

# Autonomous Vehicle Planning and Control

Wu Ning





# Session 5

Vehicle Model Predictive Controller





# Outline

## Model predictive controller

- MPC main concept
- MPC VS classical control
- MPC formulation
  - Linear MPC non-constraint case
  - Linear MPC constraint case
  - Nonlinear MPC

## MPC based vehicle motion control

- Curvilinear Model 曲线模型
- Objective function
- Constraint generation
- Horizon length
- MPC solver

## How to make MPC run faster





Part I

# Model Predictive Control

Main concept

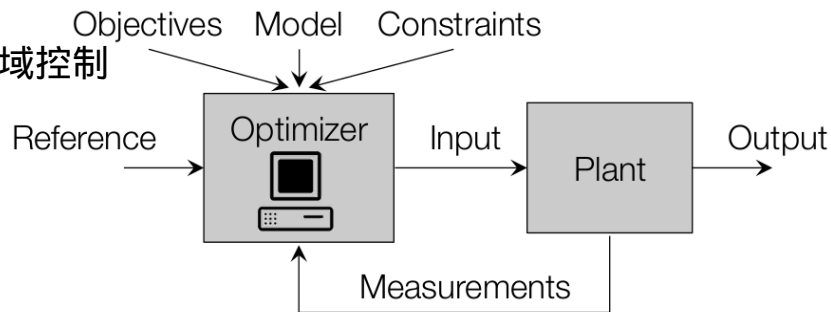




# Model Predictive Control

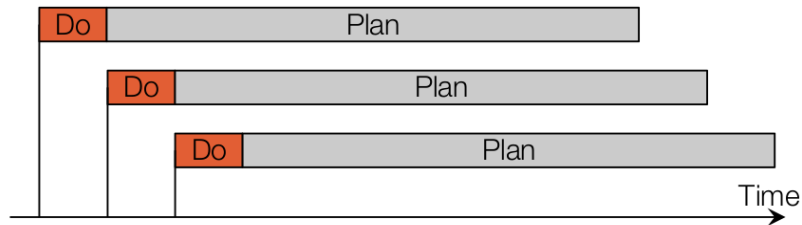
**MPC = Model Predictive Control, also known as**

- MHC = Moving Horizon Control
- RHC = Receding Horizon Control 滚动时域控制
- DMC = Dynamical Matrix Control
- GPC = Generalized Predictive Control



**Control algorithms based on**

- Receding horizon approach
- Constrained optimization





# Constraints in Control

All physical systems have **constraints**:

- Physical constraints, e.g. actuator limits
- Performance constraints, e.g. overshoot
- Safety constraints, e.g. temperature/pressure limits

最佳工作点通常在约束附近

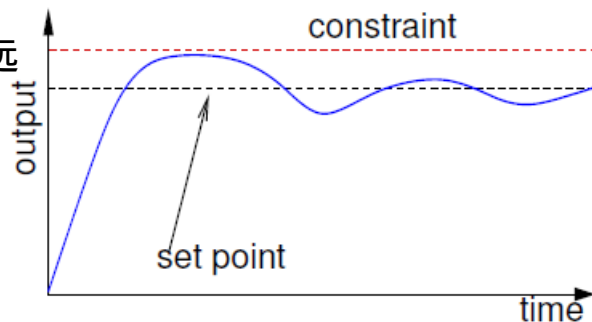
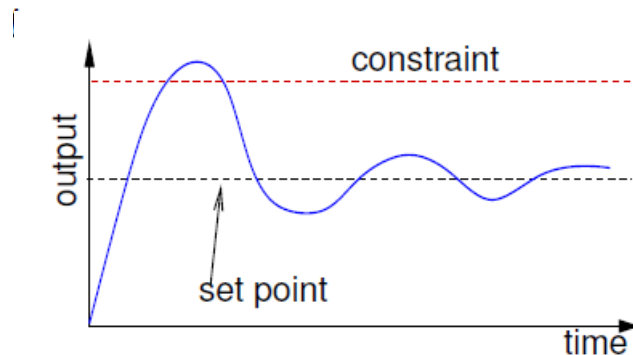
*Optimal operating points are often near constraints.*

Classical control methods:

- Ad hoc constraint management 特别约束管理
- Set point sufficiently far from constraints 设定点离约束足够远
- Suboptimal plant operation 次优

**Predictive control:**

- Constraints included in the design
- Set point optimal
- Optimal plant operation



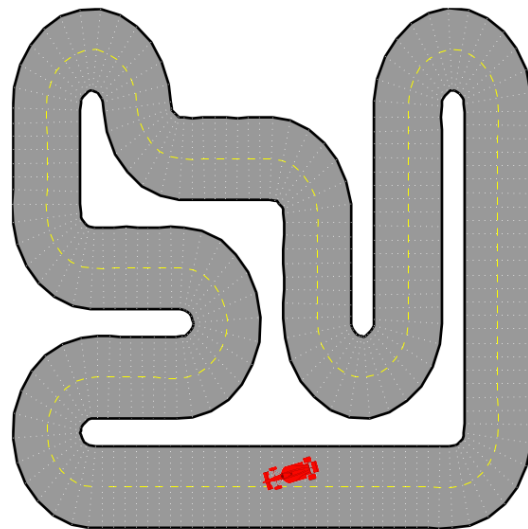
# Model Predictive Control Main Concept: Why MPC?

## Objective:

- Minimize lap time 最小化单圈时间

## Constraints:

- Avoid other cars
- Stay on road
- Don't skid 不要打滑
- Limited acceleration



# Model Predictive Control Main Concept : Why MPC?

## Objective:

- Minimize lap time

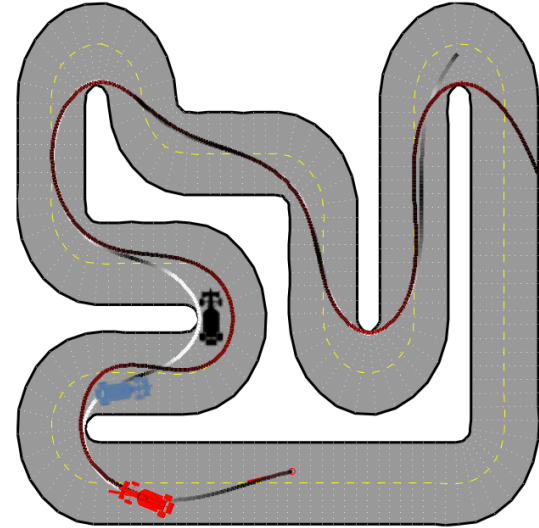
## Problem:

What to do if something unexpected happens?

- We didn't see a car around the corner!
- Must introduce feedback

## Process:

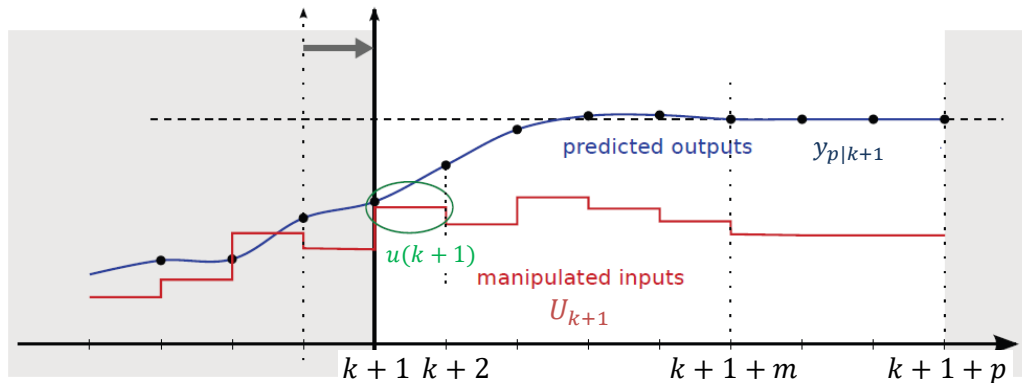
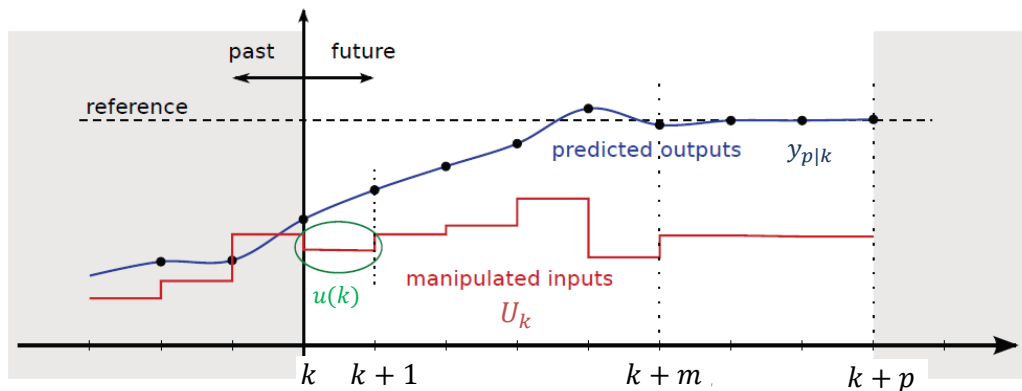
- Obtain series of planned control actions
- Apply *first* control action
- Repeat the planning procedure







# Model Predictive Control: Simple Case study



Assume we have a system

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

$$y_k = h(x_k, u_k)$$

The predicted output  $y_p$ :

$$\{y_{p_{k+1|k}}, y_{p_{k+2|k}}, \dots, y_{p_{k+p|k}}\}$$

The optimal input  $U_k$ :

$$U_k \triangleq \{u_{k|k}, u_{k+1|k}, \dots, u_{k+p-1|k}\}$$

The reference  $r_k$ :

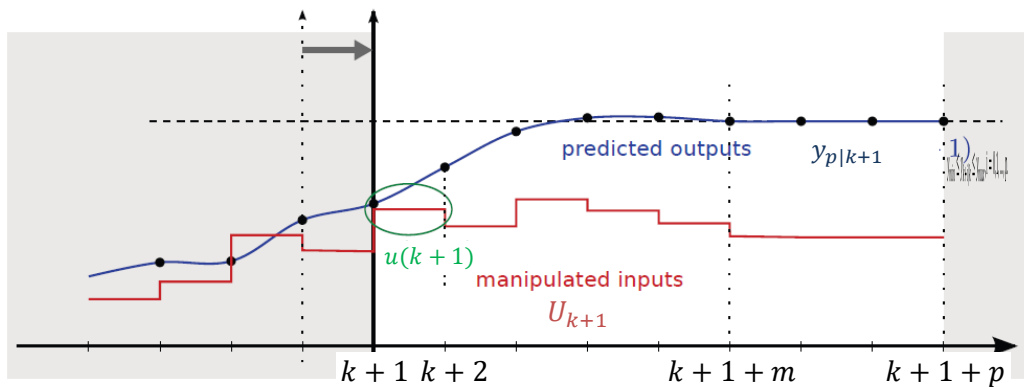
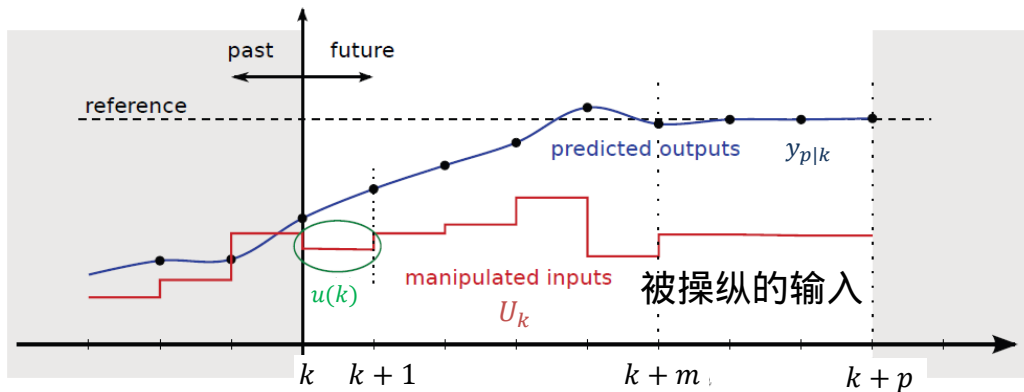
$$\{r_{k+1}, r_{k+2}, \dots, r_{k+p}\}$$

The cost function:

$$J(y_k, U_k) = \sum_{i=k+1}^{k+p} (r_i - y_{p_{i|k}})^2$$



# Model Predictive Control: Simple Case study



Optimal problem can be described as

$$\min_{U_k} J(\mathbf{y}_k, U)$$

$$s. t. \quad x_{k+1} = f(x_k, u_k)$$

$$y_k = h(x_k, u_k)$$

$$u_{min} \leq u_{k+i|k} \leq u_{max}, i = 0, 1, \dots, p-1$$

$$y_{min} \leq y_{k+i|k} \leq y_{max}, i = 0, 1, \dots, p.$$

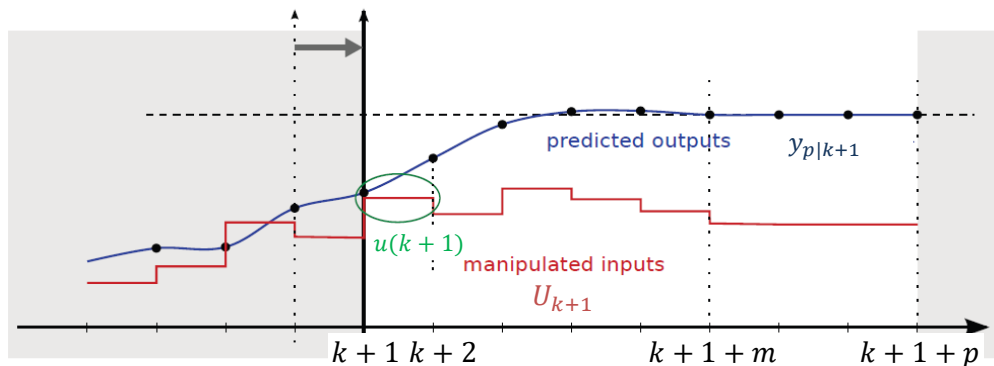
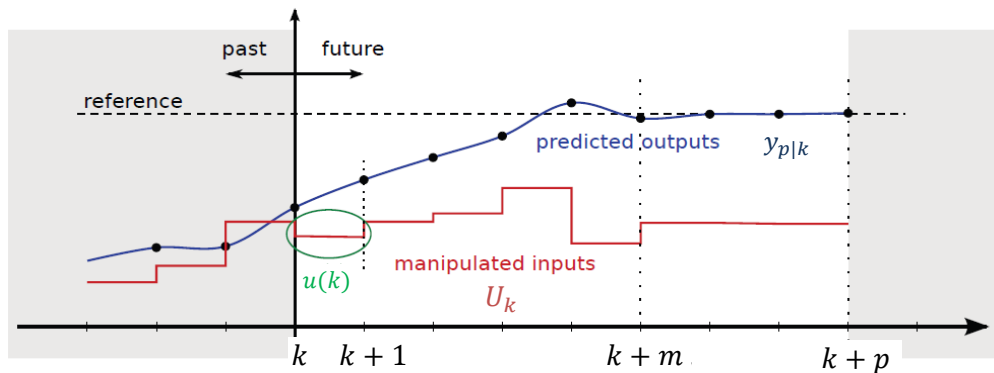
$$U_k^* \triangleq \{u_{k|k}^*, u_{k+1|k}^*, \dots, u_{k+p-1|k}^*\}$$

$$U_k^* = \arg \min_{U_k} J(\mathbf{y}_k, U_k)$$



## Model Predictive Control: Simple Case study

Can we use all the elements in  $U_k^* = \arg \min_{U_k} J(y_k, U_k)$  ?



**No, only the first one.**

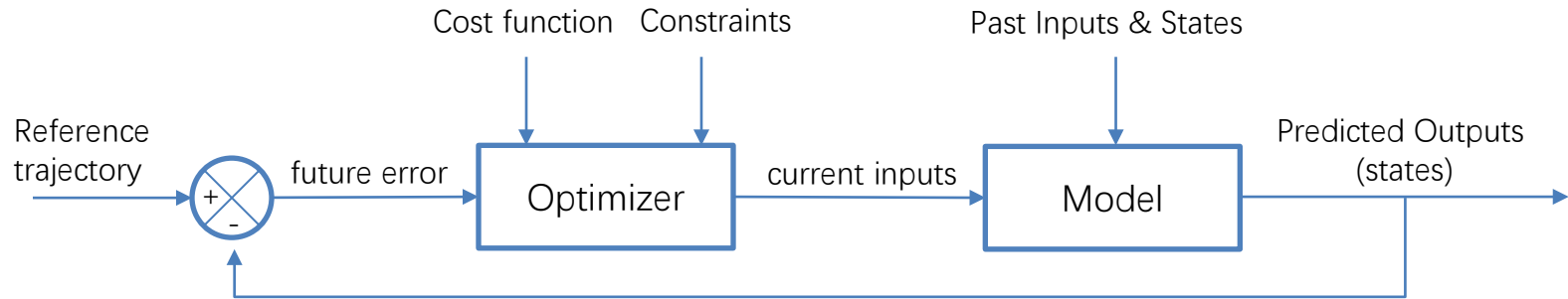
- External disturbance and model-plan mismatch 外部干扰和模型计划不匹配
- Limited horizon length

**MPC process:**

1. Predict the system future state;
2. Find the optimal solution;
3. Use the first optimal solution to the system



# Model Predictive Control Main Structure





# Model Predictive Control

Formulation





## Linear MPC – Unconstrained Case

- Linear system: 
$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \\ \mathbf{y}_k = \mathbf{C}\mathbf{x}_k \end{cases}$$

- Relation between input and states: 
$$\mathbf{x}_k = \mathbf{A}^k \mathbf{x}_0 + \sum_{j=0}^{k-1} \mathbf{A}^j \mathbf{B} \mathbf{u}_{k-1-j}$$
- Forward simulation as matrix equation:

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \cdots & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix} + \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix} \mathbf{x}_0$$



## Linear MPC – Unconstrained Case

Recall LQR cost function

$$J = \sum_{k=0}^{\infty} \mathbf{x}^T(k) \mathbf{Q} \mathbf{x}(k) + \mathbf{u}^T(k) \mathbf{R} \mathbf{u}(k)$$

- Quadratic costs:

$$J = \mathbf{x}_N^T \mathbf{P} \mathbf{x}_N + \sum_{k=0}^{N-1} (\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k)$$

$$\mathbf{P} = \mathbf{P}^T > 0$$

$$\mathbf{Q} = \mathbf{Q}^T > 0$$

$$\mathbf{R} = \mathbf{R}^T > 0$$

Positive semi-definite 半正定

- Goal: find the best control sequence  $\mathbf{u}_{0:N-1}^*$  that minimizes  $J$

$$J = \mathbf{x}_0^T \mathbf{Q} \mathbf{x}_0 + \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix}^T \begin{bmatrix} \mathbf{Q} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{P} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix} + \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}^T \begin{bmatrix} \mathbf{R} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}$$



## Linear MPC – Unconstrained Case

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}}_{\bar{S}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_z + \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}}_{\bar{T}} x_0$$

$$J(z, x_0) = x_0^T Q x_0 + \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}^T}_{\bar{Q}} \underbrace{\begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & Q & 0 \\ 0 & 0 & \dots & 0 & P \end{bmatrix}}_{\bar{Q}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}}_{\bar{S}z + \bar{T}x_0} + \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}^T}_{\bar{R}} \underbrace{\begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & R \end{bmatrix}}_{\bar{R}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{z}$$

$$J(z, x_0) = (\bar{S}z + \bar{T}x_0)^T \bar{Q} (\bar{S}z + \bar{T}x_0) + z^T \bar{R} z + x_0^T Q x_0$$

$$= \frac{1}{2} z^T \underbrace{2(\bar{R} + \bar{S}^T \bar{Q} \bar{S})}_H z + x_0^T \underbrace{2\bar{T}^T \bar{Q} \bar{S}}_F z + \frac{1}{2} x_0^T \underbrace{2(Q + \bar{T}^T \bar{Q} \bar{T})}_E x_0$$





## Linear MPC – Unconstrained Case

$$J(\mathbf{z}, \mathbf{x}_0) = \frac{1}{2} \mathbf{z}^T \underbrace{2 \left( \bar{\mathbf{R}} + \bar{\mathbf{S}}^T \bar{\mathbf{Q}} \bar{\mathbf{S}} \right)}_{\mathbf{H}} \mathbf{z} + \underbrace{\mathbf{x}_0^T 2 \bar{\mathbf{T}}^T \bar{\mathbf{Q}} \bar{\mathbf{S}}}_{\mathbf{F}} \mathbf{z} + \frac{1}{2} \mathbf{x}_0^T \underbrace{2 \left( \bar{\mathbf{Q}} + \bar{\mathbf{T}}^T \bar{\mathbf{Q}} \bar{\mathbf{T}} \right)}_{\mathbf{E}} \mathbf{x}_0$$

MPC的浓缩形式:

- Condensed form of MPC:

$$J(\mathbf{z}, \mathbf{x}_0) = \frac{1}{2} \mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{x}_0^T \mathbf{F} \mathbf{z} + \frac{1}{2} \mathbf{x}_0^T \mathbf{E} \mathbf{x}_0$$

$$\mathbf{z} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}$$

通过调零梯度获得最佳值

The optimum is obtained by zeroing the gradient

$$\nabla_{\mathbf{z}} J(\mathbf{z}, \mathbf{x}_0) = \mathbf{H} \mathbf{z} + \mathbf{F}^T \mathbf{x}_0 = 0 \rightarrow \mathbf{z}^* = -\underline{\mathbf{H}^{-1} \mathbf{F}^T} \mathbf{x}_0$$

(“batch” solution)

“批量”解决方案

unconstrained linear MPC = linear state-feedback!



## Linear MPC – Unconstrained Case

- Non-condensed form of MPC:

- Keep also  $\mathbf{x}_1, \dots, \mathbf{x}_N$  as optimization variables

$$J(\mathbf{z}, \mathbf{x}_0) = \mathbf{x}_0^T \mathbf{Q} \mathbf{x}_0 + \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix}^T \begin{bmatrix} \mathbf{R} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{P} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{x}_{N-1} \\ \mathbf{x}_N \end{bmatrix}$$

$$\mathbf{z} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}$$

- Equality constraints  
等式约束

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$$

More variables and constraints but very sparse

更多的变量和约束，但非常稀疏



## MPC and Linear Quadratic Regulator (LQR)

- “Batch” solution:  $\mathbf{z}^* = -\mathbf{H}^{-1}\mathbf{F}^T\mathbf{x}_0$ 
  - Analytically 分析地
  - Still requires to invert a large matrix  $\mathbf{H}^{-1}$
- Bellman's principle of optimality  $A \rightarrow \dots \rightarrow B \rightarrow \dots \rightarrow C$

"An optimal policy has the property that, regardless of the decisions taken to enter a particular state, the remaining decisions made for leaving that stage must constitute an optimal policy."

最优策略具有这样的性质，即不管进入某一特定状态的决策如何，离开该阶段的其余决策必定构成最优策略

- Dynamic programming LQR
  - Exploiting the sequential structure of the problem



# MPC and Linear Quadratic Regulator (LQR)

## Dynamic programming LQR

- Exploiting the sequential structure of the problem
- Bellman's principle of optimality

$$J = \mathbf{x}_N^T \mathbf{P} \mathbf{x}_N + \sum_{k=0}^{N-1} (\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k)$$

$$\begin{array}{ccccccc} \mathbf{x}_0^T \mathbf{Q} \mathbf{x}_0 & \rightarrow & \cdots & \rightarrow & \mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k & \rightarrow & \cdots & \rightarrow & \mathbf{x}_N^T \mathbf{P} \mathbf{x}_N \\ \mathbf{z}^* & & & & \mathbf{u}_0^* & \cdots & & & \mathbf{u}_k^* & \cdots & \mathbf{u}_{N-1}^* \end{array}$$

---

---



# MPC and Linear Quadratic Regulator (LQR)

## Dynamic programming LQR

将问题分解成更小的问题

- Break up the problem into sub-smaller problems

$$J_N^*(x_N) \triangleq x_N^T P x_N$$

$$J_{N-1}(x_{N-1}) \triangleq x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + J_N^*(x_N)$$

$$J = x_N^T P x_N + \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k)$$

$$x_N = A x_{N-1} + B u_{N-1}$$

immediate cost

cost-to-go

$$u_{N-1}(x_{N-1}) = \arg \min_{u_{N-1}} \underbrace{x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1}}_{\text{immediate cost}} + \underbrace{J_N^*(A x_{N-1} + B u_{N-1})}_{\text{cost-to-go}}$$

$$= \arg \min_{u_{N-1}} \underbrace{x_{N-1}^T (A^T P A + Q) x_{N-1}}_{\text{const.}} + \underbrace{u_{N-1}^T (B^T P B + R) u_{N-1} + 2 x_{N-1}^T A^T P B u_{N-1}}_{\text{cost-to-go}}$$

Zeroing the gradient:

$$2(B^T P B + R) u_{N-1}^* + 2 B^T P A x_{N-1} = 0$$

将梯度归零:

$$\underline{u_{N-1}^*(x_{N-1}) = K x_{N-1} \quad K = -(B^T P B + R)^{-1} B^T P A}$$



# MPC and Linear Quadratic Regulator (LQR)

## Dynamic programming LQR

递归解决尾部问题

- Recursively solve the tail problems

$$J = \mathbf{x}_N^T \mathbf{P} \mathbf{x}_N + \sum_{k=0}^{N-1} (\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k)$$

$$J_N^*(\mathbf{x}_N) \triangleq \mathbf{x}_N^T \mathbf{P} \mathbf{x}_N$$

$$J_{N-1}(\mathbf{x}_{N-1}) = \mathbf{x}_{N-1}^T (\mathbf{A}^T \mathbf{P} \mathbf{A} + \mathbf{Q}) \mathbf{x}_{N-1} + \mathbf{u}_{N-1}^T (\mathbf{B}^T \mathbf{P} \mathbf{B} + \mathbf{R}) \mathbf{u}_{N-1} + 2 \mathbf{x}_{N-1}^T \mathbf{A}^T \mathbf{P} \mathbf{B} \mathbf{u}_{N-1}$$

$$\mathbf{u}_{N-1}^*(\mathbf{x}_{N-1}) = \mathbf{K} \mathbf{x}_{N-1} \quad \underline{\mathbf{K} = -(\mathbf{B}^T \mathbf{P} \mathbf{B} + \mathbf{R})^{-1} \mathbf{B}^T \mathbf{P} \mathbf{A}}$$

$$J_{N-1}^*(\mathbf{x}_{N-1}) = \mathbf{x}_{N-1}^T \{ \underline{\mathbf{A}^T \mathbf{P} \mathbf{A} + \mathbf{Q} + \mathbf{A}^T \mathbf{P} \mathbf{B} \mathbf{K}} \} \mathbf{x}_{N-1}$$

$$J_{N-2}(\mathbf{x}_{N-2}) \triangleq \mathbf{x}_{N-2}^T \mathbf{Q} \mathbf{x}_{N-2} + \mathbf{u}_{N-2}^T \mathbf{R} \mathbf{u}_{N-2} + J_{N-1}^*(\mathbf{x}_{N-1})$$

...

$$\mathbf{u}_0^*(\mathbf{x}_0) = \mathbf{K} \mathbf{x}_0$$

$$\mathbf{P} = \mathbf{A}' \mathbf{P} \mathbf{A} + \mathbf{Q} - \mathbf{A}' \mathbf{P} \mathbf{B} (\mathbf{B}^T \mathbf{P} \mathbf{B} + \mathbf{R})^{-1} \mathbf{B}^T \mathbf{P} \mathbf{A}$$

Algebraic Riccati Equation

代数Riccati方程



## MPC and Linear Quadratic Regulator (LQR)

### Dynamic programming LQR

- Consider again the MPC cost function

$$\min_z \mathbf{x}_N^T \mathbf{P} \mathbf{x}_N + \sum_{k=0}^{N-1} (\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k)$$

- Update matrix  $\mathbf{P}$  and terminal gain  $\mathbf{K}$  iteratively

$$\mathbf{K} = -(\mathbf{B}^T \mathbf{P} \mathbf{B} + \mathbf{R})^{-1} \mathbf{B}^T \mathbf{P} \mathbf{A}$$

$$\mathbf{P} = \mathbf{A}^T \mathbf{P} \mathbf{A} + \mathbf{Q} + \mathbf{A}^T \mathbf{P} \mathbf{B} \mathbf{K}$$

...

$$\mathbf{u}_0^*(\mathbf{x}_0) = \mathbf{K} \mathbf{x}_0$$

“Batch” solution

$$\mathbf{z}^* = -\mathbf{H}^{-1} \mathbf{F}^T \mathbf{x}_0$$

$O(N^3)$  complexity

→ Linear complexity in horizon  $N$

(unconstrained) MPC = LQR for any choice of the prediction horizon  $N$



## Constrained Linear MPC

Convert the problem to

### Quadratic Programming (QP)

$$J(\mathbf{z}, \mathbf{x}_0) = \frac{1}{2} \mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{x}_0^T \mathbf{F} \mathbf{z} + \frac{1}{2} \mathbf{x}_0^T \mathbf{E} \mathbf{x}_0$$

$$\min_{\mathbf{z}} \quad \frac{1}{2} \mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{x}_0^T \mathbf{F} \mathbf{z} \quad (\text{quadratic objective})$$

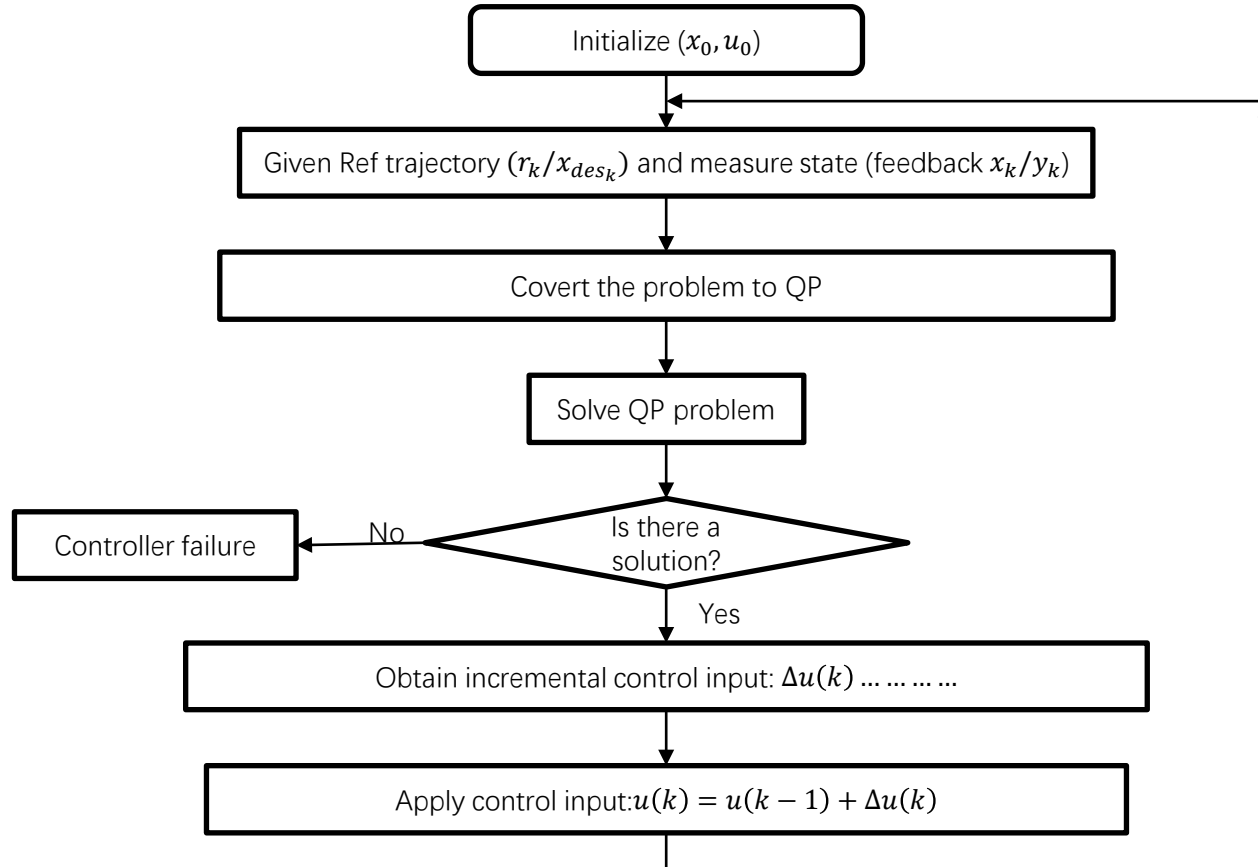
$$\text{s. t.} \quad \mathbf{G} \mathbf{z} \leq \mathbf{W} + \mathbf{S} \mathbf{x}_0 \quad (\text{linear constraints})$$

- Popular noncommercial QP solvers
  - OOQP, OSQP, qpOASES, ECOS (SOCP)
- Popular Commercial QP solvers
  - GUROBI, MOSEK (LPs, QPs, SOCPs, SDPs and MIPs ...)





# Constrained Linear MPC

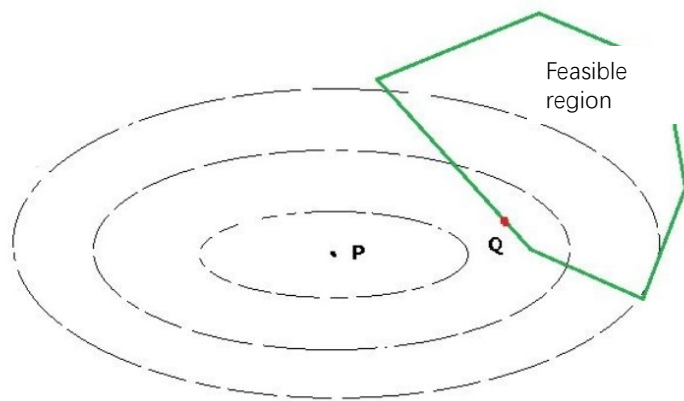




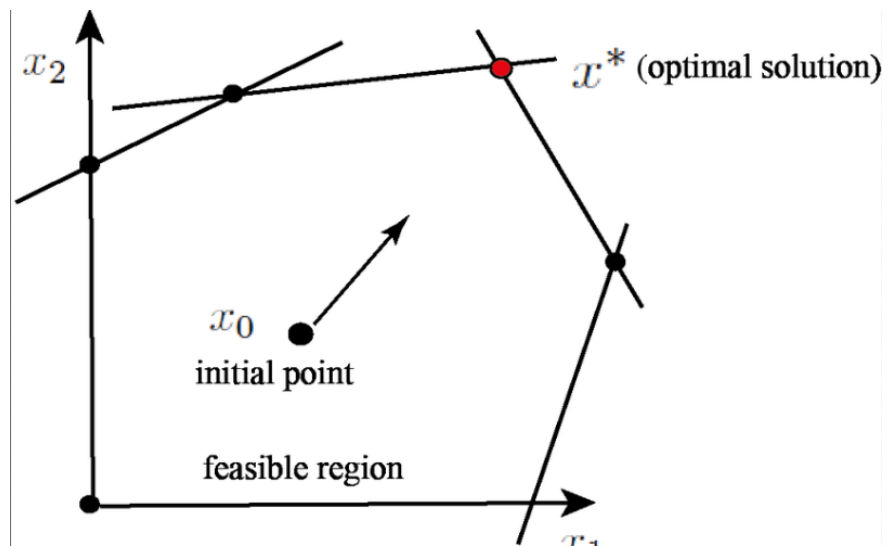
# Constrained Linear MPC

## Approaches of solving a QP problem:

- Active-set methods
  - Only inequal constraint



- Interior-point method
  - More general





## Constrained Nonlinear MPC

- Constrained (non)linear finite horizon discrete time case

$$\begin{aligned} \min_{U \triangleq \{u_{k|k}, u_{k+1|k}, \dots\}} \quad & J(\mathbf{x}_k, U) = \sum_{k=0}^{N-1} J(\mathbf{x}_k, u_k) \\ \text{s. t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k), && \text{Nonlinear process model} \\ & c_k(\mathbf{x}_k, u_k) \leq 0, && \text{inequality constraints} \\ & h_k(\mathbf{x}_k, u_k) = 0, && \text{equality constraints} \\ & \mathbf{x} \in \mathcal{X}, && \text{State constraints} \\ & u \in \mathcal{U}, && \text{input constraints} \end{aligned}$$

- No closed form solution, must be solved numerically



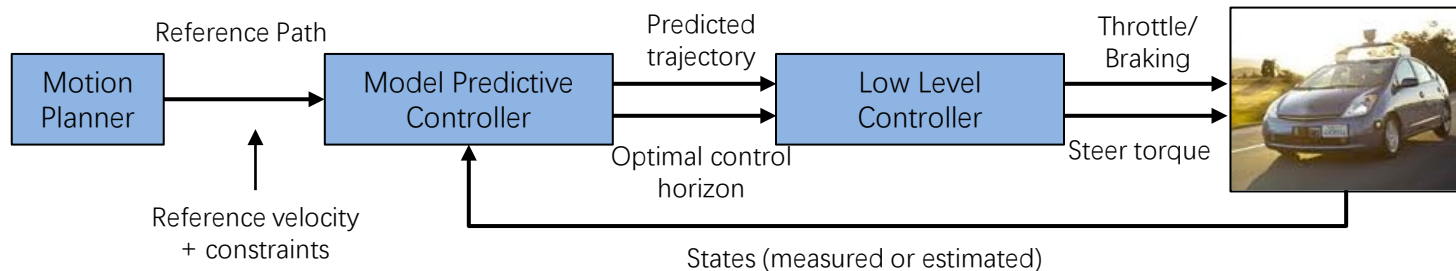
Part II

# Vehicle Motion Control





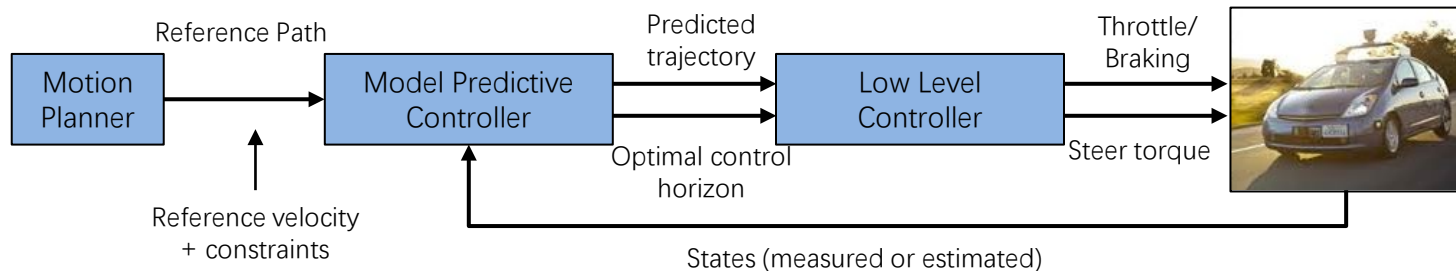
# Vehicle Motion Control: Architecture



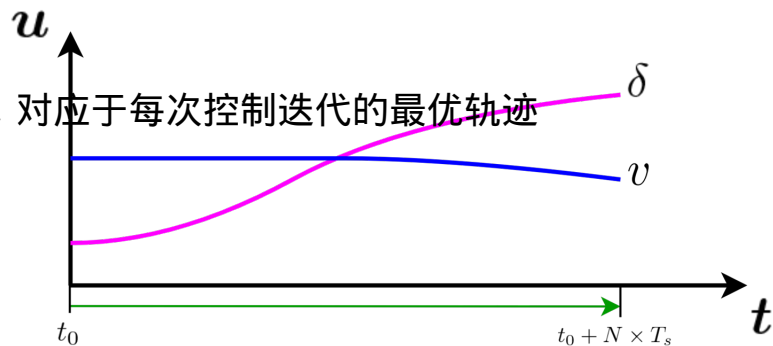
- The presented MPC framework performs **iterative trajectory optimization** over a finite time horizon. 提出的MPC框架在有限时间范围内执行迭代轨迹优化
- A **motion model of the vehicle** as process model, allows MPC to understand evolution of the vehicle state. 作为过程模型的车辆运动模型允许MPC了解车辆状态的演变
- Planner's bounds are encoded by **constraints**. 规划者的界限由约束编码
- The **objective function** shapes desired driving behavior in the most fundamental way possible. 目标函数尽可能以最基本的方式塑造期望的驾驶行为



## Vehicle Motion Control: Output



- MPC outputs a **finite optimal control horizon** corresponding to the optimal trajectory for every control iteration. **MPC输出一个有限的最优控制水平，对应于每次控制迭代的最优轨迹**  
**在MPC的具体实现中，控制范围可达其二阶导数**
- In the specific implementation of MPC, the control horizon is available up to its **second derivatives**.
  - Speed -> acceleration -> jerk
  - Steering angle -> steering rate -> steering acceleration





## Vehicle Motion Control: Model Predictive Controller

模型 – 用于传播系统以预测未来信息

- Model – used to propagate the system to predict the future information
- Cost function – Minimize
  - Deviation from desired trajectory 偏离理想轨迹
  - Minimization of control command magnitude, etc. 控制命令幅度的最小化等
- Constraints – subject to
  - Longitudinal and later dynamic models
  - Tire force limits,etc.
- Can incorporate low level controller, adding constraints for: 可以结合低级控制器，增加以下约束:
  - Engine map
  - Full dynamic vehicle model
  - Actuator models
  - Tire force models



# Vehicle Motion Control

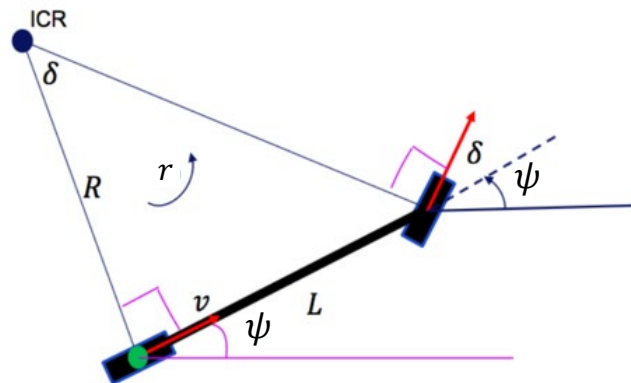
Model







# Vehicle Model (review)



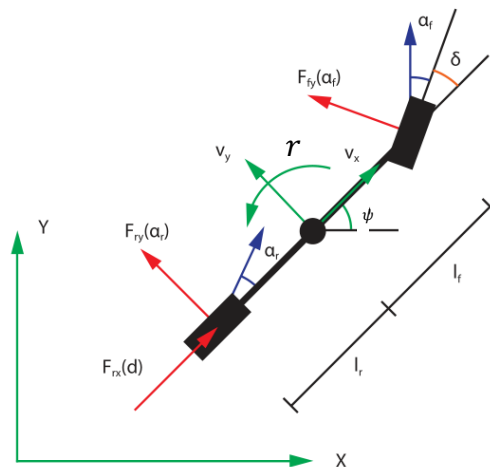
$P(p_x, p_y)$

$$\begin{cases} \dot{p}_x = v \cos(\psi) \\ \dot{p}_y = v \sin(\psi) \\ \dot{\psi} = \frac{v}{L} \tan(\delta) \\ \dot{v} = a \end{cases}$$

Kinematic Bicycle Model

Simplicity

VS.



$$\begin{aligned} \dot{x} &= v_x \cos(\psi) - v_y \sin(\psi) \\ \dot{y} &= v_x \sin(\psi) + v_y \cos(\psi) \\ \dot{\psi} &= r \\ \dot{v}_x &= \frac{1}{m} (F_{r,x} - F_{f,y} \sin(\delta) + m v_y r) \\ \dot{v}_y &= \frac{1}{m} (F_{r,y} + F_{f,y} \cos(\delta) - m v_x r) \\ \dot{r} &= \frac{1}{l_z} (F_{f,y} l_f \cos(\delta) - F_{r,y} l_r) \end{aligned}$$

Dynamic Bicycle Model

Accuracy



# Curvilinear Coordinates Vehicle Model

曲线坐标系模型是同时考虑纵向和横向运动的有效模型。

Curvilinear coordination model is an efficient model to consider both longitudinal and lateral motion.

The state and input variables are the optimization variables of MPC

State Variables:

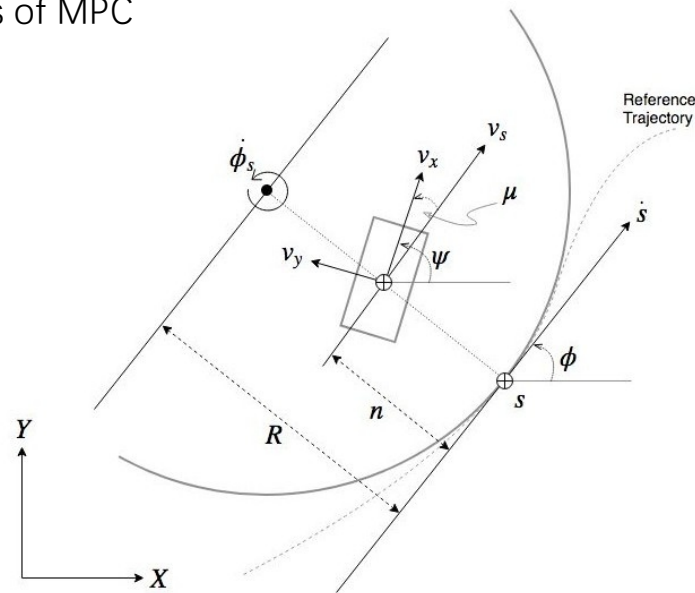
$$x = \begin{bmatrix} s \\ n \\ \mu \\ v \\ a \\ \delta \\ \dot{\delta} \end{bmatrix}$$

Progress
Lateral error
Local heading
Velocity
Acceleration in projected driving direction
Steering angle
Steering rate

Input variable:

$$u = \begin{bmatrix} u_{jerk} \\ u_{\dot{\delta}} \end{bmatrix}$$

Jerk
Steering accel





# Curvilinear Coordinates Vehicle Model

The so called arc velocity  $\dot{s}$  gives a way of describing the vehicles motion along a reference trajectory parameterized by  $s$ .

Vehicle speed along  $s$ :

$$v_s = v_x \cos \mu - v_y \sin \mu = (R - n) \dot{\phi}_s$$

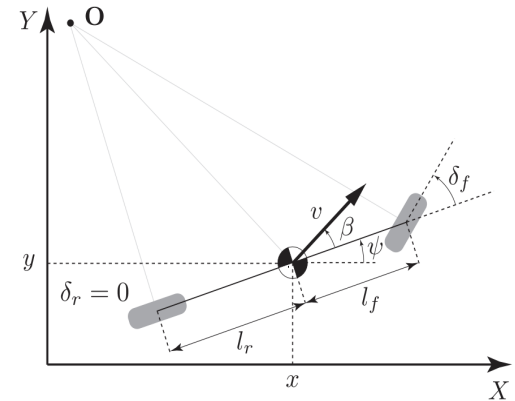
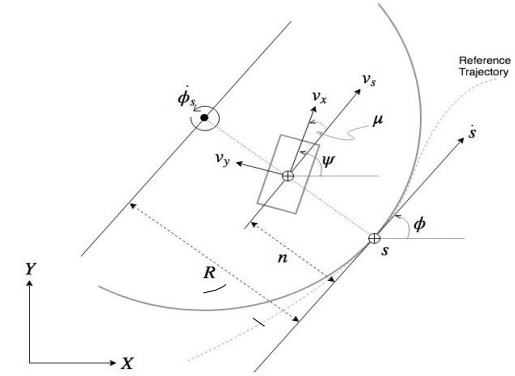
$$\left. \begin{aligned} v_s &= (R - n) \dot{\phi}_s \\ \dot{s} &= R \dot{\phi}_s \end{aligned} \right\} \Rightarrow \dot{s} = \frac{R}{R - n} (v_x \cos \mu - v_y \sin \mu)$$

$$\left. \begin{aligned} \dot{s} &= \frac{R}{R - n} (v_x \cos \mu - v_y \sin \mu) \\ \kappa &= \frac{1}{R} \end{aligned} \right\} \Rightarrow \dot{s} = \frac{v_x \cos \mu - v_y \sin \mu}{1 - n\kappa}$$

$$\left. \begin{aligned} \dot{s} &= \frac{v_x \cos \mu - v_y \sin \mu}{1 - n\kappa} \\ v_x &= v \cos \beta \\ v_y &= v \sin \beta \end{aligned} \right\} \Rightarrow \dot{s} = \frac{v \cos \beta \cos \mu - v \sin \beta \sin \mu}{1 - n\kappa} = \frac{v \cos(\beta + \mu)}{1 - n\kappa}$$

$$\dot{n} = v_x \sin \mu + v_y \cos \mu = v \cos \beta \sin \mu + v \sin \beta \cos \mu = v \sin(\beta + \mu)$$

$$\dot{\mu} = r - \dot{\phi}_s = \frac{v}{R_v} - \kappa \frac{v \cos(\beta + \mu)}{1 - n\kappa} = \frac{v \sin \beta}{l_r} - \kappa \frac{v \cos(\beta + \mu)}{1 - n\kappa}$$





# Curvilinear Coordinates Vehicle Model

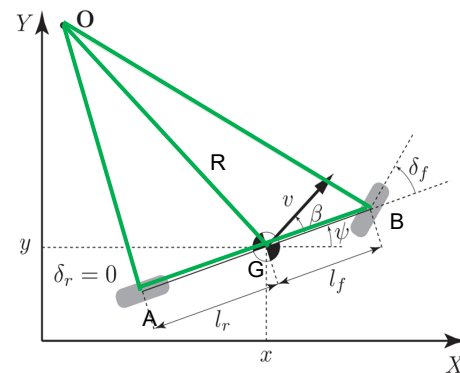
The motion model is a discretized kinematic bicycle model in curvilinear coordinates.

$$\dot{x} = \begin{bmatrix} \dot{s} \\ \dot{n} \\ \dot{\mu} \\ \dot{v} \\ \dot{a} \\ \dot{\delta} \\ \ddot{\delta} \end{bmatrix} = \begin{bmatrix} \frac{v \cos(\mu + \beta)}{1 - n\kappa} \\ v \sin(\mu + \beta) \\ \frac{v}{l_r} \sin(\beta) - \kappa \frac{v \cos(\mu + \beta)}{1 - n\kappa} \\ a \\ u_{jerk} \\ \dot{\delta} \\ u_{\ddot{\delta}} \end{bmatrix}$$

where  $\beta = \tan^{-1}(\frac{l_r}{l_r + l_f} \tan(\delta_f))$  is slip angle

$l_f$ : length from front of the car to CoG

$l_r$ : length from rear of the car to CoG



$$\sin \beta = \frac{l_r}{R} \therefore R = \frac{l_r}{\sin \beta}$$

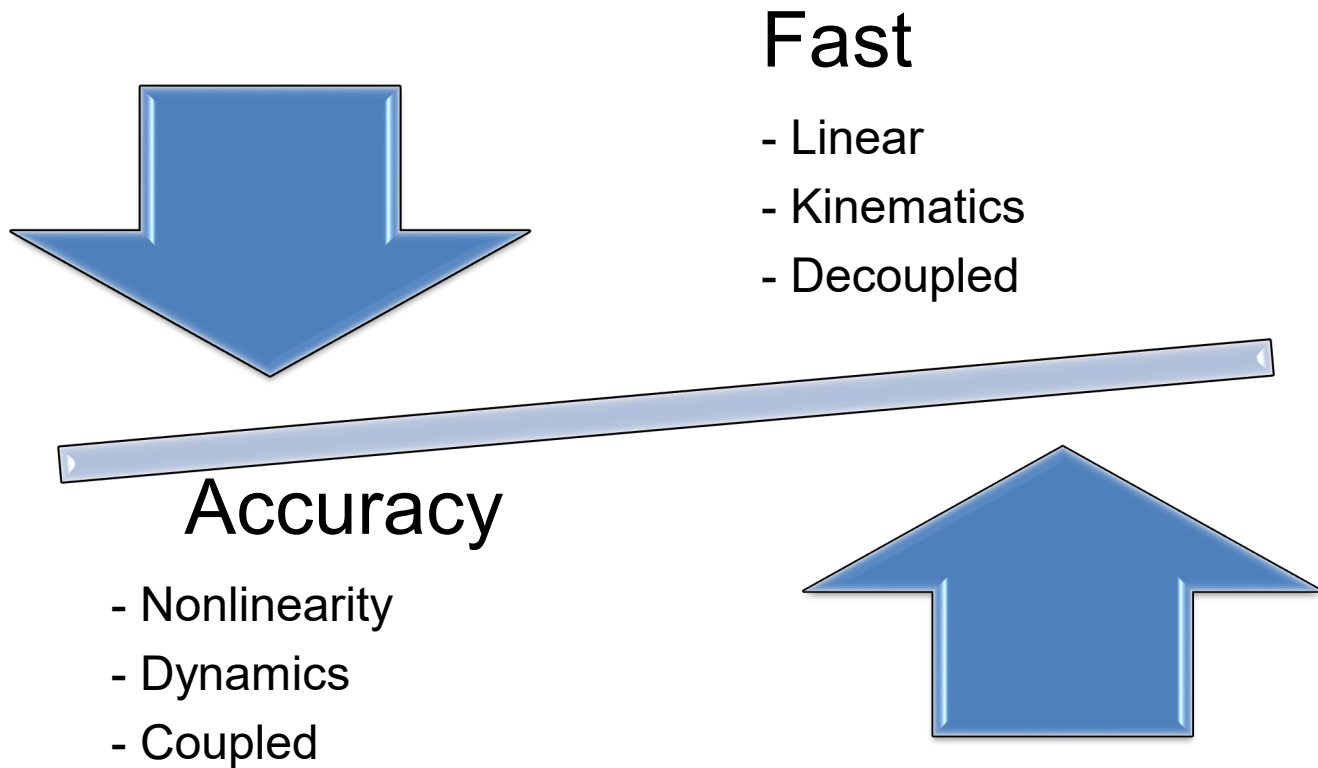
$$\frac{l_f}{\sin(\delta_f - \beta)} = \frac{R}{\sin(\frac{\pi}{2} - \delta_f)}$$

$$\frac{l_f}{\sin \delta_f \cos \beta - \cos \delta_f \sin \beta} = \frac{l_r}{\cos \delta_f \sin \beta}$$

$$\frac{l_f}{\tan \delta_f - \tan \beta} = \frac{l_r}{\tan \beta}$$



## Vehicle model choice for MPC





# Vehicle Motion Control

Objective/Cost function





## Vehicle Motion Control: Objective/Cost function

The objective function is our main mechanism to **shape driving behavior**.

The objective function has different types of components:

- **Tracking** objectives (Reference from planner)
- **Comfort** objectives (Comfortability of vehicle motion, perceived danger)
- **Safety** objectives (Constraint violation)

Different weights can be applied to separate cost terms in order to favor a certain trade-off.

$$J = w_t \cdot J_{tracking} + w_c \cdot J_{comfort} + w_s \cdot J_{safety}$$

where

$w_t$  is the weight for tracking cost,

$w_c$  is the weight for comfort cost,

$w_s$  is the weight for safety cost.



# Vehicle Motion Control: Tracking Objective/Cost function

## Tracking objective function:

- follow the planner path
- follow the “speed profile”

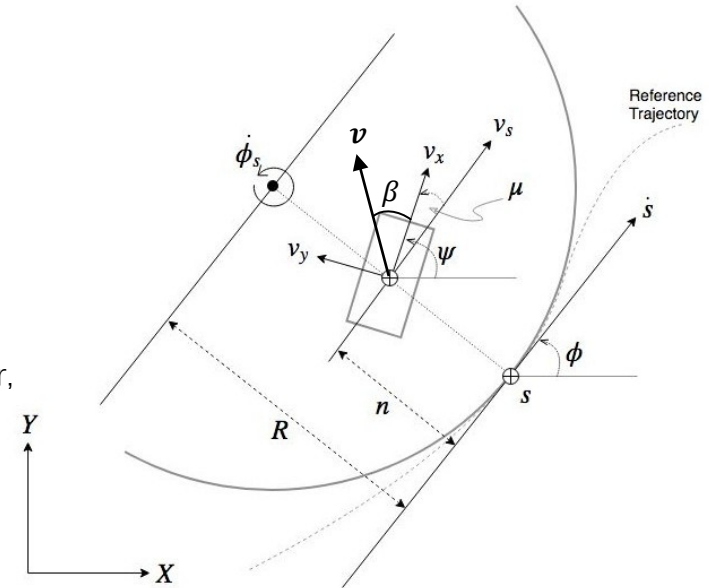
$$J_{tracking} = w_{\dot{s}} \cdot (\dot{s} - \dot{s}_{ref})^2 + w_n \cdot n^2 + w_{\mu} \cdot (\mu + \beta)^2$$

where

$w_{\dot{s}}$  is the weight for minimizing the longitudinal speed tracking error,

$w_n$  is the weight for minimizing the lateral error,

$w_{\mu}$  is the weight for minimizing the heading error.







## Vehicle Motion Control: Comfort Objective/Cost function

The comfort objective function:

$$J_{comfort} = w_{j_x} \cdot j_x^2 + w_{a_x} \cdot a_x^2 + w_{a_y} \cdot a_y^2 + w_{\dot{\delta}} \cdot \dot{\delta} + w_{\ddot{\delta}} \cdot \ddot{\delta}^2$$

Where  $w_{j_x}$  is the weight for minimizing the longitudinal jerk,

$w_{a_x}$  is the weight for minimizing the longitudinal acceleration,

$w_{a_y}$  is the weight for minimizing the lateral acceleration,

$w_{\dot{\delta}}$  is the weight for minimizing the steering rate,

$w_{\ddot{\delta}}$  is the weight for minimizing the steering acceleration.



## Vehicle Motion Control: Safety Objective

Safety cost: constraint violation ➡ Soft Constraints: Motivation

- Input constraints are dictated by physical constraints on the actuators and are usually “hard”.
- State/output constraints arise from practical restrictions on the allowed operating range and are **rarely hard**.
- Hard state/output constraints always lead to *complications in the controller implementation*,
  - Feasible operating regime is constrained even for stable systems,
  - Controller patches must be implemented to generate reasonable control action when measured/estimated states move outside feasible range because of disturbances or noise.
- In industrial implementations, typically, state constraints are **softened**.



## Vehicle Motion Control: Safety Objective

- Slack variables ( $\lambda_k$ ) are, just like states and inputs, free variables the optimizer can alter and soften the constraint.
- Violating constraints is **not cheap** and should be penalized accordingly.

Original problem:

$$\min_{U \triangleq \{u_{k|k}, u_{k+1|k}, \dots\}} J(\mathbf{x}_k, U) = \sum_{k=0}^{N-1} J_{stage}(x_k, u_k)$$

$$s.t. \quad x_{k+1} = f(x_k, u_k),$$

$$c_k(x_k, u_k) \leq 0,$$

$$x \in \mathcal{X},$$

$$u \in \mathcal{U}.$$

“Softened” problem:

$$\min_{U \triangleq \{u_{k|k}, u_{k+1|k}, \dots\}} J(\mathbf{x}_k, U) = \sum_{k=0}^{N-1} J_{stage}(x_k, u_k, \lambda_k)$$

$$s.t. \quad x_{k+1} = f(x_k, u_k),$$

$$c_k(x_k, u_k, \lambda_k) \leq 0,$$

$$x \in \mathcal{X},$$

$$u \in \mathcal{U},$$

$$\lambda \in \Lambda.$$



## Vehicle Motion Control: Safety Objective/Cost function

The safety objective function:

Can use the constraint violation => slack to estimate

$$J_{safety} = \lambda_{soft}^T E \lambda_{soft} + \bar{H} \lambda_{hard}$$

with  $E = \text{diag}(w_{\lambda_{n,soft}}, w_{\lambda_{v,soft}}, w_{\lambda_{a,soft}})$ ,  $\bar{H} = [w_{\lambda_n}, w_{\lambda_a}, w_{\lambda_s}]$ .

$$\lambda_{soft} = \begin{bmatrix} \lambda_{n,soft} \\ \lambda_{v,soft} \\ \lambda_{a,soft} \end{bmatrix}$$

Lateral error violation  
speed violation  
acceleration violation

$$\lambda_{hard} = \begin{bmatrix} \lambda_n \\ \lambda_a \\ \lambda_s \end{bmatrix}$$



## Vehicle Motion Control: Objective/Cost function

The full objective function:

$$J_{tracking} = w_{\dot{s}} \cdot (\dot{s} - \dot{s}_{ref}) + w_n \cdot n^2 + w_{\mu} \cdot (\mu + \beta)^2$$

$$J_{comfort} = w_{j_x} \cdot j_x^2 + w_{a_x} \cdot a_x^2 + w_{a_y} \cdot a_y^2 + w_{\dot{\delta}} \cdot \dot{\delta} + w_{\ddot{\delta}} \cdot \ddot{\delta}^2$$

$$J_{safety} = \lambda_{soft}^T \mathbf{E} \lambda_{soft} + \bar{\mathbf{H}} \lambda_{hard}$$

Is that enough?



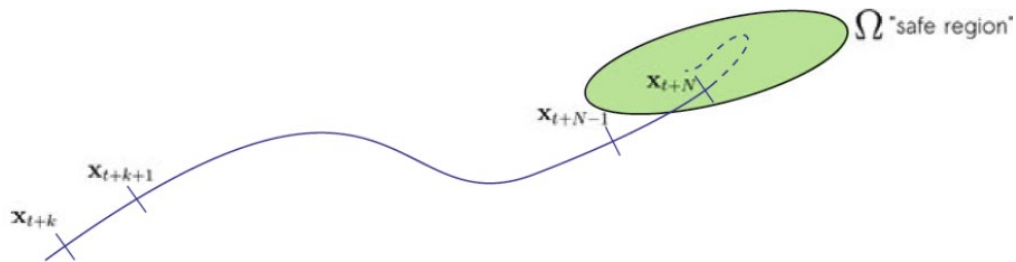
# Vehicle Motion Control: Terminal cost and constraint

## What can go wrong with “standard-finite-horizon” MPC?

- No feasibility guarantee, i.e., the MPC problem may not have a solution.  
Example: inclusion of some “hard-to-achieve” constraints.
- No stability guarantee, i.e., trajectories may not converge to the origin.  
Example: short horizon.

## How to solve?

→ Introduce terminal cost and constraint to explicitly ensure feasibility and stability.





## Vehicle Motion Control: Terminal Objective/Cost function

- The terminal cost is the cost imposed on the **final stage** of the finite prediction horizon and is generally different from regular stage objectives.
- Aim to improve closed loop stability and overall cost by designing a terminal cost that accounts for the lack of an infinite optimization horizon.
- In many cases, nonlinear MPC does not need a terminal cost and/or constraint for closed loop stability, but this is generally hard to prove.

$$\begin{aligned} \min_{U \triangleq \{u_{k|k}, u_{k+1|k}, \dots\}} J(\mathbf{x}_k, U) &= \sum_{k=0}^{N-1} J_{stage}(x_k, u_k, \lambda_k) + J_{terminal}(x_N) \\ s. t. \quad &x_{k+1} = f(x_k, u_k), \\ &c_k(x_k, u_k, \lambda_k) \leq 0, \\ &x \in \mathcal{X}, \\ &u \in \mathcal{U}, \\ &\lambda \in \Lambda. \end{aligned}$$



## Vehicle Motion Control: Terminal cost

### Choice of Terminal Set and Cost:

- Terminal constraint provides a sufficient condition for stability.
- In practise: Enlarge horizon and check stability by sampling.
- For small horizon length  $N$ , terminal constraint can be chosen as 0 for simplest choice but small RoA (region of attraction).
- For large horizon length  $N$ , RoA approaches maximum control invariant set.
- RoA without terminal constraint maybe larger than for MPC with terminal constraint but characterization of RoA extremely difficult.

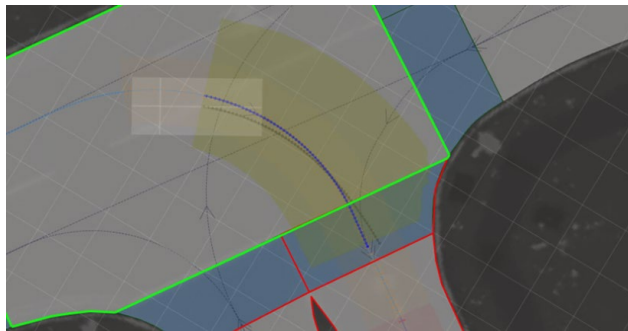




## Vehicle Motion Control: Objective/Cost function

最终成本是在最终阶段与参考路径(运动方向)不对齐的高成本

- The terminal cost imposes a high cost on not being aligned with the reference path (direction of movement) in the final stage.



- The terminal objective “ensures” that we are following the reference path closely enough to act on information beyond the control horizon.

终端目标 “ 确保 ” 我们足够紧密地跟随参考路径，以便对超出控制范围的信息采取行动。



# Vehicle Motion Control

Inequality constraints and slack





# Vehicle Motion Control: Inequality constraints

线性状态和输入不等式约束

Linear state and input inequality constraints

$$\mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{x}_{\max}$$

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max}$$

General nonlinear inequality constraints

$$\mathbf{c}^{\text{station}}(\mathbf{x}, \lambda_s) \leq 0 \quad \Longrightarrow \quad \text{Collision constraint derived from hard upper speed bound.}$$

$$\mathbf{c}^{\text{vel}}(\mathbf{x}, \lambda_{v, \text{soft}}) \leq 0 \quad \Longrightarrow \quad \text{Speed constraint derived from soft upper speed bound.}$$

$$\mathbf{c}^{\text{tube, hard}}(\mathbf{x}, \lambda_n) \leq 0 \quad \Longrightarrow \quad \text{Spatial soft/hard tube constraints imposed on car footprint.}$$
$$\mathbf{c}^{\text{tube, soft}}(\mathbf{x}, \lambda_{n, \text{soft}}) \leq 0$$

$$\mathbf{c}^{\text{a, hard}}(\mathbf{x}, \lambda_a) \leq 0 \quad \Longrightarrow \quad \text{Combined acceleration constraint on hard bounds.}$$

$$\mathbf{c}^{\text{a, soft}}(\mathbf{x}, \lambda_{a, \text{soft}}) \leq 0 \quad \Longrightarrow \quad \text{Acceleration constraint on comfort bounds.}$$

$$\mathbf{c}^{\dot{\delta}}(\mathbf{x}) \leq 0 \quad \Longrightarrow \quad \text{Speed dependent steering rate constraint.}$$



# Vehicle Motion Control: Inequality constraints and slack

松弛的

$$c^{\text{station}}(\mathbf{x}, \lambda_s) \leq 0$$

- The hard speed constraint is integrated to obtain a station constraint, representing a collision constraint. For the hard bound, a station constraint is preferred, since speed constraint violation introduces integrated station error. This approach of obtaining the station constraint is not ideal since it does not guarantee the original speed constraint to be satisfied. Preferably, MPC receives the true station constraints imposed by the environment.

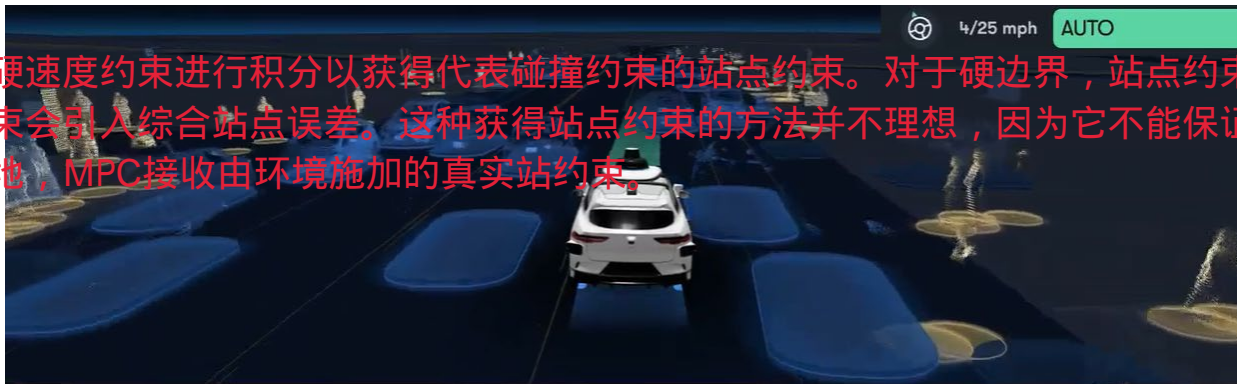
State Variables:

$$\mathbf{x} = \begin{bmatrix} s \\ n \\ \mu \\ v \\ a \\ \delta \\ \dot{\delta} \end{bmatrix}$$

$$c_k^{\text{station}}(\mathbf{x}_k, \lambda_s) = \begin{cases} s_k - s_k^{\max} - \lambda_{s,k} & \leq 0 \\ -s_k + s_k^{\min} - \lambda_{s,k} & \leq 0 \end{cases}, \quad \forall k \in \{0, \dots, N\}$$

Input variable:

$$\mathbf{u} = \begin{bmatrix} u_{\text{jerk}} \\ u_{\ddot{\delta}} \end{bmatrix}$$



对硬速度约束进行积分以获得代表碰撞约束的站点约束。对于硬边界，站点约束是优选的，因为违反速度约束会引入综合站点误差。这种获得站点约束的方法并不理想，因为它不能保证满足原始的速度约束。优选地，MPC接收由环境施加的真实站约束。



## Vehicle Motion Control: Inequality constraints and slack

$$c^{\text{vel}}(\mathbf{x}, \lambda_{v, \text{soft}}) \leq 0$$

Constraints are modeled such that they allow for slack at a certain cost.

$$v - \lambda \leq v_{\max}$$

State Variables:

$$\mathbf{x} = \begin{bmatrix} s \\ n \\ \mu \\ v \\ a \\ \delta \\ \dot{\delta} \end{bmatrix}$$

Input variable:

$$\mathbf{u} = \begin{bmatrix} u_{\text{jerk}} \\ u_{\delta} \end{bmatrix}$$

The existence of slack variables is motivated by:

- Due to suddenly changing environments/constraints, being outside of the feasible region is inevitable.
- When entering the infeasible region is the only option, we still want MPC to succeed in finding a solution that makes a trade-off in constraint violations.
- Slack helps the solver converge when initialized outside of the feasible region.

松弛变量的存在是由以下因素引起的:

由于环境/限制的突然变化, 不可避免地会超出可行区域。

当进入不可行区域是唯一的选择时, 我们仍然希望MPC能够成功地找到一个解决方案, 在违反约束的情况下进行权衡。

在可行区域之外初始化时, 松弛有助于求解器收敛。



# Vehicle Motion Control: Inequality constraints and slack

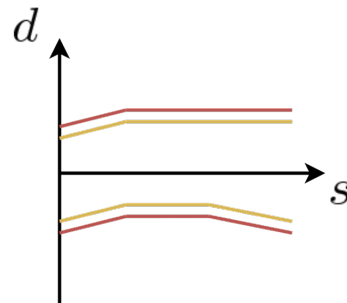
$$\begin{aligned} c^{\text{tube,hard}}(\mathbf{x}, \lambda_n) &\leq 0 \\ c^{\text{tube,soft}}(\mathbf{x}, \lambda_{n,\text{soft}}) &\leq 0 \end{aligned}$$

## Spatial tube constraints 空间管约束

Lateral position and shift of corners due to rotation of the car wrt the reference path

Width of the tube at the corners

$$c^{\text{tube,hard}}(\mathbf{x}, \lambda_n) = \begin{cases} n_k + \underline{d_{\text{car,left,front}}(\mu_k)} - \underline{d_{\text{road,left,front,k}}} - \lambda_{n,k} & \leq 0 \\ n_k + \underline{d_{\text{car,left,rear}}(\mu_k)} - \underline{d_{\text{road,left,rear,k}}} - \lambda_{n,k} & \leq 0 \\ -n_k + \underline{d_{\text{car,right,front}}(\mu_k)} + \underline{d_{\text{road,right,front,k}}} - \lambda_{n,k} & \leq 0 \\ -n_k + \underline{d_{\text{car,right,rear}}(\mu_k)} + \underline{d_{\text{road,right,rear,k}}} - \lambda_{n,k} & \leq 0 \end{cases}, \quad \forall k \in \{0, \dots, N\}$$



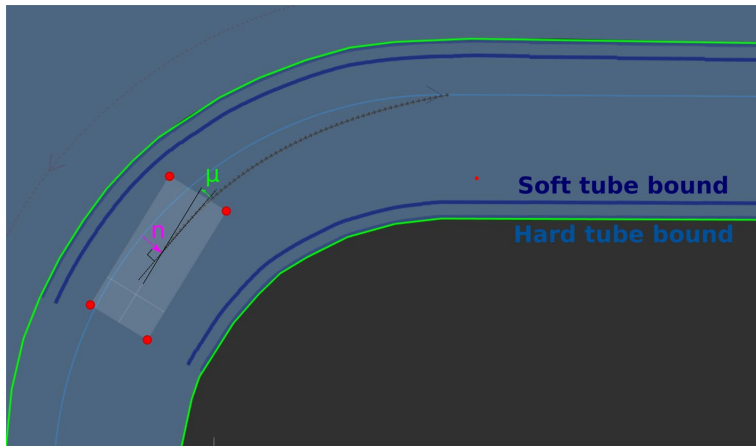
Spatial tube constraints

State Variables:

$$\mathbf{x} = \begin{bmatrix} s \\ n \\ \mu \\ v \\ a \\ \delta \\ \dot{\delta} \end{bmatrix}$$

Input variable:

$$\mathbf{u} = \begin{bmatrix} u_{\text{jerk}} \\ u_{\ddot{\delta}} \end{bmatrix}$$



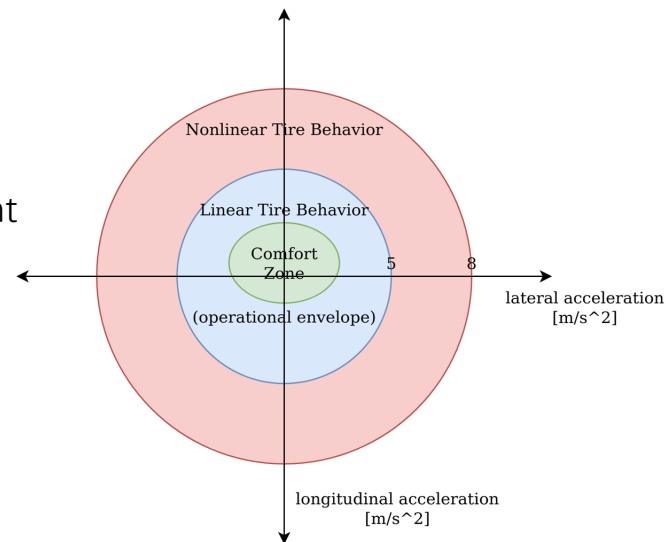


# Vehicle Motion Control: Inequality constraints and slack

$$\mathbf{c}^{a,\text{hard}}(\mathbf{x}, \lambda_a) \leq 0$$

## Max Acceleration constraint:

- Vehicle dynamics are naturally defined in total accelerations (lateral and longitudinal combined).
- An elliptical constraint is formulated to capture the constraint on total accelerations.



State Variables:

$$\mathbf{x} = \begin{bmatrix} s \\ n \\ \mu \\ v \\ a \\ \delta \\ \dot{\delta} \end{bmatrix}$$

Input variable:

$$\mathbf{u} = \begin{bmatrix} u_{jerk} \\ u_{\dot{\delta}} \end{bmatrix} \quad \mathbf{c}_k^{a,\text{hard}}(\mathbf{x}_k, \lambda_a) = \sqrt{\frac{a_y^2}{a_{y,\text{max},k}^2} + \frac{a_x^2}{a_{x,\text{max},k}^2}} - 1 - \lambda_a \leq 0, \quad \forall k \in \{0, \dots, N\}$$



# Vehicle Motion Control: Inequality constraints and slack

$$\mathbf{c}^{a,\text{soft}}(\mathbf{x}, \lambda_{a,\text{soft}}) \leq 0$$

## Comfort acceleration constraint

- Accelerations in the driving direction within comfortable bounds should be free to execute.
- Accelerations beyond comfort level are penalized with a soft bound on forward and backward accelerations individually.

State Variables:

$$\mathbf{x} = \begin{bmatrix} s \\ n \\ \mu \\ v \\ \textcolor{blue}{a} \\ \delta \\ \dot{\delta} \end{bmatrix}$$

$$\mathbf{c}_k^{a,\text{soft}}(\mathbf{x}_k, \lambda_{a,\text{soft}}) = \begin{cases} a_k - a_{\text{comfort}} - \lambda_{a,\text{soft}} & \leq 0 \\ -a_k - \text{decel}_{\text{comfort}} - \lambda_{a,\text{soft}} & \leq 0 \end{cases}, \quad \forall k \in \{0, \dots, N\}$$

Input variable:

$$\mathbf{u} = \begin{bmatrix} u_{\text{jerk}} \\ u_{\dot{\delta}} \end{bmatrix}$$





# Model Predictive Control Formulation: Summary

MPC solves the following Optimal Control Problem for every control iteration

Optimal sequence control actions

$$\min_{U \triangleq \{u_{k|k}, u_{k+1|k}, \dots\}}$$

$$J(\mathbf{x}_k, U) = \sum_{k=0}^{N-1} J_{stage}(\mathbf{x}_k, u_k, \lambda_k) + J_{terminal}(\mathbf{x}_N)$$

Objective

s. t.

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k),$$

Process model

$$c_k(\mathbf{x}_k, u_k, \lambda_k) \leq 0,$$

Inequality constraints

$$\mathbf{x} \in \mathcal{X},$$

State constraints

$$u \in \mathcal{U},$$

Actuation limits

$$\lambda \in \Lambda.$$

State Variables:

$$\mathbf{x} = \begin{bmatrix} s \\ n \\ \mu \\ v \\ a \\ \delta \\ \dot{\delta} \end{bmatrix}$$

Input variable:

$$\mathbf{u} = \begin{bmatrix} u_{jerk} \\ u_{\ddot{\delta}} \end{bmatrix}$$

- MPC aims to find an optimal solution over a finite discrete (time) horizon of  $N$  steps.



# Vehicle Motion Control

MPC solver





## Different MPC solvers

Software	Year	Target	License
ACADO Codegen	2009	NMPC	LGPL v3
qpOASES	2006	QP	LGPL v2.1
FORCES	2011	QP, QCQP	proprietary
FORCES NLP	2017	NMPC	proprietary
OSQP	2017	QP	Apache v2.0
acados	2018	NMPC	2-clause BSD



Part III

# Vehicle Motion Control

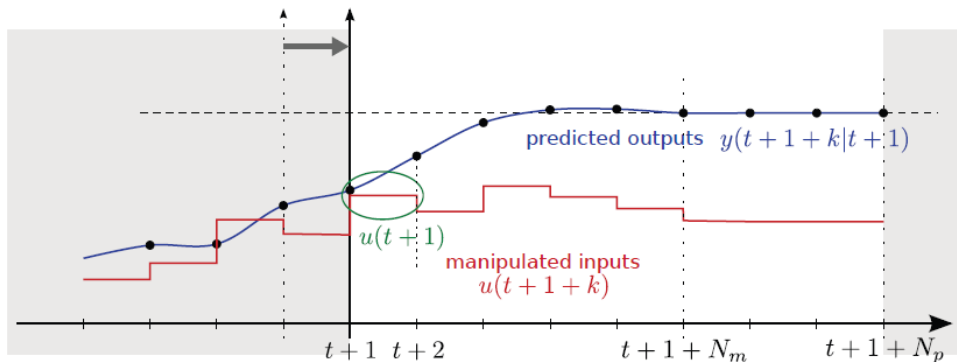
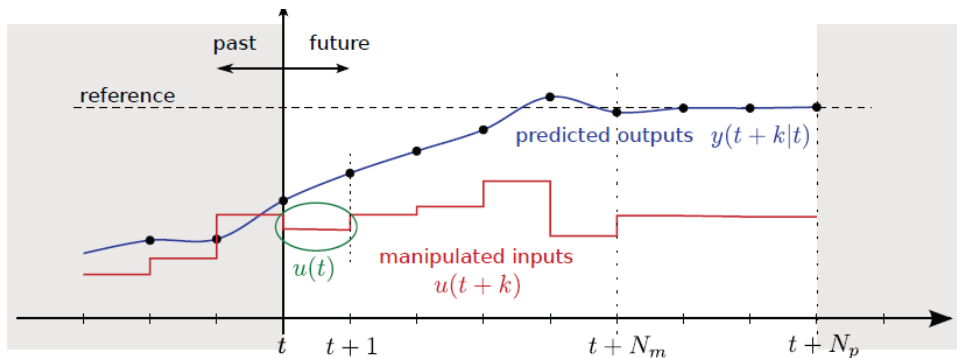
How to improve the computation time?





# MPC complexity

MPC



QP Problem  
(QP: Quadratic Programming)

$$\begin{aligned} \min_z \quad & \frac{1}{2} \mathbf{z}^T \mathbf{H} \mathbf{z} + \mathbf{x}_0^T \mathbf{F} \mathbf{z} \\ \text{s.t.} \quad & \mathbf{G} \mathbf{z} \leq \mathbf{W} + \mathbf{S} \mathbf{x}_0 \end{aligned}$$

- Number of **states**
- Number of **constraints**
- Length of the **horizon**



MPC complexity



## MPC complexity

### Autonomous vehicle application

- Fast and real time: sampling time is in milliseconds scale
- Limited memory: cost of vehicle

### How to make MPC faster?

- Reduce model order
- Shorter horizon
- Reduced Number of constraints
- Lower Precision Operations and data representation

### Any systematic method?

- Explicit MPC



显式MPC

## Explicit MPC

Can we implement constrained linear MPC **without an online QP solver**?

- **Online** optimization: given  $\mathbf{x}(t)$ , solve the problem at each time step  $t$   
(the control law  $\mathbf{u} = \mathbf{u}_0^*(\mathbf{x})$  is **implicitly** defined by the QP solver)

→ Quadratic Programming (QP)

- **Offline** optimization: solve the QP in advance for all  $\mathbf{x}(t)$  in a given range to find the control law  
 $\mathbf{u} = \mathbf{u}_0^*(\mathbf{x})$  **explicitly**

→ Multi-parametric Quadratic Programming (mpQP)



## Explicit MPC

Example

$$J^*(x) = \min_z J(z, x) = \frac{1}{2}z^2$$

$$\text{s. t.} \quad z \leq 1 + 3x$$

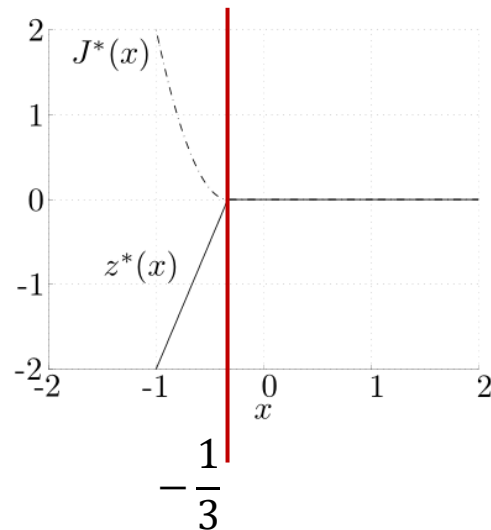
KKT condition:

$$z + \lambda = 0$$

$$\lambda(z - 3x - 1) = 0$$

$$\lambda \geq 0$$

$$z - 3x - 1 \leq 0$$



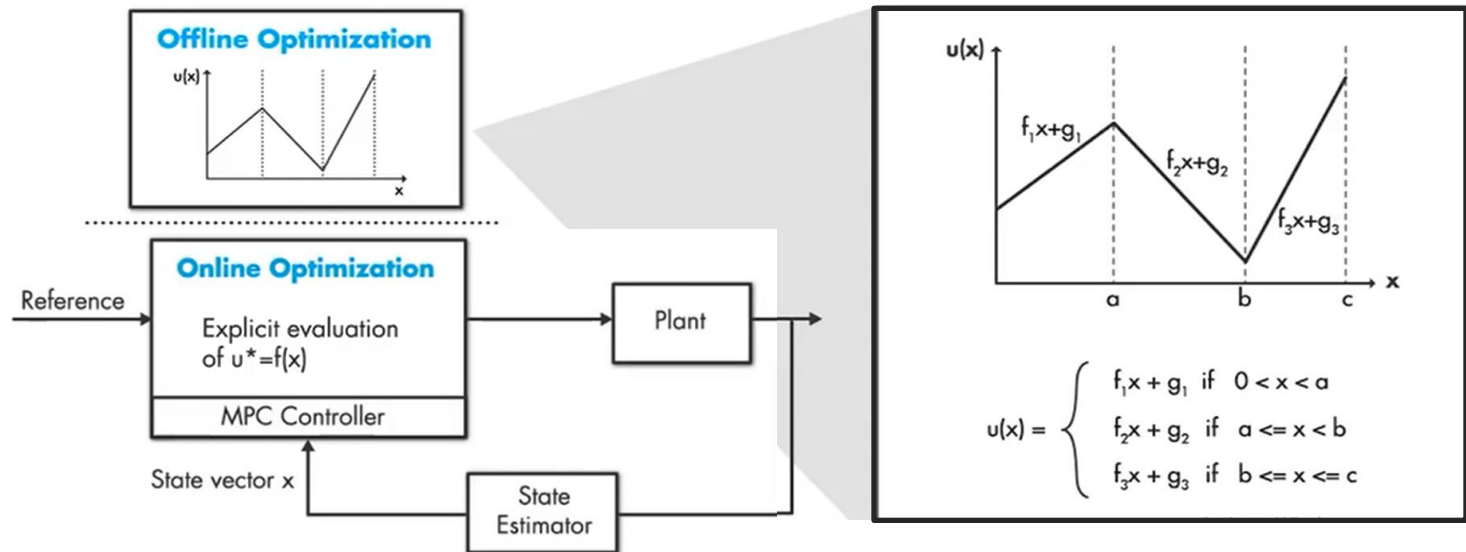
It can be proved that the multiparametric solution of a strictly convex QP is **continuous** and **piecewise affine**.





# Explicit MPC

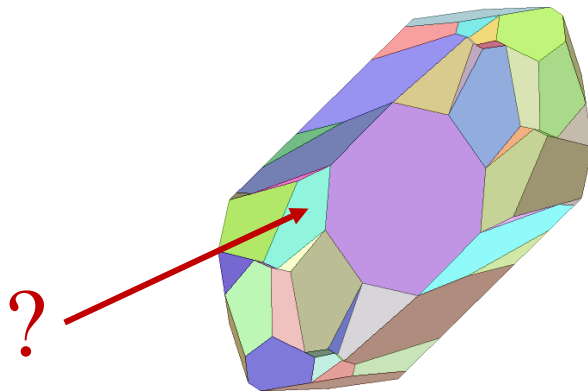
## EXPLICIT MPC





## Explicit MPC

The number of regions is (usually) **exponential** with the number of possible **combinations of active constraints**.



Too many regions make explicit MPC less attractive, due to memory (storage of polyhedra) and **throughput** requirements (time to locate  $x_0$ ).



# Vehicle Motion Control

A simple example





## Model Predictive Control: A simple example (Formulation)

MPC solves the following Optimal Control Problem for every control iteration

**Optimal sequence control actions**

**Objective**

$$\min_{U \triangleq \{u_{k|k}, u_{k+1|k}, \dots\}} J(\mathbf{x}_k, U) = \sum_{k=0}^{N-1} J_{stage}(\mathbf{x}_k, u_k, \lambda_k) + J_{terminal}(\mathbf{x}_N)$$

**s. t.**

**Process model**

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

**Inequality constraints**

$$c_k(x_k, u_k, \lambda_k) \leq 0,$$

**State constraints**

$$x \in \mathcal{X},$$

**Actuation limits**

$$u \in \mathcal{U},$$
$$\lambda \in \Lambda.$$

- MPC aims to find an optimal solution over a finite discrete (time) horizon of  $N$  steps.

# Model Predictive Control: A simple example (Model)

## Linear Dynamic Bicycle Model:

Trade off between model accuracy and solve time

$$\dot{x} = v_x \cos(\psi) - v_y \sin(\psi)$$

$$\dot{y} = v_x \sin(\psi) + v_y \cos(\psi)$$

$$m\dot{v}_x = mv_y r + (F_{r,x} - F_{f,y} \sin(\delta))$$

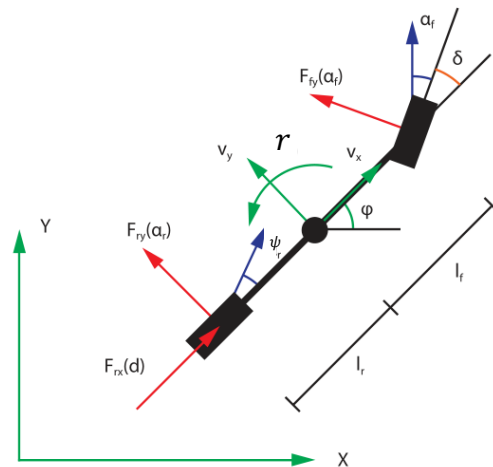
$$m\dot{v}_y = F_{r,y} + F_{f,y} \cos(\delta) - mv_x r$$

$$I_z \dot{r} = F_{f,y} l_f \cos(\delta) - F_{r,y} l_r$$

Recall tire force

$$F_{yff} = 2c_f(\delta - \theta_{vf})$$

$$F_{yrf} = 2c_r(-\theta_{vr})$$





## Model Predictive Control: A simple example (Objective/Cost function)

The objective function is our main mechanism to **shape driving behavior**.

The objective function has different types of components:

- **Tracking** objectives (Reference from planner)
- **Comfort** objectives (Comfortability of vehicle motion, perceived danger)

Different weights can be applied to separate cost terms in order to favor a certain trade-off.

$$J = w_t \cdot J_{tracking} + w_c \cdot J_{comfort}$$

position error,  $e_p$  + velocity error,  $e_v$  + heading error,  $e_\phi$

control input: steering,  $\delta, \dot{\delta}$  + accel,  $a, \dot{a}$

where

$w_t$  is the weight for tracking cost,

$w_c$  is the weight for comfort cost,



# Model Predictive Control: Summary

## Advantages of MPC

- Straightforward formulation
- Explicitly handles constraints
- Applicable to linear or nonlinear models

## Disadvantages of MPC

- Computationally expensive
- May or may not be stable
- May or may not be feasible

Thanks for Listening !