MPC based Trajectory Tracking for Autonomous Vehicles

Tu Le Cong
Dept. Control Engineering and Automation
Ho Chi Minh City University of Technology
Ho Chi Minh City, Viet Nam
Email: tu.lecongtu305@hcmut.edu.vn

Hung Nguyen Nhat
Dept. Control Engineering and Automation
Ho Chi Minh City University of Technology
Ho Chi Minh City, Viet Nam
Email: hung.nguyen2010307@hcmut.edu.vn

Hien Nguyen Vo Hong My
Dept. Control Engineering and Automation
Ho Chi Minh City University of Technology
Ho Chi Minh City, Viet Nam
Email: hien.nguyen2010260@hcmut.edu.vn

Hao Nguyen Vinh
Dept. Control Engineering and Automation
Ho Chi Minh City University of Technology
Ho Chi Minh City, Viet Nam
Email: vinhhao@hcmut.edu.vn

Abstract—**This paper investigates the comparison between Model Predictive Control (MPC) and Linear Quadratic Regulator (LQR) algorithms for real-time trajectory tracking in a four-wheel autonomous vehicle model. The vehicle is equipped with a dual-loop control system: an inner loop managed by an STM32F407 microcontroller for speed and steering angle control, and an outer loop managed by a Jetson Nano to minimize trajectory error and reduce steering oscillations. The vehicle uses RTK GPS for precise positioning and an IMU for orientation. This study implements both algorithms in a real-time system, comparing their performance, integration feasibility, and effectiveness in achieving smooth and accurate trajectory tracking.**

Index Terms—**Autonomous vehicle, trajectory tracking, MPC controller**

I. Introduction

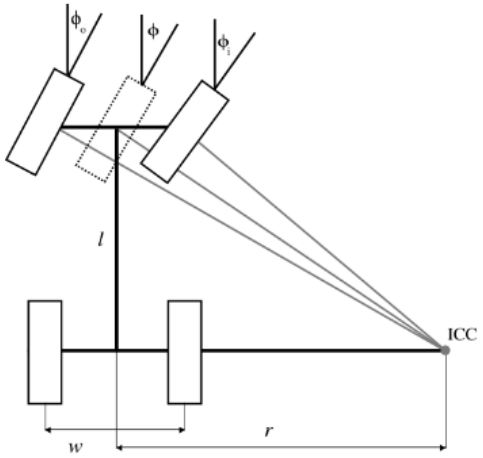
Autonomous cars have emerged as a pivotal innovation in modern transportation, promising enhanced safety and efficiency. These vehicles operate by integrating Advanced Driving Assistance Systems (ADAS) and Automatic Driving Systems (ADS), which collectively minimize human error, optimize travel time, and provide accessible transportation solutions for individuals with disabilities and the elderly [1]-[2]. ADAS features include adaptive cruise control, lane-keeping assistance, and automated emergency braking, which enhance driving safety by assisting human drivers in various tasks [3]. ADS represents a more advanced level of automation, capable of handling all driving tasks without human intervention, paving the way for fully autonomous vehicles [4].

Autonomous vehicles are equipped with a range of sophisticated technologies that enable them to sense their environment, plan their routes, and make optimal decisions. Sensing technologies, such as LiDAR, radar, and cameras, allow the vehicle to accurately perceive its surroundings [5]-[6]. The planning system processes this sensory data to chart a safe and efficient path, while the decision-making system ensures the vehicle can react appropriately to dynamic conditions, such as traffic changes or unexpected obstacles [7]. Central to these features is the trajectory tracking controller, which plays a crucial role in maintaining the vehicle's path with two essential criteria: high accuracy and smoothness. These two criteria ensure safe and comfortable rides, particularly for individuals with disabilities or the elderly who rely on autonomous vehicles for mobility [8]-[9].

Over a long period of development, many controllers have been proposed with the goal of consistently improving the accuracy and stability. There are two popular geometry controllers include Pure Pursuit, Stanley. They are commonly used due to the simple calculation, the control signal is derived based on the geometric relationships between the vehicle model and the trajectory, and the number of parameters that need to be adjusted is small. The two controllers give good results at low vehicle speeds [10]-[11]. However, they have many disadvantages that need to be supported by other controllers. Specifically for Pure Pursuit, we only need to adjust look ahead distance parameter, but it depends on the vehicle's speed. At different velocity values, we need to re-select the appropriate look ahead distance. Some solutions have been proposed to update the look ahead distance through a fuzzy control system [12]-[13] or calculated based on trajectory curvature from GPS data, vehicle speed and cross track error. Despite improvements, Pure Pursuit still has unresolved problems of poor return ability when the vehicle’s position is outside look ahead distance from the path. At the same time, Pure Pursuit is unable to respond to dynamic factors when the vehicle operates at high speeds. Therefore, adjusting look ahead distance is not enough to maintain path tracking quality. Stanley and Pure Pursuit both have the same drawback: there is no constraint on the steering angle at the controller output and does not care about the car’s heading at the target point that the vehicle is aiming for [14]. In article [15], Stanley has added a saturation at the controller’s output basing on designer's maximum value, and also added two tuning coefficients for heading error at the target point and steering angular speed. However, each set of tuning parameters is only suitable at a certain vehicle speed value and still needs to be combined with the adaptive algorithm.

To address these limitations, more advanced controllers like Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) have been introduced. Both LQR and MPC utilize the state space representation of the vehicle, which considers the vehicle's kinematic properties comprehensively. The path-tracking problem, as a multi-constrained optimization problem, must account for location error, comfort, and mechanical and electrical constraints.

The Linear Quadratic Regulator (LQR) is a popular control method that optimizes the control inputs to minimize a quadratic cost function over an infinite horizon. This cost function typically includes terms for tracking error and



control effort, balancing the need for accuracy and smoothness. LQR uses a state space representation of the vehicle's kinematic, allowing it to consider multiple state variables simultaneously. By solving the Riccati equation, LQR computes the optimal feedback gain matrix that minimizes the cost function. This approach ensures stable and efficient control, making it suitable for real-time applications.

Model Predictive Control (MPC) has gained considerable attention in the field of autonomous vehicle control due to its ability to handle multiple constraints simultaneously. MPC formulates a multivariable cost function based on future reference states and minimizes this function while respecting various constraints [16]. MPC is developed from Linear Quadratic Gaussian (LQG) algorithms, which optimize over an infinite horizon. LQG algorithms are simpler and better at dealing with disturbance rejection, whereas MPC involves finite-horizon optimization, requiring more complex online computations. Despite this complexity, MPC provides superior trajectory tracking and smoother control action changes [17]. Additionally, Traditional control methods often assume that control values will not reach actuator saturation limits, which is unrealistic in practice. For instance, when a vehicle is far from its destination, the control signals can exceed admissible values, leading to actuator saturation. Autonomous vehicles also have mechanical and electrical components that are subject to physical constraints [18]. Considering these constraints online is crucial for accurate and reliable path-tracking MPC handles actuator saturation and physical limits more effectively than traditional methods. By incorporating prediction models, receding horizon optimization, and feedback correction, MPC addresses issues related to input constraints and state admissibility, making it an attractive choice for real-time applications [19].

The main purpose of this paper is to compare MPC and LQR in real-time path-tracking applications, evaluating their performance and suitability for integration into autonomous vehicle systems. Through this comparison, we aim to highlight the strengths and limitations of each method, providing insights into their practical applications in enhancing autonomous vehicle control.

II. MODELING

A. Steering linkage model

The steering mechanism model is built based on the above model with the steering shaft from the steering wheel being connected to the control motor shaft.

The Ackerman steering condition allows the conversion from a 4-wheel kinematic model to a bicycle model. The steering angle of the bicycle model is related to the steering angle of the front wheel in the 4-wheel kinematic model as follows:

$$\emptyset_i = tan^{-1}\left(\frac{2lsin\emptyset}{2lcos\emptyset - wsin\emptyset}\right) \quad (1)$$

$$\emptyset_o = tan^{-1}\left(\frac{2lsin\emptyset}{2lcos\emptyset + wsin\emptyset}\right) \quad (2)$$

From the steering angle of the front wheel in the 4-wheel model, through the kinematic relationship of the steering mechanism, we convert to the displacement of the rack.

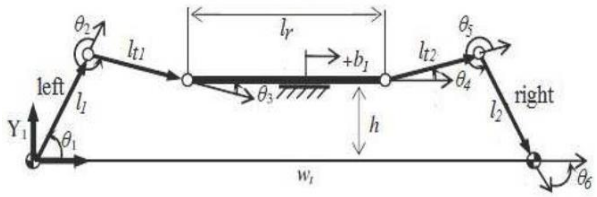


Fig. 2. Steering linkage [21]

$$x_0 = l_1 cos\theta_1 + \sqrt{l_{t1}^2 - (l_1 sin\theta_1 - h)^2} \quad (3)$$

$$x_1 = l_1 cos(\theta_1 - \emptyset_i) + \sqrt{l_{t1}^2 - (l_1 sin(\theta_1 - \emptyset_i) - h)^2} \quad (4)$$

$$\Delta x = x_1 - x_0 \quad (5)$$

Through the contact between the rack and pinion, the displacement of the rack will be converted into the rotation angle of the gear or also the rotation angle of the motor shaft.

$$\theta = 2\pi\frac{\Delta x}{\pi mZ} = 2\frac{\Delta x}{mZ} \quad (6)$$

B. Kinematic vehicle model

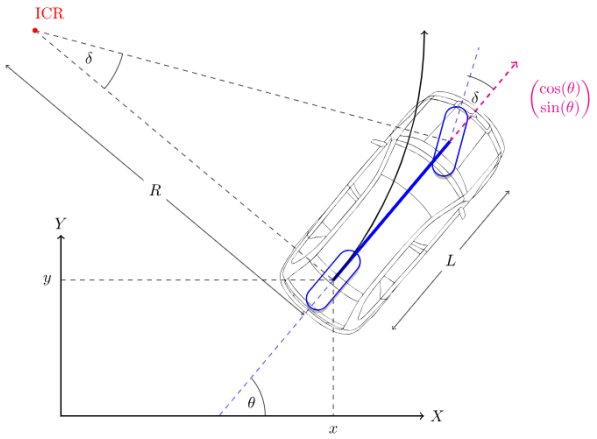


Fig. 3. Vehicle Model [22]

TABLE I
SYMBOLS AND DEFINITIONS IN VEHICLE MODEL

Symbol	Definition
x,y	The coordinates of the midpoint between the two rear wheels
θ	Heading angle
L	Distance from front to rear axle
v	Linear velocity
δ	Front wheel steering angle

The trajectory tracking controllers in this paper require the state model of the system. This paper uses a vehicle kinematic model. The vehicle's position is determined by the coordinates x and y at the center of the rear axle, obtained from a GPS device, and the heading angle θ, obtained from an IMU device.

The vehicle's kinematic equations are described as follows:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \frac{v}{L}\tan(\delta) \end{cases} \tag{7}$$

C. Linear vehicle model

The vehicle kinematics described above (8) is nonlinear and can be represented as follows:

$$\dot{X} = f(x,u) \tag{8}$$

Where x is state variable consisting [x, y, θ], the input signal u is δ, and the model parameters include [L, v]. he nonlinear equation (7) can be approximated linearly by performing a first-order Taylor expansion at the operating point (X_r, u_r):

$$\dot{X} \approx f(X_r,u_r) + \left.\frac{df(X,u)}{dX}\right|_{X=X_r}(X-X_r) + \left.\frac{df(X,u)}{du}\right|_{u=u_r}(u-u_r) \tag{9}$$

From equations (7), (8), and (9), the linearized model equations are:

$$\begin{cases} \dot{x} \approx v\cos(\theta_r) - v\sin(\theta_r)(\theta-\theta_r) \\ \dot{y} \approx v\sin(\theta_r) + v\cos(\theta_r)(\theta-\theta_r) \\ \dot{\theta} \approx \frac{v}{L}\tan(\delta_r) + \frac{v}{L}(\tan^2(\delta_r)+1)(\delta-\delta_r) \end{cases} \tag{10}$$

Discretizing the model, the state-space equations of the model can be expressed as follows:

$$\begin{aligned} \tilde{X}(k+1) &= A\tilde{X}(k) + B\tilde{u}(k) \\ \tilde{Y}(k) &= C\tilde{X}(k) \end{aligned} \tag{11}$$

where

$$A = \begin{bmatrix} 1 & 0 & -v\sin(\theta_r(k))(\Delta t) \\ 0 & 1 & v\cos(\theta_r(k))(\Delta t) \\ 0 & 0 & 1 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{v}{L}(\tan^2(\delta_r(k))+1)(\Delta t) \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\tilde{X}(k) = X(k) - X_r(k) = \begin{bmatrix} x(k) - x_r(k) \\ y(k) - y_r(k) \\ \theta(k) - \theta_r(k) \end{bmatrix},$$

$$\tilde{u}(k) = u(k) - u_r(k),$$

$$\begin{bmatrix} x_r(k+1) - x_r(k) \\ y_r(k+1) - y_r(k) \\ \theta_r(k+1) - \theta_r(k) \end{bmatrix} = \begin{bmatrix} v\cos(\theta_r(k))\Delta t \\ v\sin(\theta_r(k))\Delta t \\ \frac{v}{L}\tan(\delta_r(k))\Delta t \end{bmatrix},$$

III. Controller Design

A. Structure of Control System

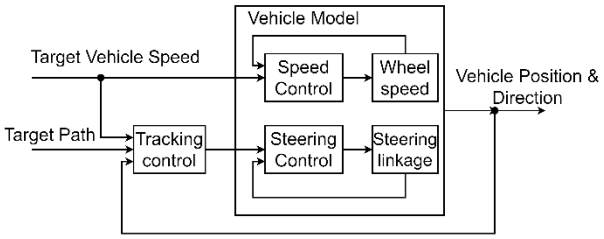


Fig. 4. Structure of Control System

Figure 4 illustrates the structure of the control system, which is divided into two control loops (two-loop control). The inner loop is responsible for controlling the speed and steering angle of the vehicle. In this paper, we use a PI controller to control the velocity and a Fuzzy PD controller to control the steering angle of the vehicle. The outer loop controls the vehicle's kinematics, ensuring that the vehicle consistently follows the predetermined trajectory. An MPC controller is used to manage the vehicle's trajectory tracking. Additionally, we use an LQR controller to compare and provide observations on the advantages and disadvantages of MPC.

Additionally, to compare predictive capabilities with an MPC controller, LQR is supplemented with a Pure Pursuit controller. The Pure Pursuit controller utilizes a lookahead point as a feedforward signal, while the signal from LQR serves as feedback. [23]

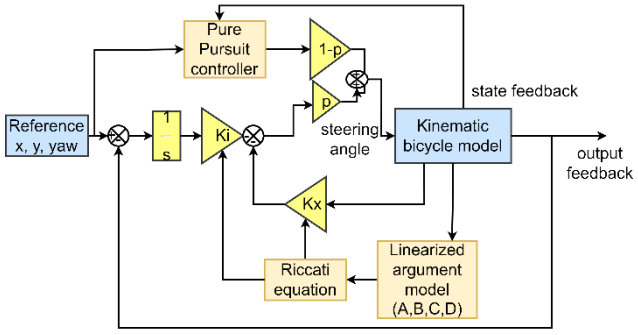


Fig. 5. LQR controller

Figure 5 illustrates the LQR controller, where the coefficient p represents the correlation coefficient between the two controllers. It demonstrates which information is weighted more heavily.

B. Steering Controller

We perform indirect steering control, through conversion from the steering angle in the bicycle model to the rotation angle of the control motor. Use a fuzzy PD controller because the control motor already has an accumulation stage inside

and aims to achieve control over a wide range of values set at the controller input.

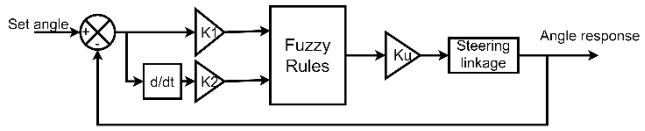


Fig. 6. Steering controller

Make selection of normalized values for the fuzzy PD controller based on the control range of steering angle from $-60°$ to $60°$, maximum motor speed and electrical power level at the engine input . Determine the language value for the control signal u at the output, the position error and the derivative of the error.

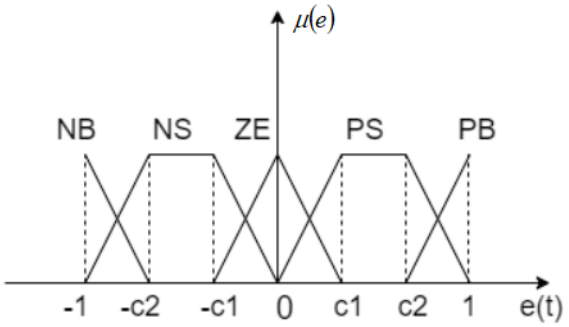


Fig. 7. Language value for angular error

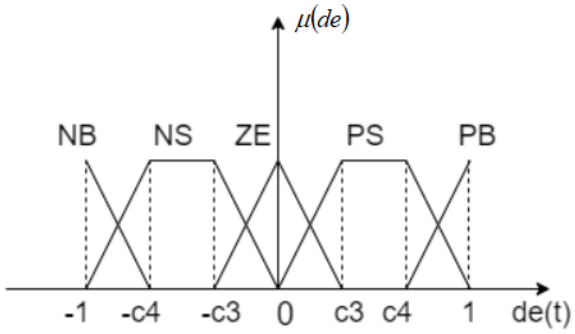


Fig. 8. Language value for derivative of angular error

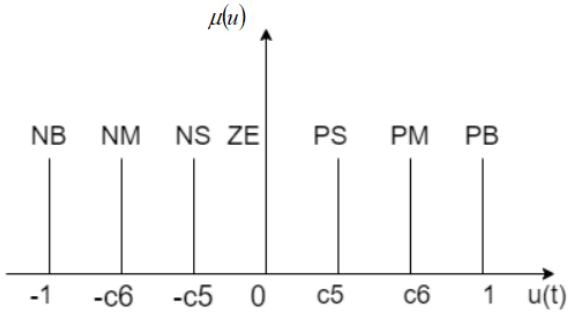


Fig. 9. Language value for control signal

TABLE II
INFERENCE RULE SYSYEM FOR FUZZY CONTROLLER

		Derivative of position error de(t)				
		NB	NS	ZE	PS	PB
position error e(t)	NB	NB	NB	NM	NS	ZE
	NS	NB	NM	NS	ZE	PS
	ZE	NM	NS	ZE	PS	PM
	PS	NS	ZE	PS	PM	PB
	PB	ZE	PS	PM	PB	PB

C. Model Predictive Control

At each sampling time, MPC predicts a sequence of vehicle behaviors within a specific prediction horizon. Based on this prediction, MPC computes a sequence of input signals over the prediction horizon to minimize the error between predicted state variables and the trajectory under predefined constraints. According to the principles of MPC, we have two distinct time domains: the real-time domain represented by the variable t, and the prediction time domain at each sampling step represented by the variable j.

To maintain generality, at each time t = k_i, the operating point for the Taylor expansion is chosen as $(X_r, u_r) = (X(k_i),0)$. To ensure system stability, MPC utilizes the augmented model of the vehicle described as follows:

$$X_e(k_i + j + 1) = A_e X_e(k_i + j) + B_e \Delta u(k_i + j)$$
$$Y_e(k_i + j) = C_e X_e(k_i + j) \quad (12)$$

In there,

$$A_e = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix}, \quad B_e = \begin{bmatrix} B \\ I \end{bmatrix}, \quad C_e = [C \quad 0],$$

$$X_e(k_i + j) = \begin{bmatrix} \tilde{X}(k_i + j) \\ \tilde{u}(k_i + j - 1) \end{bmatrix},$$

$$\Delta u(k_i + j) = \tilde{u}(k_i + j) - \tilde{u}(k_i + j - 1)$$

j = 0,1,2…N_P – 1.

The prediction horizon and control horizon in MPC are determined by two parameters Np, Nc (where Nc < Np). At sampling time t = k_i, MPC predicts Np steps of the vehicle's behavior and computes Nc control steps. We assume that $\Delta u(k_i + j) = 0$, j = Nc, Nc + 1,… Np – 1.

Assuming the validity of the linear Taylor expansion, the state matrices (A, B, C) within the prediction horizon are considered unchanged. The predicted state variables are represented as follows:

$$X_e(k_i + 1|k_i) = A_e X_e(k_i) + B_e \Delta u(k_i)$$
$$X_e(k_i + 1|k_i) = A_e^2 X_e(k_i) + A_e B_e \Delta u(k_i) + B_e \Delta u(k_i + 1)$$
$$X_e(k_i + Np|k_i) = A_e^{Np} X_e(k_i) + A_e^{Np-1} B_e \Delta u(k_i) + \ldots + A_e^{Np-Nc} B_e \Delta u(k_i + Nc - 1) \quad (13)$$

The predicted output matrix:

$$\tilde{Y} = F X_e(k_i) + G \Delta U \quad (14)$$

where

$$F = \begin{bmatrix} C_e A_e \\ C_e A_e^2 \\ \cdots \\ C_e A_e^{Np} \end{bmatrix}, \tilde{Y} = \begin{bmatrix} Y_e(k_i + 1) \\ Y_e(k_i + 2) \\ \cdots \\ Y_e(k_i + Np) \end{bmatrix},$$

$$\Delta U = \begin{bmatrix} \Delta u(k_i) \\ \Delta u(k_i + 1) \\ \cdots \\ \Delta u(k_i + Nc - 1) \end{bmatrix},$$

$$G = \begin{bmatrix} C_e B_e & 0 & 0 & \cdots & 0 \\ C_e A_e B_e & C_e B_e & 0 & \cdots & 0 \\ C_e A_e^2 B_e & C_e A_e B_e & C_e B_e & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ C_e A_e^{Np-1} B_e & C_e A_e^{Np-2} B_e & C_e A_e^{Np-3} B_e & \cdots & C_e A_e^{Np-Nc} B_e \end{bmatrix}$$

There is an issue we need to address: at the sampling time, we perform a Taylor expansion of $(X_r(k_i), u_r(k_i)) = (X(k_i),0)$. For the prediction points, how do we select the working point ($(X_r(k_i + j + 1), u_r(k_i + j + 1))$ $(j = 0, 1, 2 \ldots N_P - 1)$? According to equation (11), the Taylor expansion points for the prediction steps are related to the current Taylor expansion point.

$$\begin{bmatrix} x_r(k_i+j+1) - x_r(k_i+j) \\ y_r(k_i+j+1) - y_r(k_i+j) \\ \theta_r(k_i+j+1) - \theta_r(k_i+j) \end{bmatrix} = \begin{bmatrix} vcos(\theta_r(k_i))\Delta t \\ vsin(\theta_r(k_i))\Delta t \\ \frac{v}{L}tan(\delta_r(k_i))\Delta t \end{bmatrix},$$

From there, we establish a predicted trajectories that the vehicle needs to follow.

$$R_S = [R_{S1} \quad R_{S2} \quad \dots \quad R_{SNp}]^T \qquad (15)$$

$$R_{sj} = \begin{bmatrix} x_{ref}(k_i+j+1) - (x(k_i) + (j+1)vcos(\theta(k_i))\Delta t) \\ y_{ref}(k_i+j+1) - (y(k_i) + (j+1)vsin(\theta(k_i))\Delta t) \\ \theta_{ref}(k_i+j+1) - \theta(k_i) \end{bmatrix}$$

j = 0,1,2… Np-1

With the objective of minimizing the error between the predicted output and the instantaneous trajectory while avoiding excessive steering angle oscillations that could adversely affect the system, the cost function of MPC is formulated as follows:

$$J = [Y - R_s]^T Q_y [Y - R_s] + \Delta U^T R \Delta U \qquad (16)$$

where, Q_y and R are respectively the weighting matrices for the output variables and input signals.

$$Qy = \begin{bmatrix} Q_1 & 0 & 0 & \dots & 0 \\ 0 & Q_2 & 0 & \dots & 0 \\ 0 & 0 & Q_3 & \cdots & 0 \\ \dots & \dots & \dots & \cdots & \dots \\ 0 & 0 & 0 & \cdots & Q_{Np} \end{bmatrix},$$

$$Q_i = \begin{bmatrix} Q_x & 0 & 0 \\ 0 & Q_y & 0 \\ 0 & 0 & Q_\theta \end{bmatrix}, \qquad i = 1,2, \dots Np$$

$$R = \begin{bmatrix} R_1 & 0 & \dots & 0 \\ 0 & R_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & R_{Nc} \end{bmatrix},$$

$$R_i = [R_{\Delta\delta}], \qquad i = 1,2, \dots Nc$$

In the process of tuning parameters, two factors need to be considered: The first factor is the influence of the state variables as well as the input variables on each other through the matrices Q_i and R_i. The second factor is the influence between the prediction steps, similar to the **p** coefficient in the LQR system presented earlier. MPC can balance between using current state feedback information and future prediction information.

In this paper, we use the following weights:

$$Qy = \begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ 0 & 0 & Q & \cdots & 0 \\ \dots & \dots & \dots & \cdots & \dots \\ 0 & 0 & 0 & \cdots & 4xQ \end{bmatrix},$$

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.35 \end{bmatrix}, \qquad i = 1,2, \dots Np$$

$$R = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \ddots & \dots \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

First, we adjust the parameters correlating the state variables and input variables. We choose $Q_x = Q_y$ đ to optimize the trajectory error, and simultaneously, $Q_x = Q_y = R$ to ensure that the system does not excessively restrict steering angle oscillations during trajectory tracking. Q_θ is involved in preventing the phenomenon of swerving (steering angle oscillations) caused by position errors in x and y. Additionally, we need to balance between current information and future information. The Q matrix of the last prediction step has a weight four times larger than the previous prediction steps, highlighting the predictive nature of the MPC controller. This reflects the importance of the final prediction step in the MPC prediction sequence.

Second, for simplicity, we chose Np = Nc = 10, the prediction horizon is suitable through experimentation. The computational load is not too large, suitable for real-time systems along with the ability to forecast over a sufficiently long period for the vehicle to respond to future information.

From (14), (15), cost function is rewritten as follows:

$$J = 2[FX_e(k_i) - R_s]^T Q_y G \Delta U + \Delta U^T (G^T Q G + R) \Delta U \qquad (17)$$

The cost function is described as a quadratic function of the ΔU matrix variables, where the current state variables of the vehicle and the system matrix parameters are constants. The problem is formulated as Quadratic Programming (QP). QP algorithms are commonly used in optimization problems with constraints. To reduce computational complexity and enable real-time operation, our team opted for the classic QP Hildreth method (please refer to the source of this method) to solve the cost function with constraints, where the primary constraint in this system is the input steering angle constraint.

$$-25° \le \delta \le 25°$$

After obtaining the optimal control signal sequence ΔU, the first signal in this sequence is used as the steering angle signal that the vehicle needs to follow at each sampling time.

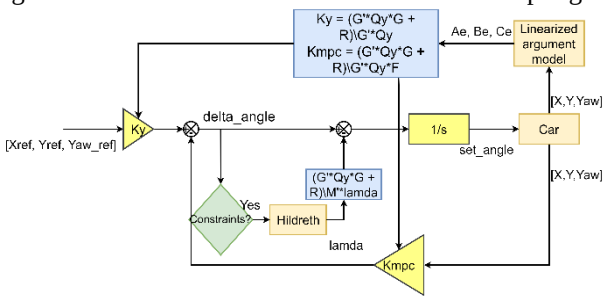


Fig. 10. MPC controller

IV. EXPERIMENTS

A. Setups

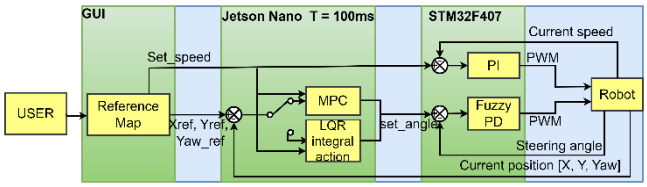


Fig. 11. Block diagram of car model's component

We use a four-wheel vehicle model to validate the MPC controller. Our system consists of three main blocks as depicted in Figure 11:

The lowest-level block (inner loop control): We use an STM32F407VGT6 to implement speed and steering control, along with corresponding Motor Driver and Motor Servo.

The next block: We employ a Jetson Nano to implement trajectory tracking control (MPC/LQR). Feedback signals are measured using an IMU ADIS16488 to determine the vehicle's orientation. We utilize the RTK GPS system NEO-M8P-0 and NEO-M8P-2 for position determination and accuracy improvement. The LORA E32-433T20D module is used as the communication protocol for GPS-RTK.

The final block: We utilize a personal laptop for monitoring and user intervention during the robot's operation.

After completing the construction of the car model, we have the following processing results:

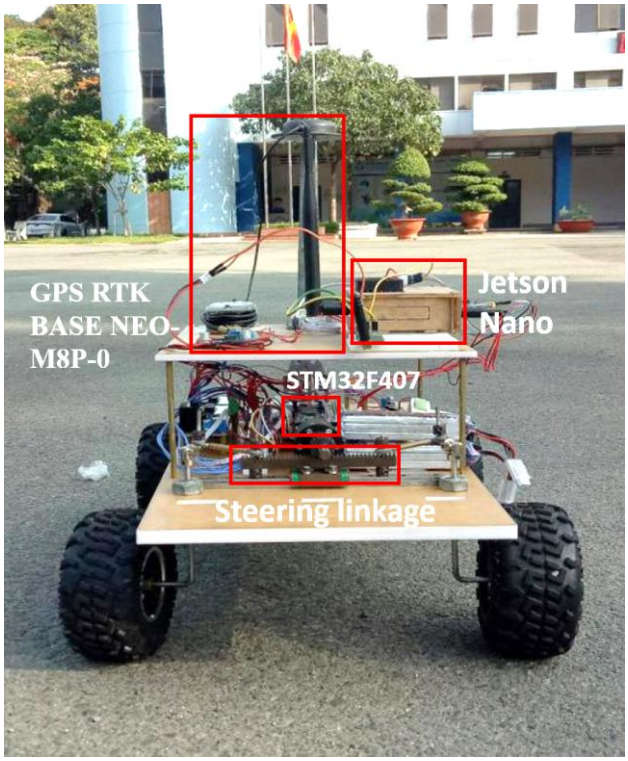


Fig. 12. Front side of car model

B. Results

To compare the trajectory tracking capability and constrained optimization ability of MPC and LQR, two types of trajectories are investigated: one with a figure-eight shape to assess trajectory tracking performance and steering angle stability provided by both controllers. The other trajectory has a square shape to evaluate constrained optimization and predictive capabilities of the controllers. These trajectories are evaluated under constant speed conditions using both simulation and experimental methods.

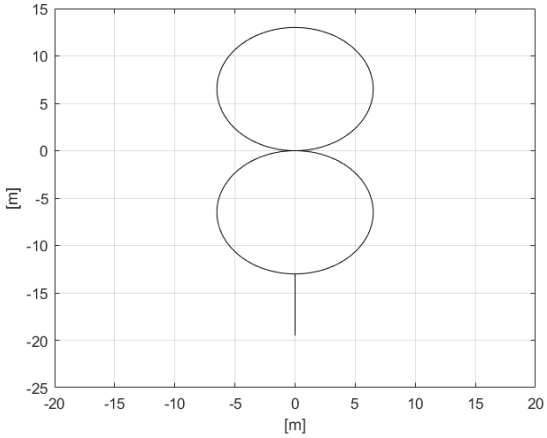


Fig. 13. Figure-eight trajectory

Figure 13 shows the figure-eight trajectory investigated in this paper, with each circle having a radius of 6.5 meters and a total path length of 245 meters. The vehicle follows the figure-eight trajectory with 3 loops per survey to validate the system's repeatability. The trajectory is evaluated with a vehicle speed v=0.6 m/s.

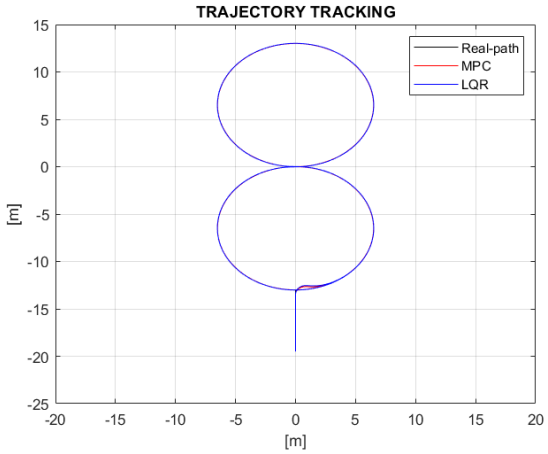


Fig. 14. Results of simulation for tracking the figure-eight trajectory

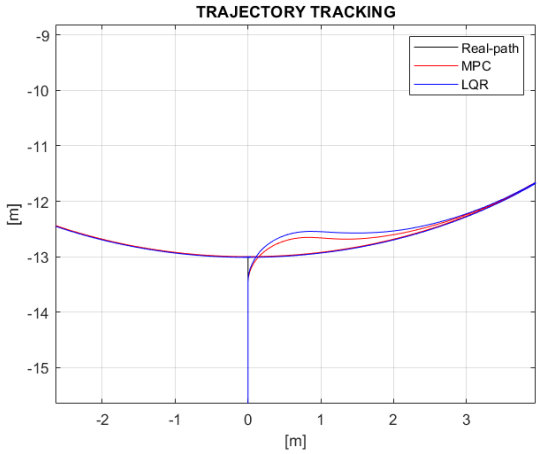


Fig. 15. Results of simulation for tracking the figure-eight trajectory

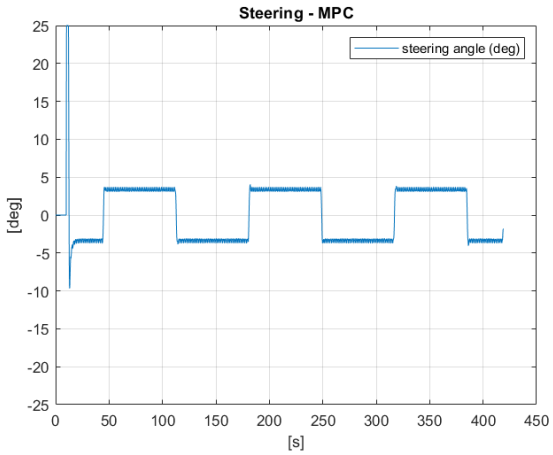


Fig. 16. Simulation results of the steering angle for MPC

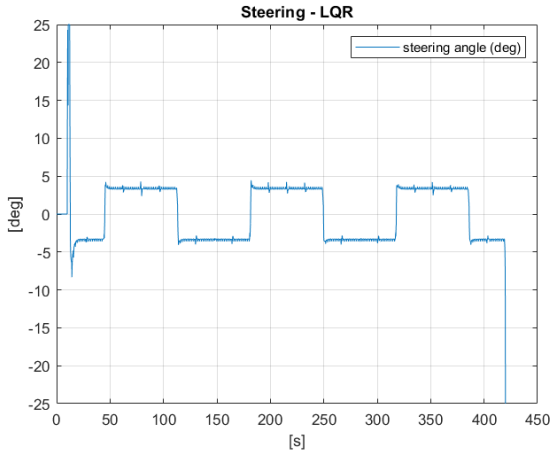


Fig. 17. Simulation results of the steering angle for LQR

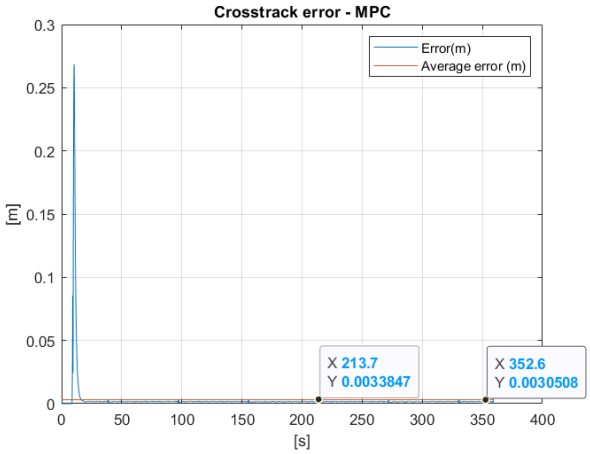


Fig. 18. Simulation results of crosstrack error for MPC

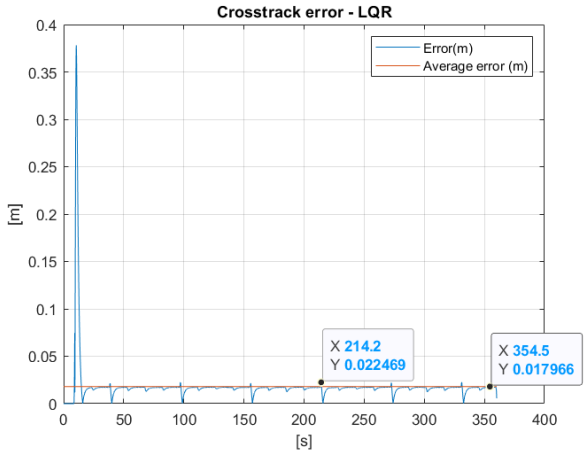


Fig. 19. Simulation results of crosstrack error for LQR.

Figure 14 to 19 show the simulation results of the two controllers when evaluating the figure-eight trajectory. Figures 16 and 17 indicate that the steering angle output from the MPC controller is more stable, while the LQR controller sometimes exhibits steering angle fluctuations. Figures 18 and 19 demonstrate that the crosstrack error of MPC is less than LQR, with the maximum errors being 0.0034 m for MPC and 0.0225 m for LQR. The average cross-track errors are 0.0031 m for MPC and 0.0178 m for LQR.

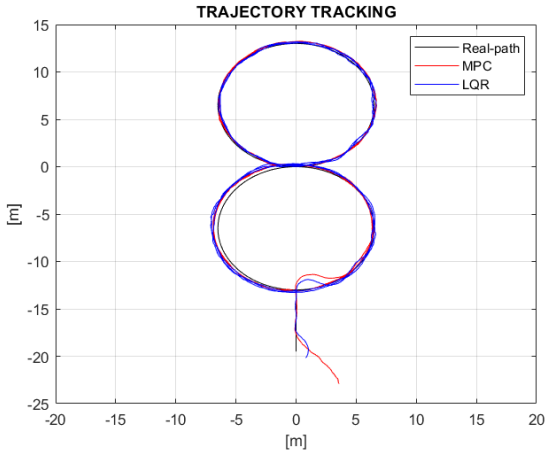


Fig. 20. Experimental results for tracking the figure-eight trajectory

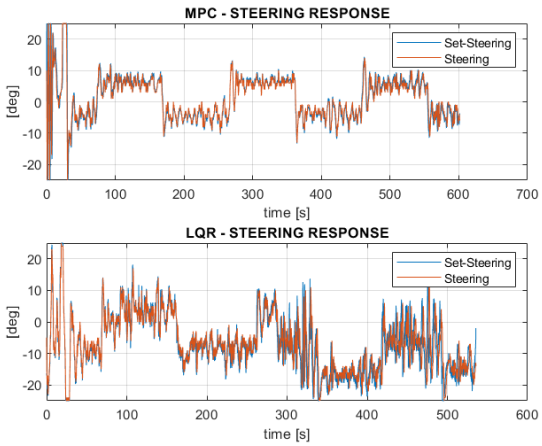


Fig. 21. Experimental steering response - figure-eight trajectory

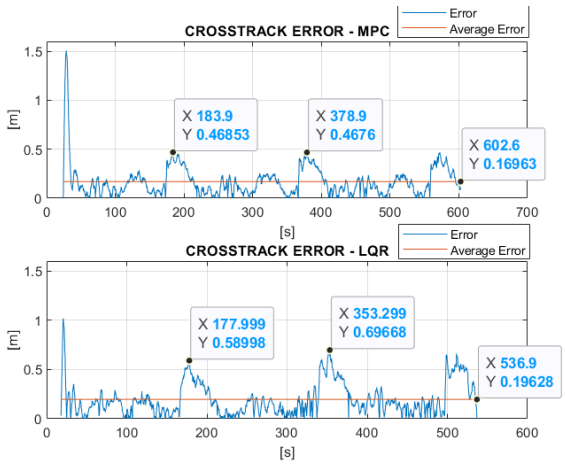


Fig. 22. Experimental cross-track error - figure-eight trajectory

Figure 20 to 22 show the experimental results of the two controllers when evaluating the figure-eight trajectory. Figure 21 demonstrates that the stability of MPC is better than LQR, specifically showing that the steering frequency of MPC is lower compared to LQR, thus improving the system. Figure 22 illustrates that the cross-track error of MPC is less than LQR, with the maximum cross-track errors being 0.469 m for MPC and 0.5899 m for LQR. The average trajectory tracking errors are 0.1696 m for MPC and 0.1963 m for LQR.

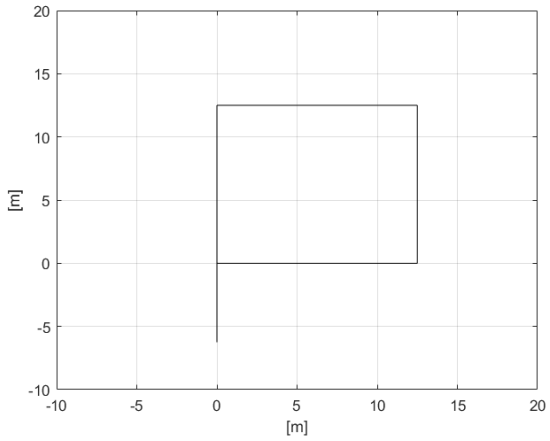


Fig. 23. The square trajectory

Figure 23 depicts a square trajectory with each side measuring 12.5 meters, totaling 100 meters in path length. The vehicle follows the square trajectory with 2 loops per survey to assess system repeatability. The trajectory is evaluated at a constant speed of v = 0.7 m/s.

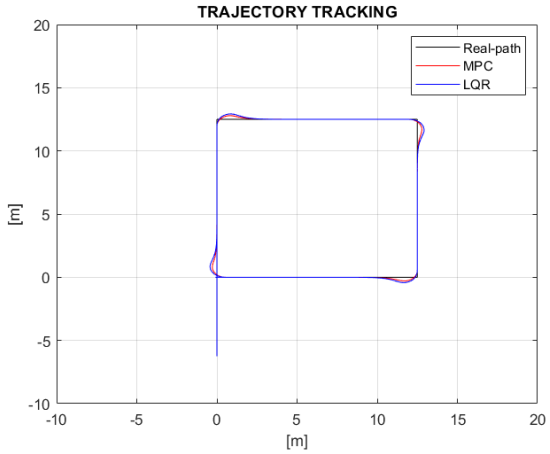


Fig. 24. Simulation results for tracking the square trajectory

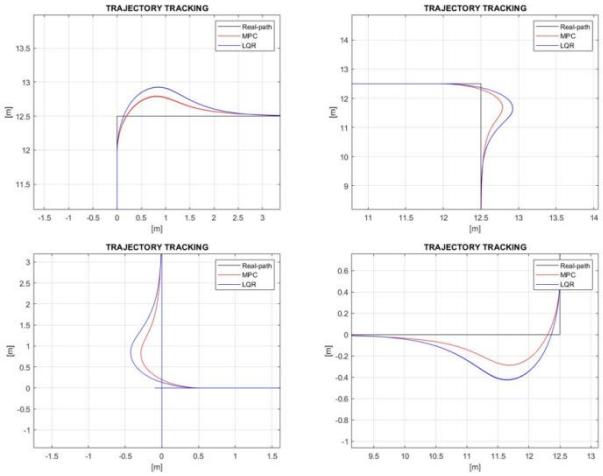


Fig. 25 . Simulation results for tracking the square trajectory – zoom in

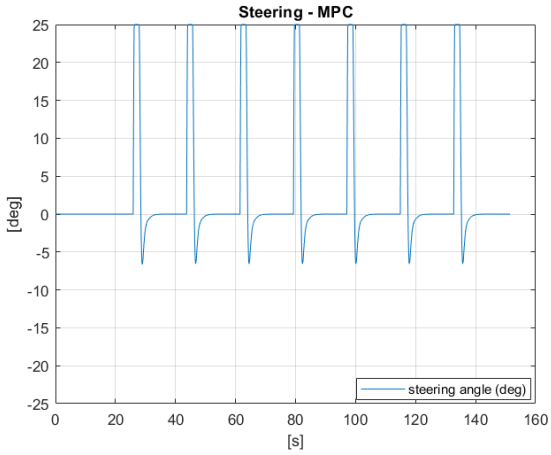


Fig. 26. Steering angle simulation - MPC

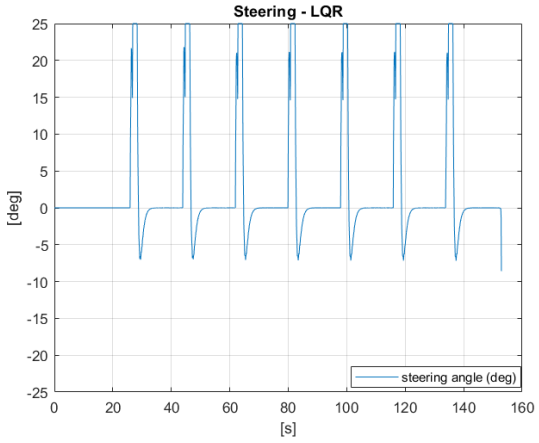


Fig. 27. Steering angle simulation - LQR

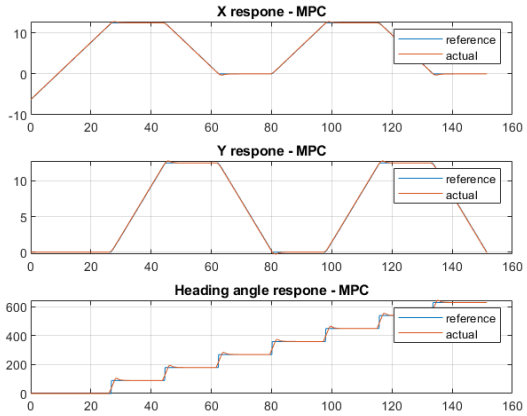


Fig. 28. State variable response simulation - MPC

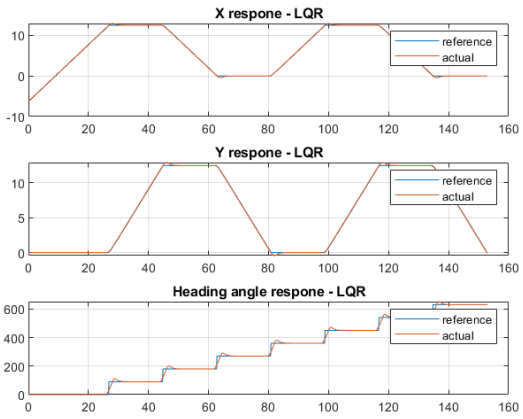


Fig. 29. State variable response simulation - LQR

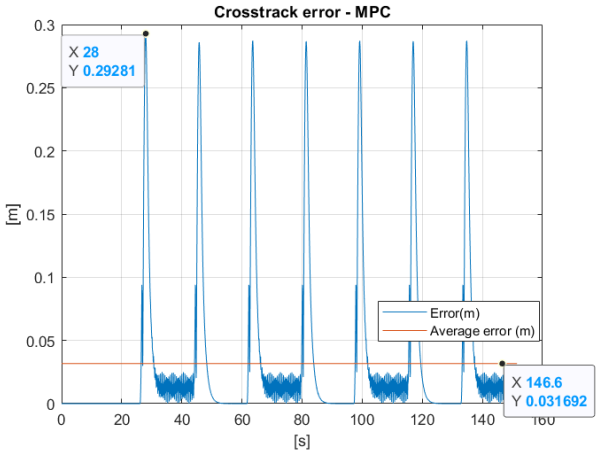


Fig. 30. Cross-track error simulation - MPC

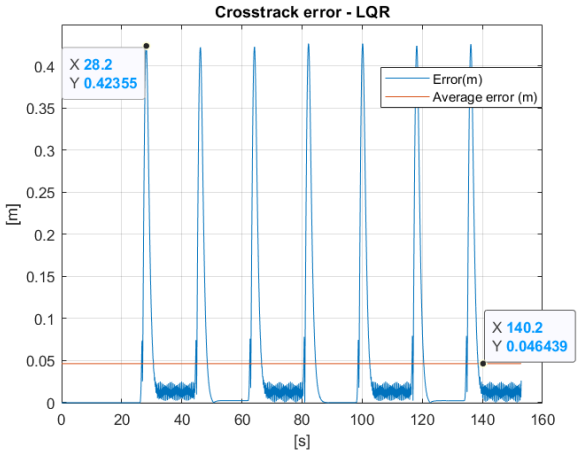


Fig. 31. Cross-track error simulation - LQR

Figures 24 to 31 show the simulation results of the two controllers when evaluating the square trajectory. Figures 24 and 25 demonstrate the predictive capability of both controllers when the vehicle tends to turn right before encountering the corners of the trajectory. Due to constraints on the steering angle, there is a spike in the response at the corners of the trajectory. However, thanks to their predictive and optimization capabilities, both controllers show improvements compared to traditional controllers. Figures 26 and 27 illustrate the steering angle responses from both controllers, with LQR showing some sharp peaks. Figures 28 and 29 depict the heading angle responses, where both controllers exhibit a tendency to react earlier to changes in the setpoint signal, resulting in reduced lag and overshoot. Figures 30 and 31 show that the maximum cross-track errors are 0.293 m for MPC and 0.424 m for LQR. Additionally, the average cross-track errors are 0.032 m for MPC and 0.046 m for LQR.

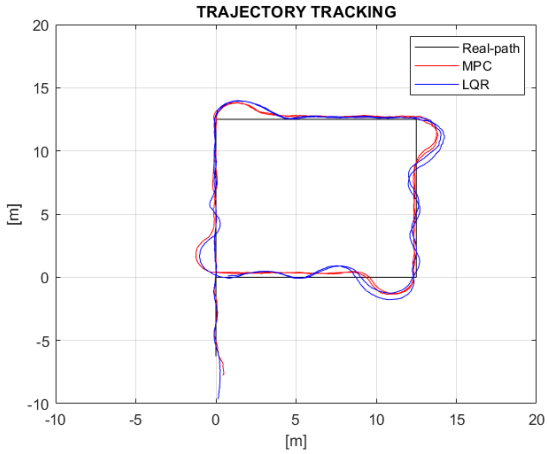


Fig. 32. Experimental results for tracking the square trajectory

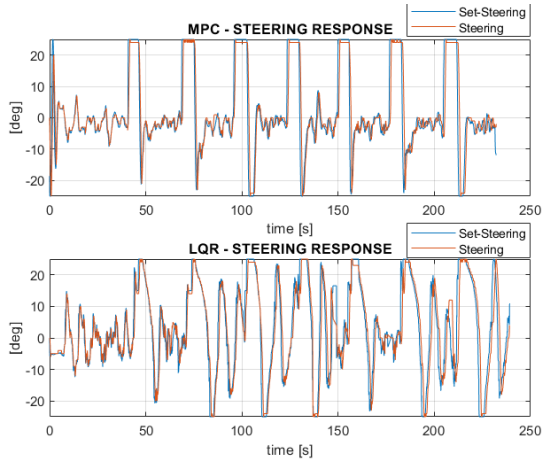


Fig. 33. Experimental steering response - square trajectory

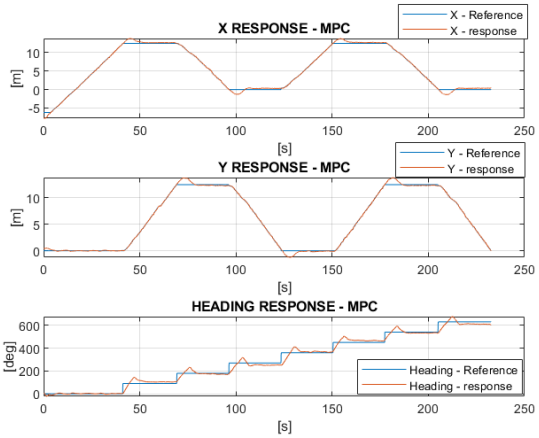


Fig. 34. Experimental state variable response - MPC

Figure 1

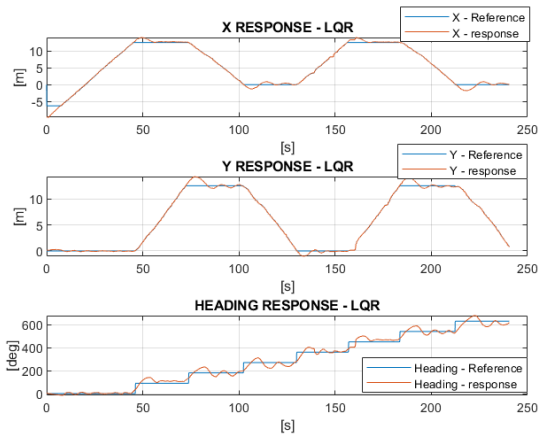


Fig. 35. Experimental state variable response - LQR

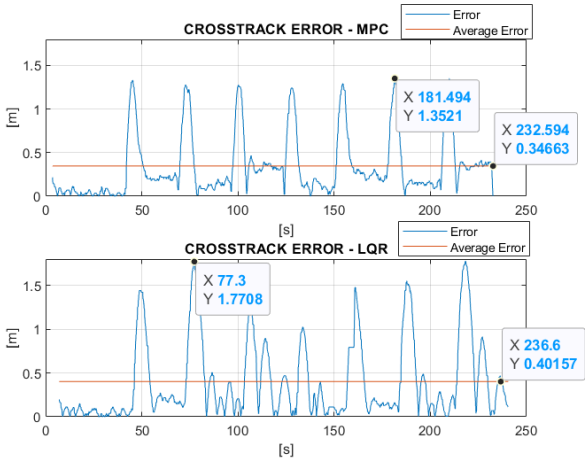


Fig. 36. Experimental trajectory tracking error - square trajectory

Figures 32 to 36 present the experimental results of the two controllers when evaluating the square trajectory. Figure 32, although not fully showcasing future prediction capability, demonstrates that the system compensates for steering angle delays and improves trajectory tracking, reducing overshoot. Figure 33 illustrates the stable steering angle response from MPC compared to LQR. Specifically, MPC shows a clear trend in steering angle at each time point, whereas LQR does not. Additionally, the steering frequency of MPC is lower than that of LQR. Figures 34 and 35 depict the heading angle responses, where MPC shows a more stable response compared to LQR. Figure 36 shows that the maximum cross-track errors are 1.352 m for MPC and 1.771 m for LQR. Moreover, the average cross-track errors are 0.3466 m for MPC and 0.4016 m for LQR.

TABLE III
RESULTS OF SURVEYED CASES

Type of trajectory		Maximum cross-track errors (m)		Average cross-track errors (m)	
Controller		MPC	LQR	MPC	LQR
Figure-eight	Simulation	0.0034	0.0225	0.0031	0.0178
	Experiment	0.469	0.5899	0.1696	0.1963
Square	Simulation	0.293	0.424	0.032	0.046
	Experiment	1.352	1.771	0.3466	0.4016

Table III shows that the trajectory tracking results of MPC are better than LQR. Both the maximum cross-track error and the average cross-track error in the two surveyed trajectories (simulation and experiment) indicate that MPC performs better in trajectory tracking compared to the LQR controller. Additionally, the steering angle response in the experimental results demonstrates that the steering angle output from MPC is more stable than LQR, with a lower steering frequency. Therefore, MPC shows better performance than LQR in two criteria: smaller cross-track errors and less steering angle oscillation.

Although there are differences between experimental and simulation results, MPC still proves to be more effective than LQR. Additionally, the computational complexity and workload of MPC need careful consideration. In this paper, the scope of investigation is limited to the kinematic vehicle model, simplifying computations and state variables for potential real-time integration into the system. Developing MPC for dynamic vehicle model with its computational complexity could lead to extensive computational demands, potentially making real-time implementation impractical. Therefore, considering LQR development as an alternative option with possibly less efficiency but significantly reduced computational workload could be worthwhile.

V. Conclusion

In this paper, we compared Model Predictive Control (MPC) and Linear Quadratic Regulator (LQR) in real-time path-tracking for autonomous vehicles, utilizing a custom-built four-wheel model. Evaluating criteria such as tracking accuracy, steering smoothness, workload, and real-time integration feasibility using a four-wheel model, MPC demonstrated better accuracy and smoother steering. While LQR offers simplicity and lower computational load. Understanding these strengths and limitations aids in selecting the optimal control strategy based on specific operational needs.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Tu Le Cong, Hung Nguyen Nhat and Hien Nguyen Vo Hong My developed the presented method, conducted the experiment, analyzed the result and wrote the manuscript. Hao Nguyen Vinh provided the original idea, supervised the project and gave critical feedback. All authors had approved the final version.

Acknowledgment

We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM and my supervisor Hao Nguyen Vinh for this study.

References

[1] E. Kassens-Noor, M. Cai, Z. Kotval-Karamchandani, and T. Decaminada, “Autonomous vehicles and mobility for people with special needs,” Transportation Research Part A: Policy and Practice, vol. 150, pp. 385–397, Aug. 2021, doi: https://doi.org/10.1016/j.tra.2021.06.014.

[2] T. Sever and G. Contissa, “Automated driving regulations – where are we now?,” Transportation Research Interdisciplinary Perspectives, vol. 24, pp. 101033–101033, Mar. 2024, doi: https://doi.org/10.1016/j.trip.2024.101033.

[3] A. Ziebinski, R. Cupek, D. Grzechca, and L. Chruszczyk, “Review of advanced driver assistance systems (ADAS),” ResearchGate,

vol. 1906, no. 1, Nov. 2017, doi: https://doi.org/10.1063/1.5012394.
[4] S. Yaakub and M. H. Alsibai, "A Review on Autonomous Driving Systems," International Journal of Engineering Technology and Sciences, vol. 5, no. 1, pp. 1–16, Jun. 2018, doi: https://doi.org/10.15282/ijets.v5i1.2800.
[5] M. Reda, A. Onsy, A. Y. Haikal , and A. Ghanbari, "Path planning algorithms in the autonomous driving system: A comprehensive review," Robotics and Autonomous Systems, vol. 174, Apr. 2024, doi: https://doi.org/10.1016/j.robot.2024.104630.
[6] M. Sagarkar, S. Damodar More, A. Singh Sant, P. Jana, and B. Pal, "Perception and Planning in Autonomous Car," Indian Institute of Technology Kharagpur, Dec. 2020, doi: https://doi.org/10.13140/RG.2.2.31312.38401.
[7] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," IEEE Transactions on Intelligent Vehicles, vol. 1, no. 1, pp. 33–55, Mar. 2016, doi: https://doi.org/10.1109/tiv.2016.2578706.
[8] C. D. Harper, C. T. Hendrickson, S. Mangones, and C. Samaras, "Estimating potential increases in travel with autonomous vehicles for the non-driving, elderly and people with travel-restrictive medical conditions," Transportation Research Part C: Emerging Technologies, vol. 72, pp. 1–9, Nov. 2016, doi: https://doi.org/10.1016/j.trc.2016.09.003.
[9] J. Cao, C. Song, S. Peng, S. Song, X. Zhang, and F. Xiao, "Trajectory Tracking Control Algorithm for Autonomous Vehicle Considering Cornering Characteristics," IEEE Access, vol. 8, pp. 59470–59484, Jan. 2020, doi: https://doi.org/10.1109/access.2020.2982963.
[10] S. Thrun et al., "Stanley: The robot that won the DARPA Grand Challenge," Journal of Field Robotics, vol. 23, no. 9, pp. 661–692, 2006, doi: https://doi.org/10.1002/rob.20147.
[11] J. Morales, J. L. Martínez, M. A. Martínez, and A. Mandow, "Pure-Pursuit Reactive Path Tracking for Nonholonomic Mobile Robots with a 2D Laser Scanner," EURASIP Journal on Advances in Signal Processing, vol. 2009, no. 1, Mar. 2009, doi: https://doi.org/10.1155/2009/935237.
[12] C. G´amez Serna, A. Lombard, Y. Ruichek, and A. Abbas-Turki, "GPS-based curve estimation for an adaptivepure pursuit algorithm," Advances in Computational Intelligence, vol. 10061, pp. 497–511, Aug. 2017, doi: https://doi.org/10.1007/978-3-319-62434-1_40.
[13] Alfredo Ollero Ojeda, A. García-Cerezo, and J. L. Martínez, "Fuzzy supervisory path tracking of mobile reports," Control Engineering Practice, vol. 2, no. 2, pp. 313–319, Apr. 1994, doi: https://doi.org/10.1016/0967-0661(94)90213-5.
[14] M. Rokonuzzaman, N. Mohajer, S. Nahavandi, and S. Mohamed, "Review and performance evaluation of path tracking controllers of autonomous vehicles," IET Intelligent Transport Systems, vol. 15, no. 5, pp. 646–670, Mar. 2021, doi: https://doi.org/10.1049/itr2.12051.
[15] G. W. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing," Proceedings of the ... American Control Conference, Jul. 2007, doi: https://doi.org/10.1109/acc.2007.4282788.
[16] H. Wang, B. Liu, X. Ping, and Q. An, "Path Tracking Control for Autonomous Vehicles Based on an Improved MPC," IEEE Access, vol. 7, pp. 161064–161073, Oct. 2019, doi: https://doi.org/10.1109/ACCESS.2019.2944894.
[17] T. M. Vu, R. Moezzi, J. Cyrus, and J. Hlava, "Model Predictive Control for Autonomous Driving Vehicles," Electronics, vol. 10, no. 21, p. 2593, Oct. 2021, doi: https://doi.org/10.3390/electronics10212593.
[18] G. V. Raffo, G. K. Gomes, J. E. Normey-Rico, C. R. Kelber, and L. B. Becker, "A Predictive Controller for Autonomous Vehicle Path Tracking," IEEE Transactions on Intelligent Transportation Systems, vol. 10, no. 1, pp. 92–102, Mar. 2009, doi: https://doi.org/10.1109/TITS.2008.2011697.
[19] S. Chen and H. Chen, "MPC-based path tracking with PID speed control for autonomous vehicles," IOP Conference Series: Materials Science and Engineering, vol. 892, p. 012034, Aug. 2020, doi: https://doi.org/10.1088/1757-899x/892/1/012034.
[20] R. Eisele, "Ackerman Steering Introduction," raw.org, Nov. 28, 2017. https://raw.org/book/kinematics/ackerman-steering/ (accessed Jul. 03, 2024).
[21] M. Nasir, Mohd Zubir Amir, Khisbullah Hudha, Mohd Azman Abdullah, and Muhammad Zahir Hassan, "MODELING AND VALIDATION OF SIX-BAR RACK AND PINIONSTEERING LINKAGE SYSTEM," 3rd International Conference on Engineering and ICT (ICEI2012) Melaka, Malaysia, Apr. 2012.
[22] "Kinematic Bicycle Model — Algorithms for Automated Driving," thomasfermi.github.io. https://thomasfermi.github.io/Algorithms-for-Automated-Driving/Control/BicycleModel.html
[23] S. Xu and H. Peng, "Design, Analysis, and Experiments of Preview Path Tracking Control for Autonomous Vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 1, pp. 48–58, Jan. 2020, doi: https://doi.org/10.1109/tits.2019.2892926.

Tu Le Cong received B.S. degree in control engineering & automation from Ho Chi Minh City University of Technology, Vietnam, in 2024. His research interest is control algorithm in embedded systems.

Hung Nguyen Nhat received B.S. degree in control engineering & automation from Ho Chi Minh City University of Technology, Vietnam, in 2024. His research interest is firmware and hardware design in embedded systems.

Hien Nguyen Vo Hong My received B.S. degree in control engineering & automation from Ho Chi Minh City University of Technology, Vietnam, in 2024. Her research interest is firmware and embedded systems.

Hao Nguyen Vinh is presently a lecturer of electrical and electronics engineering at Ho Chi Minh City University of Technology, Vietnam. He received B.S. and M.S. degrees in electrical and electronics engineering from Ho Chi Minh City University of Technology, Vietnam, in 2001 and 2003, respectively. He received Ph.D. degree in Electrical Engineering from the University of Ulsan, Korea, in 2009.
His research interests are INS/GPS positioning system, autonomous robot and adaptive control system.