Lane-Keeping of an Autonomous Vehicle using an iLQR-initialized Model Predictive Control

Sohun Patel[1] and Karthik Pythireddi[2]

[1]sohun@stanford.edu
[2]karthik9@stanford.edu

October 22, 2024

Abstract

In this project, we aim implement the lane-keep for an autonomous vehicle using MPC initialized through a reference trajectory generated by an iterative Linear Quadratic Regulator (iLQR).Our approach is to make use of a single non linear bicycle model as the dynamics, with the iLQR algorithm generating the reference state and control trajectories. We use these reference trajectories to initialize the MPC which optimizes the lateral control within the specified constraints. The overall goal of this project to evaluate the system's effectiveness in performing the lane keep during the curvy and complex roads. We notice that our controller has near-perfect tracking of an S-shaped road except for the beginning and end of the trajectories, which could be due to inaccuracies in our modeling. Nevertheless, the controller performs well in terms of a tracking perspective.

Introduction

Many OEMs and Automotive suppliers have started working on Advanced Driver Assistance Systems (ADAS) features since the early 2000s. Initially, many of the ADAS features started to appear in the high-end consumer vehicles, but because of regulatory and safety compliance, these ADAS features have become more prevalent in affordable consumer vehicles. In this project we would like to implement the Lane-Keep System (LKS) which is one of the crucial ADAS features which helps with the L2 level autonomy. The LKS enhances the driver comfort and safety. This feature also serves as a bridge to the evolution of the L2 level autonomy to L3 and L4+. Model-predictive control (MPC) is commonly used in the industry to keep a vehicle centered in the lane.

Related Work

From a LKS controller design perspective, multiple approaches have been proposed and examined in previous work. We will give a brief overview of these methods below.

2.1 Linear Quadratic Regulator

One approach to optimizing the steering action control for LKS is through the use of a linear quadratic regulator (LQR). Given a linear model of the vehicle dynamics and a cost function that can be tinkered with to penalize lane deviations and large steering angle inputs, LQR is an effective method that offers a minimized control action while tracking a reference trajectory. As demonstrated in previous work, much consideration must be given to the choice of the weighting matrices in the cost function, as they play an important role in determining the significance of each state and the cost of the control action. For instance, Sherif et al. [1] apply the LQR method for the LKS of a linear, kinematic bicycle model and propose heuristic optimization algorithms such as simulated annealing to tune and optimize these matrices in the cost function. Similarly, Guo et al. [2] apply the genetic algorithm to determine the weighting matrices. Once an optimal steering-angle control action is derived through LQR, the controller applies a calculated gain to the live state feedback to actuate the optimal steering action. Besides having to carefully choose the weighting matrices in the cost function, an additional point of consideration with LQR is that it does not account well for strict constraints in the state and action. Instead, constraints must be accounted for and added into the cost function as additional penalties.
For non-linear vehicle models, iterative LQR (iLQR) can be applied. This method works by linearizing the vehicle dynamics at each point of a reference state trajectory and using the linearized dynamics to determine the optimal control policy. Lee at al. [3] apply soft-constrained iLQR to achieve optimal and smooth steering inputs towards their linear vehicle model.

2.2 Model Predictive Control

The basic concept of model predictive control (MPC) is to forecast the system behavior and optimize that behavior to get the best possible outcome. The other part of MPC is to make use of the past recorded measurements to be able to predict the system behavior for the future i.e via the time horizon approach. MPC is an optimal control problem where the control action is obtained by solving online at each sampling interval, finite horizon in which the initial state becomes the current state of the plant. As per the work of [4], LKS has been implemented with MPC. The reference trajectory is generated by the interpolation of the five preview points. The distance between these preview points is determined by the longitudinal velocity. The MPC tracking is achieved by formulating this MPC tracking for Lane Keeping into a Quadratic Programming (QP) problem. They achieved the lane keeping by minimizing the steering angle to generate the optimal control sequence. When the ego enters a curve, it the respective lateral error is considerably high because of the poor tracking of MPC. To compensate that a reference trajectory was proposed in the form of a exponential decay function in [5], so that the respective parameters reach their optimal values in a smoother manner thereby increasing the accuracy of lateral tracking on curvy roads.

Problem Statement

We would like to use a simple non-linear bicycle model as our systems dynamics, will be applying a iLQR algorithm to generate the reference state and control trajectories. We will set up a quadratic cost function and will implement MPC for reference tracking, using the reference trajectories generated by iLQR as an initialization. We will get the respective metrics to gauge the performance

of the lane keeping like lateral tracking error, acceleration and respective trajectory plots for both iLQR and MPC.

Approach

4.1 System Dynamics

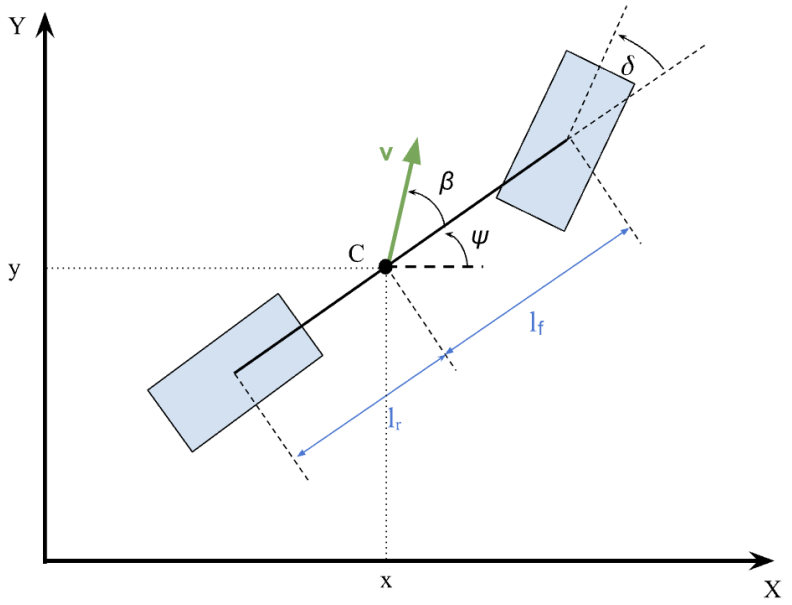


Figure 1: Simple Bicycle Kinematic Model

For the system dynamics of the vehicle we make use of the kinematic bicycle model [6] [7]. Due to its simplicity, adoption of this model is quite standard when modeling systems with respect to a vehicle in automotive industry. The model considers the following state variables and control inputs. The kinematic equations of the bicycle model are given by:

$$\dot{x} = v\cos(\psi + \beta) \tag{1}$$

$$\dot{y} = v\sin(\psi + \beta) \tag{2}$$

$$\dot{\psi} = \frac{v}{l_r}\sin(\beta) \tag{3}$$

$$\dot{v} = a \tag{4}$$

$$\beta = \tan^{-1}\left(\frac{l_r}{l_r + l_f}\tan(\delta_f)\right) \tag{5}$$

where $L = l_r + l_f$ is the wheelbase, the distance between the front and rear axle of the vehicle. x and y represent the position of the vehicle's center of mass C with respect to the XY inertial

frame, ψ represents the heading of the vehicle, and β represents the angle between the velocity of the vehicle's center of gravity and its longitudinal axis. The control inputs are the steering angle δ_f and the acceleration a that is in the same direction as the velocity v of the center of gravity. Thus, our state and control vectors are given by:

$$s = \begin{bmatrix} x \\ y \\ \psi \\ v \end{bmatrix}, \quad u = \begin{bmatrix} \delta_f \\ a \end{bmatrix}$$

The nonlinear dynamics can be represented with $\dot{s} = f(s, u)$, where f is made up of Eqns. 1, 2, 3, and 4. Eqn. 5 must also be used to calculate β from the steering angle input δ_f. Assuming a small and constant time step Δt, we can then discretize our dynamics using Euler's method:

$$s_{k+1} \approx s_k + \Delta t f(s_k, u_k) = f_d(s_k, u_k)$$

4.2 Problem Formulation

4.2.1 Reference Trajectory Generation with iLQR

First, we make use of iLQR to generate a reference control trajectory that is feasible. For lane-keeping, our state should track the position and heading of the road center for each time step. We denote this tracked state as s_{road}. If we initialize a feasible nominal trajectory $(\bar{s}, \bar{u})$ and assume that the vehicle starts in the center of the lane with initial state $s_0 = (x_{\text{road},0}, y_{\text{road},0}, \psi_{\text{road},0}, v_0)$, we can formulate a quadratic cost function as follows to optimize the control trajectory:

$$\min_{u_0,\ldots,u_{N-1}} \frac{1}{2}(s_N - s_{\text{road},N})^T Q_N (s_N - s_{\text{road},N}) + \frac{1}{2}\sum_{k=0}^{N-1} \left((s_k - s_{\text{road},k})^T Q (s_k - s_{\text{road},k}) + u_k^T R u_k\right)$$

subject to $s_{k+1} = f_d(s_k, u_k)$

Applying a second-order Taylor Series expansion to this cost function about the nominal trajectory $(\bar{s}, \bar{u})$ allows us to represent it in terms of the deviation variables $\delta s_k = s_k - \bar{s}_k$ and $\delta u_k = u_k - \bar{u}_k$. The linearized dynamics for these variables is represented by $\delta s_{k+1} = A_k \delta s_k + B_k \delta u_k$, where $A_k = \frac{\partial f_d}{\partial s}(\bar{s}_k, \bar{u}_k)$ and $B_k = \frac{\partial f_d}{\partial u}(\bar{s}_k, \bar{u}_k)$. Q_N, Q, and R are positive-definite penalty matrices that we can tune to achieve proper lane-keeping. iLQR can now be applied to this linearized objective to find a reference control trajectory $\bar{u}$.

4.2.2 MPC

Given a reference trajectory of the road s_{road}, we can formulate our nonlinear tracking MPC problem as follows, assuming a horizon window length N:

$$\begin{aligned}
\min_{\delta u_0,\ldots,\delta u_{N-1}} & \sum_{k=0}^{N-1} \left((y_k - r_k)^T Q (y_k - r_k) + \delta u_k^T R \delta u_k\right) \\
\text{subject to } & s_{k+1} = A_k s_k + B_k u_k + c_k, \\
& u_k = u_{k-1} + \delta u_k, \\
& y_k = C s_k, \quad r_k = C s_{\text{road},k} \\
& s_0 = s(t), \quad u_{-1} = u(t-1)
\end{aligned}$$

At each time-step, the optimization is solved using sequential convex programming (SCP) due to the nonlinear dynamics. Thus, for the i^{th} subproblem, with a current estimated solution of $(s^{(i)}, u^{(i)})$, we affinize the dynamics as $A_k = \frac{\partial f_d}{\partial s_k}(s_k^{(i)}, u_k^{(i)})$, $B_k = \frac{\partial f_d}{\partial u_k}(s_k^{(i)}, u_k^{(i)})$, and $c_k = f_d(s_k^{(i)}, u_k^{(i)}) - A_k s_k^{(i)} - B_k u_k^{(i)}$. For the first time-step, we "warm-up" the MPC controller by setting $(s^{(0)}, u^{(0)})$ at $t = 0$ equal to the first N values of the iLQR-generated reference trajectories. Q and R are positive definite cost matrices that we define, and C is a matrix that, when multiplied with a state vector, allows the objective function to consider only the first three values (x-position, y-position, and heading) of the state. From this MPC approach, the control input to the system is $u(t) = \delta u_0^* + u(t-1)$.

Experiments

5.1 Setup

With Python, we use the kinematic bicycle model with both vehicle-specific parameters l_r and l_f set to 1 m for simplicity and with a time-step of $\Delta t = 0.1$s to progress the dynamics. We define three different roads for the controllers to track step-by-step. The first road is a 90° elbow bend with a constant radius of curvature of 100 m, the second road is a 180° semi-circle bend with the same constant radius of curvature, and the third road is a curvy S-shaped road made up of two 180° bends of the same constant radii of curvature as the first two roads. All roads start at the origin of the inertial frame with an initial heading of 0° with respect to the x-axis of this frame. We assume that the vehicle initially starts in the center of the lane with a heading ψ matching the road's. The vehicle also starts with an initial velocity of $v_0 = 5$ m/s.

For each road, we first use the iLQR controller to determine the open-loop reference state and control trajectories that help initialize the MPC controller. Here, we tune the Q, Q_N, and R matrices and terminal time T for the iLQR controller to successfully track the trajectory. The non-linear MPC tracking controller then uses the iLQR trajectories as an initial guess for the first time-step. For the MPC controller, we use a horizon length of $N = 5$ and a maximum number of $N_{\text{SCP}} = 5$ SCP iterations at each timestep.

5.2 Results and Discussion

We were able to get some results for the Lane Keeping via iLQR-initialized MPC Control for all three roads. Since the third S-shaped road builds upon the first two and is the most complex, we only discuss tracking results for this road.

We first implemented iLQR to generated the reference state and control trajectories that help initialize the MPC controller. We gave the iLQR controller a terminal time of 99 seconds to traverse through the S-shaped road. The resulting positional trajectory of the vehicle is plotted below, where the solid line represents the iLQR's computed trajectory and the dashed line indicates the center of the reference S-shaped road. We can see that the iLQR controller is able to follow the road quite well after performing a closed-loop simulation with the bicycle model we used as the dynamics for the iLQR. However, there is some slight under steering in the first 180° bend.

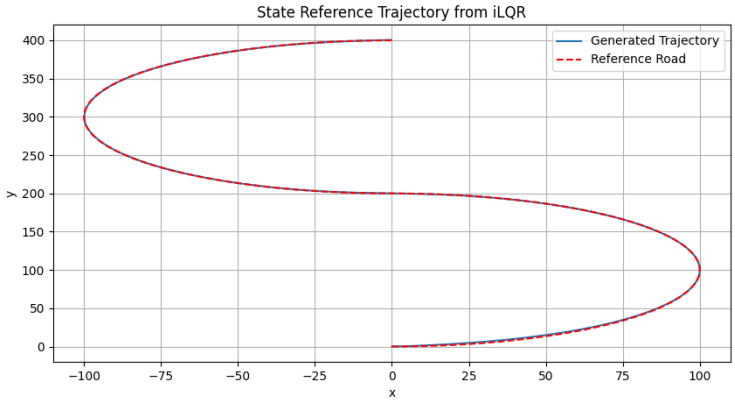
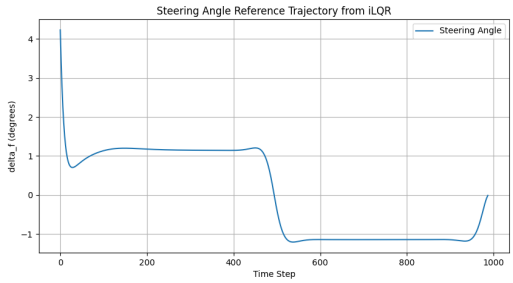
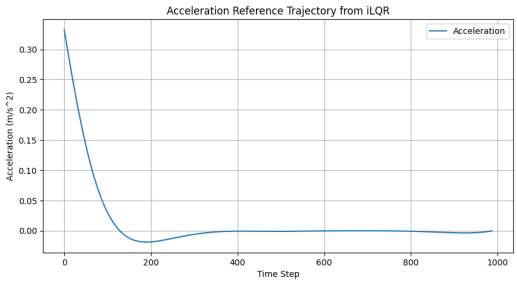


Figure 2: Reference Trajectory using iLQR

The steering angle δ_f, in degrees, shows us how the control input varies to help the ego vehicle follow the road. Especially during the curvier portions of the trajectory, we can see that the lateral control effort works to keep the vehicle in the center, as shown in the plot below. The iLQR controller compensates for the curvy road attributes by balancing between staying on the path and applying the lateral control effort via the steering angle. The longitudinal control effort to keep the vehicle at a constant velocity of 5 m/s is provided through the acceleration a. The larger acceleration associated with the beginning may explain the under steering seen in the first 180° bend.



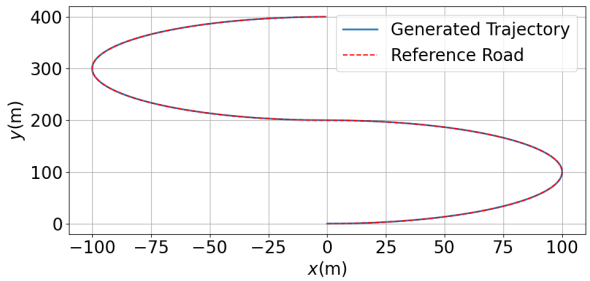
(a) Steering Angle plot for iLQR



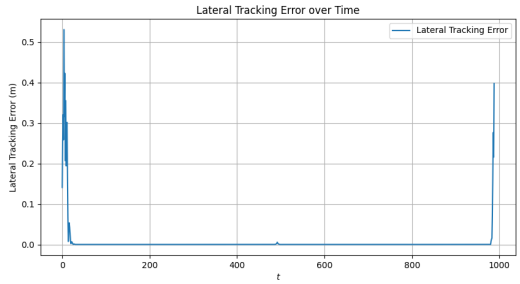
(b) Acceleration plot for iLQR

Figure 3: iLQR-generated Control Trajectories

As described in our methods above, we used these iLQR trajectories to initialize our MPC controller. After carrying out non-linear MPC to track the center of the road and simulating the vehicle's states, we calculate the lateral tracking error between the vehicle and road center at each time step. The plot of this error in in the figure below gives us insights into the performance of the control. Specifically, we see larger errors in the beginning and end of the vehicle's trajectory. This lateral tracking error may be due to inaccuracies in our modeling and problem formulation or due to the finite-horizon optimization of the MPC algorithm i.e. if the look-ahead window is too small. The lateral error can also be triggered due to the initial iLQR reference being imperfect, noise and convergence issues. From a visual inspection of the positional trajectory plot, the tracking of the lane-center from the MPC control is almost exact, with no noticeable understeering.



(a) Positional Trajectory

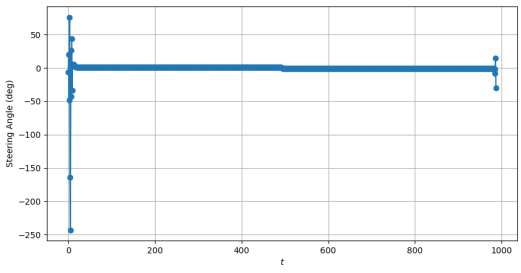


(b) Lateral Tracking Error

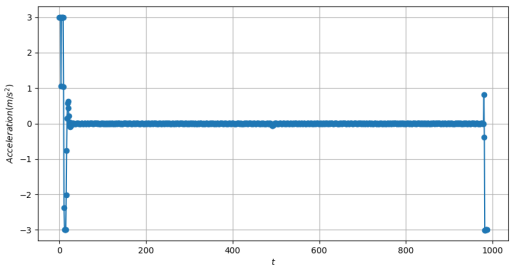
Figure 4: Positional Trajectory and Lateral Tracking Error from MPC Control

The MPC algorithm tries to track the reference trajectory by applying the control inputs in a receding horizon approach. In MPC, we compute the state and control inputs at each time step over the finite horizon. It makes use of the iLQR reference trajectories as the initial conditions to help optimize its computed tracking trajectory. Unlike iLQR, MPC allows us to directly include control constraints such as the maximum acceleration, which ensures the ego stays within an acceleration of 3 m/s^2 for comfort. We plot the MPC-generated control trajectories below. Compared to the iLQR control, the MPC control is much more noisy in the beginning and end with rapid steering, acceleration, and deceleration which could all be uncomfortable for passengers. This may be due to our inaccurate modeling of the MPC problem or due to a poor choice of our penalty matrices in the cost function. Even with this, the MPC-generated positional trajectory aligns better with the

center of the lane with less under steering compared to iLQR.



(a) Steering Angle Trajectory for MPC



(b) Acceleration Trajectory for MPC

Figure 5: MPC-generated Control Trajectories

Conclusions and Future Work

From the results, not only do we see how iLQR can be used to initialize an MPC controller, but we can also see the differences between the two control strategies for lane-keeping applications. Based on previous work, both algorithms are very useful in the planning and control strategies of the autonomous vehicles, and each have their benefit over the other. MPC can be used for quick online planning with a finite horizon and can readily integrate constraints, while iLQR can help optimize the vehicle's trajectory if the path of the vehicle is previously known.
For future work we can make use of model-based reinforcement learning (RL) techniques to learn and improve the vehicle dynamics model used in the MPC formulation. This may help reduce the noise in our resulting control trajectories and can be compared to our iLQR-reinforced MPC controller. We can also include a hybrid control architecture where the high level planning can take place by making use of RL, but for the safety controller we can stick to the traditional control strategies so that the RL-based control is within the safety bounds.

Video Presentation

Video Presentation for the Project

Github Repository

You can view "MPC_ILQR.py" in this Github repository for our relevant Python Code.

Sharing Approval

We grant the permission to share our report, video, and code with future AA203 students.

References

[1] M. M. Sherif, A. M. Ahmed, A. M. Moustafa, and M. Moness, "Optimal control of lane keeping system using simulated annealing and linear quadratic regulator," in *2019 15th International Computer Engineering Conference (ICENCO)*, pp. 1–5, 2019.

[2] L. Guo, L. Wei, P. Ge, and Z. Qin, "Lane keeping controller based on lqr optimized by genetic algorithm," in *2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, pp. 1–6, 2022.

[3] D.-H. Lee, "Lane-keeping control of autonomous vehicles through a soft-constrained iterative lqr," 2023.

[4] Y. Xu, B. Chen, X. Shan, W. Jia, Z. Lu, and G. Xu, "Model predictive control for lane keeping system in autonomous vehicle," in *2017 7th International Conference on Power Electronics Systems and Applications - Smart Mobility, Power Transfer Security (PESA)*, pp. 1–5, 2017.

[5] S. Zhang, X. Zhuan, Y. Fang, and J. Cheng, "Model-predictive optimization for lane keeping assistance system with exponential decay smoothing," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 713–718, 2021.

[6] K. M. Ng, S. A. C. Abdullah, A. Ahmad, and J. Johari, "Implementation of kinematics bicycle model for vehicle localization using android sensors," in *2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC)*, pp. 248–252, 2020.

[7] A. Ailon and S. Arogeti, "Trajectory tracking control for a kinematic bicycle model," in *2020 28th Mediterranean Conference on Control and Automation (MED)*, pp. 526–531, 2020.