MPCC Docs

Peter Werner

August 13, 2020

1 Problem Formulation

The MPCC problem as described in [LMG10] and [LDM15] is reformulated in continuous time (for the solver) as follows

$$\begin{aligned} \text{minimize} \quad & \int_0^T \begin{bmatrix} \epsilon_c^{lin}(t) & \epsilon_l^{lin}(t) \end{bmatrix} \begin{bmatrix} Qc & 0 \\ 0 & Ql \end{bmatrix} \begin{bmatrix} \epsilon_c^{lin}(t) \\ \epsilon_l^{lin}(t) \end{bmatrix} - Q_\theta \dot{\theta}(t) + u^T(t) R u(t) dt \\ \text{subject to} \quad & \dot{x} = f(x, u, \Phi) \\ & b_{lower} \preceq x(t) \preceq b_{upper} \\ & l_{lower} \preceq u(t) \preceq l_{upper} \\ & h(x, \Phi) \leq 0 \end{aligned} \tag{1}$$

given the system dynamics f and the arclength parametrization of the contour (in our case the track) Φ. Here $x(t)$ denotes the system state, $u(t)$ the inputs to the system, b the box constraints on the state, l the box constraints on the input and h captures the track boundary constraints.
The state of the system is augmented with the advancing parameter θ

$$x = \begin{bmatrix} x_{model} \\ \theta \end{bmatrix} \tag{2}$$

and the virtual input $\dot{\theta}$ is appended to the inputs from the original system dynamics.

$$u = \begin{bmatrix} u_{model} \\ \dot{\theta} \end{bmatrix} \tag{3}$$

The track boundary constraint is realized as a convex disk constraint.

$$h(x, \Phi) = (x - x_t^{lin}(\theta))^2 + (y - y_t^{lin}(\theta))^2 - r_\Phi(\hat{\theta})^2 \tag{4}$$

Here $r_\Phi(\hat{\theta})$ is the half-width of the track at the last predicted arc length.

The linearized contouring error ϵ_c^{lin} and lag error are computed as shown in fig. 1. To make the problem realtime feasible the are approximmated by linearizing the both them and the track around the previous solution $\hat{\theta}$ as:

$$\Phi(\theta) = \begin{bmatrix} x_t(\theta) \\ y_t(\theta) \end{bmatrix} \approx \Phi(\hat{\theta}) + \partial_\theta \Phi(\hat{\theta})(\theta - \hat{\theta}) \tag{5}$$

$$\Rightarrow \Phi^{lin}(\theta) = \begin{bmatrix} x_t(\hat{\theta}) + cos(\phi(\hat{\theta}))(\theta - \hat{\theta}) \\ y_t(\hat{\theta}) + sin(\phi(\hat{\theta}))(\theta - \hat{\theta}) \end{bmatrix} \tag{6}$$

this allows us to compute the errors

$$x_t^{lin}(\theta) = x(t) + \epsilon_l^{lin} cos(\phi(\hat{\theta})) + \epsilon_c^{lin} sin(\phi(\hat{\theta})) \tag{7}$$

$$y_t^{lin}(\theta) = y(t) + \epsilon_l^{lin} sin(\phi(\hat{\theta})) - \epsilon_c^{lin} cos(\phi(\hat{\theta})) \tag{8}$$

$$\Leftrightarrow \begin{cases} \epsilon_l^{lin} = cos(\phi(\hat{\theta}))(x_t^{lin}(\theta) - x(t)) + sin(\phi(\hat{\theta}))(y_t^{lin}(\theta) - y(t)) \\ \epsilon_c^{lin} = sin(\phi(\hat{\theta}))(x_t^{lin}(\theta) - x(t)) - cos(\phi(\hat{\theta}))(y_t^{lin}(\theta) - y(t)) \end{cases} \tag{9}$$

These approximations are especially good if $\epsilon_l \approx 0$ and $\hat{\theta} - \theta \approx 0$. In practice the first can be incentivized by increasing Q_l and the second by warmstarting the problem correctly.

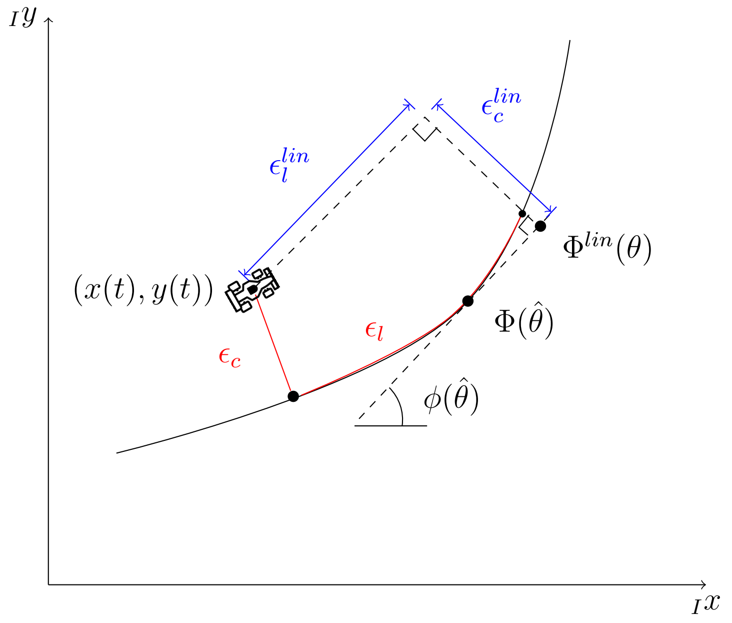

Figure 1: To compute the linearized conturing error ϵ_c^{lin} and lag error ϵ_l^{lin} the track is linearized around the last solution as the estimate of the arclength progress $\hat{\theta}$ of the MPCC. The coordinate frame $({}_I x, {}_I y)$ is the inertial reference frame the track is described in.

Taking a step back we see that MPCC basically is optimizing to move the position of a virtual point given by $\theta(t)$ as far along the track as possible while steering the model of the system to keep the contouring and lag errors small. The solver then discretizes this problem and finds approximate solutions.

2 Dynamic Model

For now the model from [LDM15] is implemented with adapted parameters. In the code the state is arranged in the following way

$$x = \begin{bmatrix} x \\ y \\ \phi \\ v_x \\ v_y \\ \omega \\ d \\ \delta \\ \theta \end{bmatrix} \tag{10}$$

where x, y are the position of the center of gravity, ϕ is the yaw angle of the car w.r.t. the world frame, v_x is the longitudinal velocity, v_y is the lateral velocity, ω is the yaw rate, θ the advancing parameter and lastly d is the integrated motor torque and δ is the steering angle. The input u is arranged as

$$u = \begin{bmatrix} \dot{d} \\ \dot{\delta} \\ \dot{\theta} \end{bmatrix} \tag{11}$$

The inputs we optimize for are chosen to be the derivatives of the commandable inputs to the system such that we can penalize their smoothness.

3 Track Parametrization

For the track the centerline is given in waypoints. To implement MPCC we need an arclenght parametrization Φ. This is realized by interpolating the waypoints using cubic splines (previously bezier curves) with a cyclic boundary condition, and creating a dense lookup table with the track location and the linearization parameters. Note that in the optimization it is not practical to pass the full parametrization since it contains functions that the used solver has dificulties dealing with such as floor and modulo. Instead the linearization parameters are precomputed offline and passed at every stage.

4 Overtaking Idea

Instead of trying to exactly figure out what positions the adversary will be at in the future and designing a corridor along the track for overtaking as done in [LDM15], we could try to augment the cost function as follows.

$$\begin{aligned}
\text{minimize} \quad & \int_0^T \begin{bmatrix} \epsilon_c^{lin}(t) & \epsilon_l^{lin}(t) \end{bmatrix} \begin{bmatrix} Qc & 0 \\ 0 & Ql \end{bmatrix} \begin{bmatrix} \epsilon_c^{lin}(t) \\ \epsilon_l^{lin}(t) \end{bmatrix} - Q_\theta \dot{\theta}(t) + u^T(t) R u(t) + O(t, x(t), \hat{x}_{adv}(t)) dt \\
\text{subject to} \quad & \dot{x} = f(x, u, \Phi) \\
& b_{lower} \preceq x(t) \preceq b_{upper} \\
& l_{lower} \preceq u(t) \preceq l_{upper} \\
& h(x, \Phi) \leq 0
\end{aligned} \tag{12}$$

with

$$O(t, x, \hat{x}_{adv}) = \sum_{i=0}^{1} \alpha_i(t) e^{-(x_{pos}(t) - \hat{x}_{adv,pos}(t) - \mu_i)^T A_i(t) (x_{pos}(t) - \hat{x}_{adv,pos}(t) - \mu_i)} \tag{13}$$

and

$$\alpha_0 > 0 \tag{14}$$

$$\alpha_1 < 0 \tag{15}$$

The idea is that instead of imposing hard collision avoidance constraints along the entire horizon, we can place these "cost bumps" around the predicted position of the adversary. Furthermore, since our certainty of the opponents behavior decreases along our prediction horizon, these cost bumps could be widened by changing the scaling matrices A_i and coefficients α_i. Additionally cutting off or blocking the opponent could be incentivized by adding a "trough" right in front of the cost peak around predicted position of the adversary. This leaves the question open about how to place these "cost bumps". Currently I am considering RL.

References

[LDM15] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[LMG10] Denise Lam, Chris Manzie, and Malcolm Good. Model predictive contouring control. In *49th IEEE Conference on Decision and Control (CDC)*, pages 6137–6142. IEEE, 2010.

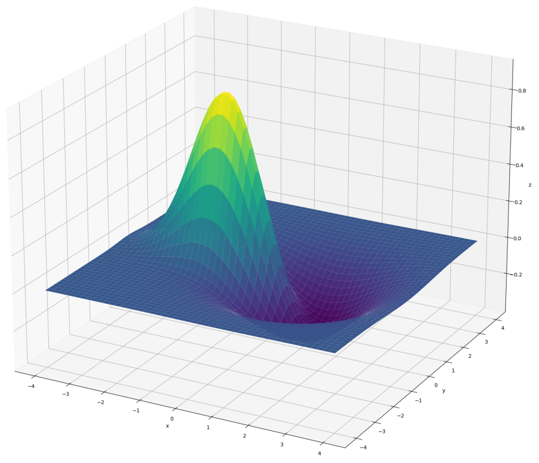

Figure 2: Example of cost bump as described in eqn 13. Here we would imagine the adversary moving from left to right along the x axis.