(1) System dynamics

$$X = (x_{EV}, v_{EV}, y_{EV}, \psi_{EV})^{\mathrm{T}}$$

$$u_{EV} = (a_{EV}, \delta_f)^{\mathrm{T}}$$

where the system state contains longitudinal position, speed, lateral position, and yaw angle of ego vehicle (EV); the control vector contains acceleration and steering angle. System dynamics are modeled using the vehicle kinematic bicycle model:

$$\dot{X} = A \cdot X + B \cdot u_{EV}$$

where:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & v_{EV} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & v_{CV} \\ 0 & \dfrac{}{l_f + l_r} \end{bmatrix}$$

(2) Cost function

$$J_{EV} = \frac{1}{2}\sum_{n=0}^{N-1}[\underbrace{\beta_1(x_{EV} - x_{des}^{EV})^2}_{\text{driving target}} + \underbrace{\beta_2(v_{CV} - v_{des}^{EV})^2}_{\text{driving efficiency}} + \underbrace{\beta_3 a_{EV}^2}_{\text{ride comfort}} + \underbrace{\beta_4(y_{EV} - y_{des}^{EV})^2}_{\text{lane-changing request}} + \underbrace{\beta_5 \psi_{EV}^2}_{\text{motion smoothness}}]$$

The cost function is formulated as a Quadratic Planning form. It aims to pursue the desired states.

(3) Constraints

$$0 \le v_{EV} \le v_{lim}$$

$$a_{min} \le a_{EV} \le a_{max}$$

$$\delta_{f_{min}} \le \delta_f \le \delta_{f_{max}}$$

$$\boldsymbol{x}_0 = (x_{EV}^0, v_{EV}^0, y_{EV}^0, \psi_{EV}^0)^T$$

For now, I didn't add any hard constraints about collision avoidance, etc.

(4) Algorithm

Discretization of system dynamics using Euler equations:

$$X_{n+1} = (A\Delta t + I) \cdot X_n + B \cdot u_{EV} \cdot \Delta t$$

Definition:

$$\boldsymbol{X} = (X_0, X_1, \dots, X_N)^{\mathrm{T}}$$

$$\boldsymbol{U}_{EV} = (\boldsymbol{u}_{EV,0}, \boldsymbol{u}_{EV,1}, \dots, \boldsymbol{u}_{EV,N-1})^{\mathrm{T}}$$

$$\mathcal{P} = diag\,(\underbrace{\boldsymbol{Q}_{EV}, \dots, \boldsymbol{Q}_{EV}}_{N}, \boldsymbol{0}_{5\times 5}, \underbrace{\theta_3, \dots, \theta_3}_{N})$$

$$\boldsymbol{Q}_{EV} = diag(\beta_1, \beta_2, 0, \beta_4, \beta_5)$$

$$\boldsymbol{P}_{EV} = diag(\beta_3, 0)$$

$$\boldsymbol{q} = (\underbrace{-\boldsymbol{Q}_{EV}\boldsymbol{x}_{des}^{EV}, \dots, -\boldsymbol{Q}_{EV}\boldsymbol{x}_{des}^{EV}}_{N}, 0, \dots, 0)^{\mathrm{T}}$$

$$\boldsymbol{x}_{des}^{EV} = (x_{des}^{EV}, v_{des}^{EV}, y_{des}^{EV}, 0)^{\mathrm{T}}$$

Thus, we can transform the cost function into a quadratic form:

$$J_{EV} = \frac{1}{2}(\boldsymbol{X}^{\mathrm{T}} \quad \boldsymbol{U}_{EV}^{\mathrm{T}})\boldsymbol{\mathcal{P}}\begin{pmatrix}\boldsymbol{X}\\ \boldsymbol{U}_{EV}\end{pmatrix} + \boldsymbol{q}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{X}\\ \boldsymbol{U}_{EV}\end{pmatrix}$$

And this problem can be easily solved using quadratic planners, such as gurobi, and scipy.

(5) Using guide

I have tried to code this controller in Python. In Python, you can just "From planner import MPCplanner" to use it.