NA 565 - Fall 2023

Model Predictive Control

September 18, 2023

UNIVERSITY OF MICHIGAN

Assignment

- HW0 completed. Congrats!
- Make sure to celebrate your small victories along the way!

Assignment

- You are now Level 0 Self-Driving Cars engineers.

SAE J3016™ LEVELS OF DRIVING AUTOMATION™

Learn more here: sae.org/standards/content/j3016_202104

	SAE LEVEL 0™	SAE LEVEL 1™	SAE LEVEL 2™	SAE LEVEL 3™	SAE LEVEL 4™	SAE LEVEL 5™
What does the human in the driver's seat have to do?	**You <u>are</u> driving** whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			**You <u>are not</u> driving** when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			**These are automated driving features**		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering **OR** brake/ acceleration support to the driver	These features provide steering **AND** brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering **OR** • adaptive cruise control	• lane centering **AND** • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

Assignment

- HW1 is up! Check Canvas.
  - Adaptive Cruise Control
  - Vehicle Trajectory Tracking (LQR & MPC)
- Start early and ask questions on Piazza.
- Each HW is a major learning milestone in this course, where you internalize the knowledge and fill in the gaps.

- We have learned LQR as a tracking controller.

- Consider a discrete-time dynamical system, with state $x_k \in \mathbb{R}^n$, and control input $u_k \in \mathbb{R}^m$:

$$x_{k+1} = A_k x_k + B_k u_k.$$

-

$$\begin{aligned} \min_{x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^m} \quad & \frac{1}{2} x_N^\mathsf{T} L x_N + \sum_{k=0}^{N-1} \frac{1}{2} x_k^\mathsf{T} Q x_k + \frac{1}{2} u_k^\mathsf{T} R u_k \\ \text{s.t.} \quad & x_{k+1} = A_k x_k + B_k u_k, k = 0, \ldots, N-1. \\ & x_0 = x_{\mathsf{init}}, x_N = x_{\mathsf{des}} \end{aligned}$$

- L,Q,R are positive definite (tunable) weight matrices.

- $$\begin{aligned} \min_{x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^m} \quad & \frac{1}{2} x_N^\mathsf{T} L x_N + \sum_{k=0}^{N-1} \frac{1}{2} x_k^\mathsf{T} Q x_k + \frac{1}{2} u_k^\mathsf{T} R u_k \\ \text{s.t.} \quad & x_{k+1} = A_k x_k + B_k u_k, k = 0, \dots, N-1. \\ & x_0 = x_{\mathsf{init}}, x_N = x_{\mathsf{des}} \end{aligned}$$
- Variables are state sequence $x_1, \dots, x_N$ and input sequence $u_1, \dots, u_{N-1}$.
- Constraints are linear dynamics equations and the initial and final state.

- The greedy algorithm, $N = 1$, only minimizes the current stage cost and ignores the effect of the input on future states.
- Except for cases where the performance is lower-bounded, e.g., maximizing submodular functions, this strategy performs poorly.
- In practice, choosing the "right" N is also part of the design and can affect the performance significantly.

- Variables are state sequence $x_1, \dots, x_N$ and input sequence $u_1, \dots, u_{N-1}$.

- Applying the control input sequence $u_1, \dots, u_{N-1}$ leads to an "open-loop" control policy as there is no feedback along the way.
- This means the LQR linear state feedback $u_k = Kx_k$ is agnostic to future changes during execution.

- Constraints are linear dynamics equations and the initial and final state.

- The state space $\mathcal{X} = \mathbb{R}^n$ and input space $\mathcal{U} = \mathbb{R}^m$ are spacial cases.
- In general, $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{U} \subset \mathbb{R}^m$, which can lead to inequality constraints.

Remark

In general, the constraint sets can be time-varying, and one can have $\mathcal{X}_k \subset \mathbb{R}^n$ and $\mathcal{U}_k \subset \mathbb{R}^m$. For example, the safe distance to the obstacle can depend on the vehicle's velocity.

- How to handle obstacles?
- How to handle input, state, or output constraints?
- What about modeling and state estimation errors?

Brainstorming

1. Obstacles
2. Input constraints
3. State or output constraints
4. Modeling and state estimation errors

- Constrained least squares formulation with equality constraints is a special case of a Quadratic Program.
-

$$
\begin{aligned}
\min_{x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^m} \quad & \frac{1}{2} x_N^{\mathsf{T}} L x_N + \sum_{k=0}^{N-1} \frac{1}{2} x_k^{\mathsf{T}} Q x_k + \frac{1}{2} u_k^{\mathsf{T}} R u_k \\
\text{s.t.} \quad & x_{k+1} = A_k x_k + B_k u_k, k = 0, \ldots, N-1. \\
& x_0 = x_{\mathsf{init}}, x_N = x_{\mathsf{des}}
\end{aligned}
$$

A *Quadratic Program* (QP) is a special kind of optimization problem with *constraints*. The cost to be minimized is supposed to be quadratic, meaning that $f : \mathbb{R}^n \to \mathbb{R}$ has the form

$$f(x) = \frac{1}{2} x^{\mathsf{T}} Q x + q x,$$

where Q is an $n \times n$ symmetric matrix, meaning that $Q^{\mathsf{T}} = Q$, and where q is a $1 \times n$ row vector.

Consider the QP

$$
\begin{aligned}
x^* = \quad & \underset{x \in \mathbb{R}^n}{\arg\min} \quad \frac{1}{2} x^\mathsf{T} Q x + q x \\
& A_{\mathsf{in}} x \preceq b_{\mathsf{in}} \\
& A_{\mathsf{eq}} x = b_{\mathsf{eq}} \\
& lb \preceq \ x \preceq ub
\end{aligned}
$$

and assume $Q^\mathsf{T} = Q$ and *positive definite* ($x \neq 0 \implies x^\mathsf{T} Q x > 0$), and that the subset of $\mathbb{R}^n$ defined by the constraints is non empty, that is

$$
C := \{x \in \mathbb{R}^n \mid A_{\mathsf{in}} x \preceq b_{\mathsf{in}},\, A_{\mathsf{eq}} x = b_{\mathsf{eq}},\, lb \preceq x \preceq ub\} \neq \emptyset.
$$

Then x^* exists and is unique (QP is convex and feasible).

- So, we could promote our old friend LQR to handle (affine) inequality constraints and solve a convex QP problem.

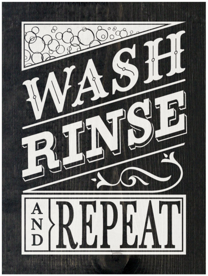
- So, we could promote our old friend LQR to handle (affine) inequality constraints and solve a convex QP problem.

- Unfortunately, this does not address the remaining challenges (which ones?).
- $\implies$ Receding Horizon Control a.k.a. Model Predictive Control.

Remark

Note that we can always clip the computed control input by LQR to satisfy the input limits (saturated LQR control), i.e., $u_{min} \leq u_k \leq u_{max}$ *(this of course changes the performance). However, satisfying the state and output constraints is not trivial.*

Main idea: If we can optimize trajectories quickly enough, then we can use our trajectory optimization as a feedback policy.



The recipe:

1. measure the current state (requires state estimator);
2. optimize a trajectory from the current state;
3. execute the first action from the optimized trajectory;
4. let the dynamics evolve for one step and repeat.

The QP-MPC algorithm is a highly efficient online control method (can be run at kilohertz!) that is defined by solving the following QP problem (constraints are affine).

$$\min_{x_k, u_k} \frac{1}{2} x_N^\intercal L x_N + \sum_{k=0}^{N-1} \frac{1}{2} x_k^\intercal Q x_k + \frac{1}{2} u_k^\intercal R u_k$$

$$\begin{aligned}
\text{s.t.} \quad & x_{k+1} = A_k x_k + B_k u_k, k = 0, \dots, N-1, \\
& x_0 = x_{\mathsf{init}}, x_N = x_{\mathsf{des}}, \\
& x_k \in \mathcal{X}, u_k \in \mathcal{U}, k = 0, \dots, N-1.
\end{aligned}$$

Remark

MPC problem is highly structured and can be implemented using sparse matrices.

- In LQR, we have a linear state feedback

$$u_k = Kx_k.$$

- In MPC, we have a complicated state feedback obtained by solving an online optimization problem

$$u_k = \mu_{\mathsf{MPC}}(x_k).$$

Performance Guarantees

- In general, there is no guarantee.
- MPC works very well in practice.
- In QP-MPC, if the constraints are not active and the horizon is long enough, we can make a case about its asymptotic behavior to be the same as LQR.
- Ensuring the feasibility of QP is important. In practice, tuning the planning horizon N is likely necessary.

Let the state be $s := [x, y, \psi, v]^\mathsf{T}$ where (x, y) is the position, ψ is the yaw angle, and v is the velocity. The vehicle is controlled by the steering angle of the front wheel δ and the acceleration a.
The equations of motion of the vehicle are given by

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \psi \\ v \end{bmatrix} = \begin{bmatrix} v\cos(\psi + \beta) \\ v\sin(\psi + \beta) \\ \frac{v}{L_r}\sin\beta \\ a \end{bmatrix}, \text{with } \beta := \arctan(\frac{L_r}{L_r + L_f}\arctan\delta),$$

where L_r and L_f are the distance from the rear or front axes to the center of the vehicle (given data).

- Let $\frac{d}{dt}s =: f(x,u)$, $u := (\delta,a)$.
- Given the reference trajectory $(\bar{s}_k, \bar{u}_k)_{k=0,2,\cdots,N-1}$, we have the linearized system as:

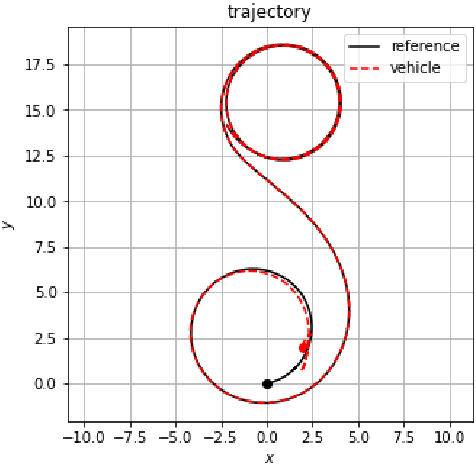
$$s_{k+1}-\bar{s}_{k+1} \approx \frac{\partial f(s_k, u_k)}{\partial s_k}|_{\bar{s}_k}(s_k-\bar{s}_k)+\frac{\partial f(s_k, u_k)}{\partial u_k}|_{\bar{u}_k}(u_k-\bar{u}_k).$$

- The perturbed state dynamics becomes:

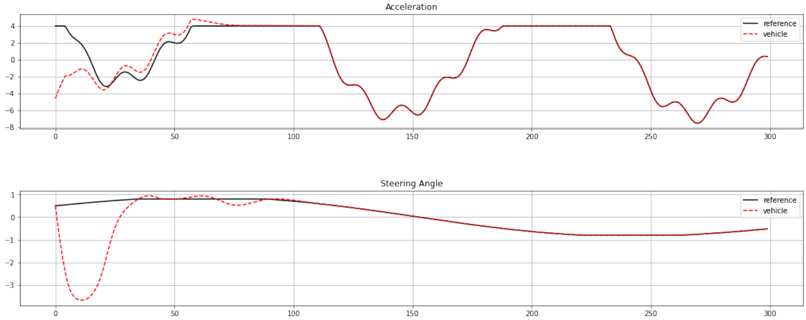
$$\delta s_{k+1} = A_k \delta s_k + B_k \delta u_k,$$

with $A_k := \frac{df(s_k,u_k)}{ds_k}|_{\bar{s}_k}$, $\frac{df(s_k,u_k)}{du_k}|_{\bar{u}_k}$, $\delta s := s - \bar{s}$, and $\delta u := u - \bar{u}$.

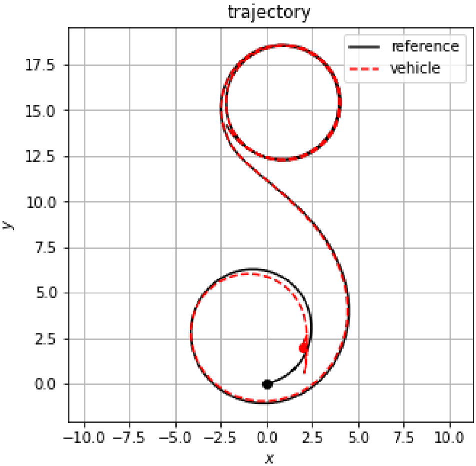
LQR trajectory tracking result.



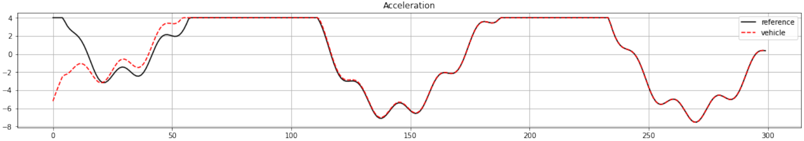
LQR input plots vs. time. Sampling time $0.05\,\mathrm{sec}$.

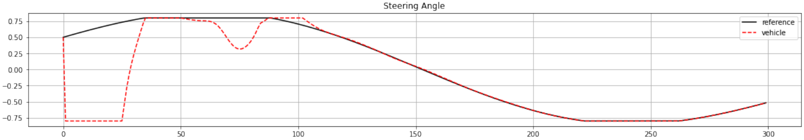


MPC trajectory tracking result.

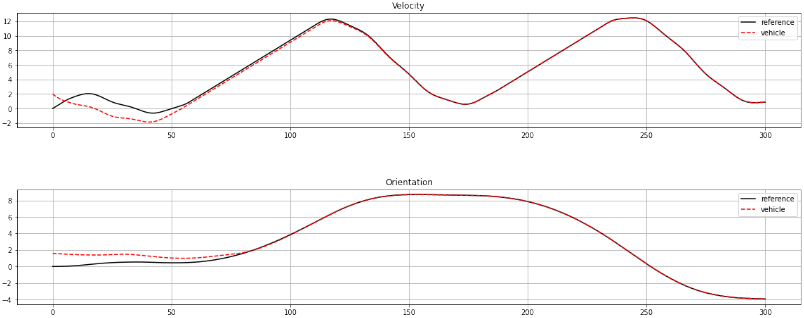


MPC input plots vs. time.





MPC velocity and yaw angle tracking plots vs. time.



Next Time

- MPC Examples in Python

References and Further Reading

- R. Tedrake, Underactuated Robotics
  https://underactuated.csail.mit.edu/
- F. Borrelli, A. Bemporad, M. Morari, Predictive Control for Linear and Hybrid Systems
  https://sites.google.com/berkeley.edu/mpc-lab/mpc-course-material
- S. Boyd,
  https://web.stanford.edu/class/ee364b/lectures/mpc_slides.pdf
- I. Kolmanovsky, AEROSP 740: Model Predictive Control (related advanced UM course; recommended if you wish to pursue this topic further. AEROSP 575: Flight and Trajectory Optimization is also related.)