深蓝学院
shenlanxueyuan.com

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
- Objective function
- Constraint generation
- Horizon length
- MPC solver

How to make MPC run faster

Part I

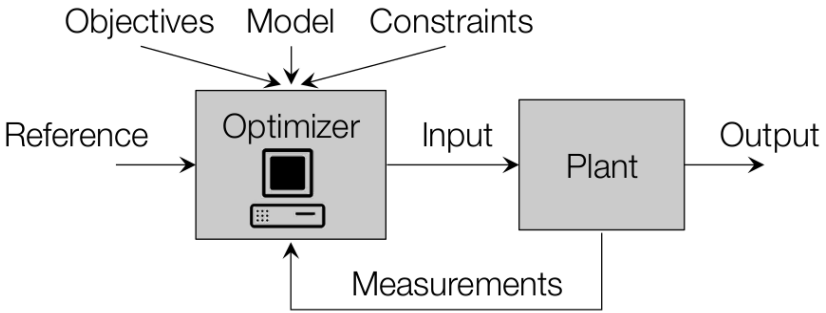
Model Predictive Control

Main concept

Model Predictive Control

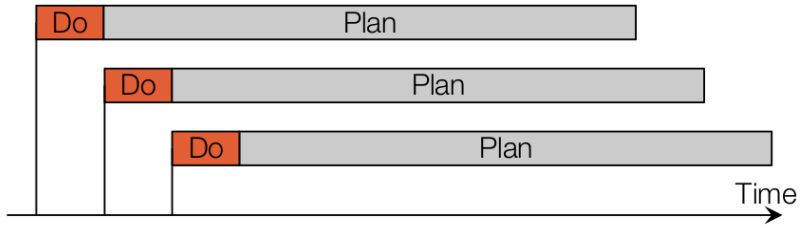
MPC = Model Predictive Control, also known as

- MHC = Moving Horizon Control
- RHC = Receding Horizon Control
- 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

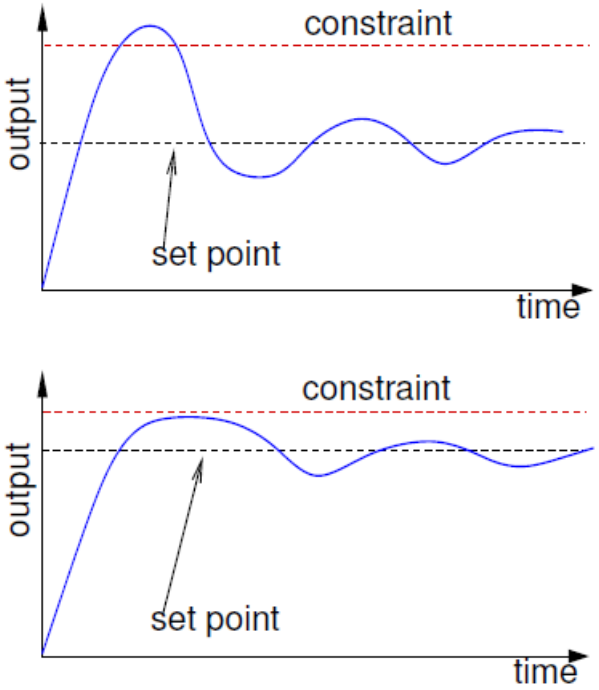
Optimal operating points are often near constraints.

Classical control methods:

- Ad hoc constraint management
- Set point sufficiently far from constraints
- Suboptimal plant operation

Predictive control:

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



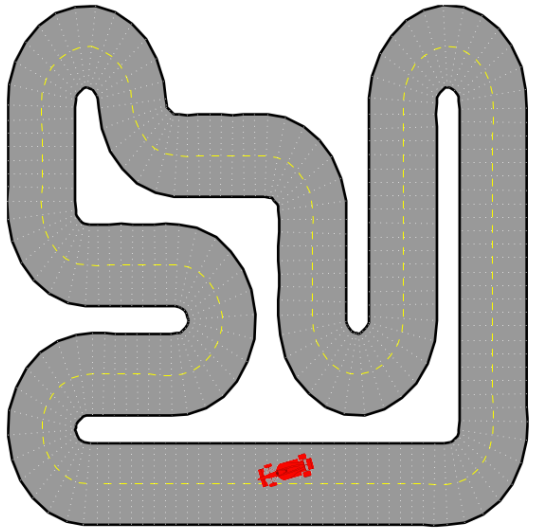
Model Predictive Control Main Concept: Why MPC?

Objective:

- Minimize lap time

Constraints:

- Avoid other cars
- Stay on road
- Don’t skid
- Limited acceleration



Model Predictive Control Main Concept : Why MPC?

Objective:

- Minimize lap time

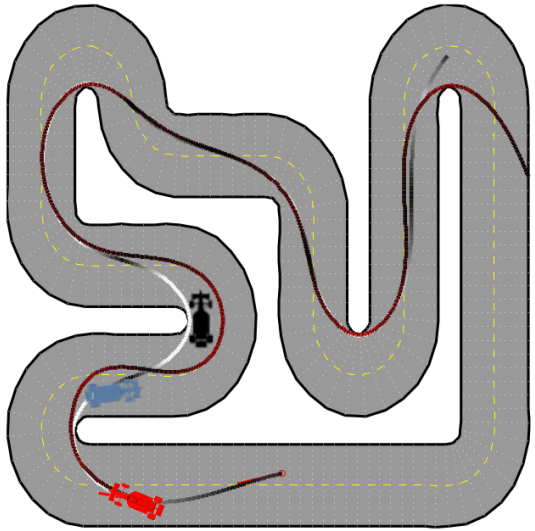
Problem:

What to do if something unexpected happens?

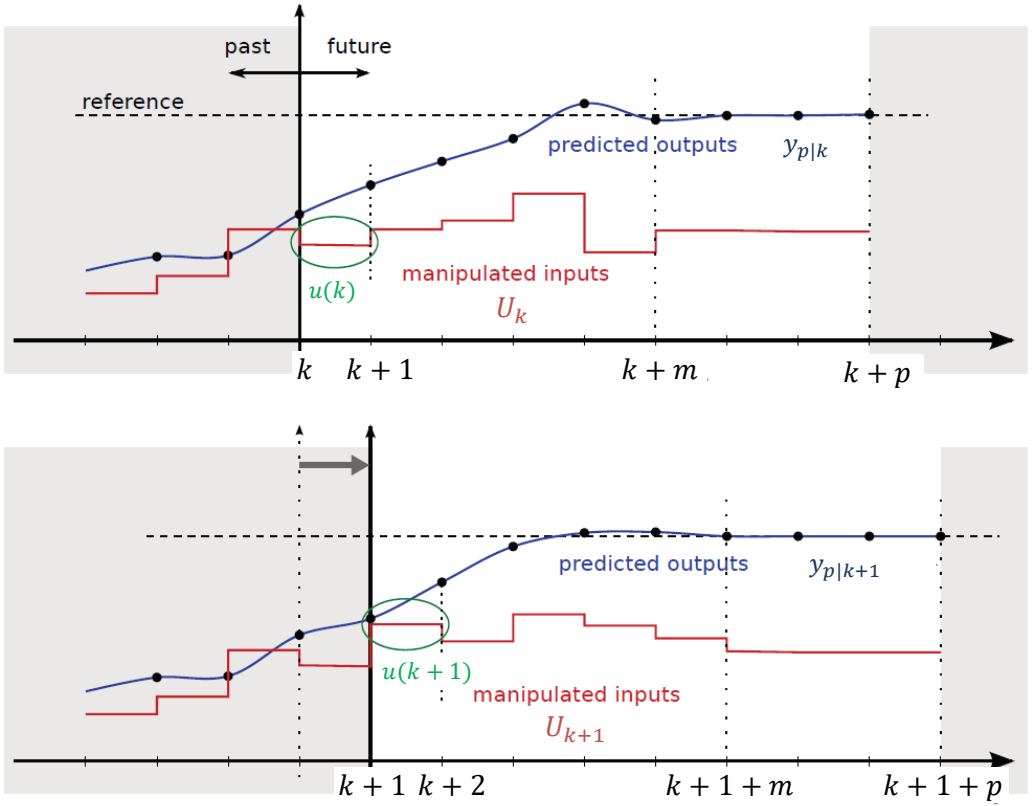
- We didn't see a car around the corner!
- Must introduce feedback

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:

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

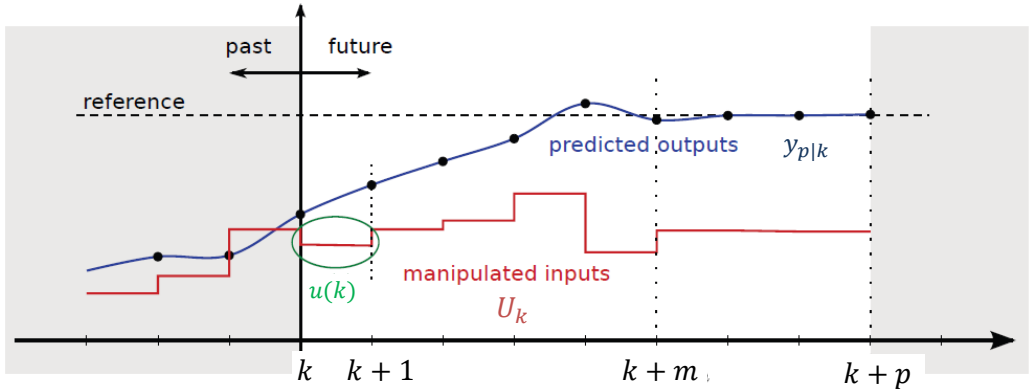
The reference r_k:

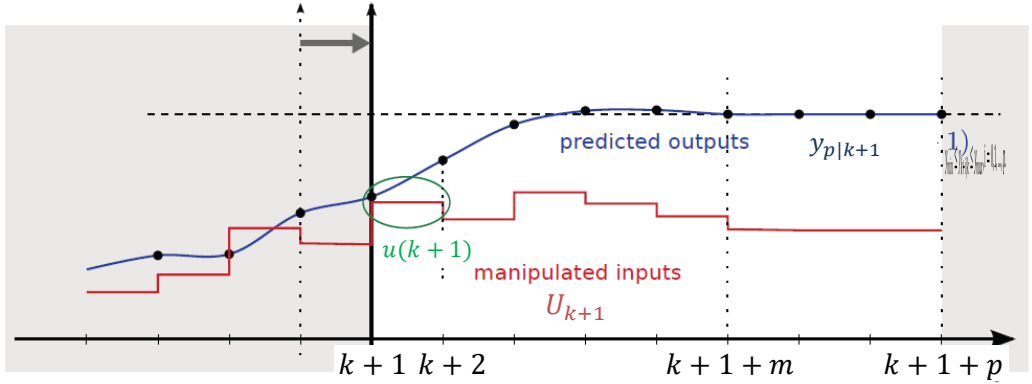
$$\{r_{k+1}, r_{k+2}, \dots, r_{k+p}\}$$

The cost function:

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

Model Predictive Control: Simple Case study





Optimal problem can be described as

$$\min_{U_k} J(\boldsymbol{y_k}, U)$$

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

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

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

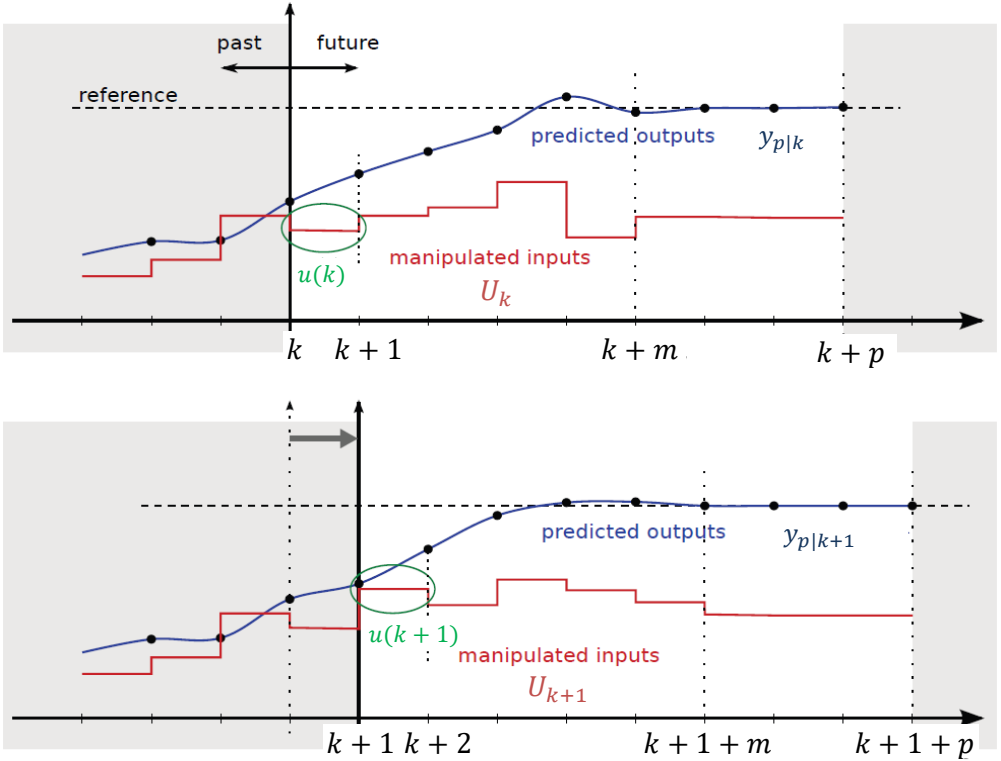
$$y_{min} \le y_{k+i|k} \le y_{max}, \mathrm{i} = 0,1, \dots, \mathrm{p}.$$

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

$$U_k^* = \arg\min_{U_k} J(\boldsymbol{y_k}, U_k)$$

Model Predictive Control: Simple Case study

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



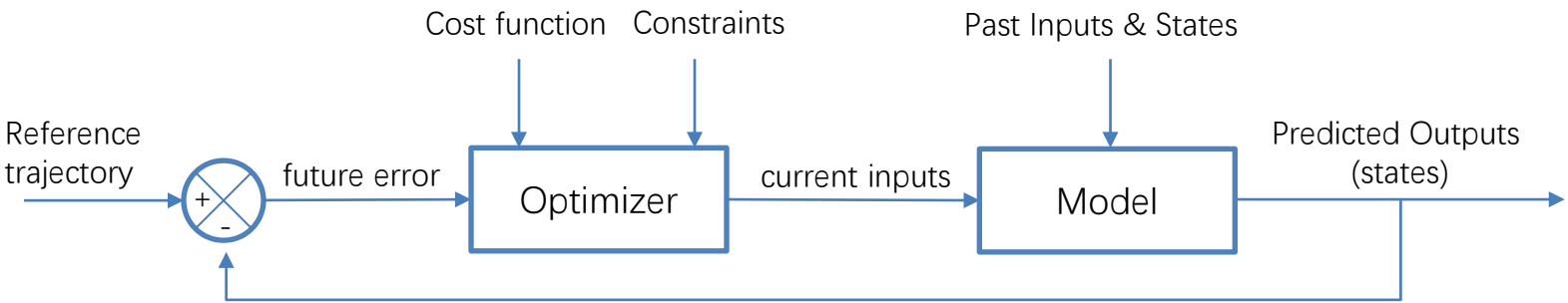
No, only the first one.

- External disturbance and model-plan mismatch
- 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: $\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k \end{cases}$

- Relation between input and states: $\boldsymbol{x}_k = \boldsymbol{A}^k \boldsymbol{x}_0 + \sum_{j=0}^{k-1} \boldsymbol{A}^j \boldsymbol{B} \boldsymbol{u}_{k-1-j}$

- Forward simulation as matrix equation:

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

Linear MPC – Unconstrained Case

Recall LQR cost function

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

- Quadratic costs:

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

$$P = P^T \succ 0$$
$$Q = Q^T \succ 0$$
$$R = R^T \succ 0$$

Positive semi-definite

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

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

Linear MPC – Unconstrained Case

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

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

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

$$= \frac{1}{2} z^T \underbrace{2\left(\bar{R} + \bar{S}^T \bar{Q}\, \bar{S}\right)}_{H} z + x_0^T \underbrace{2\bar{T}^T \bar{Q}\, \bar{S}}_{F} z + \frac{1}{2} x_0^T \underbrace{2\left(Q + \bar{T}^T \bar{Q}\, \bar{T}\right)}_{E} x_0$$

Linear MPC – Unconstrained Case

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

- Condensed form of MPC:

$$J(\boldsymbol{z}, \boldsymbol{x}_0) = \frac{1}{2}\boldsymbol{z}^T \boldsymbol{H}\boldsymbol{z} + \boldsymbol{x}_0^T \boldsymbol{F}\boldsymbol{z} + \frac{1}{2}\boldsymbol{x}_0^T \boldsymbol{E}\boldsymbol{x}_0 \qquad \boldsymbol{z} = \begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{bmatrix}$$

The optimum is obtained by zeroing the gradient

$$\nabla_{\boldsymbol{z}} J(\boldsymbol{z}, \boldsymbol{x}_0) = \boldsymbol{H}\boldsymbol{z} + \boldsymbol{F}^T \boldsymbol{x}_0 = 0 \rightarrow \boldsymbol{z}^* = -\boldsymbol{H}^{-1}\boldsymbol{F}^T \boldsymbol{x}_0$$ ("batch" solution)

unconstrained linear MPC = linear state-feedback!

Wang, Yang, and Stephen Boyd. "Fast model predictive control using online optimization." IEEE Transactions on control systems technology 18.2 (2009): 267-278.

Linear MPC – Unconstrained Case

- Non-condensed form of MPC:
  - Keep also $\boldsymbol{x_1, \dots, x_N}$ as optimization variables

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

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

  - Equality constraints

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

More variables and constraints but very sparse

Wang, Yang, and Stephen Boyd. "Fast model predictive control using online optimization." IEEE Transactions on control systems technology 18.2 (2009): 267-278.

MPC and Linear Quadratic Regulator (LQR)

- "Batch" solution: $\boldsymbol{z}^* = -\boldsymbol{H}^{-1}\boldsymbol{F}^T\boldsymbol{x}_0$
  - Analytically
  - Still requires to invert a large matrix $\boldsymbol{H}^{-1}$

- Bellman's principle of optimality A → ··· → B → ··· → 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."

  - 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

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

$$\boldsymbol{x}_0^T \boldsymbol{Q} \boldsymbol{x}_0 \rightarrow \cdots \rightarrow \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k \rightarrow \cdots \rightarrow \boldsymbol{x}_N^T \boldsymbol{P} \boldsymbol{x}_N$$

$$\boldsymbol{z}^* \qquad \boldsymbol{u}_0^* \quad \ldots \quad \boldsymbol{u}_k^* \quad \ldots \quad \boldsymbol{u}_{N-1}^*$$

MPC and Linear Quadratic Regulator (LQR)

Dynamic programming LQR

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

- Break up the problem into sub-smaller problems

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

$$\boldsymbol{x}_N = \boldsymbol{A}\boldsymbol{x}_{N-1} + \boldsymbol{B}\boldsymbol{u}_{N-1}$$

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

immediate cost

cost-to-go

$$\boldsymbol{u}_{N-1}(\boldsymbol{x}_{N-1}) = \arg\min_{\boldsymbol{u}_{N-1}} \underline{\boldsymbol{x}_{N-1}^T \boldsymbol{Q} \boldsymbol{x}_{N-1} + \boldsymbol{u}_{N-1}^T \boldsymbol{R} \boldsymbol{u}_{N-1}} + \underline{J_N^*(\boldsymbol{A}\boldsymbol{x}_{N-1} + \boldsymbol{B}\boldsymbol{u}_{N-1})}$$

$$= \arg\min_{\boldsymbol{u}_{N-1}} \underbrace{\boldsymbol{x}_{N-1}^T (\boldsymbol{A}^T \boldsymbol{P} \boldsymbol{A} + \boldsymbol{Q}) \boldsymbol{x}_{N-1}}_{\text{const.}} + \underline{\boldsymbol{u}_{N-1}^T (\boldsymbol{B}^T \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R}) \boldsymbol{u}_{N-1} + 2\boldsymbol{x}_{N-1}^T \boldsymbol{A}^T \boldsymbol{P} \boldsymbol{B} \boldsymbol{u}_{N-1}}$$

Zeroing the gradient: $2(\boldsymbol{B}^T \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R}) \boldsymbol{u}_{N-1}^* + 2\boldsymbol{B}^T \boldsymbol{P} \boldsymbol{A} \boldsymbol{x}_{N-1} = 0$

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

MPC and Linear Quadratic Regulator (LQR)

Dynamic programming LQR

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

- Recursively solve the tail problems

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

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

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

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

$$J_{N-2}(x_{N-2}) \triangleq x_{N-2}^T Q x_{N-2} + u_{N-2}^T R u_{N-2} + J_{N-1}^*(x_{N-1})$$

...

$$u_0^*(x_0) = K x_0$$

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

Algebraic Riccati Equation

MPC and Linear Quadratic Regulator (LQR)

Dynamic programming LQR

- Consider again the MPC cost function

$$\min_{z} \; \boldsymbol{x}_N^T \boldsymbol{P} \boldsymbol{x}_N + \sum_{k=0}^{N-1} \left(\boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \right)$$

"Batch" solution

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

$O(N^3)$ complexity

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

$$\boldsymbol{K} = -\left(\boldsymbol{B}^T \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R} \right)^{-1} \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{A}$$
$$\boldsymbol{P} = \boldsymbol{A}^T \boldsymbol{P} \boldsymbol{A} + \boldsymbol{Q} + \boldsymbol{A}^T \boldsymbol{P} \boldsymbol{B} \boldsymbol{K}$$
$$\dots$$
$$\boldsymbol{u}_0^*(\boldsymbol{x}_0) = \boldsymbol{K} \boldsymbol{x}_0$$

→ 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)

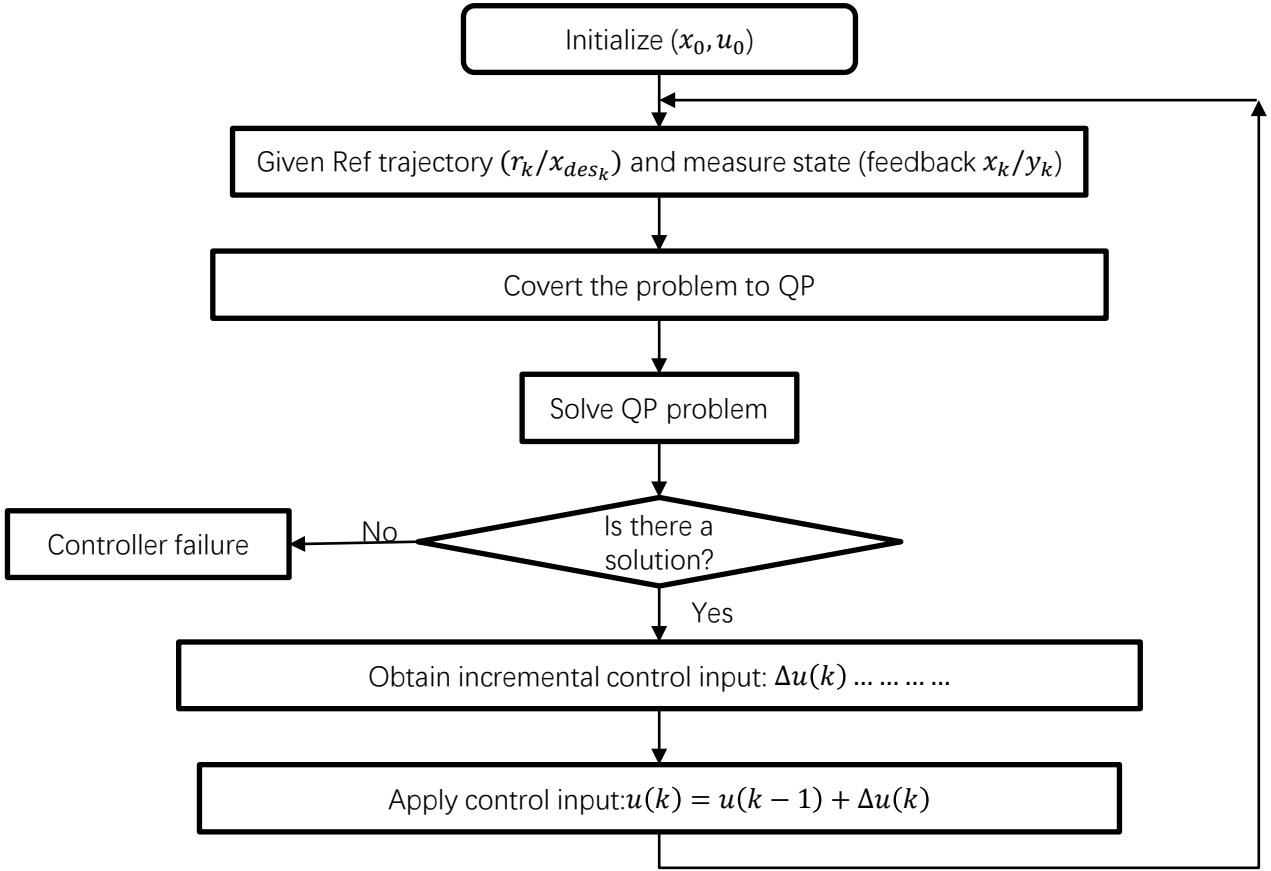
$$J(\boldsymbol{z}, \boldsymbol{x}_0) = \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{x}_0^T\boldsymbol{F}\boldsymbol{z} + \frac{1}{2}\boldsymbol{x}_0^T\boldsymbol{E}\boldsymbol{x}_0$$

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

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

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

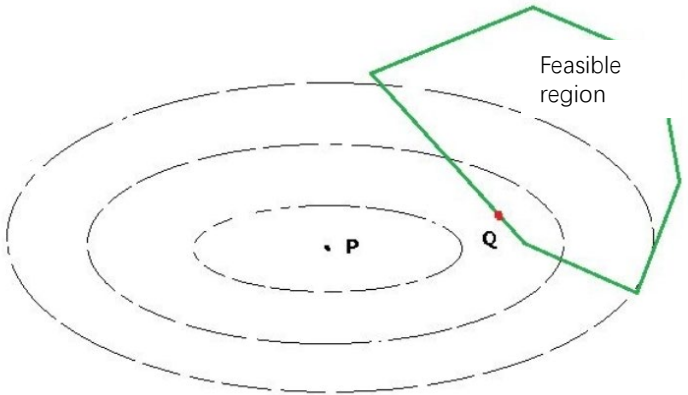
Constrained Linear MPC



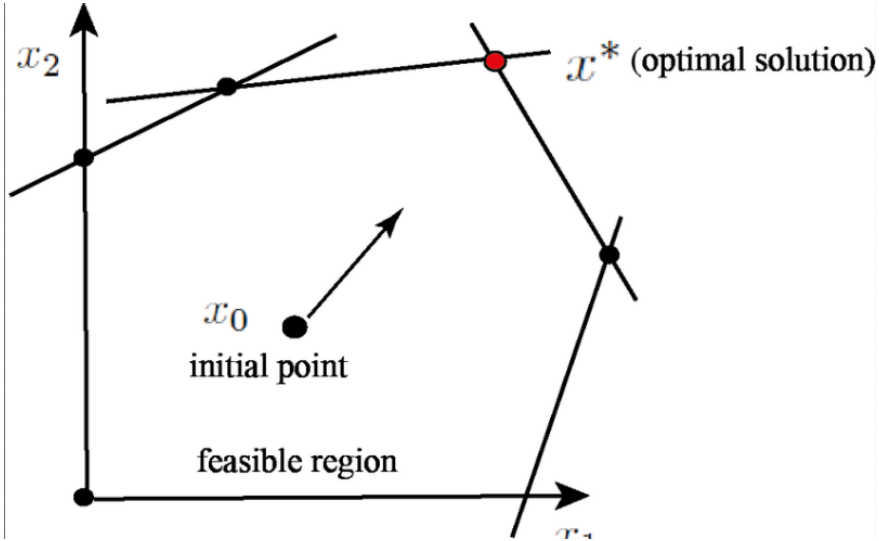
Constrained Linear MPC

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**

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

$$s.t. \quad x_{k+1} = f(x_k, u_k), \qquad \text{Nonlinear process model}$$

$$c_k(x_k, u_k) \le 0, \qquad \text{inequality constraints}$$

$$h_k(x_k, u_k) = 0, \qquad \text{equality constraints}$$

$$x \in \mathcal{X}, \qquad \text{State constraints}$$

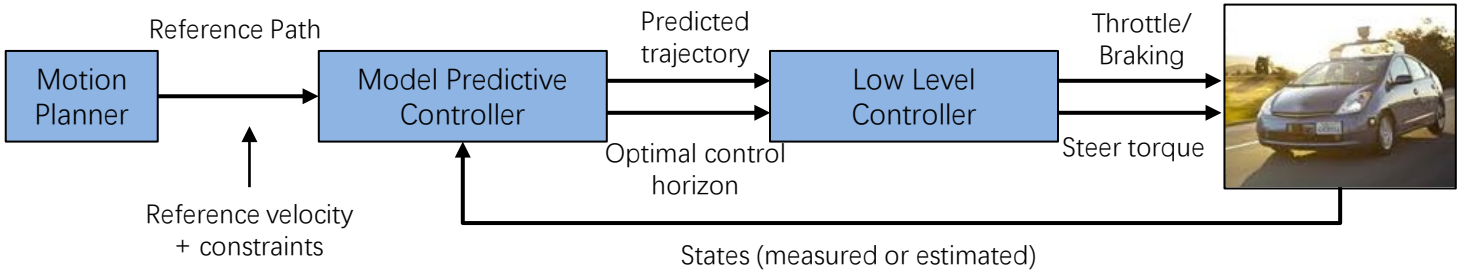
$$u \in \mathcal{U}, \qquad \text{input constraints}$$

- **No closed form solution, must be solved numerically**

Part II

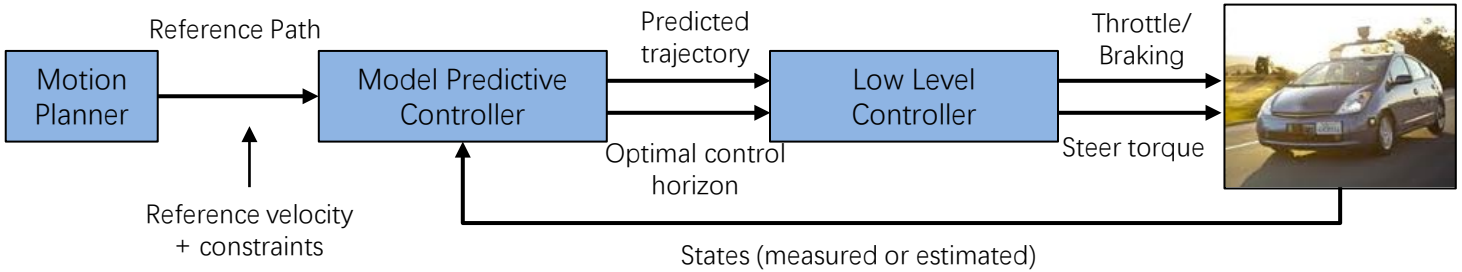
Vehicle Motion Control

Vehicle Motion Control: Architecture

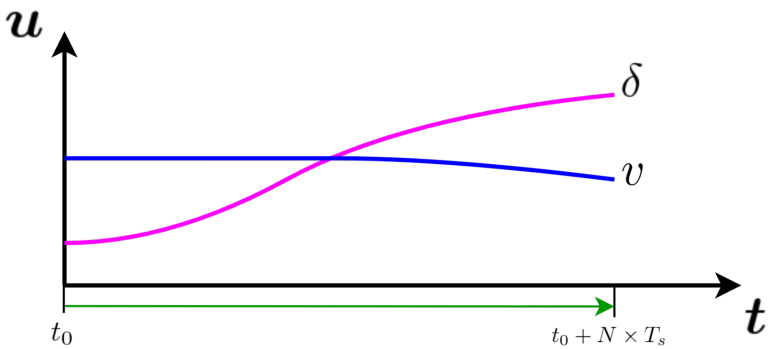


- The presented MPC framework performs **iterative trajectory optimization** over a finite time horizon.
- A **motion model of the vehicle** as process model, allows MPC to understand evolution of the vehicle state.
- Planner's bounds are encoded by **constraints**.
- The **objective function** shapes desired driving behavior in the most fundamental way possible.

Vehicle Motion Control: Output



- MPC outputs a **finite optimal control horizon** corresponding to the optimal trajectory for every control iteration.
- 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



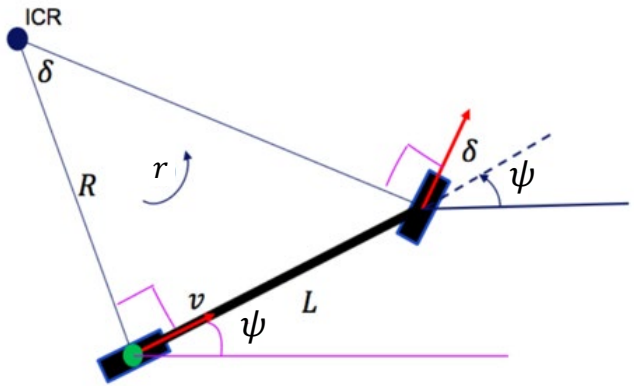
Vehicle Motion Control: Model Predictive Controller

- Model – used to propagate the system to predict the future information
- Cost function – Minimize
  - Deviation from desired trajectory
  - Minimization of control command magnitude, etc.
- Constraints – subject to
  - Longitudinal and later dynamic models
  - Tire force limits,etc.
- Can incorporate low level controller, adding constraints for:
  - Engine map
  - Full dynamic vehicle model
  - Actuator models
  - Tire force models

Vehicle Motion Control

Model

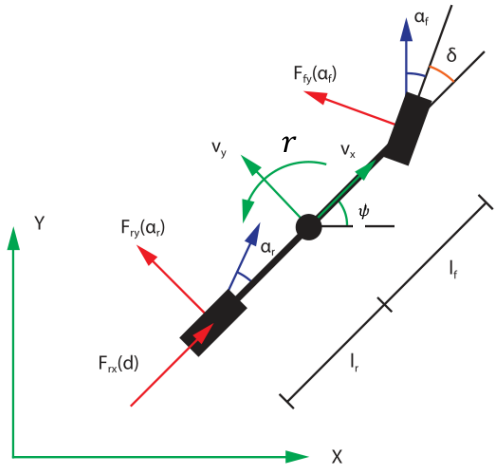
Vehicle Model (review)



$\boldsymbol{P}(p_x, p_y)$

$$\begin{cases} \dot{p_x} = v\cos(\psi) \\ \dot{p_y} = v\sin(\psi) \\ \dot{\psi} = \dfrac{v}{L}\tan(\delta) \\ \dot{v} = a \end{cases}$$

Kinematic Bicycle Model



$$\dot{x} = v_x\cos(\psi) - v_y\sin(\psi)$$
$$\dot{y} = v_x\sin(\psi) + v_y\cos(\psi)$$
$$\dot{\varphi} = r$$
$$\dot{v_x} = \frac{1}{m}\left(F_{r,x} - F_{f,y}\sin(\delta) + mv_y r\right)$$
$$\dot{v_y} = \frac{1}{m}\left(F_{r,y} + F_{f,y}\cos(\delta) - mv_x r\right)$$
$$\dot{r} = \frac{1}{I_z}\left(F_{f,y}l_f\cos(\delta) - F_{r,y}l_r\right)$$

Dynamic Bicycle Model

Simplicity

VS.

Accuracy

Curvilinear Coordinates Vehicle Model

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:

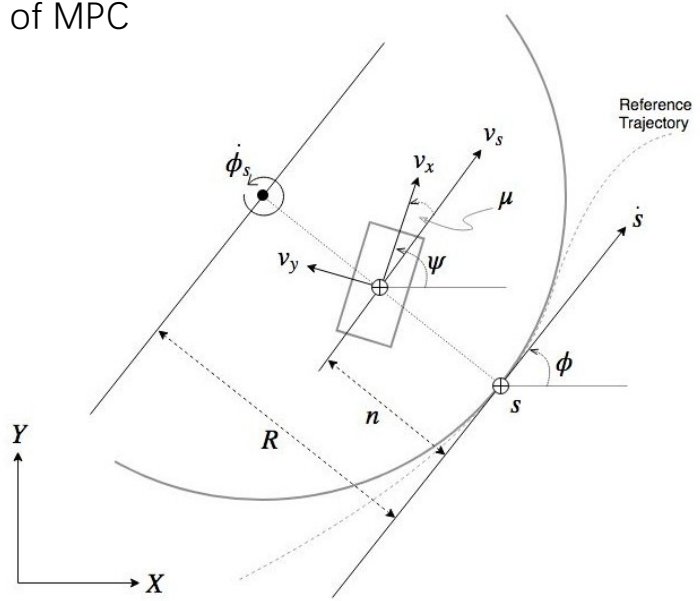
$$x = \begin{bmatrix} s \\ n \\ \mu \\ v \\ a \\ \delta \\ \dot{\delta} \end{bmatrix}$$

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

Input variable:

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

Jerk
Steering accel



A. Rucco, G. Notarstefano and J. Hauser, "An Efficient Minimum-Time Trajectory Generation Strategy for Two-Track Car Vehicles," in *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1505-1519, July 2015, doi: 10.1109/TCST.2014.2377777.

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:

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

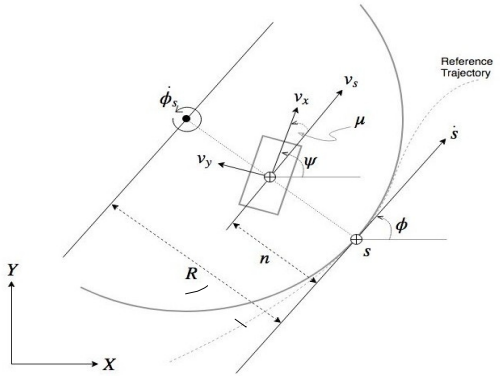
$$\left.\begin{aligned} v_s &= (R-n)\dot{\phi}_s \\ \dot{s} &= R\dot{\phi}_s \end{aligned}\right\} \Rightarrow \dot{s} = \frac{R}{R-n}\left(v_x cos\mu - v_y \sin\mu\right)$$

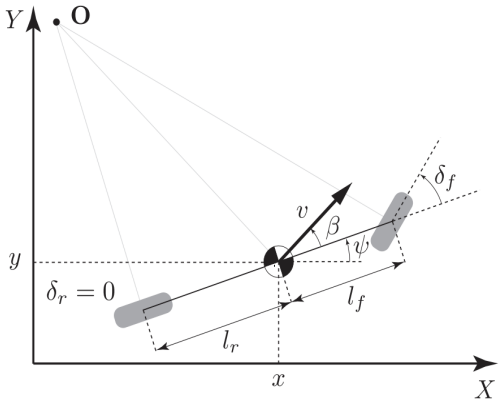
$$\left.\begin{aligned} \dot{s} &= \frac{R}{R-n}\left(v_x cos\mu - v_y \sin\mu\right) \\ \kappa &= \frac{1}{R} \end{aligned}\right\} \Rightarrow \dot{s} = \frac{v_x cos\mu - v_y \sin\mu}{1-n\kappa}$$

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

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

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





Curvilinear Coordinates Vehicle Model

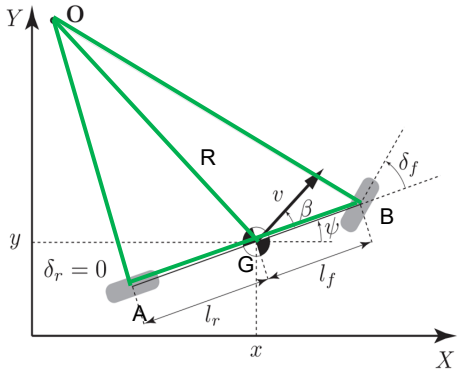
The motion model is a discretized kinematic bicycle model in curvilinear coordinates.

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

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



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

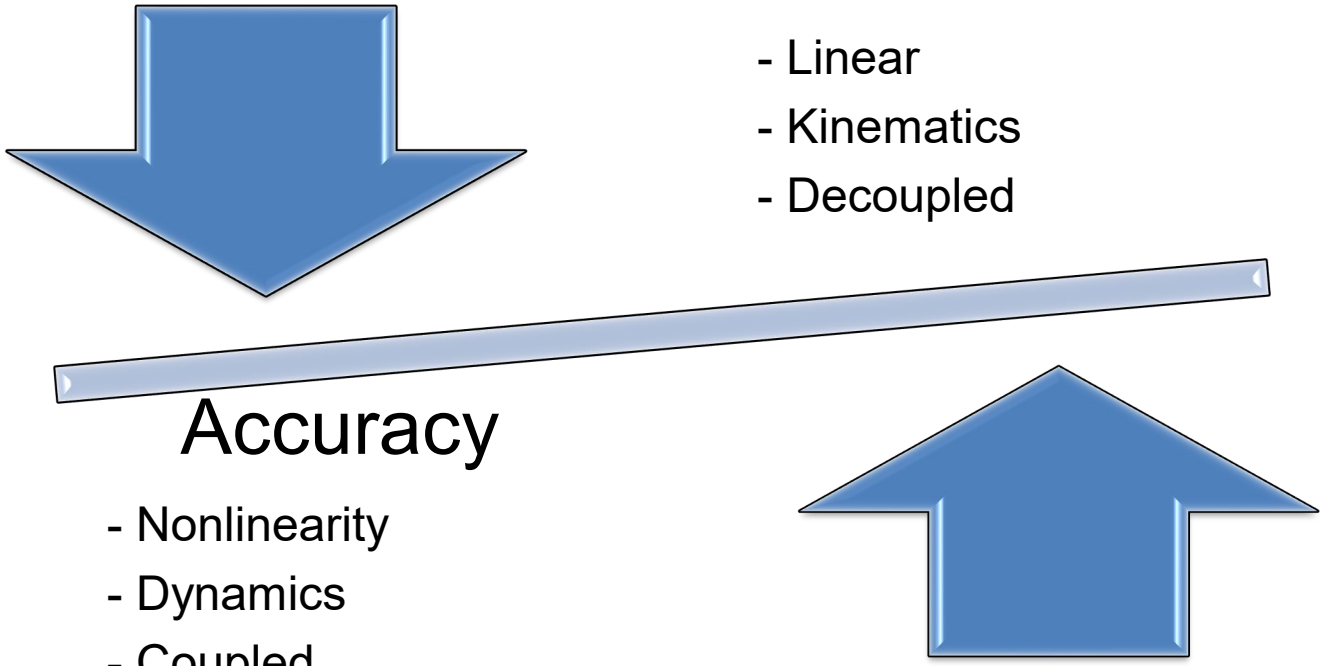
$$\frac{l_f}{\sin(\delta_f - \beta)} = \frac{R}{\sin(\frac{\pi}{2} - \delta_f)}$$

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

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

J. Kong, M. Pfeiffer, G. Schildbach and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," *2015 IEEE Intelligent Vehicles Symposium (IV)*, Seoul, 2015, pp. 1094-1099, doi: 10.1109/IVS.2015.7225830.

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”

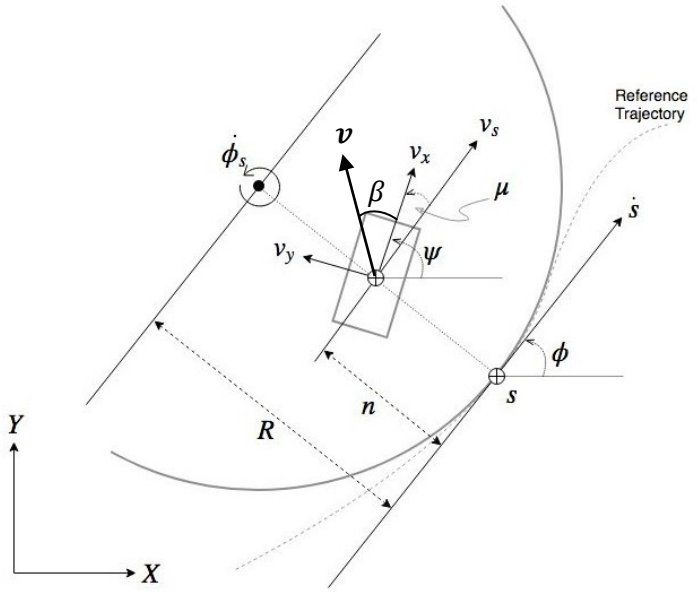
$$J_{tracking} = w_{\dot{s}} \cdot \left(\dot{s} - \dot{s}_{ref}\right)^2 + w_n \cdot n^2 + w_\mu \cdot (\mu + \beta)^2$$

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_μ 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 (λ_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:

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

$$\begin{aligned} s.t. \quad & x_{k+1} = f(x_k, u_k), \\ & c_k(x_k, u_k) \leq 0, \\ & x \in \mathcal{X}, \\ & u \in \mathcal{U}. \end{aligned}$$

"Softened" problem:

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

$$\begin{aligned} s.t. \quad & x_{k+1} = f(x_k, u_k), \\ & c_k(x_k, u_k, \lambda_k) \leq 0, \\ & x \in \mathcal{X}, \\ & u \in \mathcal{U}, \\ & \lambda \in \Lambda. \end{aligned}$$

Vehicle Motion Control: Safety Objective/Cost function

The safety objective function:

Can use the constraint violation => slack to estimate

$$J_{safety} = \boldsymbol{\lambda}_{soft}^{T} E \boldsymbol{\lambda}_{soft} + \bar{H} \boldsymbol{\lambda}_{hard}$$

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

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

Lateral error violation
speed violation
acceleration violation

$$\boldsymbol{\lambda}_{hard} = \begin{bmatrix} \lambda_n \\ \lambda_a \\ \lambda_s \end{bmatrix}$$

Vehicle Motion Control: Objective/Cost function

The full objective function:

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

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

$$J_{safety} = \boldsymbol{\lambda}_{soft}^T \boldsymbol{E} \boldsymbol{\lambda}_{soft} + \bar{\boldsymbol{H}} \boldsymbol{\lambda}_{hard}$$

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.

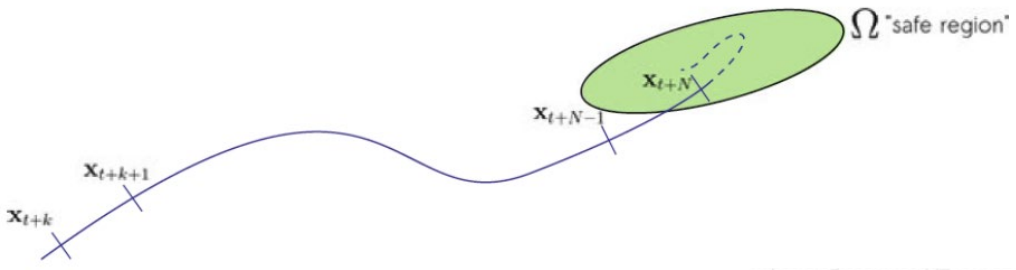
  Example: inclusion of some "hard-to-achieve" constraints.

- 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.

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

$$\begin{aligned} s.t. \quad & x_{k+1} = f(x_k, u_k), \\ & c_k(x_k, u_k, \lambda_k) \leq 0, \\ & x \in \mathcal{X}, \\ & u \in \mathcal{U}, \\ & \lambda \in \Lambda. \end{aligned}$$

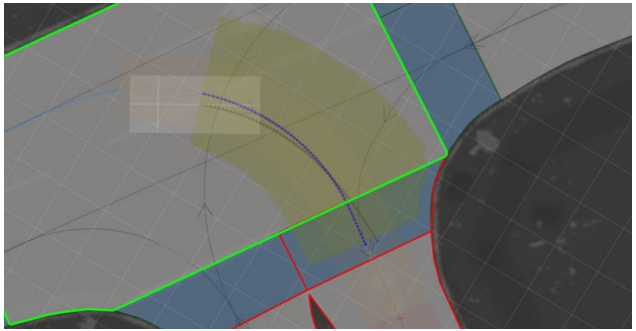
Vehicle Motion Control: Terminal cost

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

- 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.

Vehicle Motion Control

Inequality constraints and slack

Vehicle Motion Control: Inequality constraints

Linear state and input inequality constraints

$$\boldsymbol{x}_{\min} \leq \boldsymbol{x} \leq \boldsymbol{x}_{\max}$$
$$\boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}$$

General nonlinear inequality constraints

$\boldsymbol{c}^{\text{station}}(\boldsymbol{x}, \lambda_s) \leq 0$	⟹	Collision constraint derived from hard upper speed bound.
$\boldsymbol{c}^{\text{vel}}(\boldsymbol{x}, \lambda_{v,soft}) \leq 0$	⟹	Speed constraint derived from soft upper speed bound.
$\boldsymbol{c}^{\text{tube,hard}}(\boldsymbol{x}, \lambda_n) \leq 0$ $\boldsymbol{c}^{\text{tube,soft}}(\boldsymbol{x}, \lambda_{n,soft}) \leq 0$	⟹	Spatial soft/hard tube constraints imposed on car footprint.
$\boldsymbol{c}^{\text{a,hard}}(\boldsymbol{x}, \lambda_a) \leq 0$	⟹	Combined acceleration constraint on hard bounds.
$\boldsymbol{c}^{\text{a,soft}}(\boldsymbol{x}, \lambda_{a,soft}) \leq 0$	⟹	Acceleration constraint on comfort bounds.
$\boldsymbol{c}^{\dot{\delta}}(\boldsymbol{x}) \leq 0$	⟹	Speed dependent steering rate constraint.

Vehicle Motion Control: Inequality constraints and slack

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

- 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.

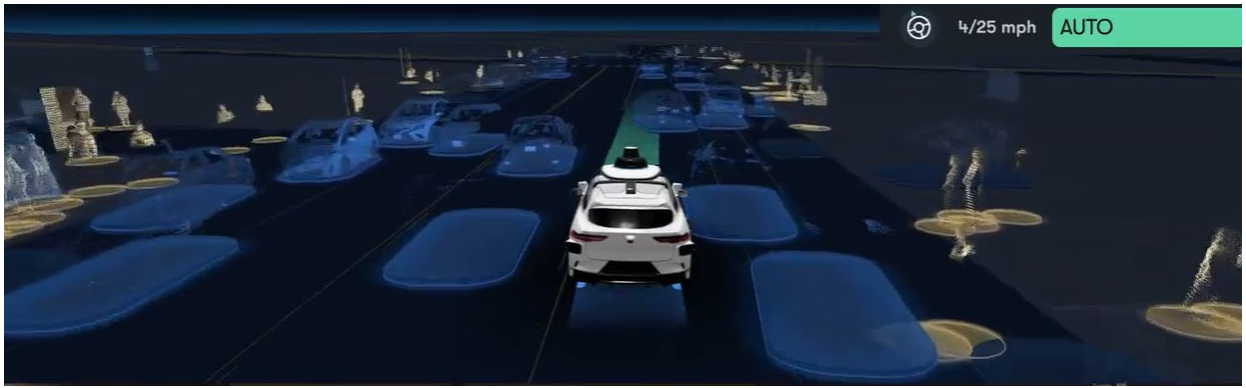
State Variables:

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

Input variable:

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

$$\boldsymbol{c}_k^{\text{station}}(\boldsymbol{x}_k, \lambda_s) = \begin{cases} s_k - s_k^{\max} - \lambda_{s,k} & \leq 0 \\ -s_k + s_k^{\min} - \lambda_{s,k} & \leq 0 \end{cases}, \qquad \forall k \in \{0, \ldots, N\}$$



Vehicle Motion Control: Inequality constraints and slack

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

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

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

State Variables:

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

Input variable:

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

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.

Vehicle Motion Control: Inequality constraints and slack

$$\boldsymbol{c}^{\text{tube,hard}}(\boldsymbol{x}, \lambda_n) \leq 0$$

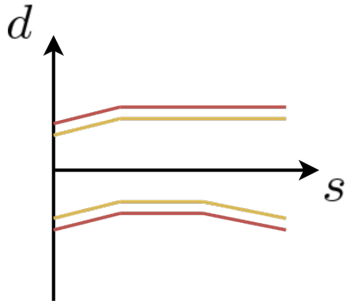
$$\boldsymbol{c}^{\text{tube,soft}}(\boldsymbol{x}, \lambda_{n,soft}) \leq 0$$

Spatial tube constraints

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

Width of the tube at the corners

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



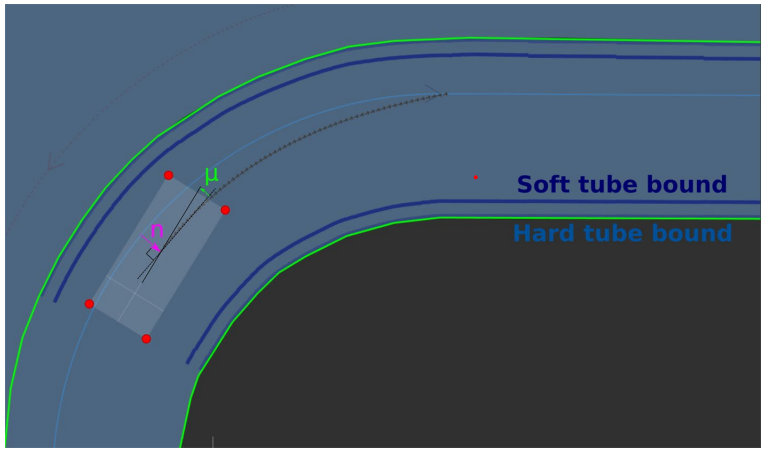
Spatial tube constraints

State Variables:

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

Input variable:

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

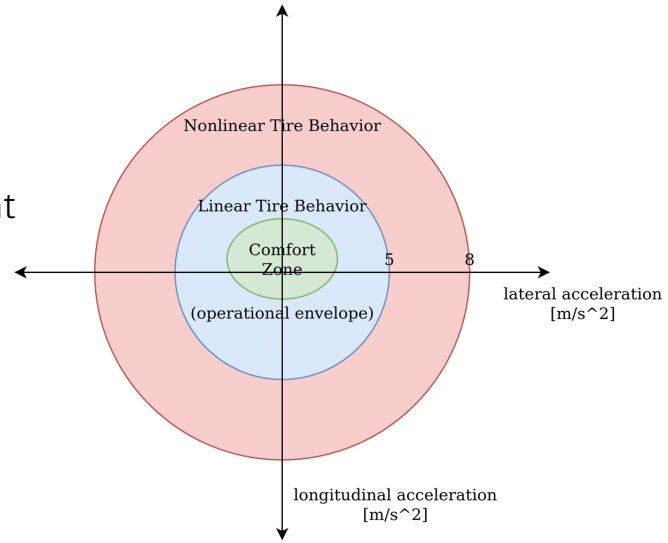


Vehicle Motion Control: Inequality constraints and slack

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

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:

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

Input variable:

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

$$c_k^{a,\mathrm{hard}}(\boldsymbol{x}_k, \lambda_a) = \sqrt{\frac{a_y^2}{a_{y,\max,k}^2} + \frac{a_x^2}{a_{x,\max,k}^2}} - 1 - \lambda_a \leq 0, \qquad \forall k \in \{0, \ldots, N\}$$

Vehicle Motion Control: Inequality constraints and slack

$$\boldsymbol{c}^{\mathrm{a,soft}}(\boldsymbol{x}, \lambda_{a,soft}) \leq 0$$

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:

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

$$\boldsymbol{c}_k^{a,\mathrm{soft}}(\boldsymbol{x}_k, \lambda_{a,\mathrm{soft}}) = \begin{cases} a_k - a_{\mathrm{comfort}} - \lambda_{a,\mathrm{soft}} & \leq 0 \\ -a_k - \mathrm{decel}_{\mathrm{comfort}} - \lambda_{a,\mathrm{soft}} & \leq 0 \end{cases}, \qquad \forall k \in \{0, \ldots, N\}$$

Input variable:

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

Model Predictive Control Formulation: Summary

MPC solves the following Optimal Control Problem for every control iteration

Optimal sequence control actions

Objective

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

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

$$c_k(x_k, u_k, \lambda_k) \leq 0, \quad \text{Inequality constraints}$$

$$x \in \mathcal{X}, \quad \text{State constraints}$$

$$u \in \mathcal{U}, \quad \text{Actuation limits}$$

$$\lambda \in \Lambda.$$

State Variables:

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

Input variable:

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

- 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

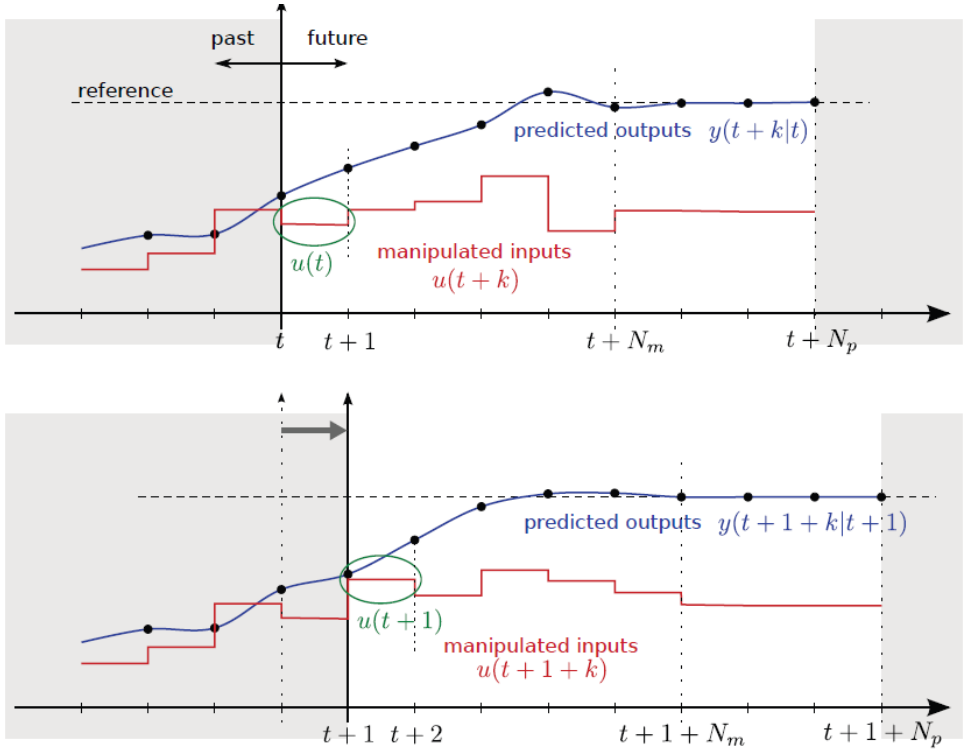
Part III

Vehicle Motion Control

How to improve the computation time?

MPC complexity

MPC



QP Problem
(QP: Quadratic Programming)

$$\min_{\boldsymbol{z}} \quad \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{x}_0^T\boldsymbol{F}\boldsymbol{z}$$

$$\text{s.t.} \quad \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{W} + \boldsymbol{S}\boldsymbol{x}_0$$

- 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

Explicit MPC

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

- Online optimization: given $\boldsymbol{x}(t)$, solve the problem at each time step t

  (the control law $\boldsymbol{u} = \boldsymbol{u}_0^*(\boldsymbol{x})$ is implicitly defined by the QP solver)

  → Quadratic Programming (QP)

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

  → Multi-parametric Quadratic Programming (mpQP)

Explicit MPC

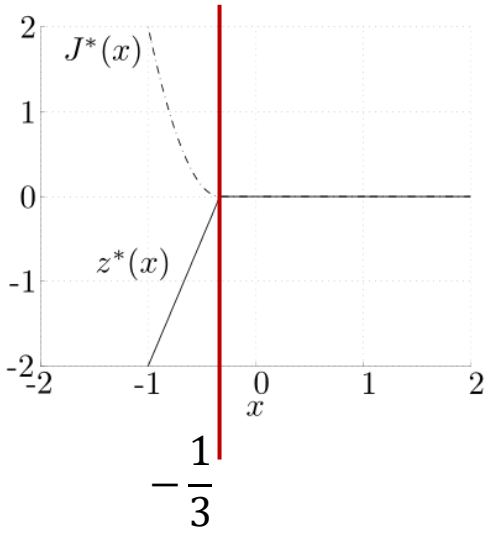
Example

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

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

KKT condition:

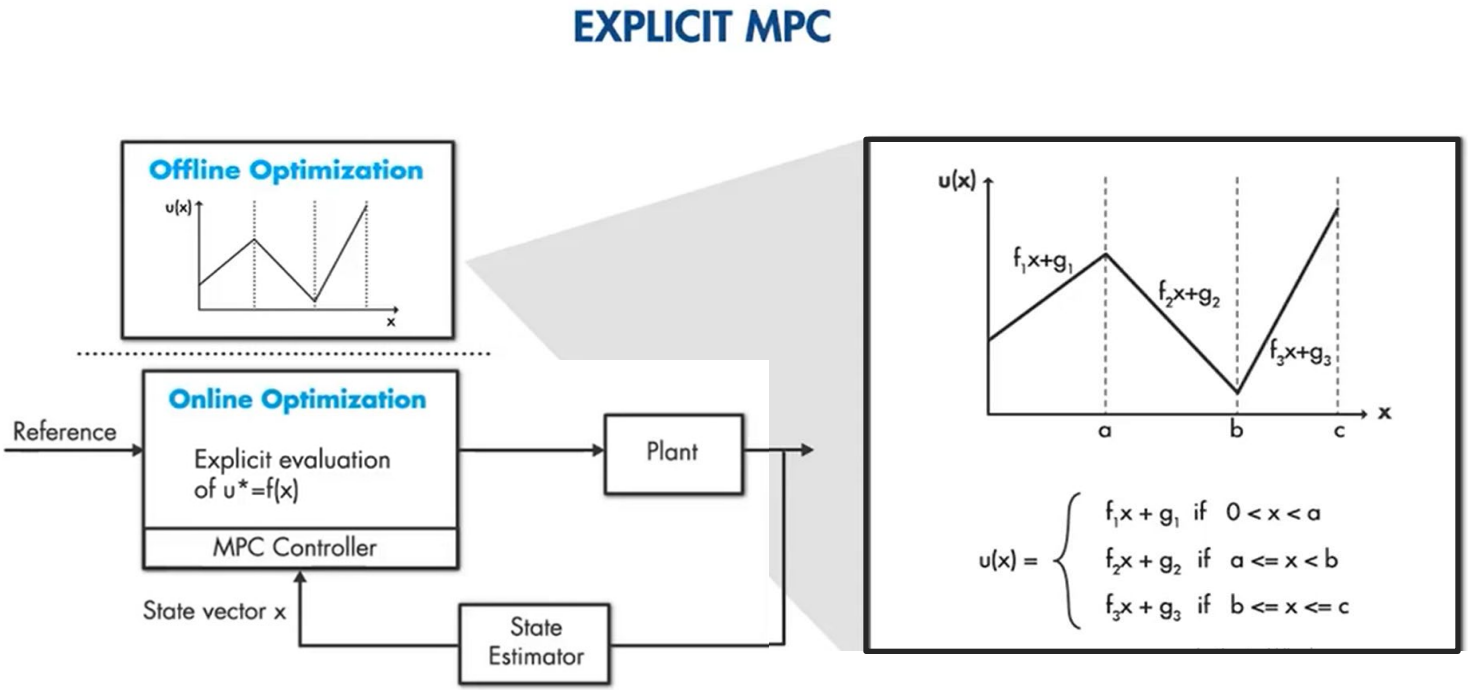
$$\begin{aligned} z + \lambda &= 0 \\ \lambda(z - 3x - 1) &= 0 \\ \lambda &\geq 0 \\ z - 3x - 1 &\leq 0 \end{aligned}$$



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

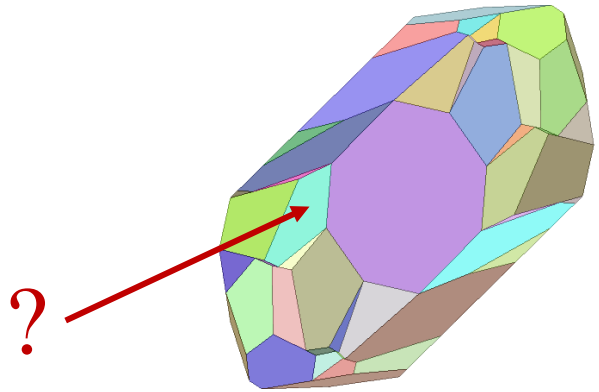
Borrelli, Francesco, Alberto Bemporad, and Manfred Morari. Predictive control for linear and hybrid systems. Cambridge University Press, 2017.

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).

Borrelli, Francesco, Alberto Bemporad, and Manfred Morari. Predictive control for linear and hybrid systems. Cambridge University Press, 2017.

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

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

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

$$c_k(x_k, u_k, \lambda_k) \leq 0, \quad \text{Inequality constraints}$$

$$x \in \mathcal{X}, \quad \text{State constraints}$$

$$u \in \mathcal{U}, \quad \text{Actuation limits}$$

$$\lambda \in \Lambda.$$

- 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

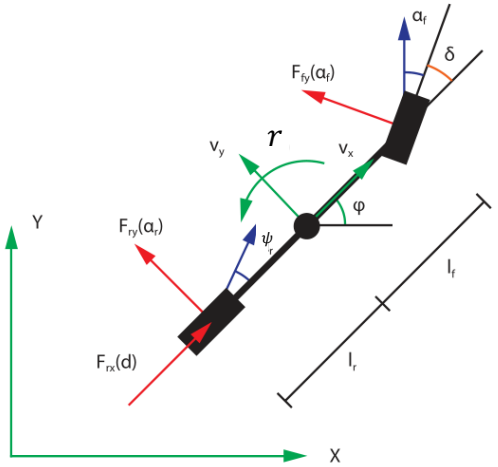
$$\dot{x} = v_x \cos(\psi) - v_y \sin(\psi)$$

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

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

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

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



Recall tire force

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

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

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.

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

$posion\ error, e_p + velocity\ error, e_v + heading\ error, e_\varphi$

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

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

深蓝学院
shenlanxueyuan.com

Thanks for Listening !