Model Predictive Control

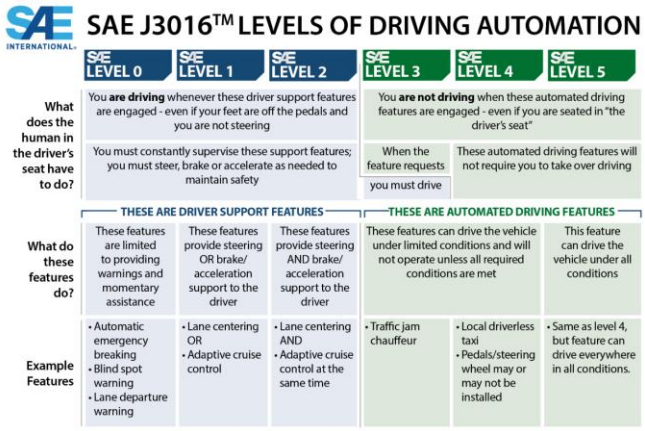
Leo Jaeeun Kim
University of Michigan
General Motors

ABSTRACT

Model Predictive Control (MPC) is a powerful optimization-based control strategy widely used in modern engineering applications, particularly in autonomous systems, robotics, and advanced driver-assistance systems (ADAS). Unlike traditional control methods, MPC explicitly incorporates system constraints and predicts future states using dynamic models to optimize control inputs over a finite horizon. This predictive nature allows for improved performance, constraint satisfaction, and adaptability to varying operating conditions. Despite its advantages, real-time implementation of MPC remains challenging due to its computational complexity and the need for efficient solvers. This paper provides an in-depth analysis of MPC, its mathematical formulation, key advantages, challenges, and applications in control systems, particularly in ADAS and autonomous vehicle domains.

SAE International defines different levels of automation in driving, ranging from SAE Level 0 (manual driving with support features) to SAE Level 5 (fully autonomous driving in all conditions). MPC plays a critical role in the transition from driver-assist systems (SAE Levels 1 and 2) to higher levels of autonomy (SAE Levels 3–5) by enabling predictive control strategies that enhance vehicle safety, stability, and decision-making capabilities. By effectively managing vehicle dynamics, constraint handling, and real-time adaptation, MPC contributes to safer and more efficient autonomous driving solutions.

This paper provides an in-depth analysis of MPC, its mathematical formulation, key advantages, challenges, and applications in control systems, particularly in ADAS and autonomous vehicle domains.



<SAE J3016 Levels of Driving Automation>

INTRODUCTION

Control systems play a crucial role in modern engineering, enabling automated decision-making and system stabilization across various applications. Traditional control techniques such as Proportional-Integral-Derivative (PID) and Linear Quadratic Regulators (LQR) have been widely used due to their simplicity and effectiveness. However, these methods often struggle with handling constraints, multi-variable interactions, and dynamic environments, which are critical in applications such as autonomous vehicles and robotics.

Model Predictive Control (MPC) has emerged as a robust alternative by leveraging system models to predict future behavior and optimize control actions accordingly. Unlike conventional controllers, MPC formulates control as an optimization problem, minimizing a predefined cost function while respecting system dynamics and constraints. This approach allows for proactive decision-making, leading to improved safety, efficiency, and adaptability.

The increasing complexity of modern autonomous systems necessitates control strategies that can manage constraints on states and inputs while ensuring optimal performance. MPC's ability to integrate real-time optimization and predictive decision-making makes it particularly suitable for applications in ADAS, robotics, and aerospace engineering. However, challenges such as computational demand, model accuracy, and real-time feasibility must be addressed to enhance its practical deployment.

This paper explores the theoretical foundations of MPC, its formulation, and computational challenges. Additionally, it discusses its applications in autonomous vehicle control, emphasizing its role in trajectory planning, collision avoidance, and motion optimization. By examining recent advancements, this study aims to highlight MPC’s potential and the ongoing efforts to improve its real-time performance for safety-critical applications

Beyond LQR: Motivation for Model Predictive Control

The Linear Quadratic Regulator (LQR) is a classical optimal control approach that minimizes a quadratic cost function to determine an optimal control policy. The weight matrices, and are positive definite and tunable, allowing flexibility in shaping the system's performance. Unlike a purely greedy algorithm, which optimizes only the immediate cost at and disregards long-term effects, LQR considers future states and control inputs over an infinite horizon. This characteristic provides improved stability and robustness compared to greedy control methods.

However, LQR has significant limitations when applied to real-world autonomous systems. One key drawback is its reliance on open-loop control sequences without feedback adjustment during execution. The linear state feedback law lacks the ability to adapt to unforeseen disturbances or dynamic environmental changes, making it unsuitable for highly dynamic applications such as autonomous driving.

Moreover, LQR does not naturally handle state and input constraints, such as maintaining a safe distance from obstacles or enforcing actuator limits. While extensions exist to incorporate affine inequality constraints by solving a convex quadratic programming (QP) problem, these do not fully address all challenges associated with real-time decision-making under uncertainty.

These limitations drive the need for Model Predictive Control (MPC). MPC extends LQR by incorporating real-time trajectory optimization as a feedback policy. The key idea is to measure the current state (requiring a state estimator), solve an online optimization problem to compute a constrained optimal trajectory, execute only the first control action, and then repeat the process at the next time step. The resulting QP-MPC framework is computationally efficient and well-structured, often leveraging sparse matrices to reduce memory and computational load. This predictive and constraint-aware approach makes MPC a superior alternative for advanced autonomous control applications.

METHOD

1) State Feedback in LQR vs. MPC

In LQR, the control law is given by a linear state feedback: $u_k = Kx_k$ where is a gain matrix derived from solving the Riccati equation. In contrast, MPC determines the control input by solving an online optimization problem: $u_k = u_{mpc}(x_k)$. This optimization-based approach allows MPC to explicitly handle constraints, but its performance depends on real-time computation and feasibility conditions. While MPC lacks a formal guarantee of convergence, it performs well in practice. When the quadratic programming (QP) constraints are inactive and the prediction horizon is sufficiently long, MPC can approximate the asymptotic behavior of LQR.

Ensuring QP feasibility is crucial for MPC implementation. Proper tuning of the prediction horizon is often necessary to balance computational efficiency and control performance.

2) Vehicle Dynamics Model

Consider a vehicle with state variables: $s := [x, y, \psi, v]^{\mathrm{T}}$, where ($x$,$y$) is the position, ψ is the yaw angle, and v is the velocity. The vehicle is controlled by the steering angle of the front wheel δ and the acceleration a.

$\frac{d}{dt}s =: f(x,u), u := (\delta, a)$, given the reference trajectory.

The equation of motion of the vehicle are given:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ \psi \\ v \end{bmatrix} = \begin{bmatrix} [v\cos(\psi) \\ v\sin(\psi) \\ \frac{v}{L_r}\tan(\delta) \\ a \end{bmatrix}$$

where L_r and L_f represent the distances from the vehicle's center to the rear and front axles, respectively.

For computational efficiency in MPC, the nonlinear vehicle dynamics are linearized around an operating point. The discrete-time approximation with a sampling time is given by:

$$x_{k+1} = x_k + h * v_k cos(\psi_k)$$

$$y_{k+1} = y_k + h * v_k sin(\psi_k)$$

$$\psi_{k+1} = \psi_k + h * \frac{v_k}{L} tan(\delta_k)$$

$$v_{k+1} = v_k + h * a_k$$

The Jacobian matrices for linearization are computed as:

$$\frac{dx}{dx} = 1\,, \qquad \frac{dx}{d\psi} = -hv_k \sin(\psi_k)\,, \qquad \frac{dx}{dv} = h * cost(\psi_k)$$

This linearized model allows the MPC optimization problem to be efficiently formulated as a Quadratic Programming (QP) problem, ensuring real-time feasibility while maintaining accuracy in trajectory tracking. By leveraging this model, MPC dynamically computes optimal control inputs while explicitly considering state and input constraints, making it a highly effective approach for autonomous vehicle control.

The MPC controller determines the optimal control sequence by minimizing a quadratic cost function while satisfying system constraints. The cost function is typically defined as:

$$J = \sum_{k=0}^{N} (x_k - x_{ref,k})^T Q (x_k - x_{ref,k}) + u_k^T \mathrm{R} u_k$$

State tracking cost: $(x_k - x_{ref,k})^T Q (x_k - x_{ref,k})$

where x_k is the system state at time step k, $x_{ref,k}$ is the reference trajectory (desired state), and Q is a positive semi-definite weight matrix that penalizes deviations from the reference state. The state tracking cost term is a quadratic measure of how far the state x_k is from the desired state $x_{ref,k}$.

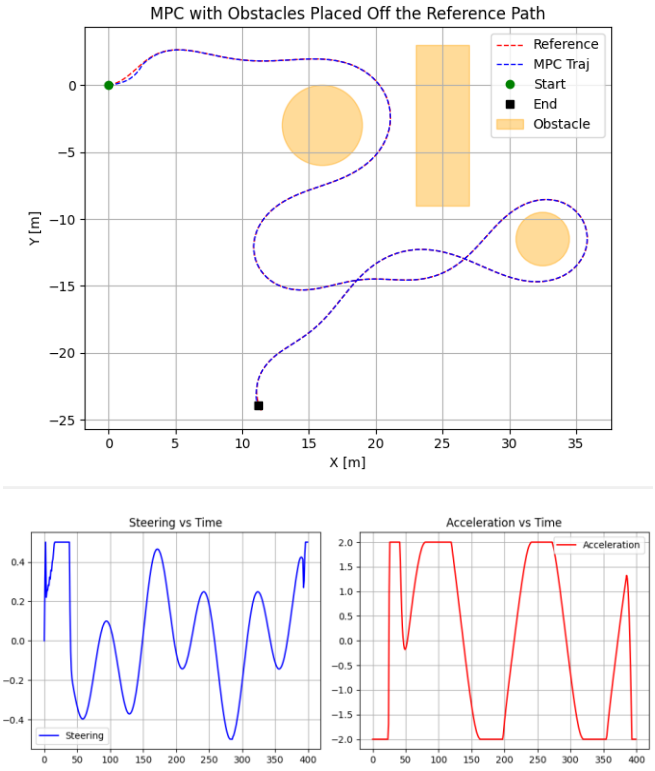
Control effort cost: $u_k^T \mathrm{R} u_k$

where u_k is the control input at time step k, and R is a positive definite weight matrix that penalizes excessive control effort.

The control effort term ensures that the control inputs are not too aggressive, preventing large or unnecessary actuator movements.

The equation sums these costs from k=0 to N, meaning it considers a finite prediction horizon of N steps into the future. The controller selects control inputs u_k to minimize this cumulative cost while obeying system constraints.

To ensure practical feasibility and safety, several constraints are imposed on the MPC controller. State constraints include position and velocity limits, as well as yaw angle restrictions to maintain stability and adherence to road boundaries. Control input constraints ensure that the steering angle and acceleration remain within allowable limits, preventing excessive maneuvers that could compromise vehicle safety. Additionally, dynamic constraints require the system to follow the linearized vehicle model equations, ensuring that the predicted motion remains physically realistic. The control horizon must also be carefully chosen to balance computational efficiency with predictive performance, allowing the controller to make optimal decisions without excessive computational burden.



The example code for MPC simulation can be found in https://github.com/woa0425/Leo_ADAS_script/blob/main/MPC/MPC_python/MPC_simulaton.py

By incorporating real-time constraint handling and predictive capabilities, MPC provides a robust solution for trajectory tracking in autonomous vehicles, outperforming traditional control approaches such as LQR.

3) Comparison of MPC and LQR in Vehicle Trajectory Tracking

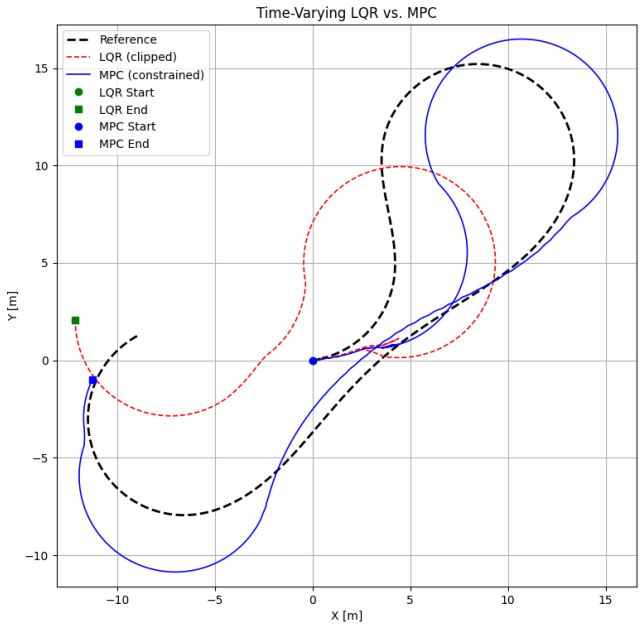
Model Predictive Control (MPC) and Linear Quadratic Regulator (LQR) are both used for trajectory tracking in vehicle control, but MPC offers several advantages in constrained and nonlinear environments.

In both Model Predictive Control (MPC) and Linear Quadratic Regulator (LQR), the weighting matrices Q and R play a crucial role in shaping control behavior by penalizing state deviations and control efforts, respectively. While LQR optimizes the control input instantaneously based on a fixed quadratic cost function, MPC extends this concept by optimizing over a finite prediction horizon, allowing it to anticipate future states while explicitly handling constraints. Increasing Q places a higher emphasis on accurate state tracking, leading to more aggressive corrections, whereas decreasing Q allows greater deviations from the reference trajectory. Conversely, increasing R discourages large control inputs, resulting in smoother actions, while decreasing R permits more aggressive control efforts to correct errors quickly. This trade-off between tracking precision and control smoothness is particularly important in applications like autonomous driving, where MPC ensures optimal decision-making over time, unlike LQR, which reacts based only on the current state.

Additionally, Qf (final state cost matrix), the terminal cost matrix, defines the penalty applied to the final state at the end of the time horizon. In finite-horizon LQR, Qf ensures that the system ends close to the desired state since there is no infinite time horizon to smooth out deviations. In MPC, optimization occurs over a moving window, but only the first control input is applied before shifting the horizon forward. Qf helps guide the system towards stability beyond the prediction horizon by heavily penalizing final-state errors. A high Qf forces the system to closely follow the reference by the end of the time window, ensuring precise terminal tracking. In contrast, a low Qf allows more flexibility at the final state, which can be useful in cases where the system continues moving beyond the optimization window. For example, in autonomous driving, when a vehicle tracks a trajectory over a 5-second prediction horizon, a high Qf ensures it remains on track by the end of the interval. If Qf is too low, the controller might not aggressively correct errors, assuming future optimization will handle them. Thus, Qf acts as a final “push” to improve long-term behavior, ensuring MPC doesn’t just optimize for short-term gains but also respects stability and convergence.

MPC explicitly handles constraints, such as steering and acceleration limits, by incorporating them directly into its optimization, whereas LQR assumes unbounded inputs. MPC also leverages future predictions, optimizing over a finite horizon to anticipate future states, while LQR optimizes only for the current state, making it purely reactive. Additionally, MPC adapts better to nonlinearities, continuously updating its

model to handle dynamic changes, whereas LQR relies on a fixed linearized approximation, limiting its accuracy in nonlinear conditions. MPC generates smoother and more stable control actions, avoiding the aggressive and oscillatory behavior sometimes seen with LQR. However, LQR can be preferable in computationally constrained applications, as it provides a closed-form solution with lower computational demand, making it suitable for high-speed execution when constraints are minimal.



The example code for comparison between LQR and MPC in vehicle trajectory tracking can be found in https://github.com/woa0425/Leo_ADAS_script/blob/main/MPC/MPC_python/MPC_VS_LQR.py

CONCLUSION

Model Predictive Control (MPC) has emerged as a highly effective control strategy for autonomous systems, outperforming traditional methods such as the Linear Quadratic Regulator (LQR) in complex, constrained, and dynamic environments. By integrating real-time optimization, state predictions, and constraint handling, MPC enables precise trajectory tracking, adaptive decision-making, and improved safety in applications like advanced driver-assistance systems (ADAS) and autonomous vehicles.

This paper has provided a detailed analysis of MPC, highlighting its formulation, advantages, challenges, and applications. A key takeaway is the critical role of weighting matrices Q, R, and Qf in shaping control behavior. Unlike LQR, which relies on a fixed state feedback gain and lacks constraint handling, MPC continuously optimizes over a finite horizon, making it more robust in real-world scenarios where constraints on steering, acceleration, and safety must be explicitly managed. However, the computational complexity of MPC remains a significant challenge for real-time applications, requiring efficient solvers and hardware acceleration for practical deployment. Future advancements in optimization algorithms, model approximation techniques, and high-performance computing will further enhance MPC's feasibility for real-time autonomous control.

In conclusion, MPC provides a powerful framework for autonomous vehicle control, balancing tracking accuracy, constraint satisfaction, and adaptive decision-making. While LQR remains a valuable technique in scenarios with minimal constraints and high-speed execution requirements, MPC's predictive capabilities make it the preferred choice for advanced, safety-critical applications. Further research into reducing computational overhead and improving real-time feasibility will continue to expand MPC's role in autonomous driving and robotics.

REFERENCES

[1] University of Michigan, NAVARCH 565 – Lecture 07: Model Predictive Control, Fall 2023.
[2] University of Michigan, NAVARCH 565 – Lecture 08: Model Predictive Control Example, Fall 2023.