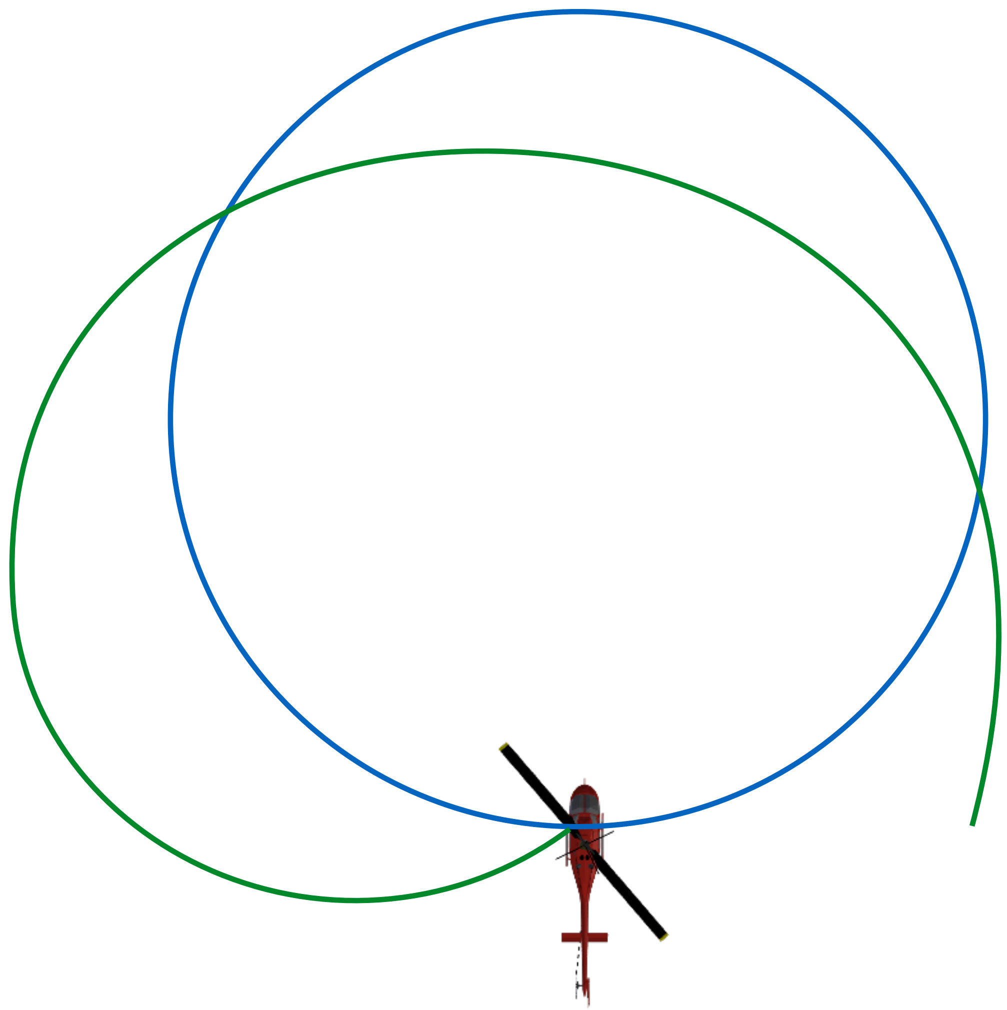
similarly calculate the value function …

Step 5: Do a backward pass



Calculate controller gains for all time steps

Step 6: Do a forward pass to get new trajectory



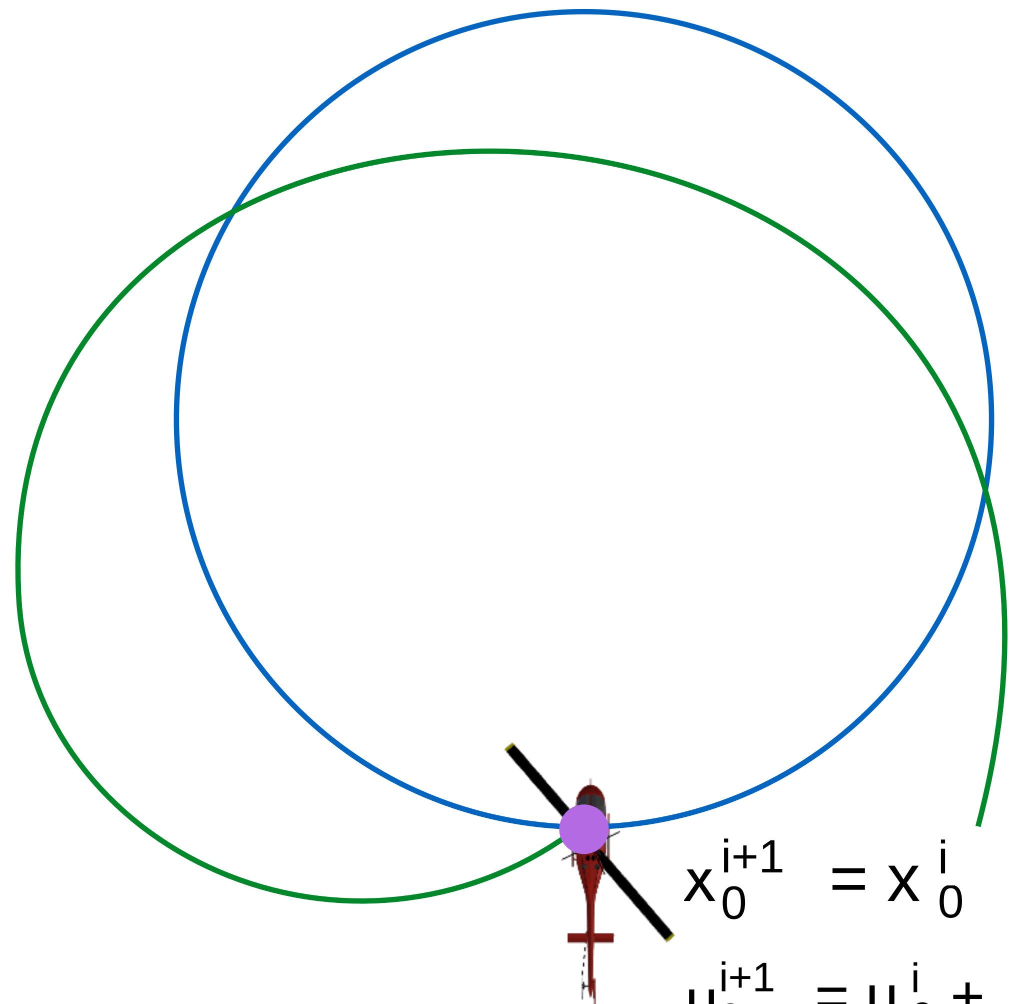
Compute
control action

$$u_t^{i+1} = u_t^i + \tilde{K}_t \quad \begin{matrix} x_t^{i+1} & & x_t^{i} \\ & 1 & \end{matrix}$$

Apply dynamics

$$x_t^{i+1} = f\left(x_t^{i+1}, u_t^{i+1}\right)$$

Step 6: Do a forward pass to get new trajectory



$$x_0^{i+1} = x_0^i$$

$$u_0^{i+1} = u_0^i + \tilde{K}_t \begin{matrix} 0 \\ 1 \end{matrix}$$

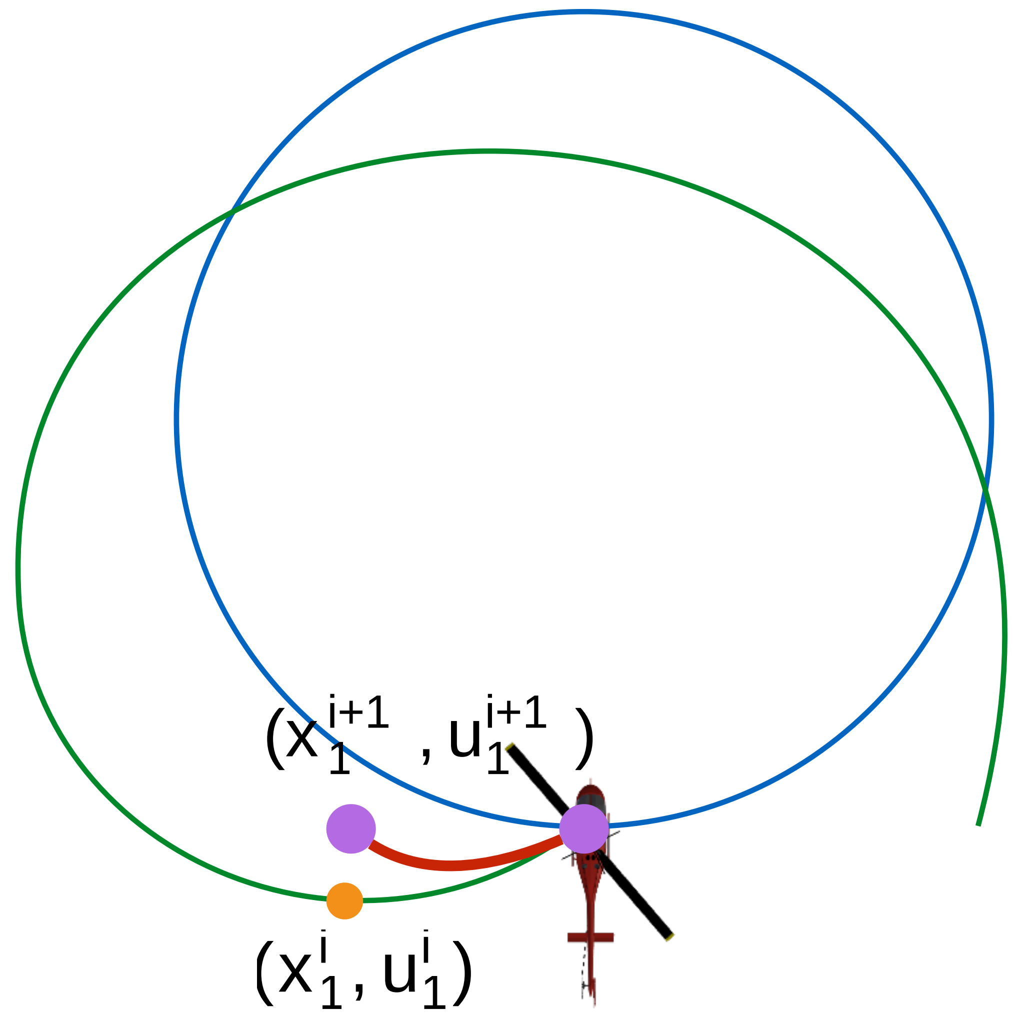
Compute control action

$$u_t^{i+1} = u_t^i + \tilde{K}_t \begin{matrix} x_t^{i+1} \quad x_t^i \\ 1 \end{matrix}$$

Apply dynamics

$$x_t^{i+1} = f\,(x_t^{i+1}, u_t^{i+1})$$

Step 6: Do a forward pass to get new trajectory



Compute

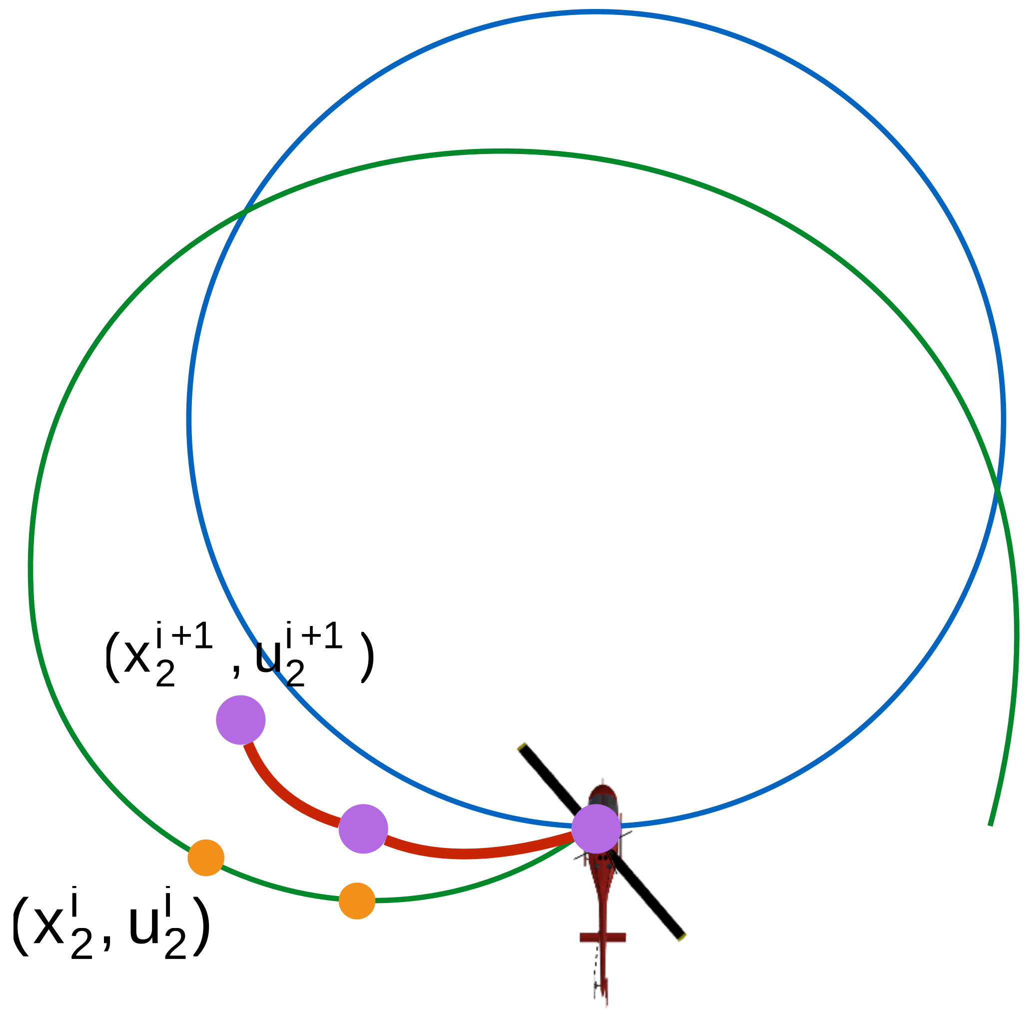
control action

$$u_t^{i+1} = u_t^i + \tilde{K}_t \begin{matrix} x_t^{i+1} & x_t^i \\ 1 & \end{matrix}$$

Apply dynamics

$$x_t^{i+1} = f\left(x_t^{i+1}, u_t^{i+1}\right)$$

Step 6: Do a forward pass to get new trajector



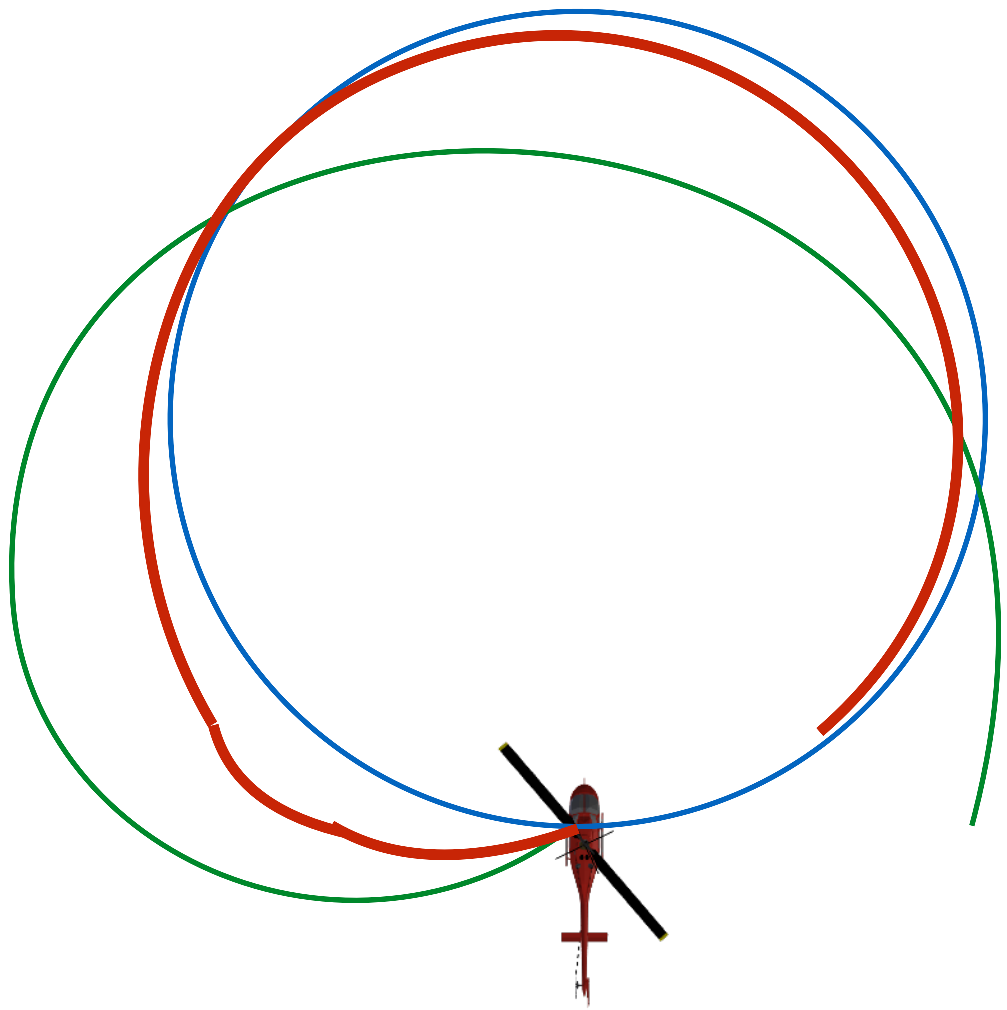
Compute
control action

$$u_t^{i+1} = u_t^i + \tilde{K}_t \begin{matrix} x_t^{i+1} & x_t^i \\ 1 & \end{matrix}$$

Apply dynamics

$$x_t^{i+1} = f\,(x_t^{i+1}, u_t^{i+1})$$

Step 6: Do a forward pass to get new trajector



Compute
control action

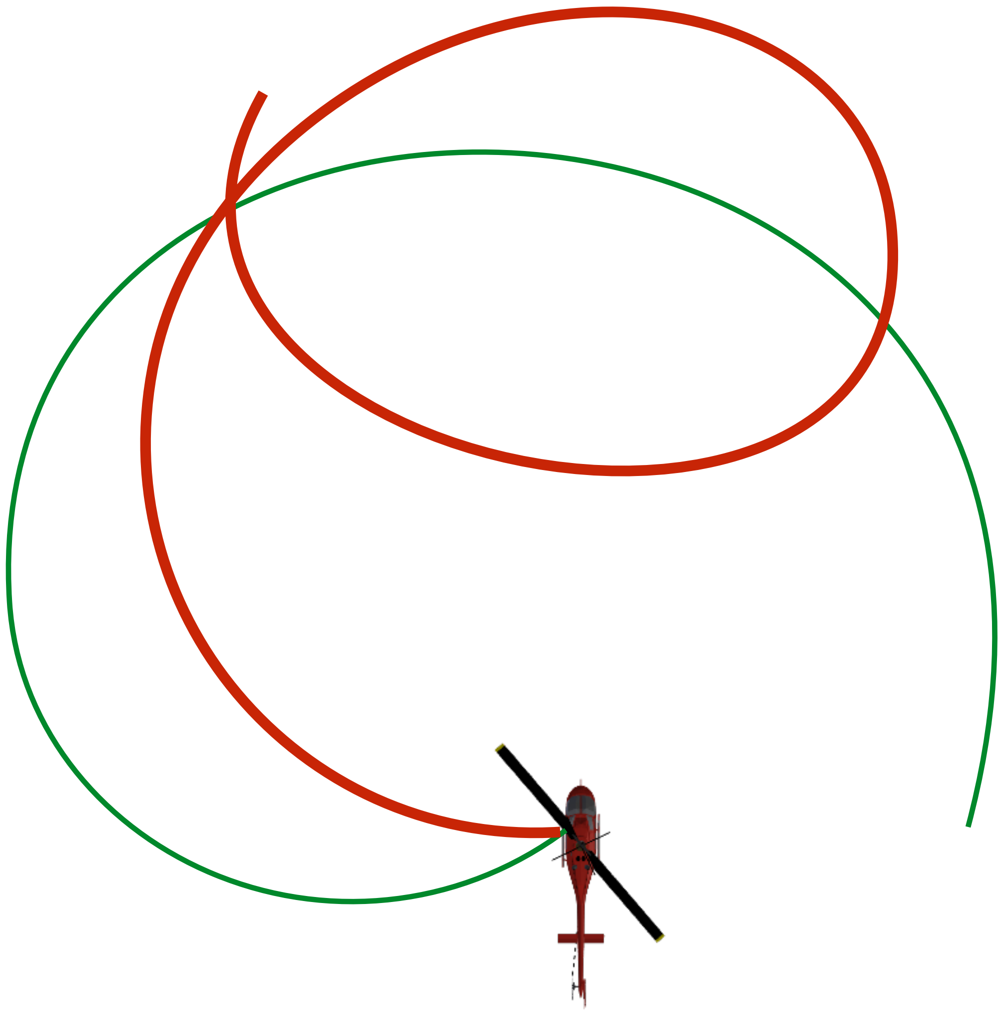
$$u_t^{i+1} = u_t^{i} + \tilde{K}_t \begin{matrix} x_t^{i+1} \quad x_t^{i} \\ 1 \end{matrix}$$

Apply dynamics

$$x_t^{i+1} = f\left(x_t^{i+1}, u_t^{i+1}\right)$$

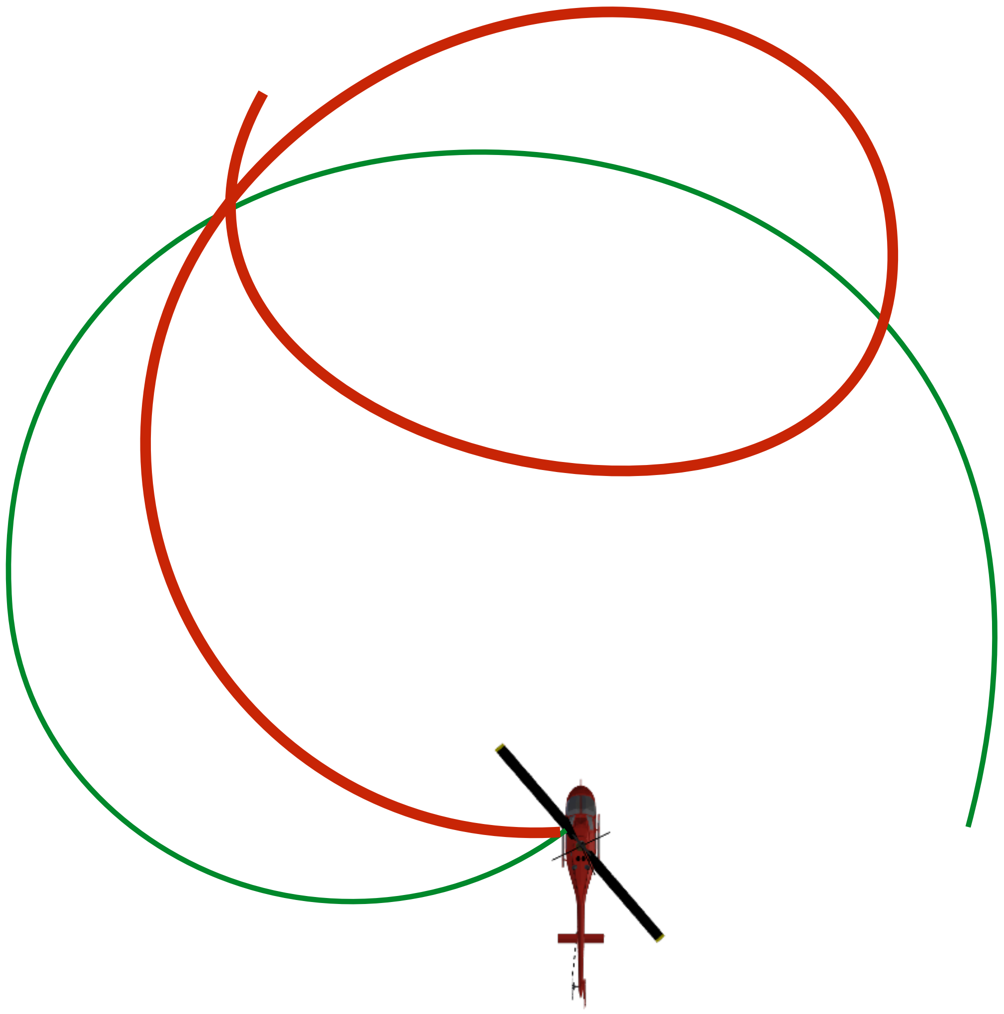
Problem: Forward pass will go bonker

Why?



Problem: Forward pass will go bonker

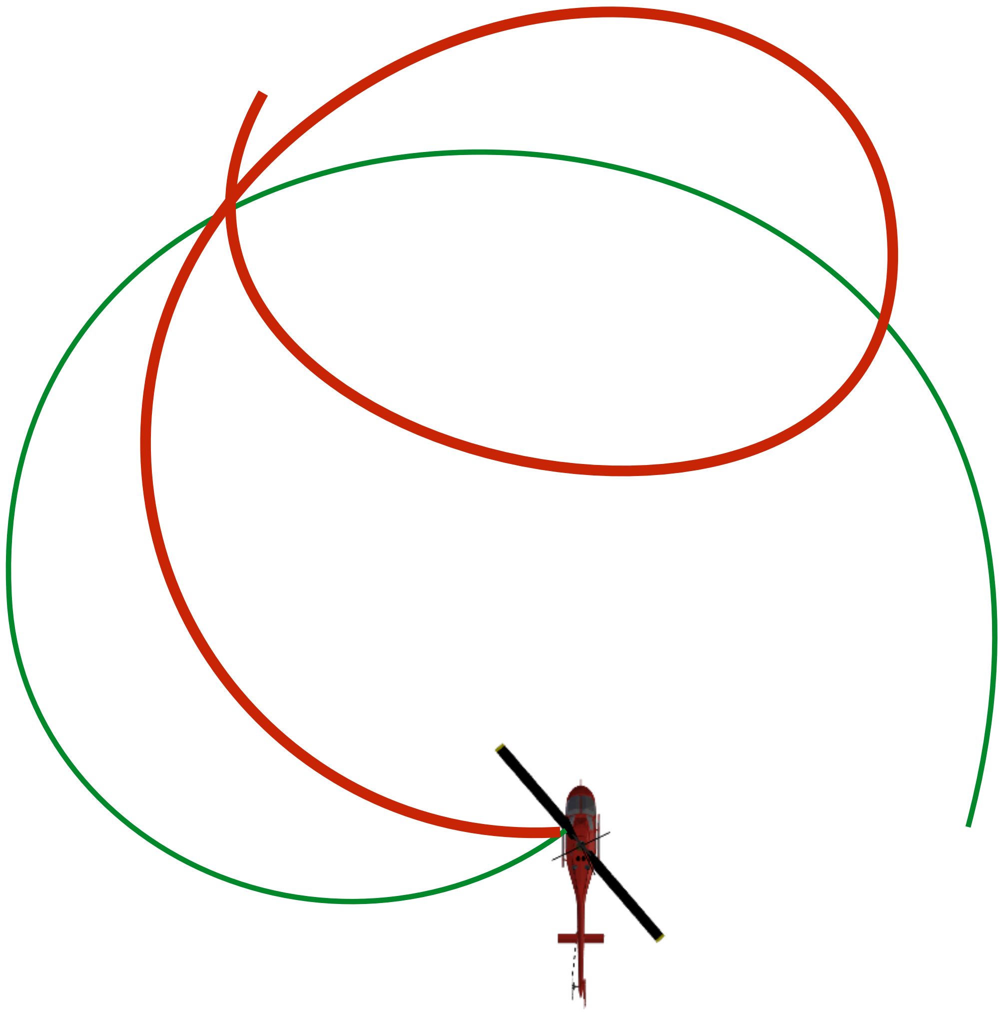
Why?



Linearization error gets bigger and bigger and bigger

Problem: Forward pass will go bonker

Why?



Linearization error gets bigger and bigger and bigger

Remedies: Change cost function to penalize deviation from linearization

Questions

Questions

1. Can we solve LQR for continuous time dynamics?

Questions

1. Can we solve LQR for continuous time dynamics?

   Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

Questions

1. Can we solve LQR for continuous time dynamics?

    Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

Questions

1. Can we solve LQR for continuous time dynamics?

    Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

    Yes! Just quadricize the cost

Questions

1. Can we solve LQR for continuous time dynamics?

Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

Yes! Just quadricize the cost

3. What if I want to penalize control derivatives?

Questions

1. Can we solve LQR for continuous time dynamics?

Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

Yes! Just quadricize the cost

3. What if I want to penalize control derivatives?

No problem! Add control as part of state space

Questions

1. Can we solve LQR for continuous time dynamics?

   Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

   Yes! Just quadricize the cost

3. What if I want to penalize control derivatives?

   No problem! Add control as part of state space

4. Can we handle noisy dynamics?

Questions

1. Can we solve LQR for continuous time dynamics?

Yes! Refer to Continuous Algebraic Ricatti Equations (CARE)

2. Can LQR handle arbitrary costs (not just tracking)?

Yes! Just quadricize the cost

3. What if I want to penalize control derivatives?

No problem! Add control as part of state space

4. Can we handle noisy dynamics?

Yes! Gaussian noise does not change the answer

Table of Controllers

	Control Law	Uses model	Stability Guarantee	Minimize Cost
PID		No	No	No
Pure Pursuit		Circular arcs	Yes - with assumptions	No
Lyapunov		Non-linear	Yes	No
LQR		Linear	Yes	Quadratic
iLQR		Non-linear	Yes	Yes

iLQR is just one technique

It’s far from perfect - can’t deal with model errors / constraints …

Model Predictive Control (MPC)

iLQR

Shooting

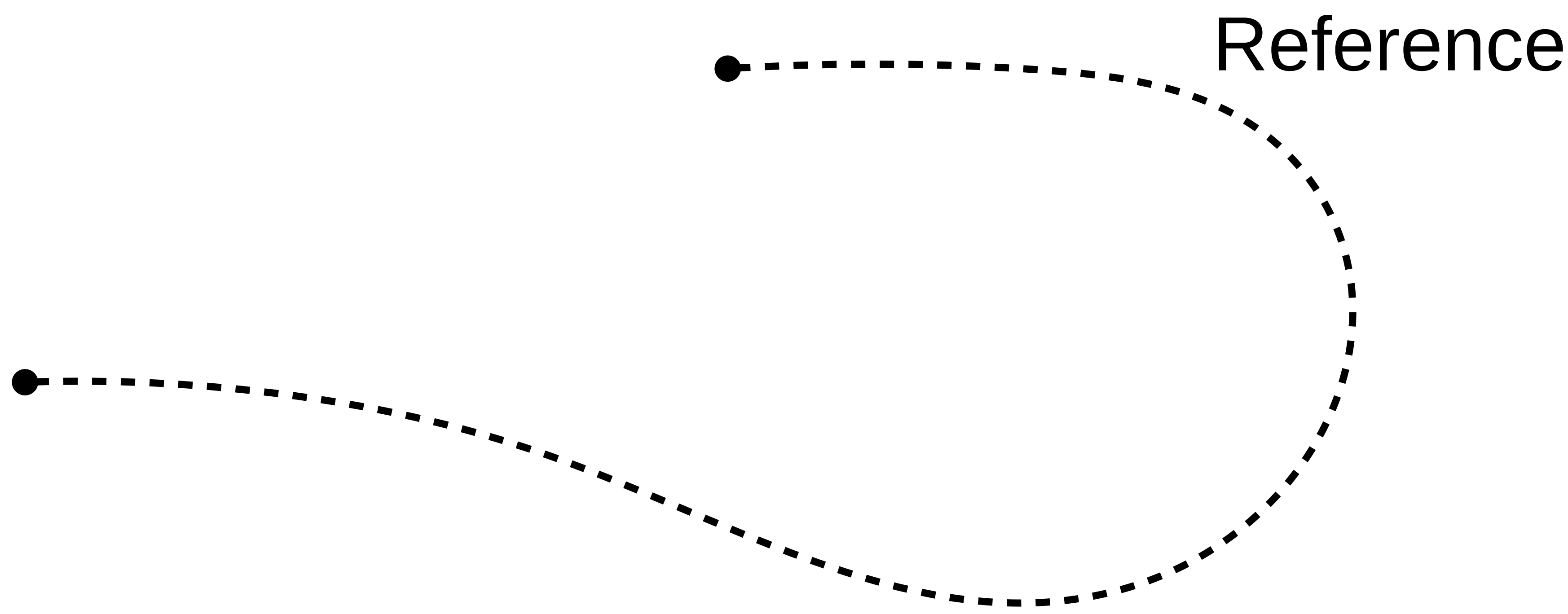
MPPI

TrajLib

DDP

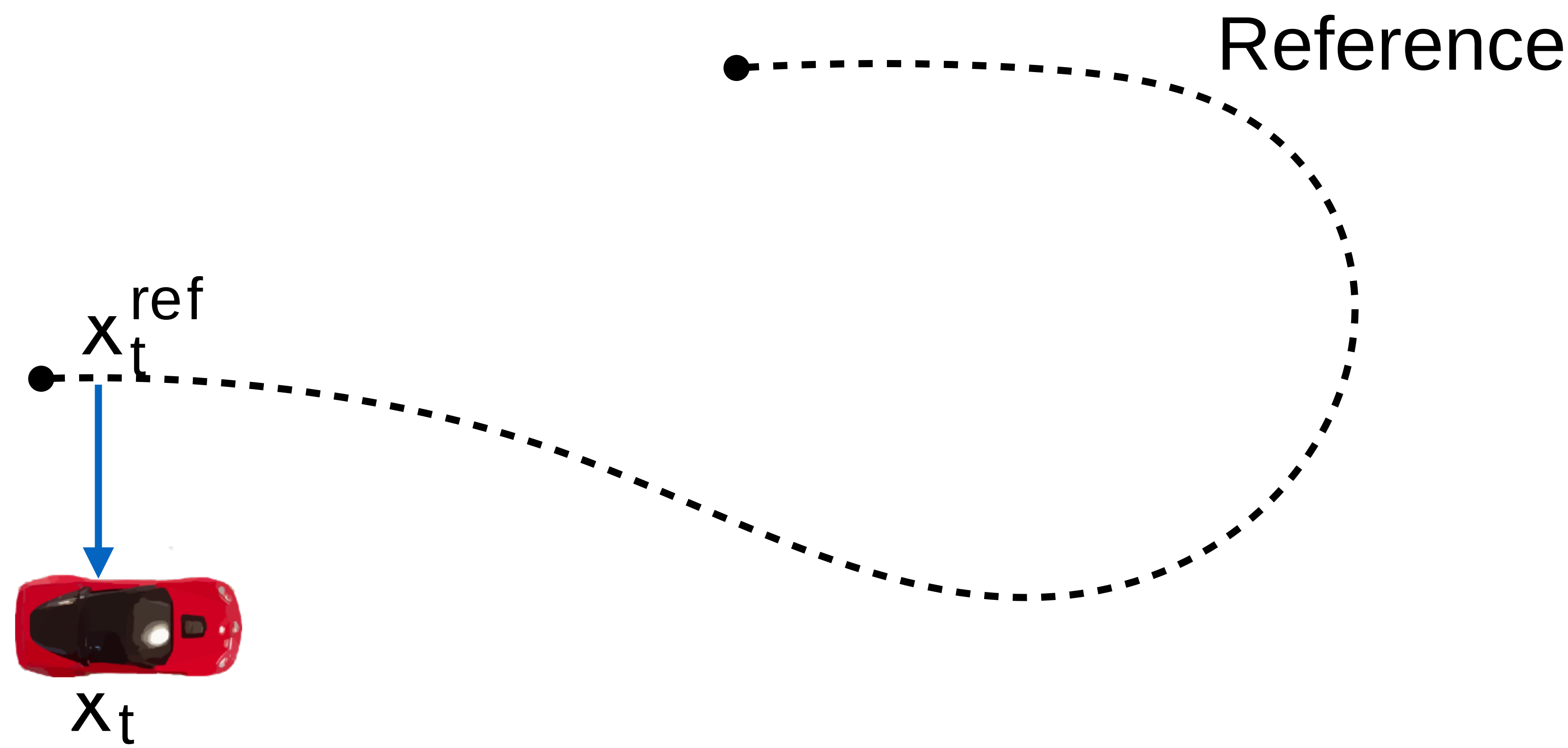
CMAES

Recap: Feedback control framework



Look at current state error and compute control actions

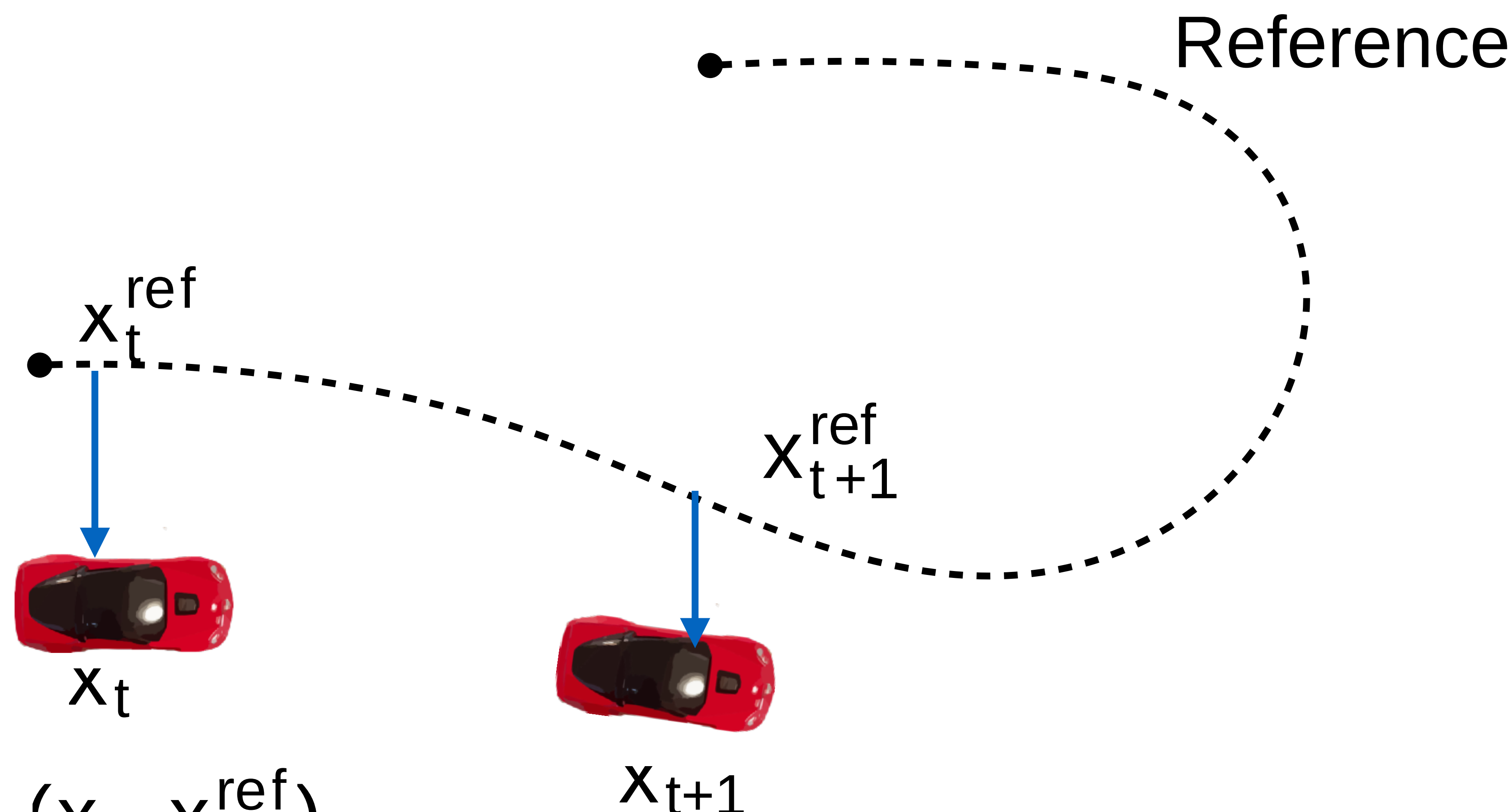
Recap:Feedback control framework



$$u_t = \quad (x_t, x_t^{ref})$$

Look at current state error and compute control actions

Recap: Feedback control framework

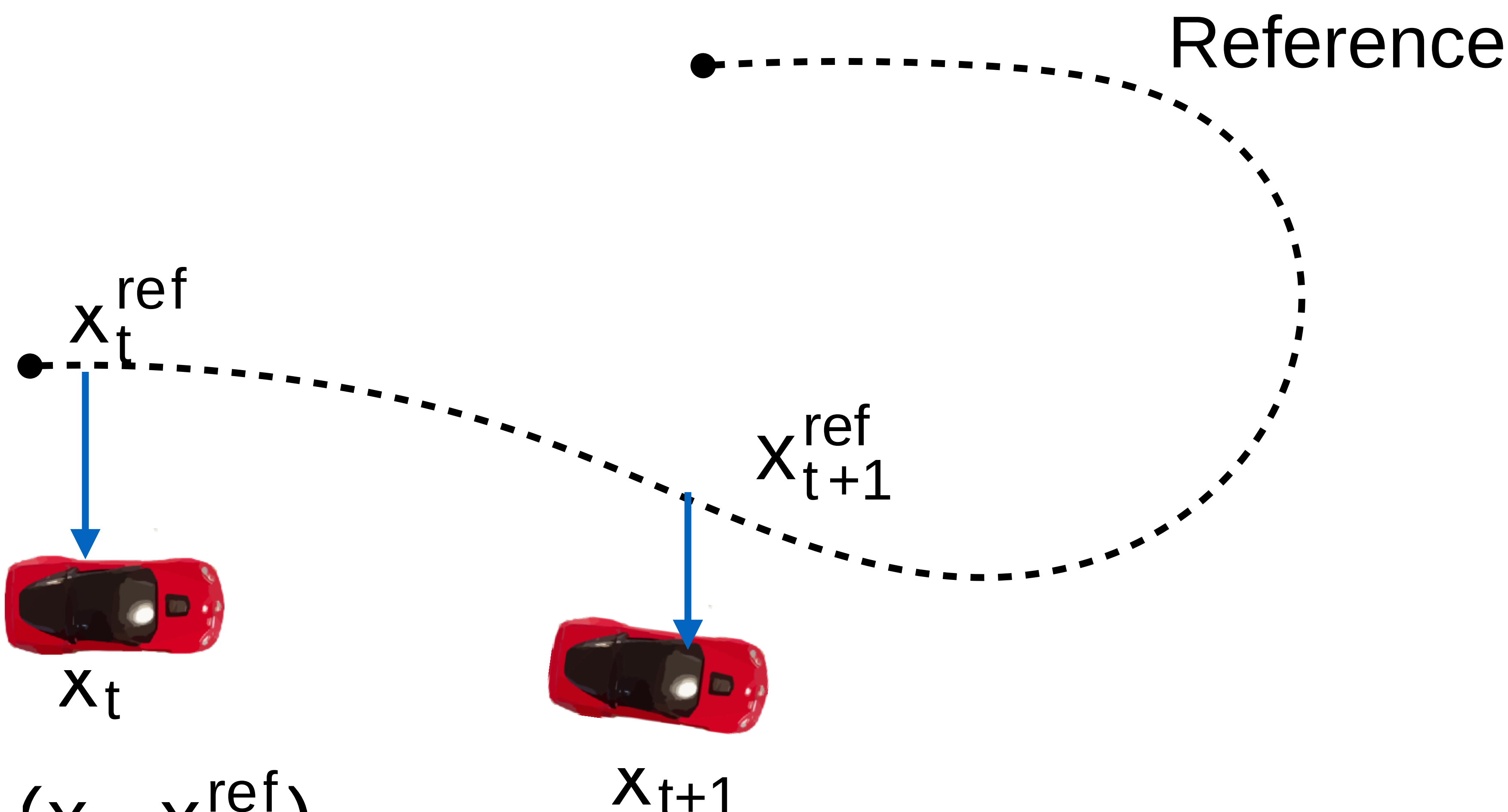


$$u_t = (x_t, x_t^{ref})$$

$$u_{t+1} = (x_{t+1}, x_{t+1}^{ref})$$

Look at current state error and compute control actions

Recap: Feedback control framework



$u_t = \quad (x_t, x_t^{ref})$

$u_{t+1} = (x_{t+1}, x_{t+1}^{ref})$

Look at current state error and compute control actions

Goal: To drive error to 0 … to optimally drive it to 0

Limitations of this framework

A xed control law that looks at instantaneous feedback

$$u_t = \quad (x_t, x_t^{ref})$$

Fixed

Reference

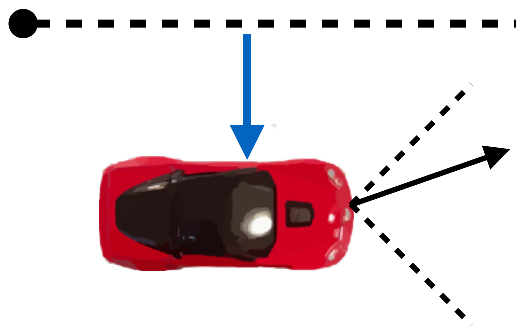
Why is it so di cult to create a magic control law?

Problem 1: What if we have constraints?

Simple scenario: Car tracking a straight line

Problem 1: What if we have constraints?

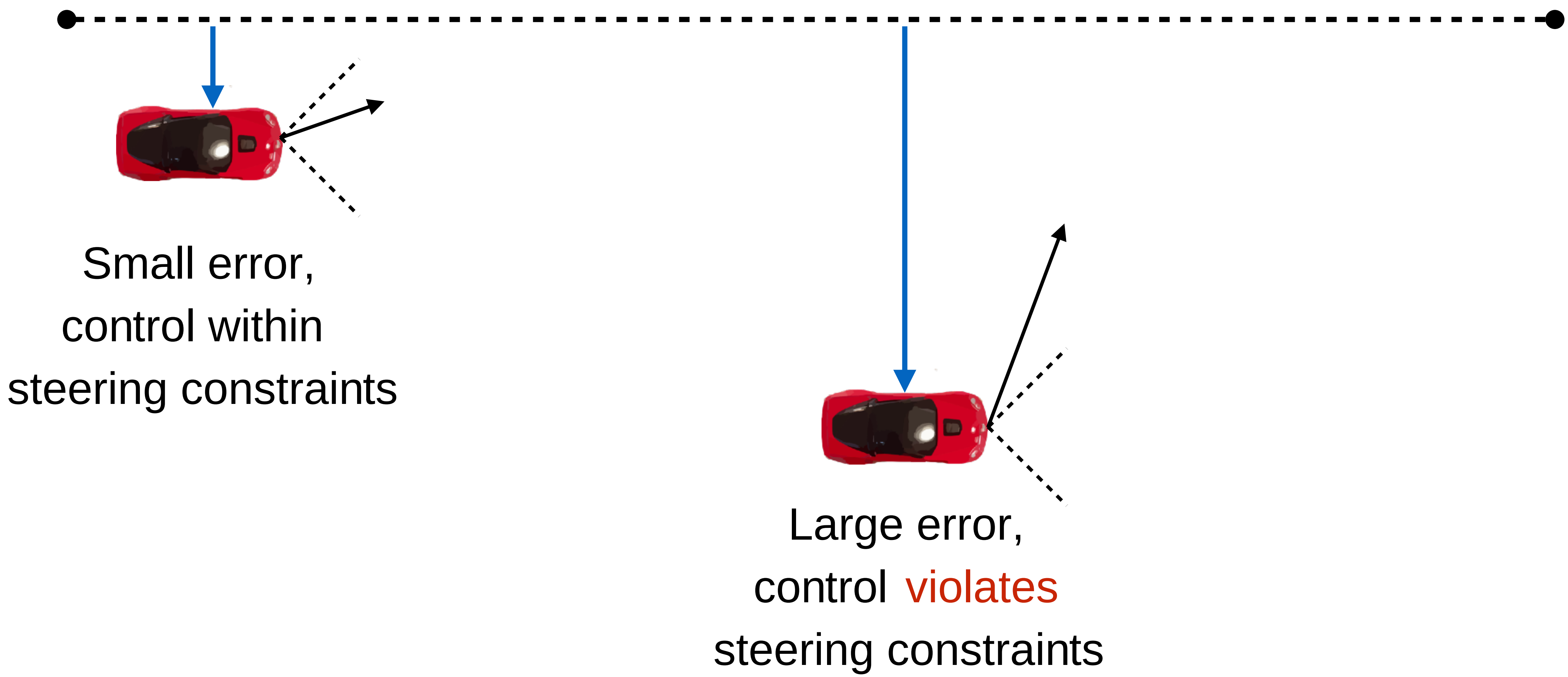
Simple scenario: Car tracking a straight line



Small error,
control within
steering constraints

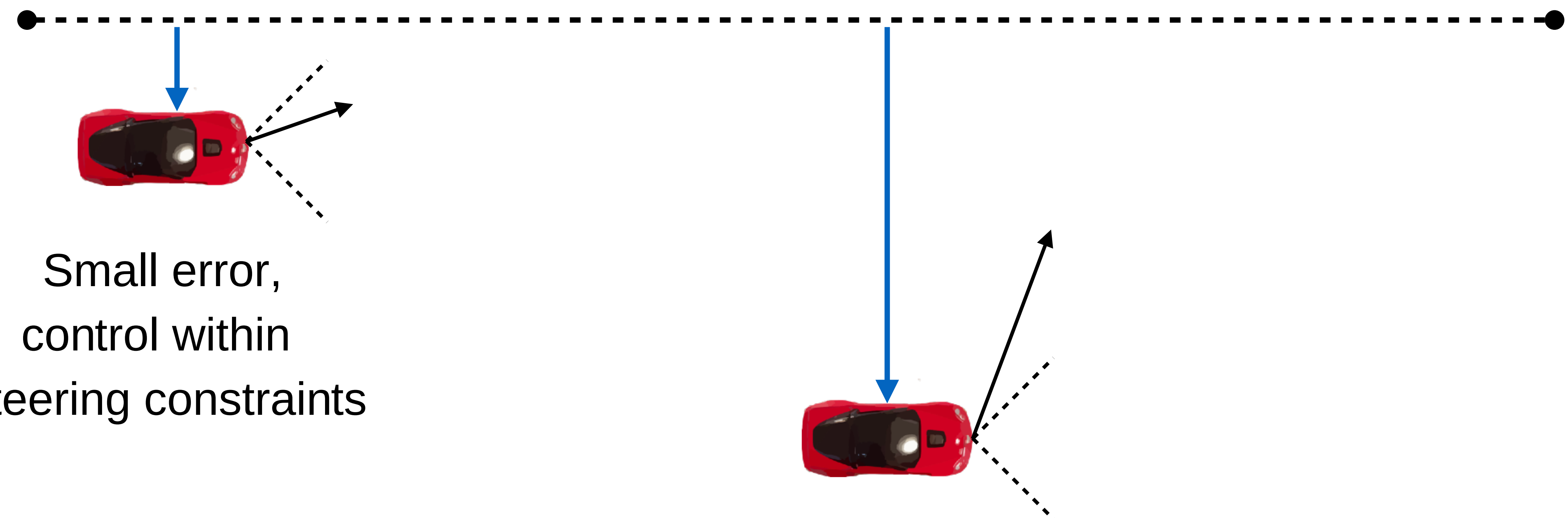
Problem 1: What if we have constraints?

Simple scenario: Car tracking a straight line



Problem 1: What if we have constraints?

Simple scenario: Car tracking a straight line



Small error,
control within
steering constraints

Large error,
control violates
steering constraints

We could "clamp control command" …
but what are the implications?

General problem: Complex models

Dynamics $x_{t+1} = f(x_t, u_t)$

Constraints $g(x_t, u_t) \quad 0$

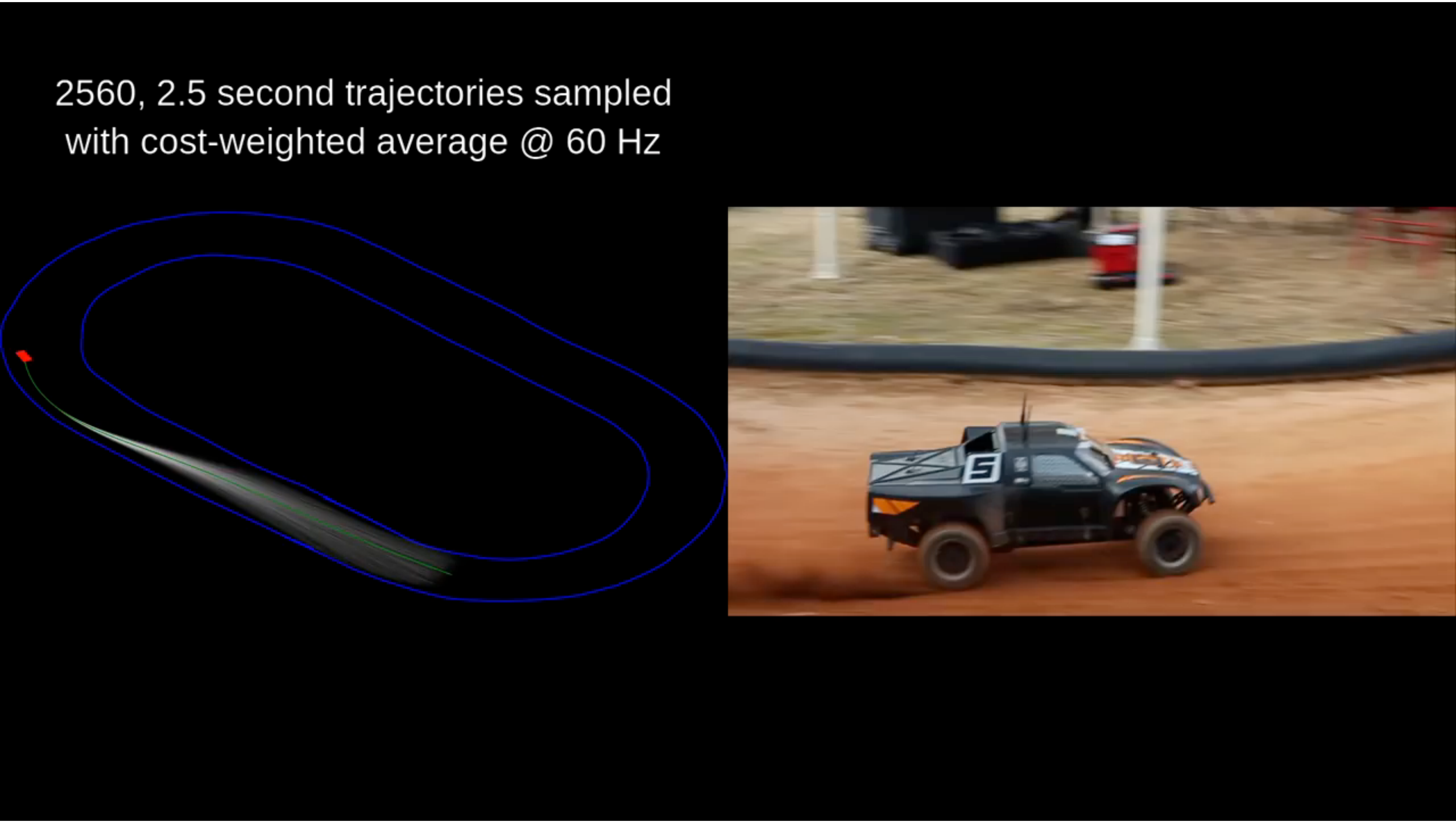
Such complex models imply we need to:

1. Predict the implications of control actions

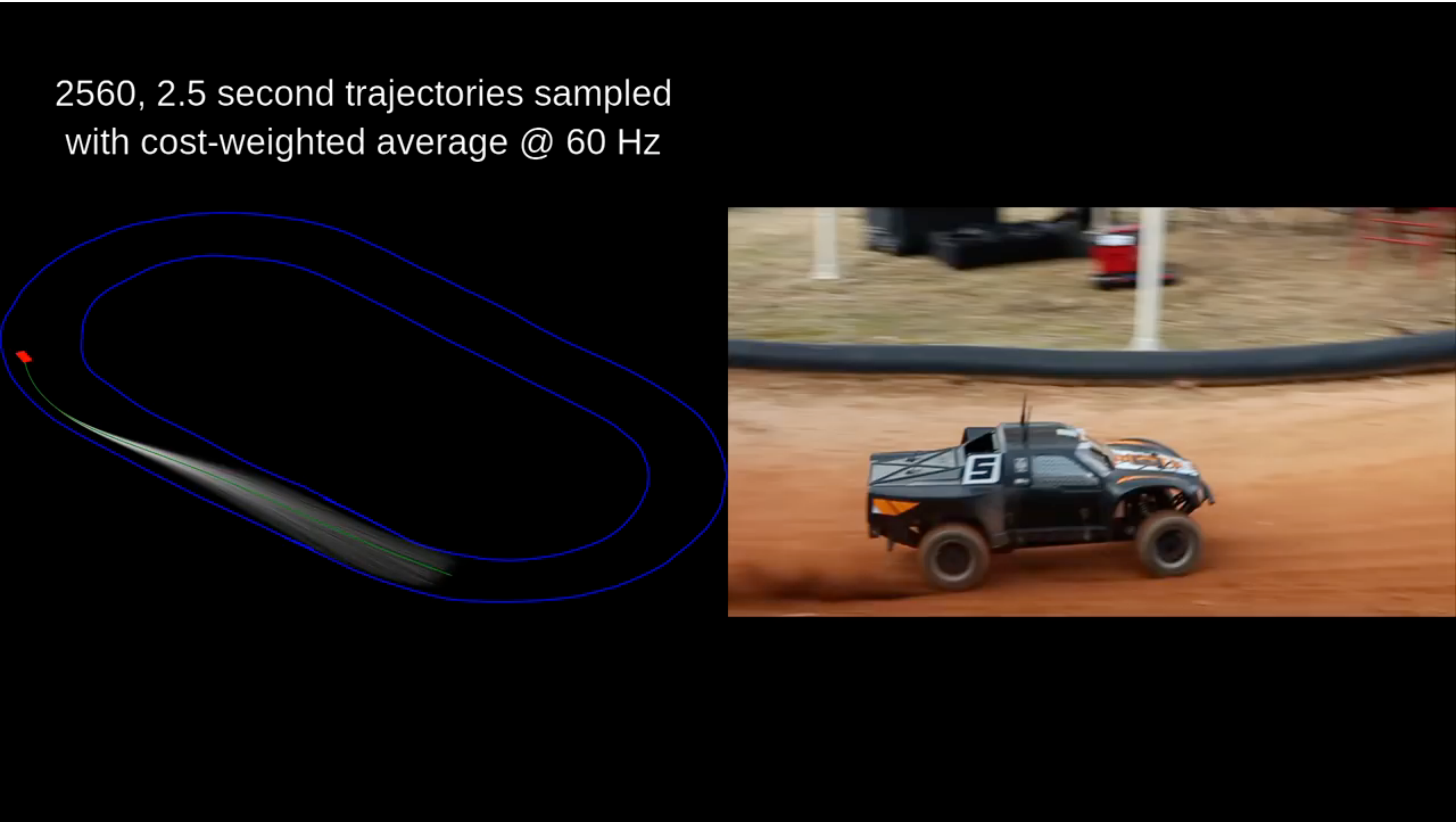
2. Do corrections NOW that would a ect the future

3. It may not be possible to nd one law - might need to predict

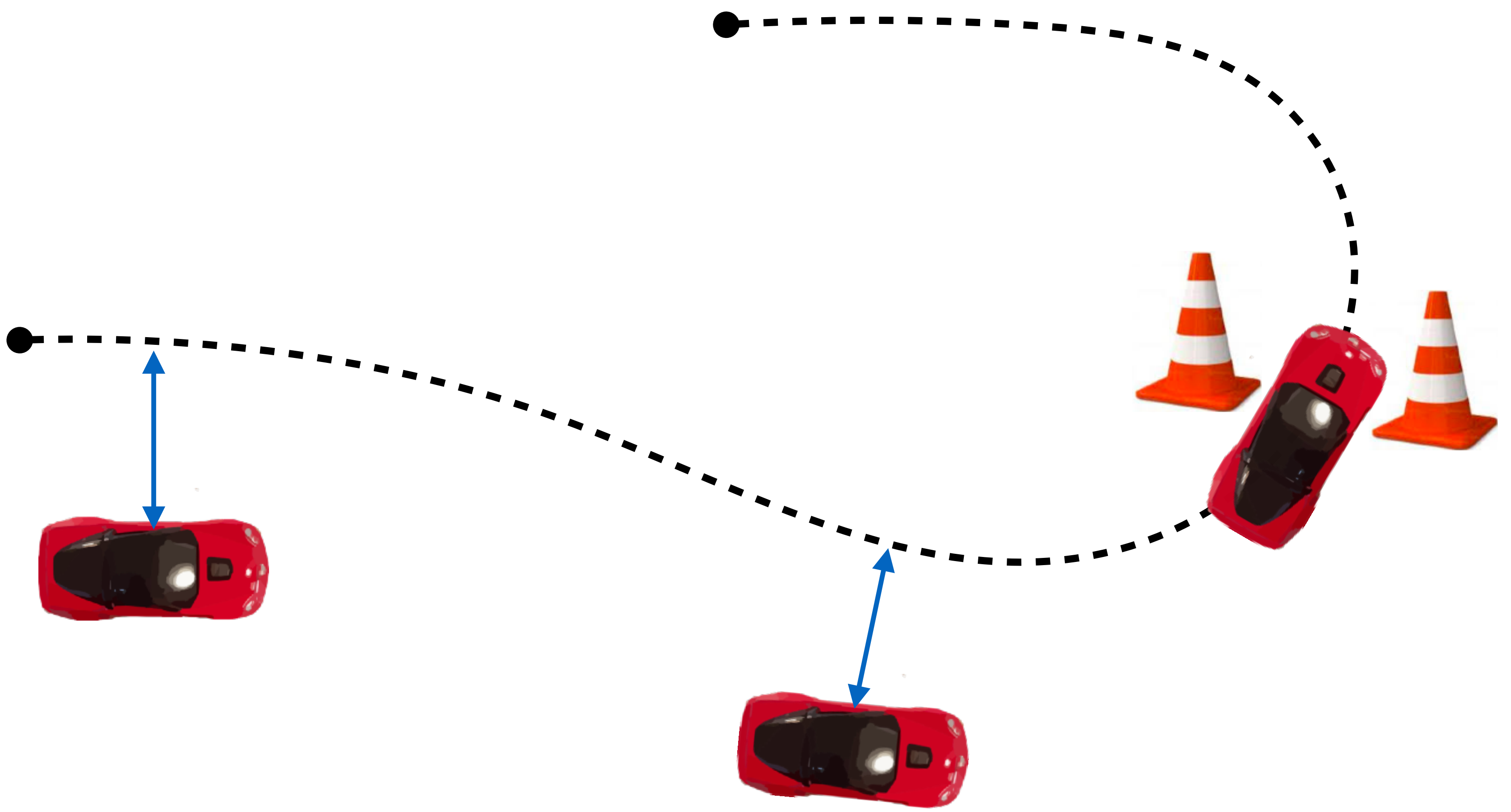
Example:Rough terrain mobility



Example:Rough terrain mobility



Problem 2: What if some errors are worse than others



We need a cost function that penalizes states non-uniformly

Key Idea:

Frame control as an optimization problem

Model predictive control (MPC)

1. Plan a sequence of control actions

2. Predict the set of next states unto a horizon H

3. Evaluate the cost / constraint of the states and controls

4. Optimize the cost

Model predictive control (MPC)

$$\min_{u_{t+1},\ldots,u_{t+H}} \sum_{k=t}^{t+H-1} J(x_k, u_{k+1})$$

(plan till horizon H)

(Cost)

$$x_{k+1} = f(x_k, u_{k+1})$$

(Predict next state with dynamics)

$$g(x_k, u_{k+1}) \quad 0$$

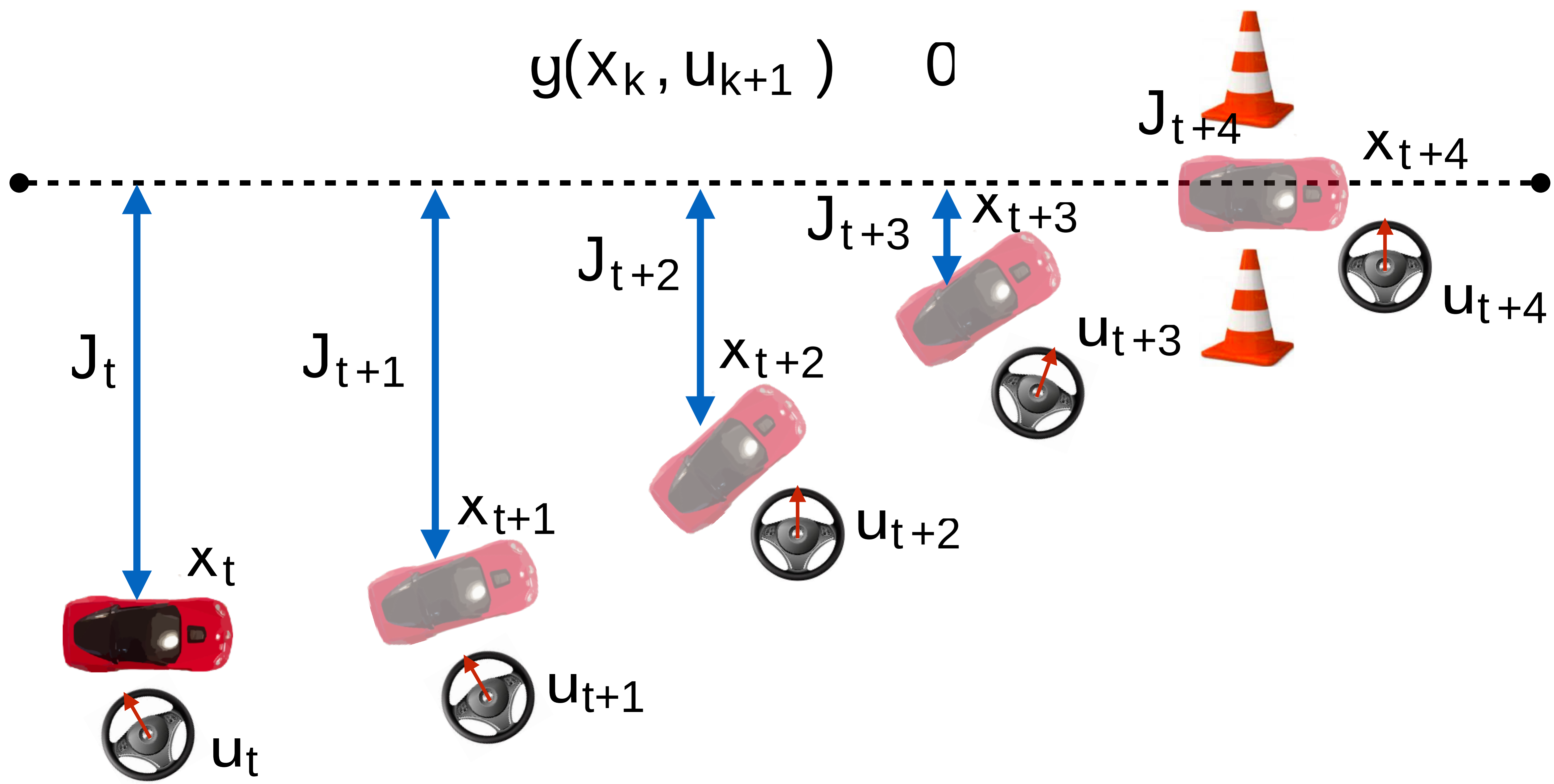
(Constraints)

Model predictive control (MPC)

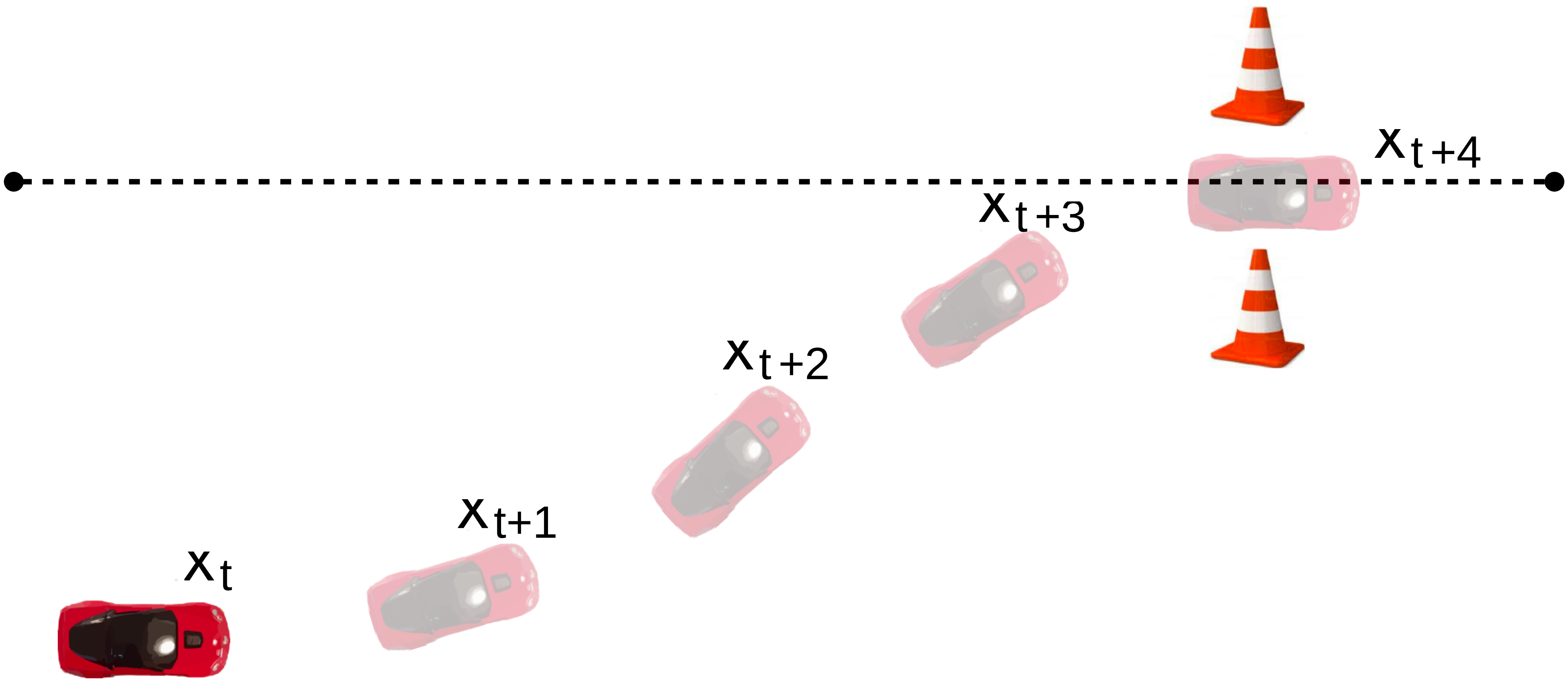
$$\min_{u_{t+1},\ldots,u_{t+H}} \sum_{k=t}^{t+H \quad 1} J(x_k, u_{k+1})$$

$$x_{k+1} = f(x_k, u_{k+1})$$

$$g(x_k, u_{k+1}) \quad 0$$

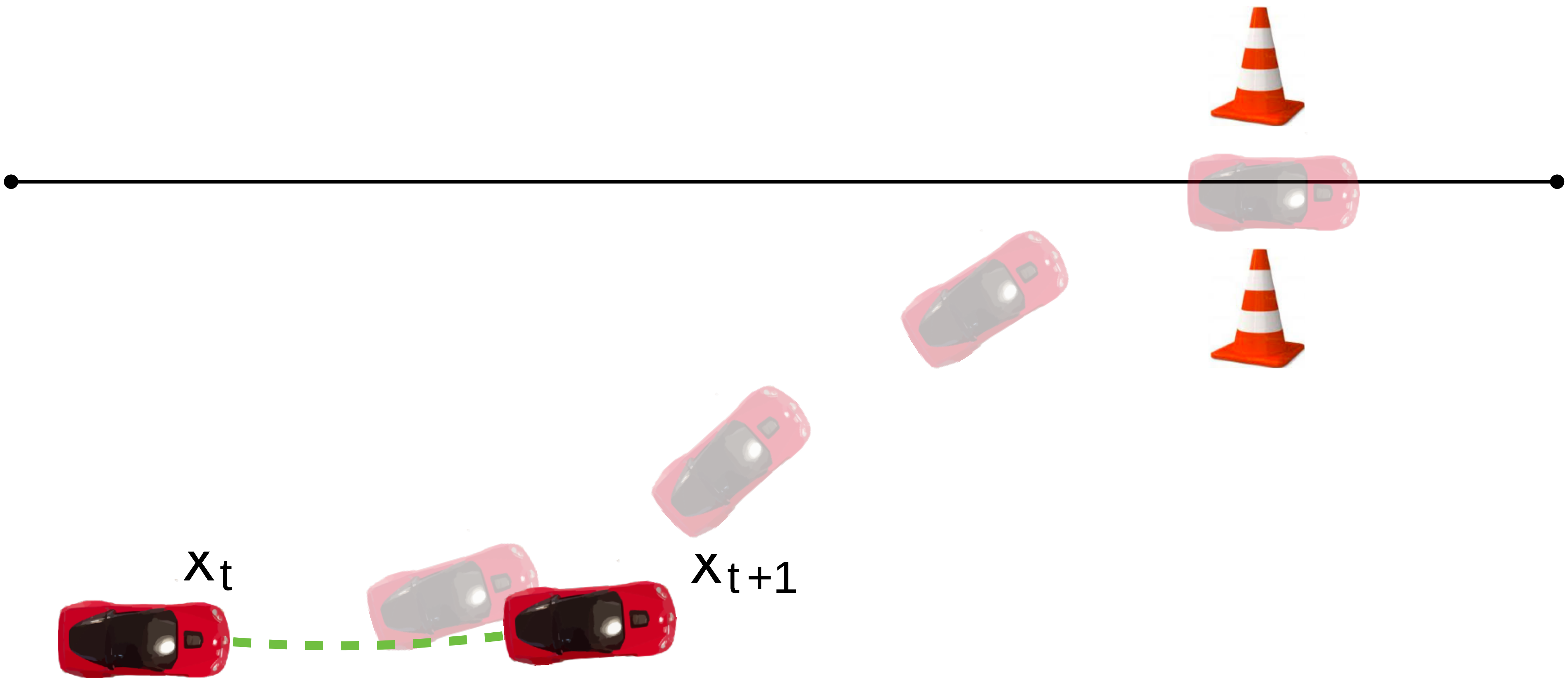


How are the controls executed?



Step 1: Solve optimization problem to a horizon

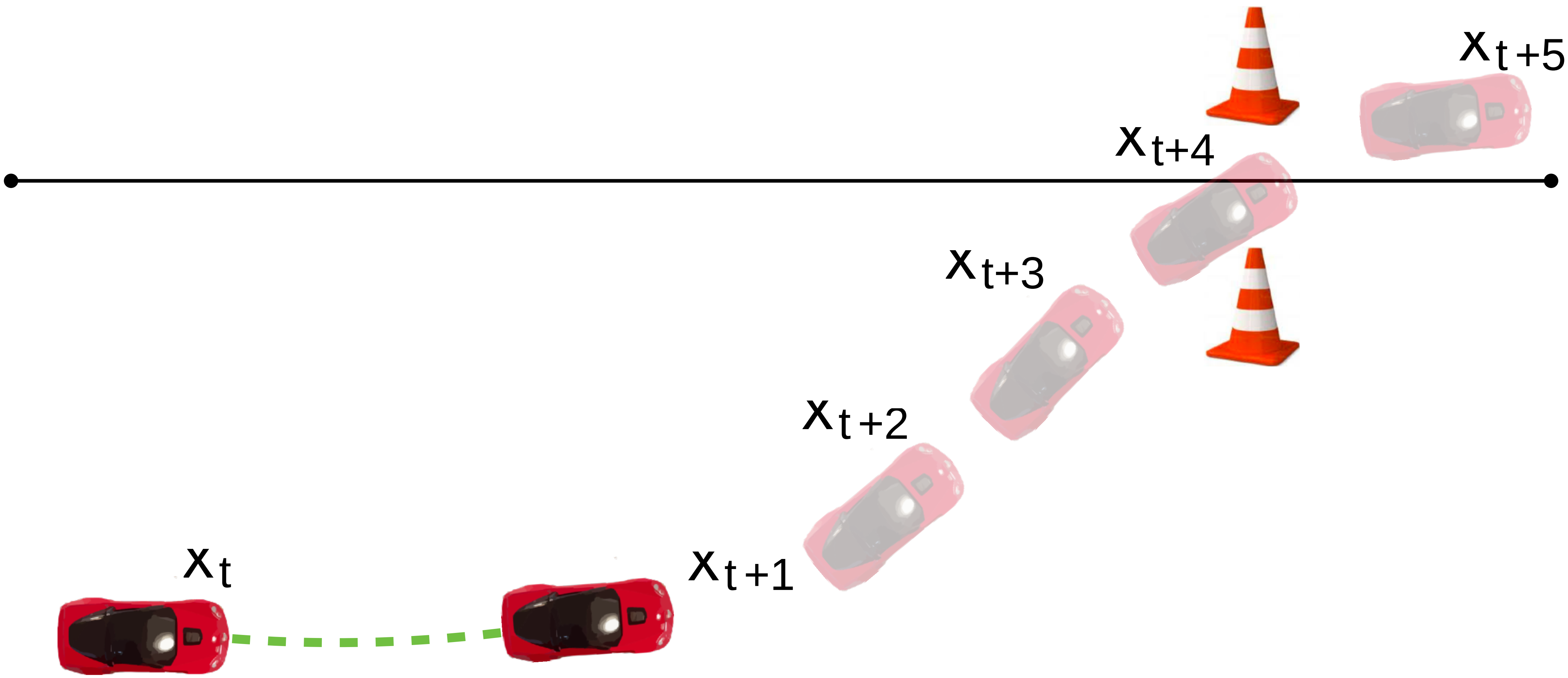
How are the controls executed?



Step 1: Solve optimization problem to a horizon

Step 2: Execute the rst control

How are the controls executed?

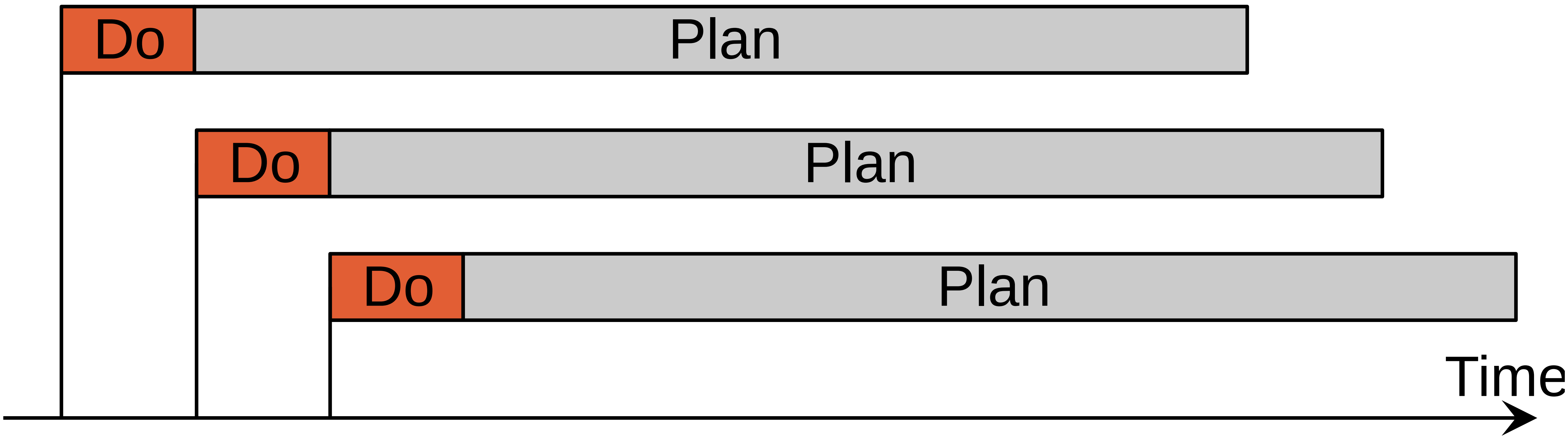


Step 1: Solve optimization problem to a horizon

Step 2: Execute the rst control

Step 3: Repeat!

MPC is a framework

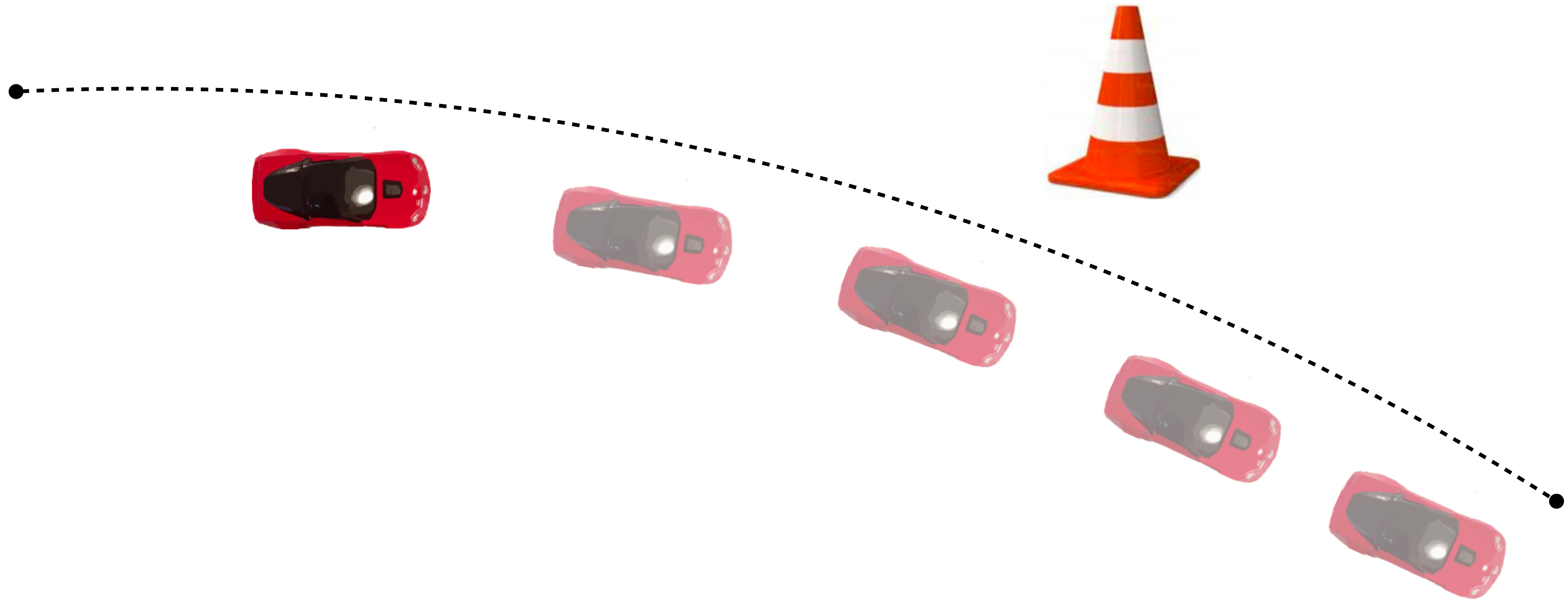


Step 1: Solve optimization problem to a horizon

Step 2: Execute the rst control

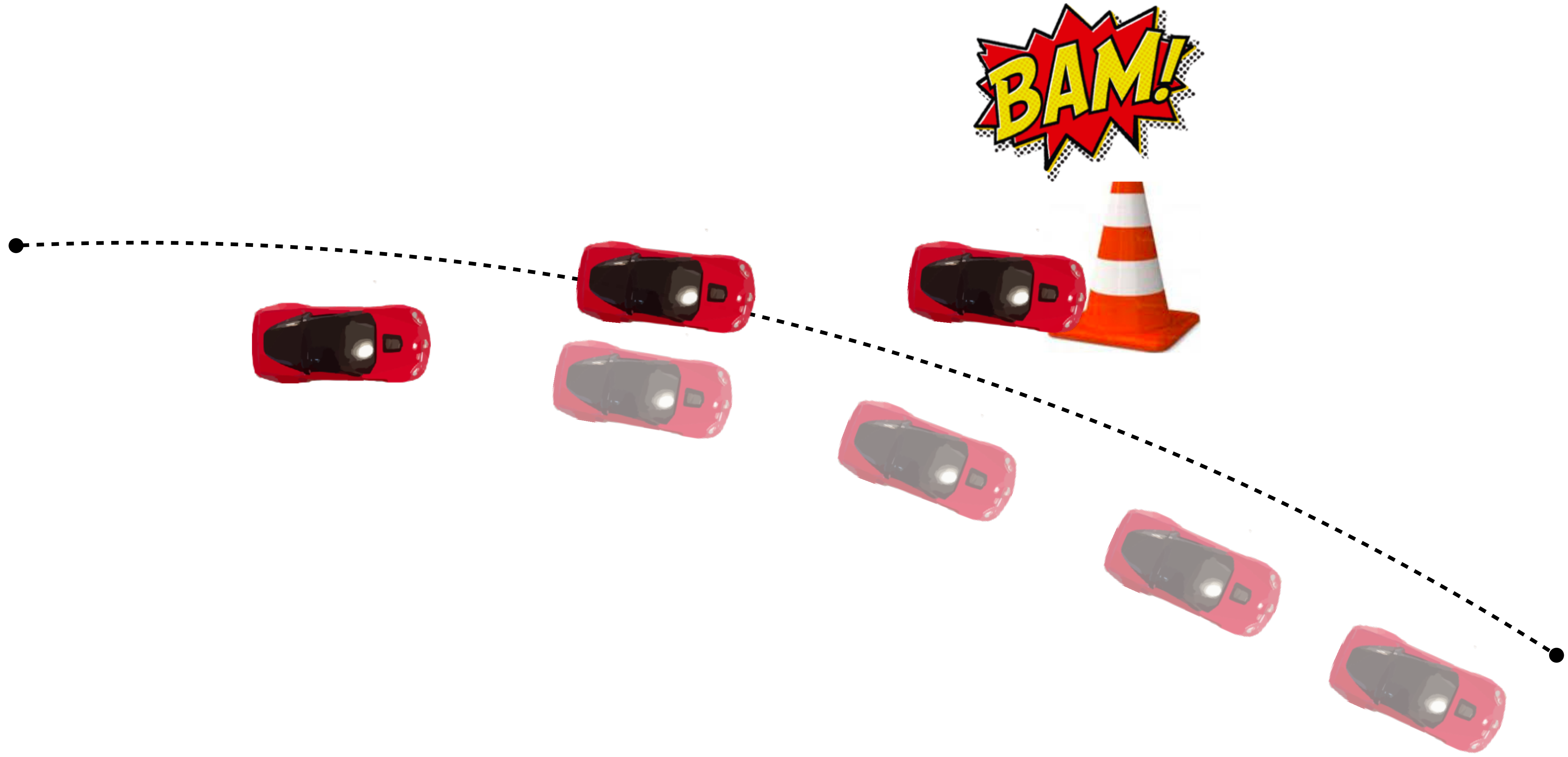
Step 3: Repeat!

Why do we need to replan?



What happens if the controls are planned once and executed?

Why do we need to replan?



What happens if the controls are planned once and executed?

Model predictive control (MPC)

$$\min_{u_{t+1},\ldots,u_{t+H}} \sum_{k=t}^{t+H-1} J(x_k, u_{k+1})$$

(plan till horizon H) (Cost)

$$x_{k+1} = f(x_k, u_{k+1})$$

(Predict next state with dynamics)

$$g(x_k, u_{k+1}) \leq 0$$

(Constraints)