Comparison between Model Predictive Control and Iterative Learning Control on Autonomous Race Tracking

Alex Wu[1] and Wei Liang[2]

***Abstract*— This is the final report for CMU 16899 final project to implement iterative learning control (ILC) and adaptive learning model predictive control (ALMPC) strategies on race tracking. The goal of this project is to compare and contrast the performances of a simple race car system running on a given track using ILC and ALMPC.**

I. INTRODUCTION

The project implemented iterative learning control (ILC) and adaptive learning model predictive control (ALMPC) strategies on tracking and compare and contrast the performances of a simple race car (agent) system running on a given track (environment) using ILC and ALMPC. The agent underwent several iterations until the control inputs converge. Initially, we will introduce constant disturbances to verify the robustness of these two control algorithms. Numerical simulations were conducted on a virtual track with different reference velocities, unknown offsets, and regional disturbances to the system dynamics. The project provided analysis at the similarities and differences between the two optimal trajectories generated by respective control algorithms.

II. BACKGROUND AND PROBLEM OVERVIEW

A. Background

As artificial intelligent and learning-based controllers became more mature, autonomous driving race cars has recently come to reality. Some questions that are always been asked such as how well a computer can do, and is a computer better than humans? Here in the paper we will introduce two methods to control a simplified race car models around a L shape track. Then we will discuss about the similarities and differences between the results.

The goal of autonomous racing is to drive a car as fast as possible around a track. In real-world, the vehicle is operating at its performance limits in a race, so the vehicle dynamics become highly nonlinear given the tire-road friction and road surface conditions. One control method for race tracking is ILC. Since the vehicle is racing multiple laps around the same sequence of turns and the track is not changing from lap-to-lap, ILC can be used to gradually determine the control input to correct the tracking errors and reduce lap time [1]. Another approach is learning-based model predictive control (LMPC) that exploits information from the previous laps to improve the performance of the closed loop system over iterations [2], [3].

B. Problem Overview

Existing ILC methods for highly dynamic vehicle driving applications focus on determining the proper steering inputs for a transient driving maneuver. The longitudinal acceleration is typically assumed constant and is not affected by the iterative learning controller [1].

ALMPC can be employed on uncertain constrained linear systems performing iterative tasks [4] and stable linear time invariant systems [5] in the presence of bounded disturbances. In autonomous driving area, ALMPC is more popular to solve lane keeping problems that minimize the distance from lane centerline and the steady state heading angle error [6]. ALMPC has not been deployed on autonomous race yet. In this project, we will develop and deploy a controller using ALMPC to minimize the lap time and tracking errors and compare its performance with ILC.

In this study, we proposed an ILC approach combined with MPC to introduce the longitudinal acceleration as the control input to the vehicle dynamics system. Secondly, we proposed an ALMPC approach to solve autonomous racing problems and compare the results with ILC and LMPC methods. In the discussion, we compare the performances and talk about the differences of ILC, LMPC, and ALMPC in autonomous racing applications.

III. PROJECT DESIGN

A. Vehicle Model and Control Problem

The vehicle dynamics is described by:

$$x_{t+1} = f(x_t, u_t), \tag{1}$$

where $f(x_t, u_t)$ is the vehicle dynamic state update equation. The details of the vehicle dynamics can be found in [7, Chapter 2]. The states and inputs are:

$$\begin{aligned} x &= [v_x, v_y, w_z, e_\psi, s, e_y] \\ u &= [\delta, a] \end{aligned}$$

where v_x and v_y are longitude and lateral velocities, w_z is the yaw rate, and s is the travelled distance along the centerline of the track. There are Gaussian noises on the measurements of v_x, v_y, and w_z. The states e_ψ and e_y are the heading angle and lateral distance error between the vehicle and the centerline of the track. The control inputs include steering angle δ and longitude acceleration a. Note that the initial position is identical to the finish line position on the track.

[1]Alex Wu is with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA chichiaw@andrew.cmu.edu

[2]Wei Liang is with the School of Architecture, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA weiliang@andrew.cmu.edu

B. *Controller Design*

- ILC: ILC is usually used for determine a steering input for highly dynamic vehicle trajectories. Yet it can improve the tracking performance through the iterations, ILC still needs a pre-defined racing trajectory as a reference. A common form of ILC control input is

$$u_t^{j+1} = u_t^j - \mathbf{L}e_j, \tag{2}$$

where $\mathbf{Q}$ and $\mathbf{L}$ matrices will be obtained us proportional-derivative or quadratically optimal ILC controller.
At each sampling time, a constrained finite–time optimal control problem for a horizon of N is solved to minimize a given cost function. The optimization problem to be solved at each time step t at each iteration j is given by,

$$\min_{u_t^j} \sum_{k=t}^{t+N-1} \|x_{t+k+1} - x_{ref,t+k+1}\|_Q + \|u_{t+k|t}\|_R \tag{3}$$

$$\text{s.t. } x_{t+k+1} = A^{j-1}x_{t+k} + B^{j-1}u_{t+k|t} + E^{j-1}\theta_{t+k} + w_{t+k} \tag{4}$$

$$x_{k+t|t} \in \mathcal{X}, u_{t|t} \in \mathcal{U}, x_{t|t} = x_t^j$$

where (4) is the linearized model of the vehicle dynamics based on last ($(j-1)$th) iteration by least squared method. The detail of the linearization can be found in [10], [12]. C^{j-1} is the residual array. The reference x_ref can be a racing trajectory generated by an expert or the centerline of the track. The Q matrix is a positive semi-definite matrix and the R matrix is a positive definite matrix.
The proposed control policy is given by,

$$u_t^j = u_{t|t}^{*,i} - Le^j, \tag{5}$$

where $L \in \mathbb{R}^{n_u \times n_x}$ is the learning function matrix that changes through the iterations. It is chosen to be the optimal LQR gain for system (4) with parameters Q and R. The linear programs (LP) arising in the iterative MPC algorithm are solved in OSQP Python module [11]. A feasible initial trajectory is created by a PID controller to initialize the convex safe set of states and L. The PID controller is given by,

$$u(x) = [\delta, a]^T = -\begin{bmatrix} k_{p,e_Y}e_Y + k_{p,e_\psi}e_\psi \\ k_{p,v_{ref}}(v_x - v_{ref}) \end{bmatrix}$$

- LMPC:
LMPC is an advanced version of a standard MPC controller. Our team implemented a LMPC controller described in [8], [9], [10]. The LMPC controller is capable of generating a optimal trajectory(shortest lap time) without a pre-computed reference, and it is assuming that there will be no model mismatch in the controller, where it will lead us to the ALMPC controller described in the next subsection. The key difference between a LMPC controller and a standard MPC controller is the construction of the terminal costs. In a LMPC controller, the terminal cost is defined based on previous iteration data. At a state along the jth iteration of a LMPC controller, a safe set is defined as follow,

$$\mathcal{CL}_l^j(x) = \{\bar{x} \in \mathbb{R}^n : \exists\lambda \in \mathbb{R}^{K(j-l+1)}, \lambda \geq 0, \mathbf{1}\lambda = 1, D_i^j(x)\lambda = \bar{x}\}$$

$$D_i^j(x) = \{x_{t_1^l}^l, ..., x_{t_k^l}^l, ..., x_{t_1^j}^j, ..., x_{t_k^j}^j\},$$

$$S_i^j(x) = \{x_{t_1^l+1}^l, ..., x_{t_k^l+1}^l, ..., x_{t_1^j+1}^j, ..., x_{t_k^j+1}^j\}$$

CL is a defined local convex safe set. It is constructed using a subset of the stored data points. In particular, the local convex safe set around x is defined as a convex hull of the K-nearest neighbors to x. D matrix contains these K closest points(states) from the previous l to j - 1 iterations, and a S matrix stores how these k closest points propagate according to the previous data. Then a local convex Q-function is defined to further estimate the terminal costs of a LMPC controller. A Q function and J(cost-to-go) are defined as follow.

$$Q_l^j(\bar{x}, x) = \min_\lambda \mathbf{J}_l^j(x)\lambda$$

$$\text{s.t.} \lambda \geq 0, \mathbf{1}\lambda = 1, D_i^j(x)\lambda = \bar{x}$$

$$\mathbf{J}_l^j(x) = [J_{t_1^l \to T^l}^l(x_{t_1^l}^l), ..., J_{t_M^l \to T^l}^l(x_{t_M^l}^l), ..., J_{t_1^j \to T^j}^j(x_{t_1^j}^j), ... J_{t_M^j \to T^j}^l(x_{t_M^j}^j)]$$

This Q function can be thought of as the smallest combination of previous cost-to-go values. A cost-to-go is defined simply as the time the race car needs to take to get to the finish line.
Lastly, a local LMPC is trying to solve the following finite time optimal control problem:

$$J_{t\to t+N}^{LMPC}(x_t^j, z_t^j) = \min_{U_t^j, \lambda_t^j} [\Sigma_{k=t}^{t+N-1} h(x_{k|t}^j) + \mathbf{J}_l^{j-1}(z_t^j)\lambda_t^j]$$

$$s.t \quad x_{t|t}^j = x_t^j$$

$$\lambda_t^j \geq 0, \mathbf{1}\lambda_t^j = 1, D_i^j(z_t^j)\lambda = x_{t+N|t}^j$$

$$x_{k+1|t}^j = A_{k|t}^j x_{k|t}^j + B_{k|t}^j u_{k|t}^j + C_{k|t}^j$$

where, U is a set of control inputs and h(x) is the stage cost.

$$h(x) = \begin{cases} 1 & \text{If} \quad x \notin X_{finished} \\ 0 & \text{Else} \end{cases}$$

The above A, B and C matrices describe the system dynamics. If the system is a non-linear system, then further linearization needs to be done. As one can perceive, the above optimal control problem is similar to that of a MPC. The only distinctive part is the way a LMPC controller construct the terminal cost, which is described in this section.

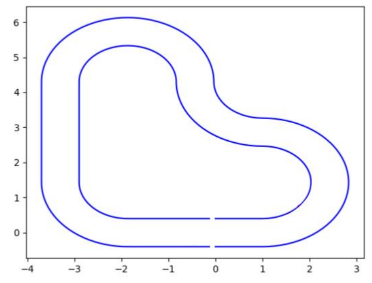


Fig. 1: L-shaped Race Track

- ALMPC: ALMPC is a modified version of a LMPC controller, where an unknown offset is added to the system. An ALMPC controller is able to learn the possible value set of the offset and update this believe at every following iteration. There for the system dynamics is defined in the following equation:

$$x_{t+1} = Ax_t + Bu_t + E\theta_a + w_t$$

Here θ_a is the unknown offset of the system. In a ALMPC controller, a ”Feasible Parameter Set” is used to estimate the possible range this unknown offset. At each iteration in an ALMPC controller, the following equation is used to update the belief of this feasible parameter set,

$$\theta^i = \{\theta \in R^p, x_t^i - Ax_{t-1}^i - Bu_{t-1}^i - E\theta \in W\}$$

where W is the variance of the noise w within the dynamics system. With this ALMPC approach, the controller is able to converge to a steady solution even when there is an unknown constant offset to the model.

IV. SIMULATION ENVIRONMENT

A. Track

We used Python to build both ILC and ALMPC controller and simulated the project. We referred to Berkeley MPC Lab Learning Model Predictive Control (LMPC) for autonomous racing[1] to build our controllers. Due to time constraint, the whole simulation is in 2D and in Python. A 19.22m long and 0.8m wide L-shape track is used for the simulation, as shown in Fig. 1. The iterative racing lap trahjectories are shown in red and a race car is described using a red rectangle running on the track, as shown in the later sections. During our experiment, we tried on different tracks, such as gargle-shaped, and round-shape tracks. However, as the complexity of the track increases, it is more likely that the solution can not converge to an optimal track. Therefore, through some attempts, we decided on using this L-shaped track, which can enlarge the differences between two algorithms.

[1]`https://github.com/urosolia/RacingLMPC`

B. Comparison Formulation and Disturbance Design

There will be two aspects of comparisons. First aspect is the comparison of the robustness. Even though a tracking environment is relatively ”steady” compared with real-world driving situations, there are still disturbances such as wind resistance and road frictions. Therefore, the robustness of these algorithms is crucial. Secondly, we are also curious on the optimal trajectories defined by different algorithms. A more detailed analysis will be focused on why a certain feature in the trajectory is aligned with the optimization logic of a certain algorithm.

We will design the disturbance as follows: First, we added Gaussian noises to the dynamic space, v_x, v_y, and w_z with a variance 0.01. Secondly, we built another simulation environment by adding general offset to the first two states v_x and v_y. The third environment, with the second condition enabled, will be an additional constant disturbance over a certain segment of the track, $s \in [5, 8]$.

V. ILC SIMULATIONS AND RESULTS

A. Initialization

The Q matrix is positive semi-definite with diagonal elements as $[1, 1, 1, 1, 0, 100]$. The R matrix is positive definite with diagonal elements as $[1, 10]$. The reference trajectory defined in this simulation is the centerline of the track. We can see that there is no weight on the travelled distance along the centerline of the track in the state space, but the weight of lateral error is high given the sensitivity of the state and to make sure the direction of the vehicle dynamics is in line with the track. The penalty of longitudinal acceleration is also large to secure the vehicle to stay in the track while in highly dynamics and large curvatures.

The values of PID controller initialization is $k_{p,e_Y} = 0.6$, $k_{p,e_\psi} = 0.9$, and $k_{p,v_x} = 1.5$.

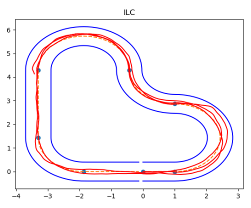
We simulated the proposed ILC approach with two reference longitudinal velocities, 0.8m/s and 1.6m/s.

B. Simulation with Regular Reference Longitudinal Velocity

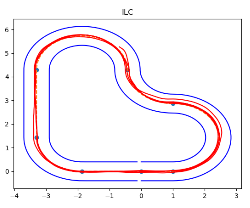
From Fig. 2(a) to (d), the racing trajectories converge to the reference centerline track through the iterations of update of dynamics and learning function matrix L. Fig. 3(a) indicates the convergence of root mean square of lateral and heading angle error through the iterations. Nonetheless, there is an ”elbow point” at the 3rd iteration, as the convergence slows down and the heading angle error starts oscillation. When the ILC approach was firstly introduced, there is a significantly drop of lap time from 36s to 28s, but the lap time did not improve through the iterations and oscillates around 28s.

C. Simulation with High Reference Longitudinal Velocity

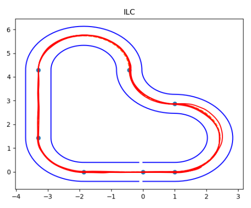
To further reduce the lap time and observe if the system can stay robust under a higher intensity of dynamics, we increase the reference longitudinal velocity from 0.8m/s to 1.6m/s. Compared to the previous session, Fig. 4(a) to (d) show a more discursive pattern, but the racing trajectories still converge to the reference track. The elbow point of



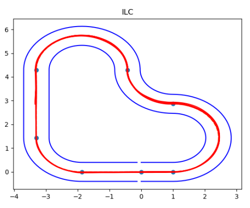
(a) Laps of first iteration.



(b) Laps of third iteration.

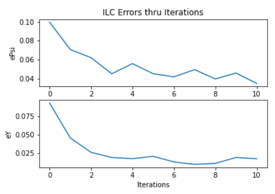


(c) Laps of fifth iteration.

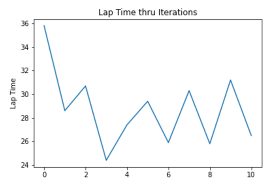


(d) Laps of Seventh iteration.

Fig. 2: Simulation results through the iterations of ILC approach with regular reference longitudinal velocity.



(a) Error convergence.



(b) Iterations of lap time.

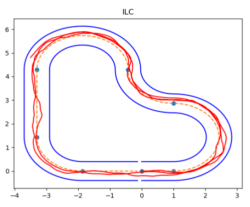
Fig. 3: Error convergence and Lap time trend through the iterations of ILC approach with regular reference longitudinal velocity.

convergence appear at the second iteration and both the final lateral and heading angle error are larger than the final errors with regular reference longitudinal velocity in Fig. 5(a). However, the lap time in Fig. 5(b) was reduced to 22.5s after 3 iterations that is lower than the final lap time with regular reference longitudinal velocity. Higher reference velocity brings more uncertainty and larger tracking error, but since the velocity is larger, the vehicle can finish a lap faster than before.

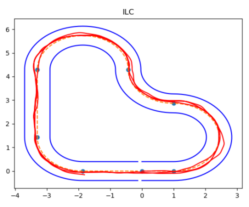
The simulations indicate the robustness of the proposed ILC approach under certain circumstances of longitudinal velocities. For more extreme vehicle dynamics and larger longitudinal velocities, LMPC and ALMPC will be applied in the next section.

D. Simulation with Unknown Offsets in System Dynamics

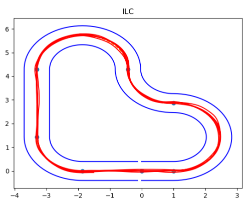
After the baseline simulations, we tested ILC approach with offsets to the first two states, v_x and v_y. The offset was added at 0.01m and 0.03m. The ILC approach learned the parameter very quick and still converged to the reference track, as shown in Fig. 6. However, the proposed ILC approach was not able to handle larger unknown offsets. As a comparison, ALMPC was able to tolerate offset 10 times larger.



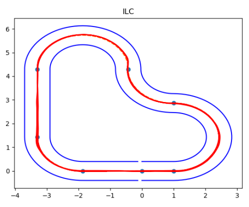
(a) Laps of first iteration.



(b) Laps of second iteration.

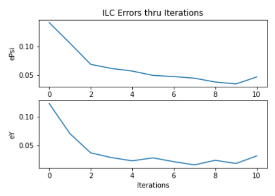


(c) Laps of third iteration.

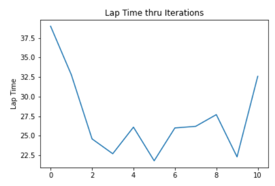


(d) Laps of fifth iteration.

Fig. 4: Simulation results through the iterations of ILC approach with high reference longitudinal velocity.



(a) Error convergence.



(b) Iterations of lap time.

Fig. 5: Error convergence and Lap time trend through the iterations of ILC approach with high reference longitudinal velocity.

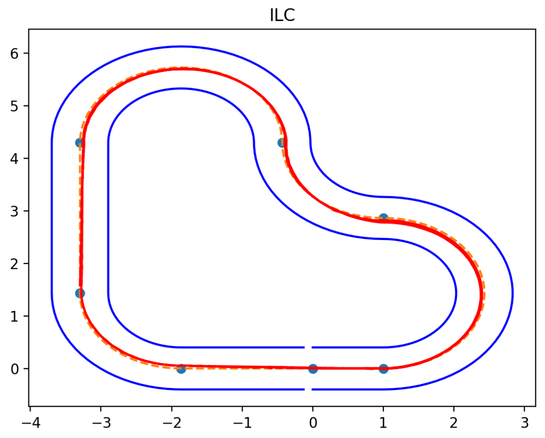


Fig. 6: Last (10th) iteration of ILC with unknown offset

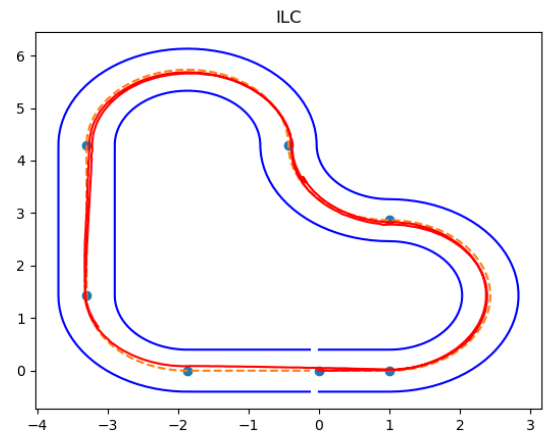


Fig. 7: Last (10th) iteration of ILC with regional offset

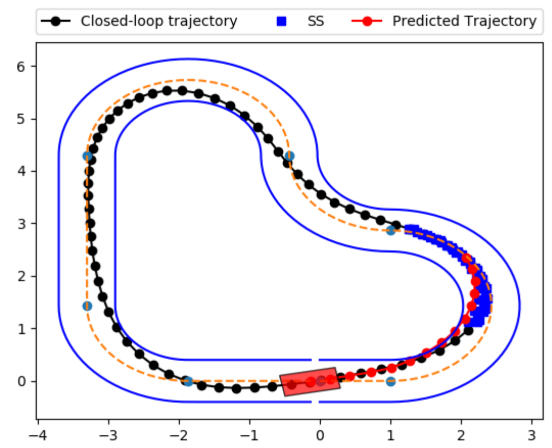


Fig. 8: LMPC Result

E. *Simulate with Additional Regional Disturbances*

Furthermore, we introduced constant disturbance along a section of the track. We introduced constant offsets at a segment of the track, $s \in [5, 8]$. The offset was added as 0.01m for v_x and 0.03m for v_y. The results shown in Fig. **??** indicates that ILC is able to handle this situation, same as ALMPC. However, this simulation condition was built based on the second simulation environment that means the system ILC applied to has a smaller general offsets than the system for ALMPC.

In summary, the proposed ILC shows a significant converges of tracking errors to a reference racing trajectory and the ability to reduce the lap time to a certain level. Notwithstanding, the proposed ILC approach may not be an ideal solution to race time oriented problem meanwhile maintain robustness. In this case, we proposed an ALMPC approach based on the LMPC approach in [10] that shows outperforms ILC and LMPC in regard to noises and offsets.

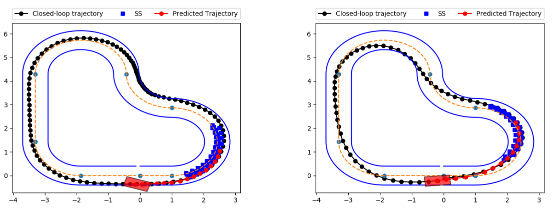
VI. LMPC AND ALMPC SIMULATIONS AND RESULTS

A. *Standard LMPC Results – Base line comparison*

The initialization of the convex safe set of the states were still constructed by the PID controller in this section. The first analysis will be focused on the results from LMPC and ALMPC controllers. The results are relatively similar, so we will discuss them side by side. In our simulation, we first used a LMPC controller to control the race car. Here, we set our iteration to 43 laps, and used 4 past trajectories to construct our local safe set. The result was very promising as shown in figure 8. As one can perceive, the controller was able to converge to a optimal trajectory, where the lap time was around 5.7s.

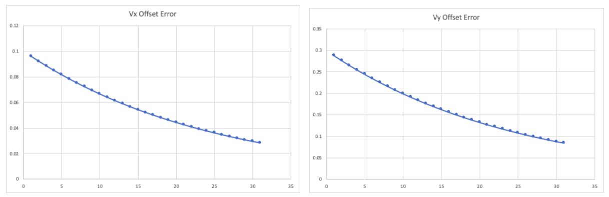
B. *Simulate with Unknown Offset in System Dynamics*

We introduced some offsets in the first two terms of the state, i.e. v_x and v_y. The offsets were set at 0.1m and 0.3m. The results of the two algorithms are shown in Fig. 9. First of all, the offsets were set at 0.1m and 0.3m for easy comparison between the two algorithms, since a LMPC controller was not able to handle bigger unknown offsets. In the scenarios that a LMPC controller could still obtain a converged trajectory, it was not optimal, as shown in Fig. 9. The final lap time of a LMPC controller was at around 10.5s, where it was 6.5s for the ALMPC controller. However, with the ALMPC controller, the controller was able to learn the offsets and apply this learned value to get optimal trajectory. The error between estimated offsets and true offsets were shown in figure 10.



(a) LMPC Result (b) ALMPC Result

Fig. 9: System Dynamics with offsets.



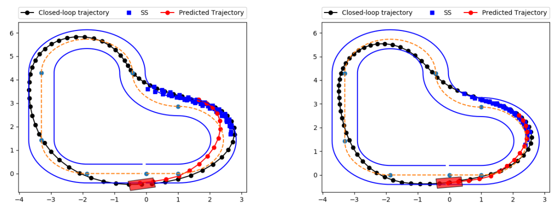
(a) Offset Error in Vx (b) Offset Error in Vy

Fig. 10: Error between ground truth offsets and predicted offsets.

C. *Simulate with Additional Regional Disturbances*

Furthermore, we introduced constant disturbance along a section of the track. We introduced constant offsets on

v_x(+0.5) and v_y(-0.01), at $s \in [5, 8]$. The results are shown in Fig. 11. Here the LMPC controller seems to have a similar result to the one for ALMPC. This was not always the case. More experiment will need to be done in order to justify it. An initial guess of this is that the disturbances somehow eliminated the effects of the offsets, so an LMPC controller can also obtain a relatively good result. Even though the results are similar, an ALMPC controller is more robust to the disturbances. A bigger disturbances will cause the LMPC to fail. In general, we proved that an ALMPC was more robust and better at handling model mismatch, compared to an LMPC controller.



(a) LMPC Result (b) ALMPC Result

Fig. 11: System Dynamics with offsets and disturbances.

VII. CONCLUSIONS

Two algorithms gave rise to two different optimal trajectories. The reason is due to the fact that the underlying optimizations are different. For an ILC controller, it reduces the overall lap costs by minimizing the error in local y direction at each time stamp along the track. In LMPC and ALMPC controllers, they minimize the total lap costs by reducing the total number pf time stamps. In short, they should be used in different scenarios. An ILC controller should be used to follow a trajectory while an ALMPC controller can be used to generate a shortest lap time trajectory. As for robustness, an ALMPC controller is more robust and less prone to model uncertainties and disturbances. It is able to estimate the unknown offset within the system and applied the estimation back to the control inputs. To sum up, both control algorithms have different features and optimizing logic. When it comes to controlling a race car on a track to minimize the lap time, an ALMPC controller is better.

REFERENCES

[1] N. R. Kapania and J. C. Gerdes, "Path tracking of highly dynamic autonomous vehicle trajectories via iterative learning control," in *Proceedings of the American Control Conference*, vol. 2015-July. Institute of Electrical and Electronics Engineers Inc., jul 2015, pp. 2753–2758.

[2] U. Rosolia, A. Carvalho, and F. Borrelli, "Autonomous racing using learning Model Predictive Control," in *Proceedings of the American Control Conference*. Institute of Electrical and Electronics Engineers Inc., jun 2017, pp. 5115–5120.

[3] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli, "Repetitive learning model predictive control: An autonomous racing example," in *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, vol. 2018-Janua. Institute of Electrical and Electronics Engineers Inc., jan 2018, pp. 2545–2550.

[4] M. Bujarbaruah, X. Zhang, U. Rosolia, and F. Borrelli, "Adaptive MPC for Iterative Tasks," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 2018-Decem. Institute of Electrical and Electronics Engineers Inc., jan 2019, pp. 6322–6327.

[5] M. Bujarbaruah, X. Zhang, and F. Borrelli, "Adaptive MPC with Chance Constraints for FIR Systems," in *Proceedings of the American Control Conference*, vol. 2018-June. Institute of Electrical and Electronics Engineers Inc., aug 2018, pp. 2312–2317.

[6] M. Bujarbaruah, X. Zhang, H. E. Tseng, and F. Borrelli, "Adaptive MPC for Autonomous Lane Keeping," jun 2018. [Online], *available at* `http://arxiv.org/abs/1806.04335`

[7] Rajamani, Rajesh, *Vehicle Dynamics and Control*, Springer US, 2012.

[8] Rosolia, Ugo and Borrelli, Francesco,"Learning Model Predictive Control for Iterative Tasks: A Computationally Efficient Approach for Linear System," *IFAC-PapersOnLin*, Vol. 50, No. 1, 3142–3147,(jul 2017). Elsevier B.V.

[9] Rosolia, Ugo and Borrelli, Francesco,"Learning model predictive control for iterative tasks. A data-driven control framework," *IEEE Transactions on Automatic Control*, Vol. 63, No. 7, 1883–1896,(jul 2018). Institute of Electrical and Electronics Engineers Inc.

[10] Rosolia, Ugo and Borrelli, Francesco,"Learning How to Autonomously Race a Car: A Predictive Control Approach," *IEEE Transactions on Control Systems Technology*, Vol. 28, No. 6, 2713–2719,(nov 2020). Institute of Electrical and Electronics Engineers Inc.

[11] Stellato, Bartolomeo and Goran Banjac,"OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, 12, 637–672 2020.

[12] Carvalho, Ashwin and Gao, Yiqi and Gray, Andrew and Tseng, H. Eric and Borrelli, Francesco,"Predictive control of an autonomous ground vehicle using an iterative linearization approach," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2335–2340,2013. Institute of Electrical and Electronics Engineers Inc.