Trajectory Tracking Control of an Autonomous Vehicle using Model Predictive Control and PID Controller

James Clark School of Engineering
Nazrin Gurbanova
UID: 120426469
Rohith Vikram Saravanan
UID: 119474198

(2023)

Table of Contents

I. Abstract *3*

II. Introduction *3*

III. Mathematical Modeling of Vehicle *5*

A. Bicycle Model Kinematics **5**

B. Bicycle Model Dynamics **8**

IV. Design of the controllers *10*

A. PID Design **10**

B. MPC design **11**

V. Simulation Results and Analysis *14*

A. Results of the PID controller **14**

B. Results of the MPC controller **17**

VI. Conclusion *21*

VII. References *22*

I. Abstract

Over the years, there has been a noticeable increase in vehicle accidents, including car crashes and traffic congestion, primarily attributed to the growing volume of traffic. Recent advancements in autonomous driving technology offer a promising solution to mitigate both traffic congestion and vehicle accidents. Autonomous driving vehicles possess the capability to plan a path and determine their trajectory based on various environmental factors. These vehicles utilize a controller to manage and guide the vehicle along the planned trajectory. This paper focuses on comparing the trajectory-tracking abilities of two controllers: a Proportional-Integral-Derivative (PID) controller and a Model Predictive Controller (MPC). Both controllers are employed for trajectory tracking along a unit circular path, and their performance is compared. The simulation results, including position graphs and error graphs, of both controllers are analyzed and compared.

II. Introduction

An autonomous vehicle can navigate its surroundings with precision, ensuring safe and collision-free driving with partial or no human intervention. The vehicle is usually equipped with sensors, and a control system, and uses mapping and localization, artificial intelligence, and machine learning techniques to navigate in the environment. Autonomous vehicles have several advantages. Firstly, they improve the safety of driving as autonomous driving reduces possible errors that can be made by human drivers. Even if there are a large number of vehicles present, they help prevent traffic congestion. Autonomous vehicles can drive for people unable to drive. They also improve the fuel efficiency of the vehicle by using optimized driving patterns.

Over the years, the increasing number of vehicles has led to an increase in road accidents, traffic congestion, and pollution in the environment. More than 90 percent of the accidents occurring on the road are due to human errors. To avoid the mishaps caused by the increasing number of vehicles, there has been continuous research in the field of autonomous driving cars [1]. The main research focus for these vehicles is to build a strong control framework that is able to withstand adverse external conditions and, at the same time, a flexible framework that gives us collision-free trajectory guidance, taking care of the road boundaries, other objects, and vehicles, as well as the traffic regulations [1].

Most of the research in path and trajectory planning and guidance is being conducted in the field of robotics. Various algorithms are proposed to solve these trajectory planning problems. These algorithms include sampling-based algorithms, which determine the fastest trajectory by randomly sampling the waypoints for the trajectory; on the other hand, there are algorithms like A*, where the trajectory is planned by finding the heuristic distance between the start and goal points [2]. A path is a set of instructions that includes the directions that need to be followed for

the vehicle, which is placed at the initial location, to reach the destination location. A trajectory is a path that additionally considers kinematic variables such as the velocity acceleration components as well as the vehicle dynamics to reach the destination [2].

Generally, there are three approaches that are followed for the planning techniques. The first approach is a sampling-based approach, where the state spaces representing the various states of the system are randomly sampled and used as waypoints to determine the trajectory. This approach includes the probabilistic roadmap (PRM) method and the randomly exploring random tree (RRT) method. These methods are usually used for high-dimensional workspaces where exhaustive searching is not recommended. The second approach is the decoupling method. For complex environments, this method separates the global path planning from the local obstacle avoidance, hence using two planners to form the trajectory. The global planner considers the whole environment with the start and goal points and considers larger obstacles and terrain while planning the trajectory. Some of the global planning algorithms include Dijkstra and the A* algorithm. The local planner considers the immediate obstacles and makes decisions based on them to plan the trajectory. The local planning algorithms include potential field and reaction control algorithms [4]. The third approach to planning is to provide formulation by considering the initial and boundary conditions along with the mathematical restrictions that need to be followed to find the trajectory. Following the mathematical restrictions and conditions, we get the desired solution for planning the trajectory, making the trajectory more accurate [3].

One part of the planning strategies is the Model Predictive Control which is a widely used simple and efficient method where dynamic changes can be adapted with the help of regular updates [4]. The main purpose of this optimization technique is that it considers the constraints [2].

A wide range of studies have been carried out about the implementation of Model Predictive Control in the path tracking of autonomous vehicles. Firstly, in the low-level trajectory tracking MPC can be used instead of traditional controllers [5]. One case can be the study where a model with complex and non-linear vehicle dynamics is changed into a model that has linear and time-varying behavior, the MPC algorithm is utilized to follow the trajectory of lane changes on the road [6]. These models require vehicle data such as tire information, inertia characteristics, and vehicle mass. Within MPC, linearization can lead to minimized execution time. However, in some dynamic situations for vehicles requiring lateral and control with longer prediction horizons, MPC may fail to give the desired results [7]. The last study worth mentioning is the one where the researchers identified model mismatch as a disturbance that is measurable and caused because of changing conditions of the road and small-angle presumptions. An MPC controller was proposed which uses a differential evolution algorithm to address this issue [8].

The paper compares the behavior of two controllers in the path tracking of a vehicle model. The second section illustrates the bicycle model's kinematics and dynamics presenting derivations of the equations that are to be utilized in the simulation part. Subsequently, the third section shows

the methods for the simulation thus explaining the way two controllers, namely PID and MPC, work. Having designed the controllers, Section 4 presents the result obtained from the simulation. The last section which is the conclusion highlights what has been achieved from the study briefly, comparing the results done by our group and the paper; moreover, what difficulties we encountered while applying the techniques to get the results.

III. Mathematical Modeling of Vehicle

A. Bicycle Model Kinematics

We are using a kinematic bicycle model to compare the trajectory tracking control of the PID controller with the model predictive controller. A bicycle model that can take in the lateral, longitudinal, and yaw-angle motions of the vehicle is considered. The bicycle model is taken from the paper [1].

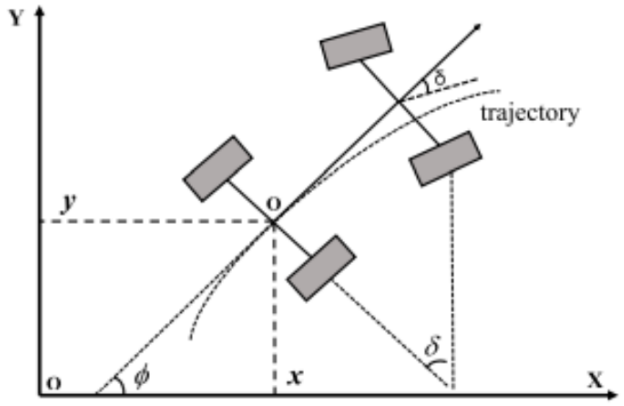


Fig. 1. 1 Vehicle Kinematic Model

The inertial coordinates of the vehicles are represented in Fig 1.1. The figure denotes the x and y coordinates representing the position of the center of the rear wheel axle of the vehicle. The figure illustrates the vehicle's orientation, denoted by ϕ, and the steering angle, represented by δ. A system with position coordinates represented as $\chi = [x\ y\ \phi]^T$ and control variables represented as $u = [v\ \delta]^T$ is defined. The vehicle dynamic model having three degrees of freedom is defined by the equations 1.1,1.2 and 1.3.

$$\dot{x} = v\cos\phi \qquad (1.1)$$

$$\dot{y} = v\sin\phi \qquad (1.2)$$

$$\dot{\phi} = \frac{v\tan\delta}{l} \qquad (1.3)$$

Where the l represents the longitudinal distance between the front and rear wheels. The differential equation for the above dynamic model is written as

$$\dot{\chi} = f_1(\chi, u) \qquad (1.4)$$

The reference trajectory, which is the trajectory we want the robot to follow, the velocity component equation for reference trajectory is represented by equation 1.5.

$$\dot{\chi_r} = f_1(\chi_r, u_r) \qquad (1.5)$$

Where χ_r represents the position coordinates of the reference trajectory and $u_r$represents the reference control output.

Taylor series expansion:

$$f(x) = f(a)\frac{f\prime(a)}{1!}(x-a) + \frac{f\prime\prime(a)}{2!}(x-a)^2 + \frac{f\prime\prime\prime(a)}{3!}(x-a)^3 + \cdots\cdots \quad (1.6)$$

By Taylor series expansion $f(x)$ can be written as:

$$\sum_{n=0}^{\infty}\frac{f^n(a)}{n!}(x-a)^n \quad (1.7)$$

Where, $f^{(n)}(a)$ - n^{th} derivative of f; $\quad n!$ − factorial of n

Expanding equation 1.4 using the Taylor series expansion at the desired reference point:

$$\dot{\chi} = f_1(\chi_r, u_r) + \frac{\partial(\chi\ ,u\ )}{\partial\chi}(\chi-\chi_r) + \frac{\partial(\chi,u)}{\partial u}(u-u_r) \qquad (1.8)$$

Combining the equation 1.5 and 1.8 we get:

$$\dot{\tilde{\chi}} = \begin{bmatrix}\dot{x}-\dot{x}_r\\ \dot{y}-\dot{y}_r\\ \dot{\phi}-\dot{\phi}_r\end{bmatrix} + \begin{bmatrix}0 & 0 & -v\sin\phi_r\\ 0 & 0 & v\cos\phi_r\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}x-x_r\\ y-y_r\\ \phi-\phi_r\end{bmatrix} + \begin{bmatrix}\cos\phi_r & 0\\ \sin\phi_r & 0\\ \frac{\tan\phi_r}{l} & \frac{v_r}{l\cos^2\delta}\end{bmatrix}\begin{bmatrix}v-v_r\\ \delta-\delta_r\end{bmatrix} \qquad (1.9)$$

In the matrix:

$$\begin{bmatrix}0 & 0 & -v\sin\phi_r\\ 0 & 0 & v\cos\phi_r\\ 0 & 0 & 0\end{bmatrix}$$

The first column of the matrix represents the partial derivative of $\dot{x_r}, \dot{y_r}, \dot{\phi_r}$ with respect to x_r. The first column of the matrix represents the partial derivative of $\dot{x_r}, \dot{y_r}, \dot{\phi_r}$ with respect to y_rThe first column of the matrix represents the partial derivative of $\dot{x_r}, \dot{y_r}, \dot{\phi_r}$ with respect to ϕ_r

In the matrix:

$$\begin{bmatrix} cos\phi_r & 0 \\ sin\phi_r & 0 \\ \frac{tan\phi_r}{l} & \frac{v_r}{lcos^2\delta} \end{bmatrix}$$

The first column of the matrix represents the partial derivative of $\dot{x_r}, \dot{y_r}, \dot{\phi_r}$ with respect to v_r. The first column of the matrix represents the partial derivative of $\dot{x_r}, \dot{y_r}, \dot{\phi_r}$ with respect to δ_r.

Here the v_r represents the reference velocity the vehicle needs to follow to move in the right direction of the trajectory. Here the δ_r represents the reference steering angle the vehicle needs to align to move in the right trajectory direction. The x_r, y_r, ϕ_r represents the x and y reference position and the reference orientation of the vehicle.

$$\begin{bmatrix} \dot{x} - \dot{x}_r \\ \dot{y} - \dot{y}_r \\ \dot{\phi} - \dot{\phi}_r \end{bmatrix} + \begin{bmatrix} 0 & 0 & -vsin\phi_r \\ 0 & 0 & vcos\phi_r \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \phi - \phi_r \end{bmatrix}$$

Adding the above vector with the matrix we get:

$$\begin{bmatrix} x - x_r \\ y - y_r \\ \phi - \phi_r \end{bmatrix} \left[I + \begin{bmatrix} 0 & 0 & -vsin\phi_r \\ 0 & 0 & vcos\phi_r \\ 0 & 0 & 0 \end{bmatrix} \right] = \begin{bmatrix} 1 & 0 & -vsin\phi_r \\ 0 & 1 & vcos\phi_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \phi - \phi_r \end{bmatrix}$$

By solving the above equation for a time interval T, we get equation 1.10.

$$\bar{\chi}(k+1) = A_1(k,t)\bar{\chi}(k) + B_1(k,t)\bar{u}(k) \quad (1.10)$$

The expression $\bar{\chi}(k+1)$ represents the higher-order derivative of $\bar{\chi}(k)$. Equation 8 is in the form of $\chi = Ax + Bu$, which is a state-space equation. Here, the coefficient matrix A is denoted as $A_1(k,t)$, the coefficient matrix B is denoted as $B_1(k,t)$ the state variable is represented by $\bar{\chi}(k)$, and the input variable is represented by $\bar{u}(k)$. The respective values of these terms are represented as follows:

$$A_{1k,t} = \begin{bmatrix} 1 & 0 & -vsin\phi_r T \\ 0 & 1 & vcos\phi_r T \\ 0 & 0 & 1 \end{bmatrix}$$

$$B_{1k,t} = \begin{bmatrix} cos\phi_r T & 0 \\ sin\phi_r T & 0 \\ \frac{tan\phi_r T}{l} & \frac{v_r}{lcos^2\delta} \end{bmatrix}$$

$$\bar{\chi}(k) = \begin{bmatrix} x(k) - x_r(k) \\ y(k) - y_r(k) \\ \phi(k) - \phi_r(k) \end{bmatrix}.$$

$$\bar{u}(k) = \begin{bmatrix} v(k) - v_r(k) \\ \delta(k) - \delta_r(k) \end{bmatrix}$$

B. Bicycle Model Dynamics

The dynamic model of the vehicle is used in the simulation part of the trajectory tracking of the vehicle using MPC controller. To use the vehicle information, the state space equation for the dynamic equation needs to be obtained. That equation is utilized on MATLAB to create the plant model as is shown in the next section. To represent lateral vehicle dynamics, the two degrees of freedom are shown by the vehicle lateral position y and the vehicle yaw angle. The vehicle lateral position is measured along the lateral axis of the vehicle to the point O which is the center of rotation of the vehicle. What is more, the yaw angle is considered with respect to the X axis [9].

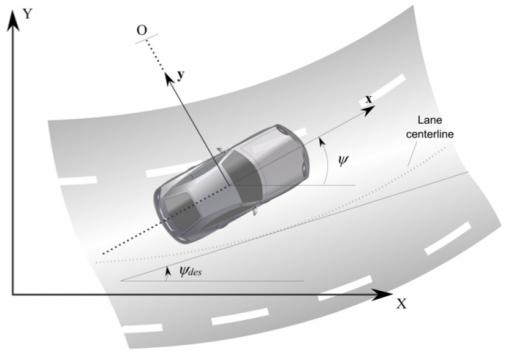


Fig. 2. 1 Lateral Vehicle Dynamics

Using Newton's 2nd law for the motion through the y-axis

$ma_y = F_{yf} + F_{yr}$

F_{yf} – the front wheel tire lateral force

F_{yr} – the rear wheel tire lateral force

Centripetal acceleration $V_x\dot{\psi}$ and the acceleration due to motion in y axis contribute to a_y, therefore equation can be written as following as well: $a_y = \ddot{y} + V_x\dot{\psi}$ (2.1)

After substitutions of the equations above, the lateral translational motion of the vehicle:

$$m(\ddot{y} + V_x\dot{\psi}) = F_{yf} + F_{yr} \quad (2.2)$$

Moment balance:

$$I_z\ddot{\psi} = l_f F_{yf} - l_r F_{yr} \quad (2.3)$$

l_f – distance from the front tire to center of the vehicle

l_r– distance from the rear tire to center of the vehicle

The slip angle for the front wheel

$$\alpha_f = \delta - \theta_{Vf} \quad (2.4)$$

θ_{Vf} – angle between velocity vector and longitudinal axis of the vehicle model
δ – steering angle

The slip angle for the rear wheel: $\alpha_r = -\theta_{Vr}$ (2.5)

Front wheels' lateral tire force: $F_{yf} = 2C_{\alpha f}(\delta - \theta_{Vf})$ (2.6)
Note: Multiplication by 2 is due to possessing two front wheels.
Rear wheels' lateral tire force

$$F_{yr} = 2C_{\alpha r}(-\theta_{Vr}) \quad (2.7)$$

δ – steering angle
$C_{\alpha f}$ / $C_{\alpha r}$–cornering stiffness
θ_{Vf}/ θ_{Vr} – velocity angle.
For the calculation of the θ_{Vf} and θ_{Vr}, following equations can be used:

$$tan\left(\theta_{Vf}\right) = \frac{V_y + l_f\dot{\psi}}{V_x} \quad (2.8)$$

$$tan\left(\theta_{Vr}\right) = \frac{V_y - l_r\dot{\psi}}{V_x} \quad (2.9)$$

Considering $V_y = \dot{y}$, with the small angle approximations:

$$\theta_{Vf} = \frac{\dot{y} + l_f\dot{\psi}}{V_x} \quad (2.10)$$

$$\theta_{Vr} = \frac{\dot{y} - l_r\dot{\psi}}{V_x} \quad (2.11)$$

The state space model can be given as follows by plugging 2.4, 2.5, 2.10, 2.11 into 2.2 and 2.3:

$$\frac{d}{dt}\begin{Bmatrix} y \\ \dot{y} \\ \psi \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{\alpha f} + 2C_{\alpha r}}{mV_x} & 0 & -V_x - \frac{2C_{\alpha f}\ell_f - 2C_{\alpha r}\ell_r}{mV_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2\ell_f C_{\alpha f} - 2\ell_r C_{\alpha r}}{I_z V_x} & 0 & -\frac{2\ell_f^2 C_{\alpha f} + 2\ell_r^2 C_{\alpha r}}{I_z V_x} \end{bmatrix} + \begin{Bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2\ell_f C_{\alpha f}}{I_z} \end{Bmatrix} \delta$$

IV. Design of the controllers

A. PID Design

A PID controller is designed for the trajectory control of the mobile robot. A reference trajectory path is created, and the vehicle is controlled using the PID controller to move along that reference trajectory. The reference trajectory is a unit circle with a center at (*0,0*). The trajectory is generated using a ramp input, taking the sine of the ramp input for the x reference values and the cosine of the ramp input for the y reference values. Each input value is represented as (x_r, y_r), where (x_r, y_r) represents a reference target position. The position error of the vehicle is then calculated by subtracting the actual position coordinates from the reference coordinates, as given in Equation 3.1.

$$e = \sqrt{(x - x_r)^2 + (y - y_r)^2} \quad (3.1)$$

The actual positions of the vehicle are obtained from the output of the vehicle kinematic model. The position error is mathematically calculated and provided as input to the PID controller. The PID controller calculates the velocity value to proportionally manage the error. The calculated velocity from the PID controller is then used as the input velocity for the vehicle kinematic model, computed using Equation 3.2.

$$v = Kpe + Ki\int edt + Kd\,\frac{de}{dt} \quad (3.2)$$

The vehicle's relative angle, representing the angle the vehicle must follow to reach the reference trajectory, is calculated by subtracting the reference position angle from the actual position angle. This relative angle is determined by Equation 3.3.

$$\phi_r = tan^{-1}\left[\frac{y_r - y}{x_r - x}\right] \quad (3.3)$$

Additionally, the vehicle's relative steering angle, indicating the steering the vehicle must make to follow the reference path, is calculated using Equation 3.4.

$$\delta_r = K_h(\phi_r - \phi) \quad (3.4)$$

The vehicle kinematic model takes the longitudinal velocity (v) and steering angle (δ) as control parameters and produces the position parameters x, y, ϕ as outputs. The control block diagram of the model is illustrated in Figure 1. The PID controller is tuned using suitable gain values $Kp, Ki,$ and Kd as provided in Table 1, sourced from the paper.

For fine-tuning the PID controller's gain parameters, the Ziegler-Nichols or Cohen-Coon approaches were employed. Although these approaches are primarily intended for linearized models, we utilize them for PID controller tuning. In cases where the PID controller does not adequately compensate for system non-linearities, a more sophisticated approach is required. In this context, we have applied the model predictive control approach.

B. MPC design

The lateral controller is designed based on the model predictive control theory. The model predictive control can predict the output of the system in the future according to the prediction model, the current state quantity of the system and the future control quantity and can solve problems with various constraints in a rolling manner. The MPC control process mainly includes the establishment of the prediction model, the deduction of the prediction equation, the design of the objective function, the addition of constraints, and the rolling optimization solution [10].

The first-order quotient difference is used to obtain the discrete state space equation:

$$\xi(k+1) = A(k)\xi(k) + B(k)u(k) \quad (3.5)$$

$$A(k) = I + TA(t); B(k) = TB(t);$$

T – sampling period, $t -$ sampling time, N_p – prediction time domain, $k = t, t+1, \dots, t+N_p$, I – unit matrix

$$\{\tilde{\xi}(k+1|t) = \tilde{A}_{k,t}\tilde{\xi}(k|t) + \tilde{B}_{k,t}\Delta u(k|t), \quad (3.6)$$
$$\tilde{\lambda}(k|t) = \tilde{C}_{k.t}\tilde{\xi}(k|t) \quad (3.7)$$

$\tilde{\xi}(k)$ is the state matrix composed of the state quantity at time k and the control quantity at time k-1, $\tilde{\xi}(k|t) = \begin{bmatrix} \xi(k|t) \\ u(k-1|t) \end{bmatrix}$

$\Delta u(k|t)$ – the control increment at time k of the system

$$\Delta u(k|t) = u(k|t) - u(k-1|t). \qquad (3.8)$$

$\tilde{A}_{k,t} = \begin{bmatrix} A_{k,t} & B_{k,t} & 0 & I \end{bmatrix}$, $\tilde{B}_{k,t} = \begin{bmatrix} B_{k,t} & I \end{bmatrix}$

$\tilde{\eta}(k|t)$ – the output of the system at time k

Assume that the prediction time domain of the MPC controller is N_p, the control step size is N_c, and $N_c \leq N_p$. The predicted output:

$$Y(k|t) = \Psi_k \tilde{\xi}(k|t) + \Theta_k \Delta U(k|t) \qquad (3.9)$$

$Y(k|t)$ – the output matrix in the prediction time domain

$$Y(k) = \begin{bmatrix} \eta(k+1|t) \\ \eta(k+2|t) \\ \dots \\ \eta(k+N_p|t) \end{bmatrix}$$

$\Delta U(k)$ – the control increment in the control time domain N_c

$$\Delta U(k) = \begin{bmatrix} \Delta u(k|t) \\ \Delta u(k+1|t) \\ \dots \\ \Delta u(k+N_c-1|t) \end{bmatrix}$$

$$\Psi_k = \begin{bmatrix} \bar{C}_{k,t}\bar{A}_{k,t} \\ \bar{C}_{k,t}\bar{A}^2{}_{k,t} \\ \dots \\ \bar{C}_{k,t}\bar{A}^{N_p}{}_{k,t} \end{bmatrix}$$

$$\Theta_t = \begin{bmatrix} \bar{C}_{k,t}\bar{A} & 0 & \dots & 0 \\ \bar{C}_{k,t}\bar{A}_{k,t}\bar{B}_{k,t} & \bar{C}_{k,t}\bar{B}_{k,t} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \bar{C}_{k,t}\bar{A}^{N_p-1}{}_{k,t}\,\bar{B}_{k,t} & \bar{C}_{k,t}\bar{A}^{N_p-2}{}_{k,t}\,\bar{B}_{k,t} & \dots & 0 \end{bmatrix}$$

As for the longitudinal and lateral motion control of the subject vehicle in the cooperative control system, the cost function:

$$J(k) = \sum_{i=1}^{N_p} \left|\left|\eta(k+i|t) - \eta_{ref}(k+i|t)\right|\right|_Q^2 + \sum_{i=1}^{N_c-1} \left|\left|\Delta U(k+i|t|\right|\right|_R^2 + \rho\varepsilon^2 \quad (3.10)$$

$\sum_{i=1}^{N_p} \left|\left|\eta(k+i|t) - \eta_{ref}(k+i|t)\right|\right|_Q^2$ → the objective of minimizing the error between the predictive output points and the desired trajectory points

$\sum_{i=1}^{N_c-1} \left|\left|\Delta U(k+i|t|\right|\right|_R^2 + \rho\varepsilon^2$ → reflects the consideration given to the size of U(t) reflects the consideration given to the size of U(t)

$\eta(k+i|t)$ – the actual output of the system

$\eta_{ref}(k+i|t)$ – the reference output of the system

Q, R – the weight coefficient matrix

ρ – the relaxation factor weight coefficient

ε – the relaxation factor

To find the optimal U, we minimize J which can be expressed as

$$J(\xi(t), u(k-1), \Delta U) = \Delta U(k)^T H_t \Delta U(k) + G_t \Delta U(k) \quad (3.11)$$

$$H_t = \Theta^T Q \Theta + R$$

$$G_t = 2E_t^T Q \Theta_t$$

$$\eta_{ref}(t) = \left[x_p(i), y_p(i), \phi_p(i)\right]^T$$

The input and output constraints for the designing process:

Input constraints

$$-1 \leq x \leq 1;$$

$$-1 \leq y \leq 1;$$

$$-3.14 \leq \phi \leq 3.14;$$

$$-3.14 \leq \delta \leq 3.14$$

Output constraint

$$0 \leq v \leq 10$$

Prediction horizon=10 Control horizon = 2 Sampling time = 0.01s

V. Simulation Results and Analysis

A. Results of the PID controller

The trajectory tracking performance of the two controllers is studied using MATLAB Simulink. The controller gain values that were used for the tuning of the PID controller to make the vehicle follow the reference path are given in Table 1.

TABLE 1

CONTROLLER GAINS

Controller gains	Values
Kp	5.5
Ki	3
Kd	0.5
K	10

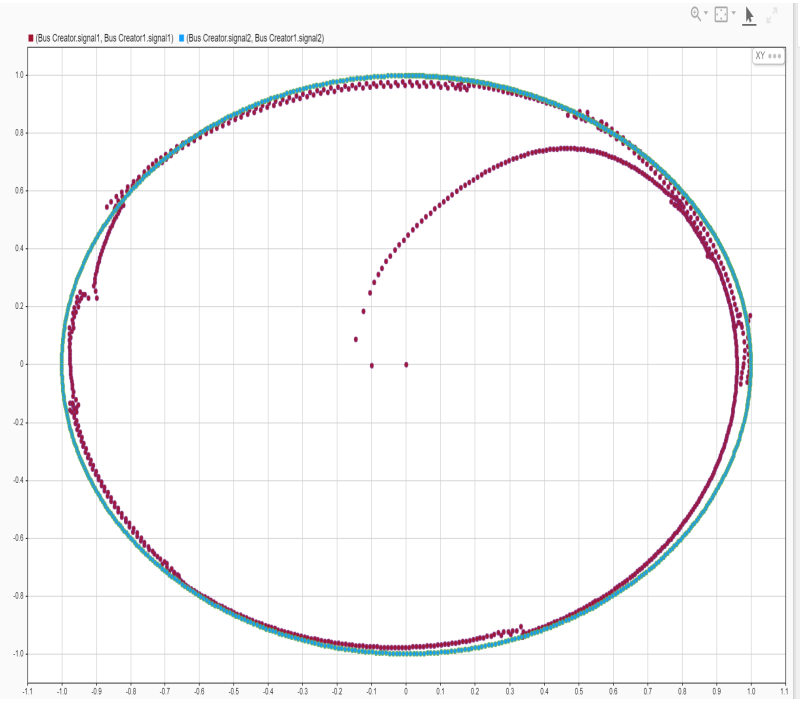


Fig 5. 1 Trajectory tracking using PID controller

The figure above represents the trajectory tracking of the PID controller designed to guide the vehicle along a reference path. In our simulation model, the reference trajectory is a unit circle centered at (0,0). The vehicle, initially positioned at (0,0) and under the control of the PID controller, successfully tracks the reference path, with only minor deviations observed at the left and right ends of the trajectory.

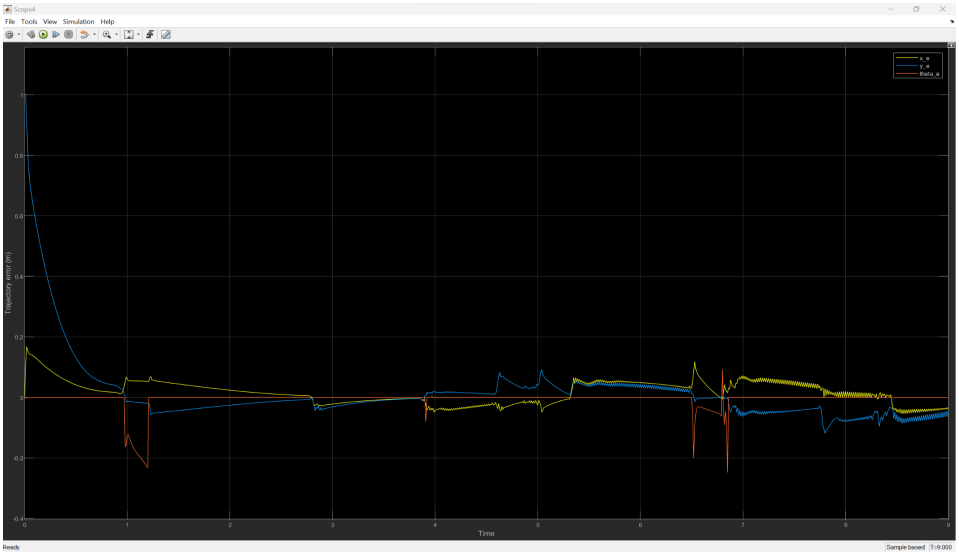


Fig 5. 2 x, y, phi error values

The figure above represents the reference trajectory tracking error in the x, y, and phi positions of the vehicle over time.

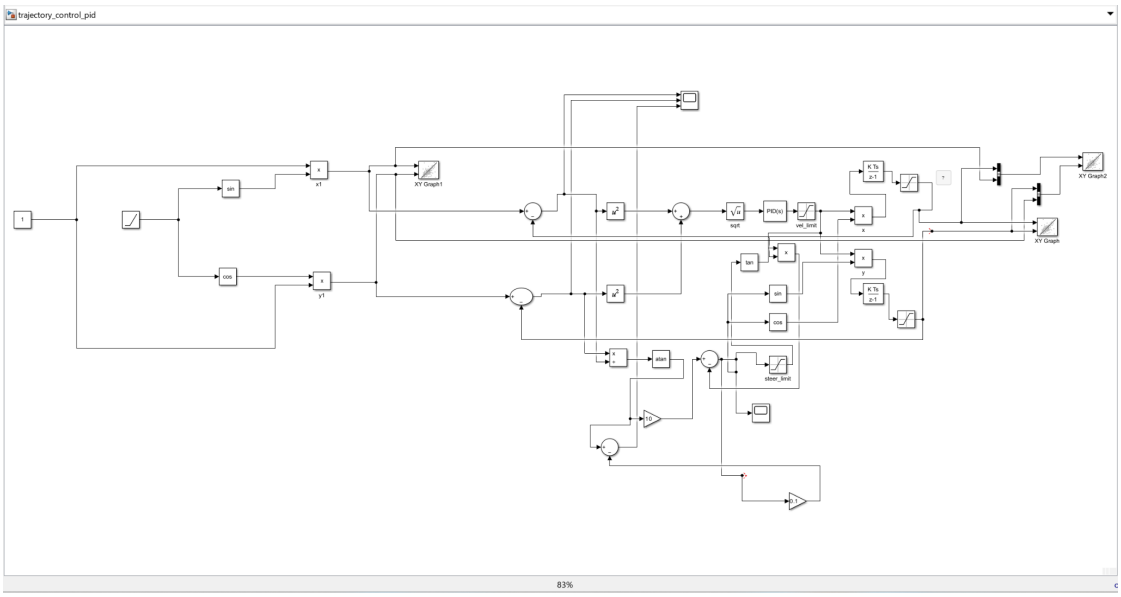


Fig 5. 3 Block diagram for the path tracking using PID control

The figure above represents the control block diagram used to simulate the results of trajectory tracking using a PID controller. The reference trajectory is generated using the sine and cosine values of the ramp input, and the radius of the reference trajectory is obtained from a constant block. The x and y coordinates of the trajectory are calculated by taking the cosine and sine of the ramp input times the radius. The actual x and y coordinates obtained from the kinematic bicycle model are subtracted from the reference x and y values using a sum block. The x and y differences are squared using separate square blocks. Then, the squared differences are added using a sum block. To calculate the position error of the vehicle, the square root of the output from the sum block is obtained using a square root block. The position error is fed into the PID controller, which produces the x and y velocity components for the vehicle, enabling the vehicle model to track the reference trajectory. The PID controller is tuned using the controller gain values provided in the paper. The velocity output from the PID controller is then subjected to a saturation block to ensure that the velocity remains within the preferred limit. The linear velocity produced by the PID controller serves as a control input for the vehicle kinematic model.

To calculate the reference orientation of the robot, the difference in y values is divided by the difference in x values using a divide block. Subsequently, the arctangent (atan) block is employed to obtain the arctangent of the divided value, representing the reference orientation of the robot. The calculated reference orientation is then increased tenfold using a gain block. The actual orientation value of the vehicle kinematic model is subtracted from the increased reference orientation to derive the reference steering angle for the vehicle model. Limits are applied to the reference steering angle using a saturation block. This reference steering angle is then employed as a control input for the vehicle kinematic model.

The control inputs provided to the vehicle model are used to generate velocity component values that direct the vehicle along the reference trajectory. The x velocity component of the vehicle is calculated by multiplying the linear velocity generated by the PID controller by the cosine of the reference orientation. Similarly, the y velocity component of the vehicle is calculated by multiplying the linear velocity generated by the PID controller by the sine of the reference orientation. The orientation component of the vehicle is determined by multiplying the linear velocity generated by the PID controller by the tangent of the reference steering angle divided by the length of the vehicle model.

These x, y, and orientation components are then subtracted from their corresponding reference values to calculate the position error value and the reference steering angle value, respectively.

B. Results of the MPC controller

Model predictive control has been used in the second stage of the study. As is given in the reference paper [11], Fig 5.4 has been created in Simulink Fig 5.5.

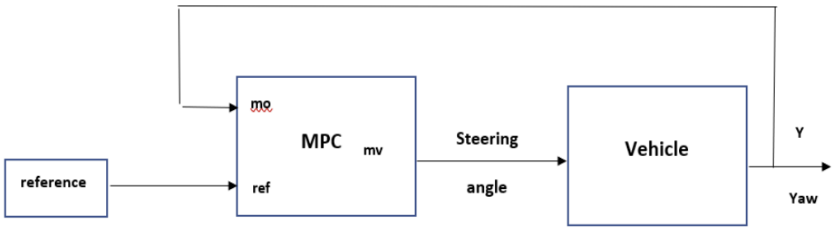


Fig 5. 4 Block diagram of Model Predictive Control

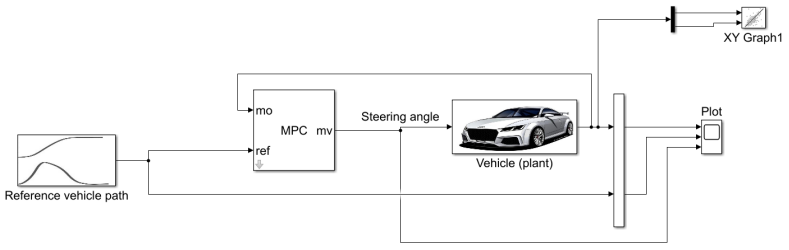


Fig 5. 5 Block diagram of Model Predictive Control (Simulink)

The MPC controller block requires the plant model which is the vehicle model that has been defined in the previous section and the reference vehicle path to follow the reference trajectory for the model. Reference block is created using waypoints in the Driving Scenario Design app of MATLAB with the help of which we were able to the get reference yaw angles for the simulation. After exporting, the data is held in the param file and utilized for the MPC controller's reference input. The state space block for the vehicle is used as described in the dynamics of the vehicle section of this report.

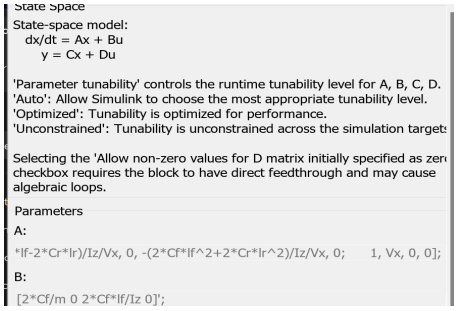
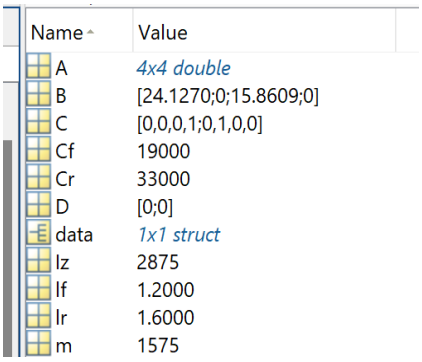


Fig 5. 6 Vehicle dynamics equation in the simulink block

The vehicle parameters have been assigned as follows:

TABLE II
VEHICLE PARAMETERS

Total mass of the vehicle(m)	1575kg
Yaw moment of inertia of vehicle(I_z)	$2875mNs^2$
Longitudinal distance from center of gravity of front tires(I_f)	1.2m
Longitudinal distance from center of gravity of rear tires(I_r)	1.6m
Cornering stiffness of front tire (C_{af})	19000N/rad
Cornering stiffness of rear tire(C_{ar})	33000N/rad



As regards the tuning of the MPC, the values given are used to track the behavior of the vehicle in following the path given Fig 5.7.

Prediction horizon=10 Control horizon = 2 Sampling time = 0.01s

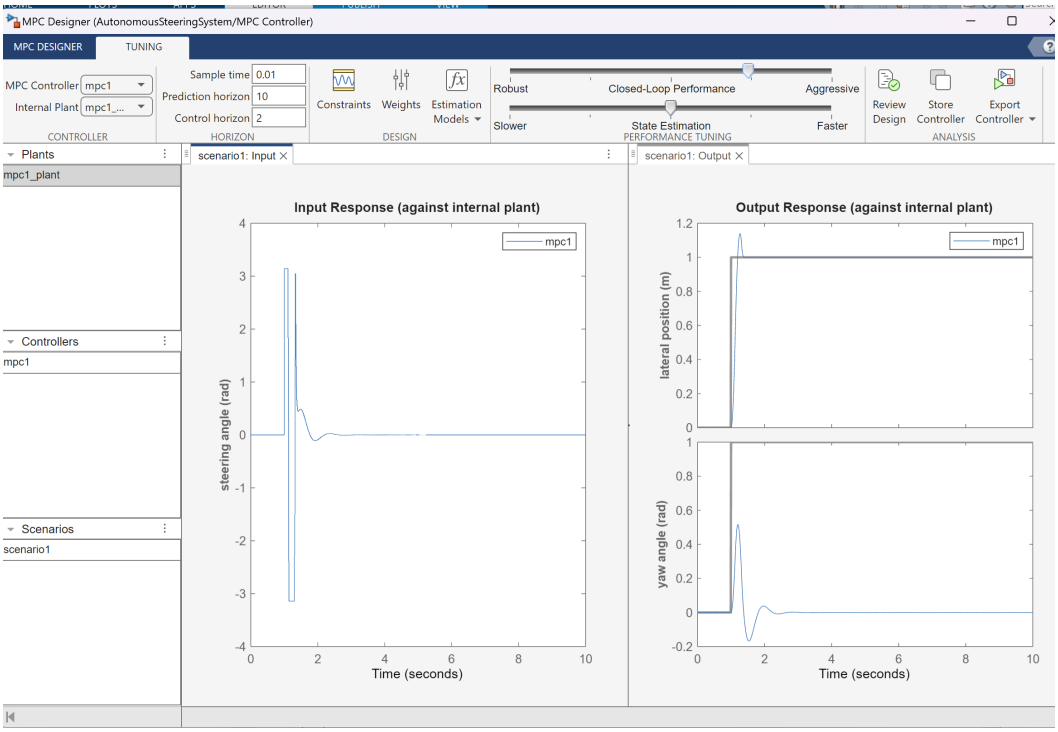


Fig 5. 7 Tuning of Model Predictive Control

The prediction horizon should be long to include system dynamics such as delays and lags adequately, while to avoid late responses it also ought to be short. It is a common practice to choose it as the minimum settling time and time horizon over which the disturbances might be predicted. Moving onto control horizon, it needs to be as long as the time to prevent frequent input switching and enable steady-state operation of the system.

The constraints for the input and output values are used to tune the MPC (Fig 5.8).

Constraints (mpc1)

Input and Output Constraints

Channel	Type	Min	Max	RateMin	RateMax
▼ Inputs					
u(1)	*MV*	-3.14159265…	3.141592653…	-3.14159265…	3.141592653…
▼ Outputs					
y(1)	*MO*	-3.14159265…	3.141592653…		
y(2)	*MO*	-10	10		

Fig 5. 8 Assigning constraints to MPC controller

In Fig 5.9 the lateral positions graph for the reference and controlled values have been plotted. As can be seen, MPC manages following the paths with satisfactory performance showing a small error between. At the end, the yaw angle given and obtained are compared as is illustrated in Fig 5.10.

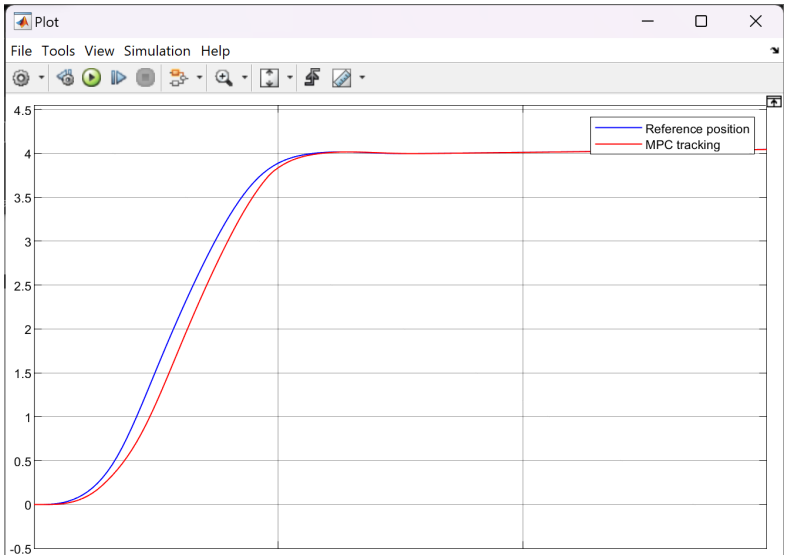


Fig 5. 9 Lateral positions of reference and MPC tracking

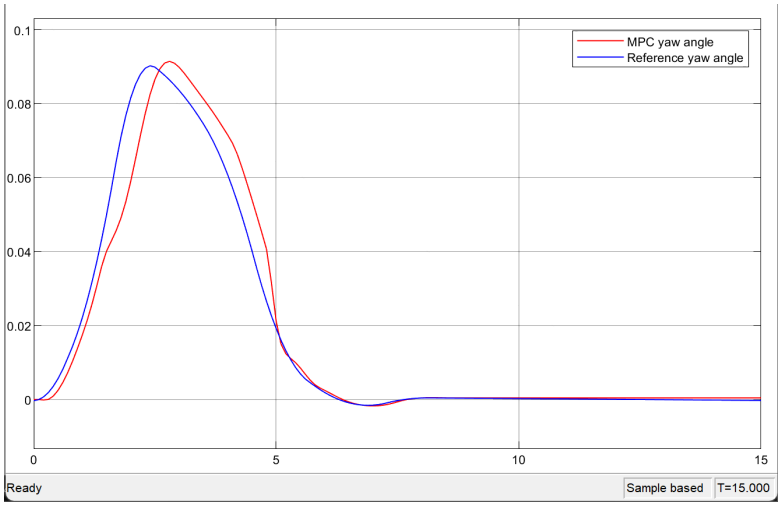


Fig 5. 10 Yaw angles of reference and MPC

VI. Conclusion

In conclusion, we have studied the trajectory tracking of an autonomous vehicle using PID and MPC controllers. For the PID controller, we utilized a unit circular reference trajectory, and for the MPC controller, a step input trajectory was employed. The trajectory tracking by the PID controller was satisfactory; the actual path closely matched the reference path with slight deviations towards the left and right ends of the trajectory. In contrast, the trajectory tracking with the MPC controller was more accurate, as the actual path closely aligned with the desired reference path. Moreover, the error in position coordinates for the PID controller was comparatively higher when compared to the yaw error of the MPC controller.

- We did not use a unit circle as a reference trajectory for the MPC controller because we were unable to figure out the proper computational equations for generating the circular coordinates from the output yaw angle.
- Thus, the MPC controller does a better job at trajectory tracking when compared to the PID controller. This is because the MPC controller considers various constraints and dynamic design parameters of the vehicle to predict the system's behavior, tune the controller, and generate an output yaw angle such that the vehicle accurately follows the reference path.

VII. References

[1] T. Weiskircher, Q. Wang and B. Ayalew,, ""Predictive Guidance and Control Framework for (Semi-)Autonomous Vehicles in Public Traffic," in IEEE Transactions on Control Systems Technology, vol. 25, no. 6, pp. 2034-2046, Nov. 2017, doi: 10.1109/TCST.2016.26421".

[2] D. N. Godbole, V. Hagenmeyer, R. Sengupta, and D. Swaroop, "Design of emergency manoeuvres for automated highway system: Obstacle avoidance problem".

[3] D. N. Godbole, V. Hagenmeyer, R. Sengupta, and D. Swaroop, "Design of emergency manoeuvres for automated highway system: Obstacle avoidance problem".

[4] I. Papadimitriou and M. Tomizuka, ""Fast lane changing computations using polynomials," in Proc. Amer. Control Conf., vol. 1. Denver, CO, USA, Jun. 2003, pp. 48–53".

[5] Y. G. J. K. H. a. F. B. A. Gray, "Robust predictive control for semi-autonomous vehicles with an uncertain driver model".

[6] S. J. Abele, ""Socio-Economic Impacts Of Intelligent Safety Systems (SEISS)," 2005.".

[7] G. Prokop, "Modeling human vehicle driving by model predictive online optimization".

[8] M. A. Forrest, "Autonomous Cars and Society.".

[9] R. Rajamani, ", Vehicle Dynamics and Control, Mechanical Engineering Series, 15 DOI 10.1007/978-1-4614-1433-9_2, © Rajesh Rajamani 2012".

[10] D. S. Pae, G. H. Kim, T. K. Kang, and M. T. Lim, "Path planning based on obstacle-dependent Gaussian model pre- dictive control for autonomous driving".

[11] Anagha Anil1 and V.R Jisha, "Trajectory Tracking Control of an Autonomous Vehicle using Model Predictive Control and PID Controller".

[12] Yunlong Bai ,1 Gang Li ,1 Hongyao Jin,1 and Ning Li2, "Research on Lateral and Longitudinal Coordinated Control of Distributed Driven Driverless Formula Racing Car under High-Speed Tracking Conditions".

[13] Yunlong Bai ,1 Gang Li ,1 Hongyao Jin,1 and Ning Li2, "Research on Lateral and Longitudinal Coordinated Control of Distributed Driven Driverless Formula Racing Car under High-Speed Tracking Conditions".