Non-linear Vehicle Model and Linear MPC Controller in Autoware

Sujeendra Ramesh
rames154@umn.edu

December 15, 2024

Contents

1 Abstract

This report presents the development and analysis of a non-linear vehicle model and a Model Predictive Control MPC (Model Predictive Control) framework implemented within the **Autoware** autonomous driving platform. The study focuses on dynamic modeling, control system design, and simulation analysis to achieve precise lateral control in autonomous vehicles.

The vehicle dynamics are represented by two complementary models. A non-linear vehicle model, incorporating tire forces and slip angles, is implemented on the simulation side to mimic real-world nonlinearities. For the controller, a linearized state-space model derived from the vehicle dynamics is used, enabling the design of an MPC framework suitable for real-time applications.

The MPC controller employs bilinear discretization to ensure dynamic feasibility while minimizing a quadratic cost function for trajectory tracking and control effort. Simulations are conducted to analyze the interaction between the non-linear simulation model and the linearized control system, highlighting the importance of accurate modeling for robust control design in autonomous driving scenarios.

2 Dynamic Modeling

This section presents the mathematical formulations for the linear and non-linear vehicle models utilized in the project.

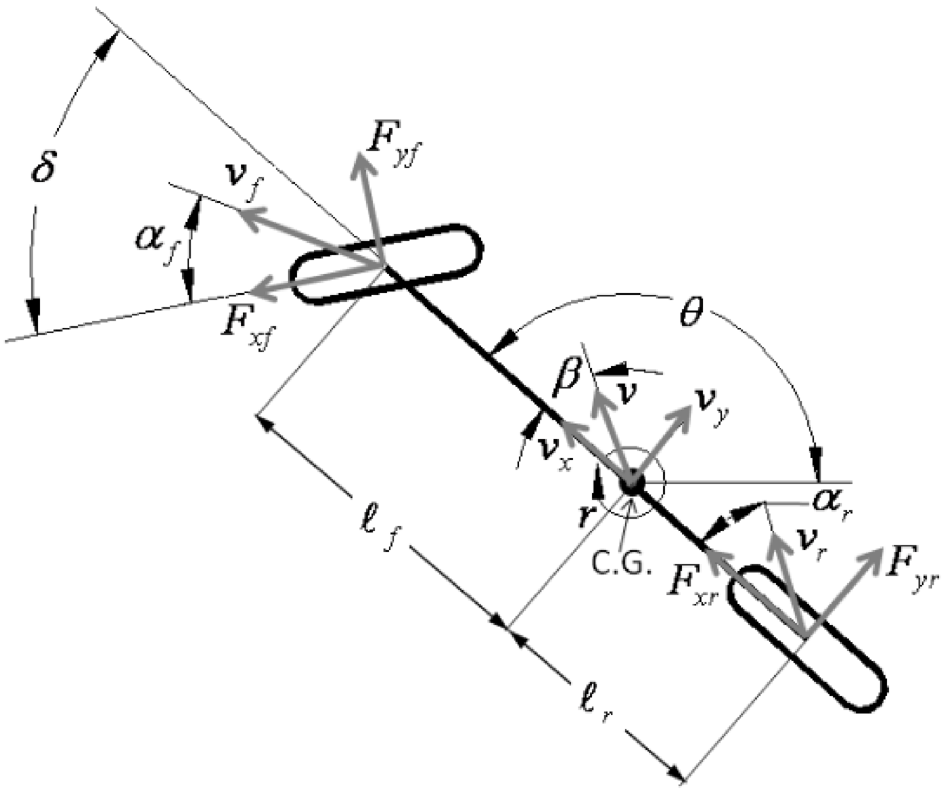

Figure 1: Dynamic Model

The equation of motion considering forces in the lateral direction is given as:

$$F_{yf}\cos(\delta) - F_{xf}\sin(\delta) + F_{yr} = m(\dot{v}_y + v_x r). \tag{1}$$

Considering motion in the plane, a center of gravity (C.G.) along the centerline of the vehicle, and yaw inertia I_z, the yaw moment balance gives:

$$\ell_f(F_{yf}\cos(\delta)) - \ell_r(F_{yr} - F_{xf}\sin(\delta)) = I_z\dot{r}, \tag{2}$$

where r is the angular rate about the yaw axis.

Without the constraint on lateral slip, the slip angles of the tires are given as:

$$\alpha_f = \tan^{-1}\left(\frac{v_y + \ell_f r}{v_x}\right) - \delta, \tag{3}$$

$$\alpha_r = \tan^{-1}\left(\frac{v_y - \ell_r r}{v_x}\right). \tag{4}$$

Modeling the force generated by the wheels as linearly proportional to the slip angle, the lateral forces are defined as:

$$F_{yf} = -2c_f\alpha_f, \tag{5}$$

$$F_{yr} = -2c_r\alpha_r. \tag{6}$$

Assuming a constant longitudinal velocity, $\dot{v}_x = 0$, allows the simplification:

$$F_{xf} = 0. \tag{7}$$

Substituting Eqs. (5) and (6) into Eqs. (1) and (2), and solving for $\dot{v}_y$ and $\dot{r}$, we get:

$$\dot{v}_y = \frac{-2c_f\left[\tan^{-1}\left(\frac{v_y+\ell_f r}{v_x}\right) - \delta\right]\cos(\delta) - 2c_r\tan^{-1}\left(\frac{v_y-\ell_r r}{v_x}\right)}{m} - v_x r, \tag{8}$$

$$\dot{r} = \frac{-2l_f c_f\left[\tan^{-1}\left(\frac{v_y+\ell_f r}{v_x}\right) - \delta\right]\cos(\delta) + 2l_r c_r\tan^{-1}\left(\frac{v_y-\ell_r r}{v_x}\right)}{I_z}. \tag{9}$$

This gives the **dynamic bicycle model**.

Key parameters include:

- **Tire Cornering Stiffness** (c_f, c_r): Front and Rear tire cornering stiffness.
- **Vehicle Mass and Inertia** (m, I_z): Distributed between front and rear axles.
- **Wheelbase** (l_f, l_r): Distance from Center of gravity to axles.
- **Steer** (δ): Steering angle.
- **Slip Angles** (α_f, α_r): Front and rear slip angles respectively.
- **Velocities** ($\mathbf{v}_y, \mathbf{v}_x$): Lateral and Longitudinal Velocities.
- **Tire Forces** ($\mathbf{F}_{yf}, \mathbf{F}_{yr}$): Front and Rear tire forces.

2.1 Linearized Dynamic Bicycle Model

To apply linear control methods to the dynamic bicycle model, the model must be linearized. Applying small-angle assumptions to Eqs. (8) and (9) gives:

$$\dot{v}_y = \frac{-2c_f v_y - 2c_f\ell_f r}{m v_x} + \frac{2c_f\delta}{m} + \frac{-2c_r v_y + 2c_r\ell_r r}{m v_x} - v_x r, \tag{10}$$

$$\dot{r} = \frac{-2\ell_f c_f v_y - 2\ell_f^2 c_f r}{I_z v_x} + \frac{2\ell_f c_f\delta}{I_z} + \frac{2\ell_r c_r v_y - 2\ell_r^2 c_r r}{I_z v_x}. \tag{11}$$

Collecting terms, the linearized equations become:

$$\dot{v}_y = \frac{-(2c_f + 2c_r)}{mv_x} v_y + \left[\frac{2l_r c_r - 2l_f c_f}{mv_x} - v_x\right] r + \frac{2c_f}{m}\delta, \tag{12}$$

$$\dot{r} = \frac{2\ell_r c_r - 2\ell_f c_f}{I_z v_x} v_y + \frac{-(2\ell_f^2 c_f + 2\ell_r^2 c_r)}{I_z v_x} r + \frac{2\ell_f c_f}{I_z}\delta. \tag{13}$$

This represents the **linearized dynamic bicycle model**.

2.2 Linear Vehicle Model in Path Coordinates

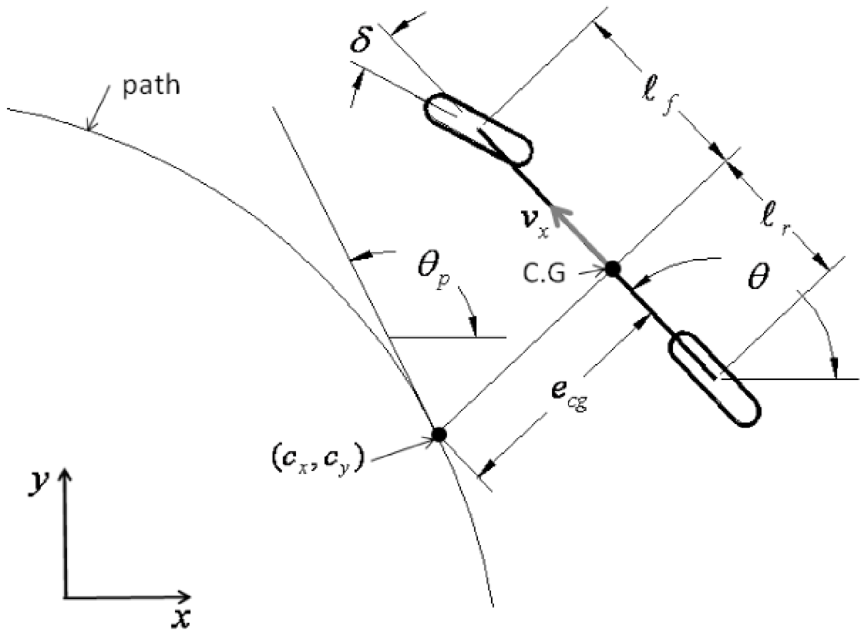

Figure 2: Dynamic Model in path coordinates

As with the kinematic bicycle model, it is useful to express the dynamic bicycle model with respect to the path. With the constant longitudinal velocity assumption, the yaw rate derived from the path $r(s)$ is defined as:

$$r(s) = \kappa(s)v_x.$$

The path-derived lateral acceleration $\dot{v}_y(s)$ follows as:

$$\dot{v}_y(s) = \kappa(s)v_x^2.$$

Letting e_{cg} be the orthogonal distance of the C.G. to the path, we have:

$$\ddot{e}_{\text{cg}} = (\dot{v}_y + v_x r) - \dot{v}_y(s) = \dot{v}_y + v_x(r - r(s)) = \dot{v}_y + v_x\dot{\theta}_e.$$

$$\dot{e}_{\text{cg}} = v_y + v_x \sin(\theta_e),$$

where θ_e is defined as $\theta - \theta_p(s)$. Substituting $(e_{\text{cg}}, \theta_p)$ into Eqs. (12) and (13) and the state space model in tracking error variables is therefore given by:

$$\begin{bmatrix} \dot{e}_{\text{cg}} \\ \ddot{e}_{\text{cg}} \\ \dot{\theta}_e \\ \ddot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2c_f+2c_r}{m\cdot v_x} & \frac{2c_f+2c_r}{m} & \frac{2l_r\cdot c_r-2l_f\cdot c_f}{m\cdot v_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2l_r\cdot c_r-2l_f\cdot c_f}{I_z\cdot v_x} & \frac{2l_f\cdot c_f-2l_r\cdot c_r}{I_z} & -\frac{2l_f^2\cdot c_f+2l_r^2\cdot c_r}{I_z\cdot v_x} \end{bmatrix} \begin{bmatrix} e_{\text{cg}} \\ \dot{e}_{\text{cg}} \\ \theta_e \\ \dot{\theta}_e \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2c_f}{m} \\ 0 \\ \frac{2l_f\cdot c_f}{I_z} \end{bmatrix} \delta + \begin{bmatrix} 0 \\ \frac{2l_r\cdot c_r-2l_f\cdot c_f}{m\cdot v_x} - v_x \\ 0 \\ -\frac{2l_f^2\cdot c_f+2l_r^2\cdot c_r}{I_z\cdot v_x} \end{bmatrix} r(s) \tag{14}$$

The linear vehicle model represents the dynamics in a simplified form suitable for Model Predictive Control (MPC). The equations are derived using small-angle approximations and assuming near-linear operating conditions as shown before.

State-Space Representation: From the equation 14 we have below state representation and A, B, and W are found accordingly.

$$\dot{x} = Ax + Bu + W \tag{15}$$

where x represents the state vector [position error, velocity error, heading angle error, heading angle velocity error], u represents the input vector [steering angle], and W denotes curvature-dependent terms. We need to discretize the matrix using Bilinear technique to find the A_d, B_d and W_d which is used to find **x(k+1)** i.e next state from the current state. Therefore the discretized matrices are:

$$A_d = (I - 0.5\Delta tA)^{-1}(I + 0.5\Delta tA) \tag{16}$$

$$B_d = ((I - 0.5\Delta tA)^{-1}\Delta t)B \tag{17}$$

$$W_d = (I - 0.5\Delta tA)^{-1}\Delta t\kappa v_x W \tag{18}$$

We will use these matrices in our MPC control design for next state prediction.

2.3 Non-linear Vehicle Model

As we discussed at the simulation side of the Autoware specifically in the RViz(Robot Visulaization) side, the vehicle model is implemented in non linear form. The non-linear vehicle model incorporates realistic dynamic behavior, including tire forces and lateral slip angles. This model is essential for validating controller performance under more complex scenarios.

Non-linear Dynamics:

$$\dot{x} = Ax + Bu + W + Ff \tag{19}$$

where f accounts for non-linear tire forces modeled using Pacejka-like formulations.

$$f = \begin{bmatrix} f_{alpha_f} \\ f_{alpha_r} \end{bmatrix} = \begin{bmatrix} -c_2\alpha_f^2 \cdot \text{sgn}(\alpha_f) + c_3\alpha_f^3 \\ -c_2\alpha_r^2 \cdot \text{sgn}(\alpha_r) + c_3\alpha_r^3 \end{bmatrix}$$

,

$$F = \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & \frac{1}{m} \\ 0 & 0 \\ \frac{l_f}{I_z} & \frac{-l_r}{I_z} \end{bmatrix}$$

3 System Specifications

The parameters used in the system are as follows:

- Vehicle Mass, $m = 2400\ kg$
- Moment of Inertia, $I_z = 4670.46\ kg \cdot m^2$
- Distance to Front Axle, $l_f = 1.395\ m$

- Distance to Rear Axle, $l_r = 1.395\ m$
- Tire Cornering Stiffness, $c_f = c_r = 155494.663\ N/rad$
- Maximum Velocity, $v_x = 15\ Km/h$, it also depends on path planner of the Autoware

These parameters were either directly measured or estimated through simulation in Autoware's vehicle model and path planner, ensuring that they accurately reflect the performance characteristics of the vehicle in various operational scenarios. The vehicle's behavior, including its response to steering inputs, acceleration, and braking, is largely influenced by these parameters. Additionally, the parameters are fine-tuned through simulation to account for real-world deviations and dynamic changes.

4 Control System Design

The control system design focuses on implementing a Model Predictive Control (MPC) framework based on the linearized vehicle model while ensuring robustness and feasibility for nonlinear dynamics. The design steps are outlined below:

In the linear MPC formulation, all motion and constraint expressions are linear. For the path following problem, let's assume that the system's motion can be described by a set of equations, denoted as (20). The state evolution and measurements are presented in a discrete state space format, where matrices A, B, and C represent the state transition, control, and measurement matrices, respectively.

$$x_{k+1} = Ax_k + Bu_k + w_k, y_k = Cx_k \tag{20}$$

$$x_k \in R^n, u_k \in R^m, w_k \in R^n, y_k \in R^l, A \in R^{n\times n}, B \in R^{n\times m}, C \in R^{l\times n} \tag{21}$$

Equation 20 represents the state-space equation, where x_k represents the internal states, u_k denotes the input, and w_k represents a known disturbance caused by linearization or problem structure. The measurements are indicated by the variable

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} A^1 \\ A^2 \\ A^3 \\ \vdots \\ A^n \end{bmatrix} x_0 + \begin{bmatrix} B & 0 & \dots & & 0 \\ AB & B & 0 & \dots & 0 \\ A^2B & AB & B & \dots & 0 \\ \vdots & \vdots & & & 0 \\ A^{n-1}B & A^{n-2}B & \dots & AB & B \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{n-1} \end{bmatrix} \tag{22}$$

$$+ \begin{bmatrix} I & 0 & \dots & & 0 \\ A & I & 0 & \dots & 0 \\ A^2 & A & I & \dots & 0 \\ \vdots & \vdots & & & 0 \\ A^{n-1} & A^{n-2} & \dots & A & I \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{n-1} \end{bmatrix} \tag{23}$$

The state transition and measurement equations in 20 are iterative, moving from time k to time $k+1$. By propagating the equation starting from an initial state (x_0, u_0) and control pair along with a specified horizon of N steps, one can predict the trajectories of states and measurements as shown in equation 22 and 23.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} C & 0 & \dots & & 0 \\ 0 & C & 0 & \dots & 0 \\ 0 & 0 & C & \dots & 0 \\ \vdots & & & \ddots & 0 \\ 0 & \dots & 0 & 0 & C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \tag{24}$$

$$X = Fx_0 + GU + SW, Y = HX \tag{25}$$

The equation 25 is similar to state state matrix. We know the matrix elements from previously derived dynamic model equations.
Now that G, S, W, and H are known, we can express the output behavior Y for the next n steps as a function of the input U. This allows us to calculate the control input U so that $Y(U)$ follows the target trajectory Y_{ref}.

4.1 Objective function

The next step is to define a cost function. The cost function generally uses the following quadratic form:

$$J = (Y - Y_{ref})^T Q(Y - Y_{ref}) + (U - U_{ref})^T R(U - U_{ref}) \tag{26}$$

where U_{ref} is the target or steady-state input around which the system is linearized for U.
This cost function is the same as that of the LQR controller. The first term of J penalizes the deviation from the reference trajectory. The second term penalizes the deviation from the reference (or steady-state) control trajectory. The Q and R are the cost weights Positive and Positive semi-semidefinite matrices.

4.2 Constraints

The main advantage of MPC controllers is the capability to deal with any state or input constraints. The constraints can be expressed as box constraints, such as "the tire angle must be within ±30 degrees", and can be put in the following form:

$$u_{min} < u < u_{max} \tag{27}$$

4.3 Implementation

The controller is implemented using a quadratic programming (QP) solver (osqp). The steps are:

1. Linearize the non-linear model around the current state.
2. Discretize the linearized model to derive A_d, B_d, W_d.
3. Solve the QP problem to obtain the optimal control sequence.
4. Apply the first control input and update the state.

5 Robustness Analysis

In this section, we evaluate the robustness of the control system by varying the cornering stiffness (c_α) to observe its effect on the **control effort energy** and **lateral error**:

- **Control Effort Energy**: Defined as $R\delta^2$, where δ is the steering angle, and $R = 10$. It reflects the energy required to maintain the desired trajectory.
- **Lateral Error**: Denoted as e_{cg}, representing the orthogonal distance from the vehicle's center of gravity to the desired path.

To test robustness, we analyze two scenarios: overestimating and underestimating c_α. Each case highlights the trade-offs between control energy and tracking accuracy.

5.1 Overestimated Cornering Stiffness (c_α)

When c_α is overestimated as shown in the Figure 3 and 4:

- **Control Effort Energy**: Increases due to higher stiffness, requiring more frequent and larger steering adjustments.
- **Lateral Error**: Reduced with a maximum of around 0.6 m, as the system aggressively minimizes deviations.

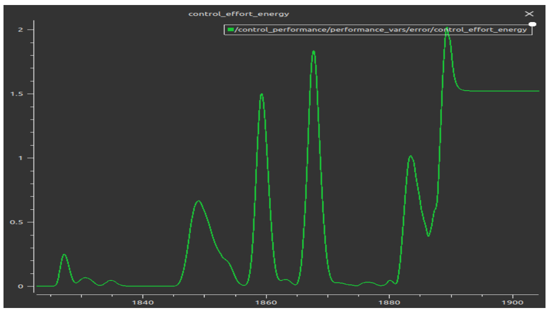

Figure 3: Control Effort Energy (Overestimated c_α)

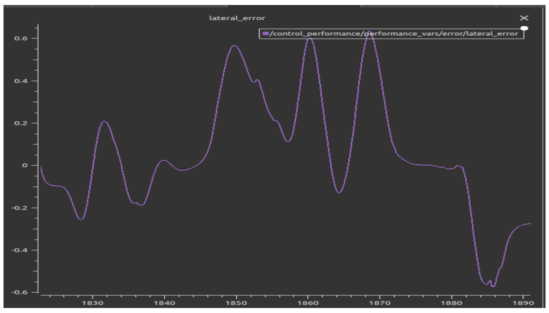

Figure 4: Lateral Error (Overestimated c_α)

5.2 Underestimated Cornering Stiffness (c_α)

When c_α is underestimated as shown in the Figure 5 and 6:

- **Control Effort Energy**: Decreases as the vehicle requires fewer steering adjustments due to lower stiffness.
- **Lateral Error**: Increases with a maximum of approximately 0.8 m, indicating degraded tracking performance.

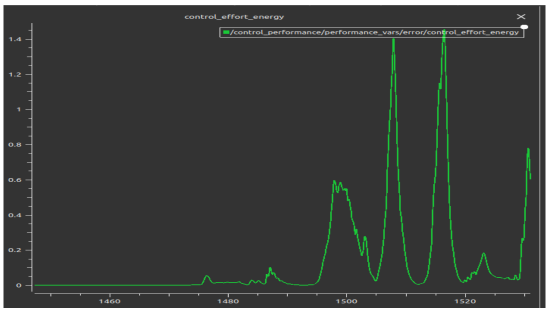

Figure 5: Control Effort Energy (Underestimated c_α)

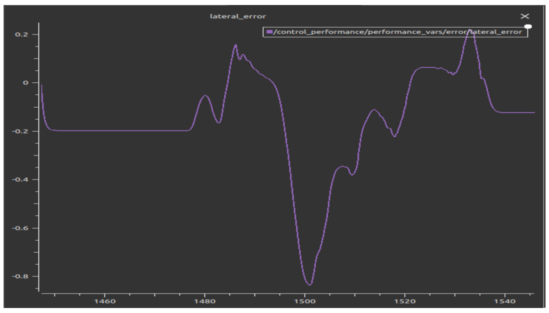

Figure 6: Lateral Error (Underestimated c_α)

5.3 Summary

The analysis shows that with overestimated c_α, energy usage is higher but lateral error is smaller. Conversely, underestimated c_α results in lower energy but larger lateral error. Both cases deviate by approximately 0.2 magnitude in energy and tracking error compared to nominal parameters.

6 Simulation Results and Analysis

The simulation results aim to evaluate the performance of the system under various conditions using both linear and non-linear vehicle models. These results provide insights into the vehicle's lateral dynamics, trajectory tracking, and steering responses, emphasizing the accuracy and stability of the control system. Each subsection focuses on specific aspects of the vehicle's behavior during the simulation.

6.1 Map Overview

The simulation environment, shown in Figure 7, includes scenarios that involve both lane changes and navigating around a circular path. These diverse scenarios are used to evaluate the control system's ability to handle complex driving tasks.

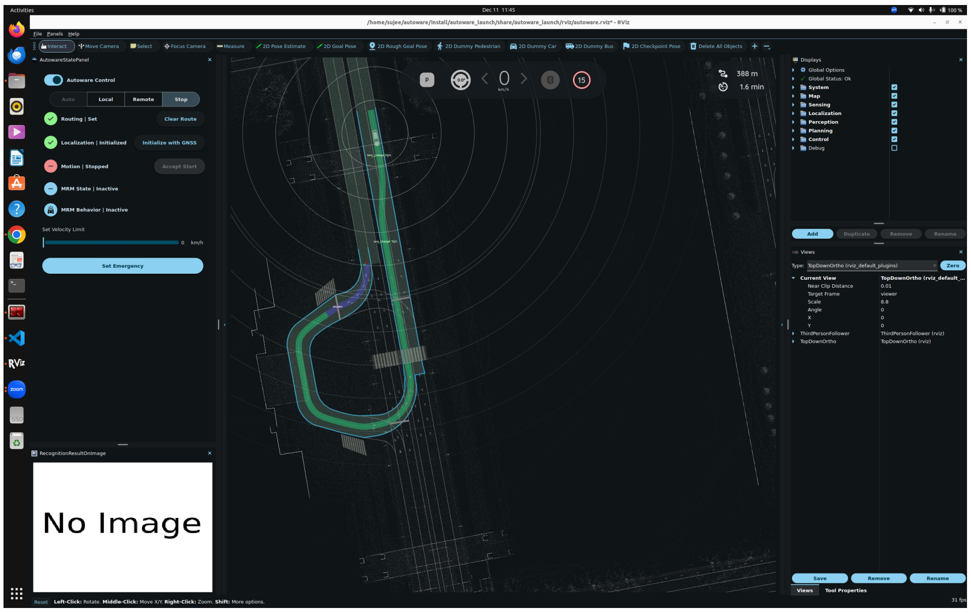

Figure 7: Map

The layout tests the vehicle's trajectory-following capability and dynamic stability as it adapts to various turns and maneuvers, providing a comprehensive assessment of the control system's performance under different driving conditions.

6.2 Lateral Error and Curvature

Figure 8 shows the lateral error (e_{cg}) and curvature values over time. This plot illustrates the tracking performance and curvature adherence of the vehicle during simulation. As we can see that as the curvature changes from $\kappa = 0$ to non zero value the error increases and the max error reaches 0.7m. However, the control system remains robust, quickly correcting the vehicle's trajectory and bringing it back on track within a fraction of a second as the curvature changes.

6.3 Trajectory Tracking and Error Metrics

Figure 9 compares the vehicle's planned trajectory with its actual pose. The error is most noticeable at the curve entry and exit points. The motion planning algorithm generates the trajectory from the start to the goal pose, while the odometry sensor provides the vehicle's position in the Frenet frame. This data is plotted on the same graph to highlight the path deviation during the simulation

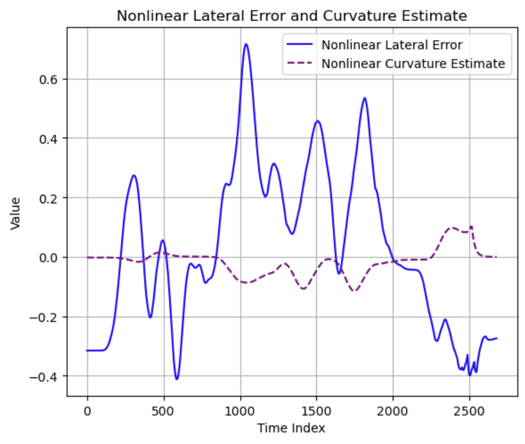

Figure 8: Lateral error and curvature over time.

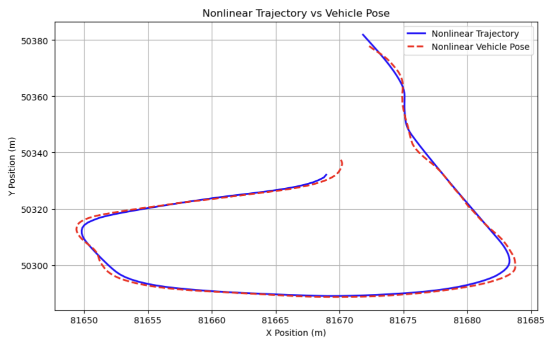

Figure 9: Trajectory tracking and error metrics.

6.4 Lateral Error Velocity and Heading Metrics

Figure 10, 11 and 12 shows the Lateral error velocity ($\dot{e_{cg}}$), Heading error (θ_e), and Heading error velocity ($\dot{\theta}_e$) over time. These values are derived from the state equations used in the model. As observed, the absolute heading error reaches a maximum of 0.3 radians, primarily during the curve entry and exit phases, which is a typical scenario in trajectory tracking.

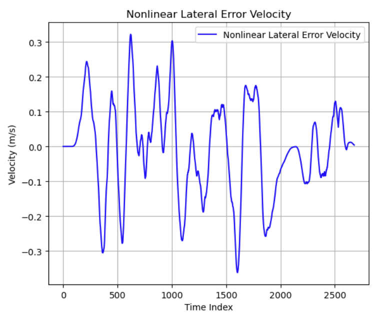

Figure 10: Lateral error velocity over time.

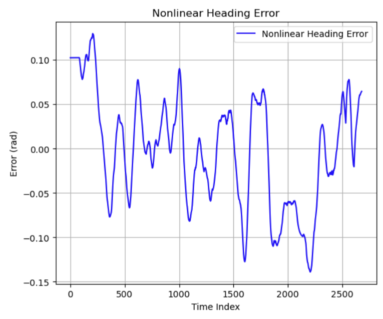

Figure 11: Heading error over time.

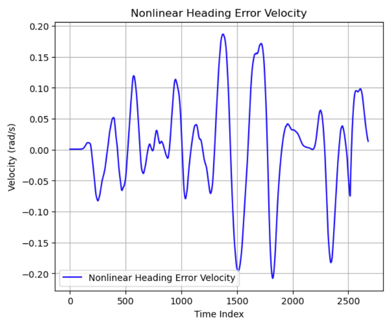

Figure 12: Heading error velocity over time.

6.5 Tire Angle Comparison

The comparison of tire angle response between the linear and non-linear vehicle models, shown in Figure 13, reveals that while the time delay between tire angles is minimal in both models, the non-linear model exhibits more steering noise. This results in less smooth steering compared to the linear model, which shows a more stable response with fewer fluctuations.

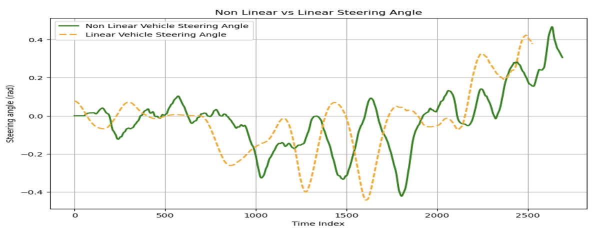

Figure 13: Comparison of tire angles between linear and non-linear models.

7 Conclusion

This report demonstrates the implementation and evaluation of both linear and non-linear vehicle models for Model Predictive Control (MPC) within the Autoware platform, with a focus on lateral control for autonomous vehicles. The key findings include:

- **Superior Performance of Non-Linear Model:** The non-linear model consistently outperformed the linear model, especially in handling dynamic vehicle behaviors, such as tire forces and slip angles, resulting in more accurate trajectory tracking.
- **MPC Robustness:** Both linear and non-linear MPC controllers demonstrated robustness to disturbances and system uncertainties, although the non-linear MPC showed superior resilience in more complex scenarios.
- **Real-World Applicability:** The non-linear model's ability to closely simulate real-world vehicle dynamics makes it more suitable for deployment in autonomous driving systems, where handling real-world complexities is essential.

In future work, additional research into adaptive MPC controllers and further tuning of the vehicle dynamics could enhance the robustness and efficiency of the control system, particularly in highly dynamic or unstructured environments.

8 References

1. Jeon, W., Chakrabarty, A., Zemouche, A., Rajamani, R. (2021). Simultaneous state estimation and tire model learning for autonomous vehicle applications. IEEE/ASME Transactions on Mechatronics, 26(4), 1941-1950.
2. Snider, J. M. (2009). Automatic steering methods for autonomous automobile path tracking. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08.
3. Funke, J., Brown, M., Erlien, S. M., Gerdes, J. C. (2016). Collision avoidance and stabilization for autonomous vehicles in emergency scenarios. IEEE Transactions on Control Systems Technology, 25(4), 1204-1216.
4. Autoware Control Design
5. Autoware Vehicle Model

9 Appendix: ROS, Autoware, and Vehicle Simulation Details

9.1 ROS and Autoware Integration

Robot Operating System (ROS) serves as the backbone for Autoware, providing the middleware needed for efficient communication between various components of an autonomous vehicle system. Autoware utilizes ROS to handle tasks such as message passing, sensor integration, and modular development of software packages. By leveraging ROS, Autoware can seamlessly connect perception, planning, and control modules.

9.2 RViz and Vehicle Coordinates in Simulation

RViz is a visualization tool within ROS that allows users to interactively view data from the autonomous vehicle in both real-time and simulation environments. In the context of Autoware, RViz is used to display the vehicle's trajectory, sensor data, and pose in a simulated or real-world coordinate frame. This visualization provides critical insights into the vehicle's behavior and its planned trajectory.

Vehicle coordinates within RViz are derived from the state provided by the non-linear vehicle model and the planned trajectory. The following equations demonstrate how the coordinates X and Y are calculated.

9.3 Coordinate Calculation

The desired trajectory X_{des} and Y_{des} are computed from the path planning module, which provides a time-dependent trajectory. The equations below outline how the actual coordinates X and Y are determined using the non-linear vehicle model's output:

$$\begin{aligned}
\text{Yaw Angle: } \theta(t) &= \int_0^t \kappa V_x \, dt \\
f_1(t) &= V_x \cdot \cos(\theta(t)) \\
f_2(t) &= V_x \cdot \sin(\theta(t)) \\
X_{\text{des}} &= \int f_1(t) \, dt \\
Y_{\text{des}} &= \int f_2(t) \, dt \\
X &= X_{\text{des}} - e_1 \cdot \sin(e_2 + \theta(t)) \\
Y &= Y_{\text{des}} + e_1 \cdot \cos(e_2 + \theta(t))
\end{aligned}$$

Here:

- e_1 is the lateral error from the desired trajectory.
- e_2 is the heading error.
- $\theta(t)$ represents the yaw angle derived from the vehicle model.
- V_x is the longitudinal velocity of the vehicle.
- κ road curvature.

The non-linear vehicle model provides the next state of the vehicle, which is then used to compute X, Y, and θ. These values are subsequently fed into the RViz simulation environment to visualize the vehicle's motion and validate the trajectory tracking. By continuously updating these coordinates, the vehicle's pose and path alignment can be monitored effectively in real-time.

9.4 Additional Resources

All the relevant code implementations and demonstration videos have been uploaded to a shared drive for reference. Access them using the following link:
`https://drive.google.com/drive/folders/1nrLzc-StMBUQn3rdMJOat16VVo2AxnNP?usp=sharing`