IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.DOI

A Comprehensive Analysis of Model Predictive Control for Lane Keeping Assist System

JAMES GARCIA[1],EVANDRO TEIXEIRA[1], ANDRÉ MURILO[2], AND RAFAEL RODRIGUES[1]

[1]Department of Automotive Engineering, University of Brasilia, Brasilia, Federal District, Brazil
[2]Department of Automatics, Federal University of Lavras, Lavras - Minas Gerais. Brazil

Corresponding author: Evandro Teixeira (e-mail: evandroleonardo@unb.br). ORCID: 0000-0002-4781-6782 (Evandro Teixeira)

This work was supported by the Fundação de Desenvolvimento da Pesquisa (FUNDEP) [Grant Number 27192.02.01/2020.10-00]

ABSTRACT

Lane Keeping Assist System (LKAS) enhances comfort and safety while driving. It plays a significant role in the Advanced Driver Assistance System (ADAS) and future Automated Driving (AD). The LKAS solution aims to help the driver keep the vehicle within the road lines, preventing unintentional lane departure. Despite LKAS being an important solution for comfortable driving, robust LKAS steering control is still lacking, requiring constant driver intervention or premature LKAS deactivation. LKAS require optimal control solutions with real-time constraints. This paper comprehensively analyzes Model Predictive Control (MPC) for real-time LKAS applications. Classical and parameterized MPC schemes with distinct Quadratic Programming (QP) solvers are combined to evaluate LKAS closed-loop control performance and real-time constraints. A sideslip and lateral speed bicycle modes were used to evaluate classical, trivial, and exponential MPC schemes. Experimental results highlight the three MPC and QP-appropriate solutions with satisfactory reference tracking without steering command and real-time constraints violation.

INDEX TERMS Advanced Driver Assistance Systems, Lane Keeping Assist System, Model Predictive Control, MPC parameterization, Quadratic Programming.

I. INTRODUCTION

VEHICLE safety encompasses regulations, actions, and technological solutions designed to prevent traffic accidents [1]. Today, there is a global effort involving multiple sectors, including transport, health, education, etc., to address safety on roads [2]. Carmakers and governments worldwide are actively seeking innovative solutions to enhance the safety of all road users. Active safety systems such as Anti-lock Braking Systems (ABS), Electronic Stability Control (ESC), Autonomous Emergency Braking (AEBS), and Lane Assist Systems (LAS) are employed to prevent accidents from occurring altogether, sometimes actively assisting the driver in mitigating the impact. These Advanced Driver Assistance Systems (ADAS) are often implemented to be forgiving of human error.

The Lane Assist System is a type of active safety technology that proactively assists the driver in keeping the vehicle within its lane [3]. It promotes comfort and safety by supporting the driver with lateral control tasks [4]. Lane Keeping Assist System (LKAS) solutions detect unintentional lane drift by adjusting the steering to keep the vehicle within its lane [5]. A forward-looking camera serves as a sensor input for LKAS, detecting lines and lane features. Information from the car driving scene, such as lateral offset, heading angle, lane curvature, etc., is retrieved and forwarded to the LKAS controller [6]. This information is used to make subtle adjustments to maintain the vehicle in the middle of the detected lane. Additional inputs, such as sidelight activation and hands on the steering wheel, are checked to avoid unauthorized vehicle intervention [7].

LKAS provides a more confident driving experience on narrow roadways. It often reduces unintentional lane departure resulting from driver fatigue, high workload, momentary distraction, or somnolence [8]. Through an adaptive learning mechanism, LKAS adjusts itself, taking into account the driver's style and avoiding potential conflicts between steering actuation and driver intervention. LKAS proves particularly useful in long trips or heavy traffic conditions [9].

Despite the recognized benefits of LKAS, there are barriers to implementing an effective lane-keeping assistance solution. Weather conditions can sometimes impact LKAS sensors, degrading traffic scene data such as pavement marks and speed limit signs. Sunlight, dirt, heavy rain, snow, and shadow make lane mark recognition a challenging task [10]. Furthermore, inappropriate or early steering system intervention often leads to discomfort while driving, resulting in premature LKAS deactivation. To avoid undesired steering actuation, some researchers have dedicated efforts to improving lateral displacement control using various methods, including Linear Quadratic Regulator (LQR) [11], Proportional Integral Derivative (PID) [12], Fuzzy Control [13], End-to-End Learning [14], Robust h_∞ Controller [15], and Model Predictive Control (MPC) [1] [16].

Model Predictive Control (MPC) appears to be an effective solution for LKAS system control. It can adapt to different driving scenarios while satisfying predefined constraints. State variables and control outputs can be constrained to adhere to comfort and safety limits, preventing abrupt steering interventions. MPC handles predictions and state constraints, making it a robust candidate for LKAS control. MPC estimates future vehicle behaviour, rejecting potential disturbances and enabling smooth conduction within the road lane. Weather conditions and variations in driving style are inherently considered in this control technique.

In this paper, a comprehensive analysis of Model Predictive Control (MPC) for the Lane Keeping Assist System (LKAS) is conducted. Specifically, classical and parameterized MPC schemes with various Quadratic Programming solvers (QPs) are investigated for closed-loop control performance and real-time application constraints. A potential advantage of parameterized MPC-like strategies lies in their significant reduction of the computational workload associated with real-time optimization problem-solving. Therefore, this article presents a thorough examination of solutions that involve the selection of an appropriate MPC parameterization for the control sequence along with a suitable QP solver. This allows for a simultaneous reduction in computational time while ensuring optimal performance and addressing the operational constraints of LKAS.

In this context, the contributions of this paper aim to address the following points:

- Design and implement MPC-based strategies for LKAS applications, employing two distinct representations of vehicle dynamic models that adhere to the operational constraints of the system.
- Develop techniques aimed to reduce the computational burden of MPC through appropriate control sequence parameterization. The paper will present two theoretical approaches: a trivial and an exponential parameterization technique.
- Experiments and comparative analysis of diverse QP-type solvers to evaluate both controller efficacy and resulting computational time.
- Application of parameterization techniques across various established QP solvers, attended by comprehensive comparisons regarding LKAS performance and computational burden among different MPC methodologies.

II. RELATED WORK

Several research initiatives have focused on the application of Model Predictive Control for Lane Keeping Assist Systems. For instance, Salt et al. (2021) designed a Linear Parameter Variation (LPV) model to balance computational effort and vehicle dynamic modelling accuracy [16]. Similarly, An et al. (2020) highlighted an MPC formulation, including driver behaviour and a co-driver, to control the steering system whenever the vehicle leaves the lane [17]. Farag (2021) presented an MPC solution including trajectory tracking, comfort, and trip time as sub-goals of the MPC cost function [18]. Additionally, Rathai (2017) showed a novel framework for lane assist systems ensuring vehicle stability in the presence of disturbances like road curves, bank angles, and longitudinal speed variation [19]. Liu (2015) presented another solution including uncertainties and disturbances due to driver behaviors [20].

Despite the valuable contributions of the research as mentioned earlier, they fail to investigate crucial issues related to practical automotive applications. Some of them do not consider closed-loop control analysis subject to real-time constraints [19]. Furthermore, QP solutions with detailed performance analysis are limited [17]–[20] sometimes without real-time performance analysis [19]. This lack makes LKAS-MPC-based solutions difficult for real-world automotive applications [21], [22]. Most often, prediction horizon, cost function and real-time constraints require computing resources [23] not available in most automotive Electronic Control Units (ECUs) [24]. One key characteristic of MPC is the implicit determination of the control law by solving the constrained optimization problem online [25]. Indeed, the computational burden associated with MPC-type applications stands as a significant drawback of such strategies. This challenge constrains the advancement of feasible MPC implementations that are crucial for embedded systems, such as automotive ECUs and other electronic devices.

III. VEHICLE MODEL

The lateral vehicle dynamics will be fully described using a two-degree-of-freedom (2-DOF) bicycle model [26] (see Figure 1). To simplify the model, certain assumptions have been made, namely, the exclusion of roll and pitch dynamics, and the use of small angles and constant longitudinal speed [27]–[29]. The decision not to consider such dynamics may impact the computation of tire load distribution, especially in cases of abrupt manoeuvres. In such scenarios, the assumption of linearity in the bicycle model may no longer be valid. However, concerning LKAS, simplifying model conditions doesn't considerably interfere with the vehicle model behaviour. This is primarily because, in most cases, the vehicle dynamics operate within a linear operating zone,

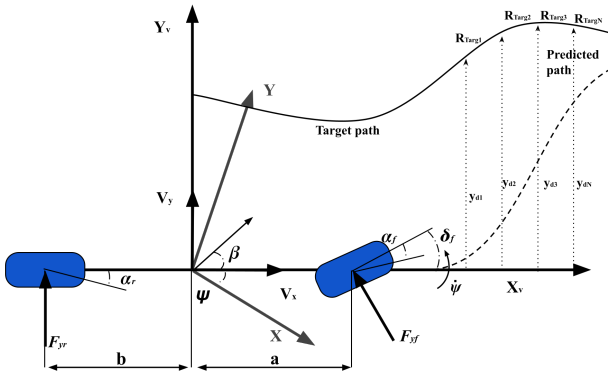

FIGURE 1: Bicycle vehicle model

significantly minimizing the impact of model simplifications on the system's overall performance. Thus, the lateral vehicle position and yaw angle are represented by the 2-DOF model and are derived from the following differential equations:

$$\begin{aligned} ma_y &= -m\,v_x\,\dot{\psi} + F_{F_y} + F_{R_y} \\ I_z\ddot{\psi} &= l_F\,F_{F_y} - l_R\,F_{R_y} \end{aligned} \tag{1}$$

Where m represents the vehicle mass; $\dot{\psi}$ and $\ddot{\psi}$ devote the yaw speed and acceleration about the z-axis at the centre of gravity (CG) of the vehicle; a_y represents the lateral acceleration and v_x is the longitudinal vehicle speed; Iz is the inertial moment around the z-axis; and a and b are front and rear axles distance from CG respectively. The lateral tire forces F_{F_y} and F_{R_y} can be expressed by the following equations:

$$\begin{aligned} F_{F_y} &= 2\,C_{\alpha_f}\,\alpha_f \\ F_{R_y} &= 2\,C_{\alpha_r}\,\alpha_r \end{aligned} \tag{2}$$

In the given expression C_{α_f} and C_{α_r} represent the front and rear cornering stiffness respectively. Since slip angle (α) is crucial to compute wheel course and true vehicle direction, the front (α_f) and rear (α_r) slip angle can be defined as follows:

$$\alpha_f = \delta - \frac{v_y + a\,\dot{\psi}}{v_x} \quad ; \quad \alpha_r = -\frac{v_y - b\,\dot{\psi}}{v_x} \tag{3}$$

For an effective LKAS MPC-based controller design, vehicle dynamics modelling is required using state-space notation. A bicycle vehicle model can be formulated by combining equations 2 and 3 as follows:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,u(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) \end{aligned} \tag{4}$$

Where $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{B}$ are the state, input and output matrixes respectively. The state vector is represented by $\mathbf{x}(t)$ whilst $u(t)$ and $\mathbf{y}(t)$ are the input and output commands. Two mathematical formulations can be used to represent a Linear Time-Invariant (LTI) system model in state space. The sideslip bicycle model formulation is presented as follows:

$$\mathbf{A} = \begin{bmatrix} 0 & vx & vx & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{C_{\alpha f}+C_{\alpha r}}{mv_x} & \frac{aC_{\alpha f}-bC_{\alpha r}}{mv_x^2} - 1 \\ 0 & 0 & \frac{bC_{\alpha r}-aC_{\alpha f}}{I_z} & -\frac{(b^2C_{\alpha r}+a^2C_{\alpha f})}{I_z v_x} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \frac{C_{\alpha f}}{m} \\ \frac{aC_{\alpha f}}{I_z} \end{bmatrix} \quad ; \quad \mathbf{C} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \tag{5}$$

In the mentioned LTI system, the mathematical formulation considers the sideslip angle (β) at the CG. For constant longitudinal speed and small steering angular variations [30], the sideslip angle can be written as $\beta \approx v_y/v_x$. Thus, the update state space vector can be expressed as $\mathbf{x}(t) = [\ y\ \psi\ \beta\ \dot{\psi}\ ]^T$, where y is the lateral position, β is the sideslip angle, ψ and $\dot{\psi}$ are the yaw angle and yaw rate respectively. The lateral speed bicycle vehicle model, where the sideslip angle (β) is replaced by lateral speed ($\dot{y}$), can be written as follows [17]:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2C_{\alpha f}+2C_{\alpha r}}{mv_x} & 0 & -v_x - \frac{2aC_{\alpha f}-2bC_{\alpha r}}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2aC_{\alpha f}-2bC_{\alpha r}}{I_z v_x} & 0 & -\frac{2a^2C_{\alpha f}+2b^2C_{\alpha r}}{I_z v_x} \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ \frac{2C_{\alpha f}}{m} \\ 0 \\ \frac{2aC_{\alpha f}}{I_z} \end{bmatrix} \quad ; \quad \mathbf{C} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \tag{6}$$

Where $\mathbf{x}(t) = [\ y\ \dot{y}\ \psi\ \dot{\psi}\ ]$ is the state vector containing the lateral position and yaw angle along with their respective derivatives. The target lateral displacement (y_d) can be expressed in terms of inertial coordinates ($[X(Rtarg), Y(Rtarg)]$) as follows:

$$\mathbf{y}_d = Y(R_{targ})cos(\psi) - X(R_{targ})sin(\psi) \tag{7}$$

IV. LANE KEEPING MPC-BASED CONTROLLER

In the context of the LKAS MPC-based controller design for a Linear Time-Invariant (LTI) vehicle model, following Alamir's design rules and approach [31], the vehicle model discretization can be expressed as follows:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}_d\,\mathbf{x}(k) + \mathbf{B}_d\,u(k) \\ \mathbf{y}(k) &= \mathbf{C}_d\mathbf{x}(k) \end{aligned} \tag{8}$$

Where k represents the actual instant, $\mathbf{A}_d \in \mathbb{R}^{n.n}$ and $\mathbf{B}_d \in \mathbb{R}^{n.n_u}$ are discretised space state matrices from lateral dynamics (see Equation 4). A prediction map, used to

estimate the vehicle's future states, is obtained by computing the N sequences of upcoming actions. Thus, the n-step ahead control inputs $\tilde{u}(k)$ can be calculated as follows:

$$\tilde{u}(k) = (u(k) \quad u(k+1) \cdots u(k+N-1))^T \in \mathbb{R}^{N.n_u} \quad (9)$$

A prediction map for n-steps ahead allows estimating the future states trajectory $\tilde{x}(k|\tilde{u}(k))$ based on a sequence of future actions $\tilde{u}(k)$. Thus, the future state trajectory starting from $x(k)$ can be written as follows:

$$\tilde{x}(k|\tilde{u}(k)) = (x(k+1) \quad \cdots \quad x(k+N))^T \in \mathbb{R}^{N.n} \quad (10)$$

Given the matrices $\mathbf{A}_d$ and $\mathbf{B}_d$, a general expression for $\tilde{x}(k|\tilde{u}(k))$ can be written as:

$$x(k+i) = \Phi_i x(k) + \Psi_i \tilde{u}(k) \quad (11)$$

where the matrices Φ_i and Ψ_i are given by:

$$\mathbf{\Phi}_i = \mathbf{A}_d{}^i \quad (12)$$

$$\mathbf{\Psi}_i = \begin{pmatrix} \mathbf{A}_d{}^{i-1}\mathbf{B} & \cdots & \mathbf{A}_d\mathbf{B} & \mathbf{B} \end{pmatrix}^T \begin{pmatrix} \mathbf{\Pi}_1^{(n_u,N)} \\ \mathbf{\Pi}_2^{(n_u,N)} \\ \vdots \\ \mathbf{\Pi}_i^{(n_u,N)} \end{pmatrix} \quad (13)$$

Where $\mathbf{\Pi}_i^{(n_u,N)}$ is the matrix used to select the i^{th} part of dimension n_u from a vector containing the concatenation of N vectors and can be defined as follows:

$$\mathbf{\Pi}_i^{(n,N)} = \begin{pmatrix} 0_{n\times n(i-1)} & I_{n\times n} & 0_{n\times n(N-i)} \end{pmatrix} \quad (14)$$

LKAS MPC-based control aims to regulate the vehicle's lateral displacement (y) by following the target path. Similarly, the first state variables are the outputs to be tracked, regardless of the type of vehicle dynamics. Thus, the output equation of the system can be formalized as follows:

$$\mathbf{y}(k) = \mathbf{C}_r \mathbf{x}(k) \quad (15)$$

$$\mathbf{C}_r = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \quad (16)$$

Where $\mathbf{C}_r$ is the regulated output matrix.

A. COST FUNCTION

In MPC control theory, a cost function is a mathematical expression used to balance tracking accuracy and control effort. Typically, a cost function is employed to evaluate MPC performance over an n-step ahead prediction horizon [32]. The cost function used to design the LKAS MPC-based controller can be expressed as follows:

$$\begin{aligned} J(k) &= \sum_{i=1}^{N} ||y_r(k+i) - y_d(k+i)||^2_{Q_y} \\ &+ \sum_{i=1}^{N} ||\Pi_i^{(n_u,N)} \tilde{u}(k) - u_d||^2_{Q_u} \end{aligned} \quad (17)$$

Where Q_u and Q_y are the weighting matrices used to penalise control command (u) and the path error ($y_r - y_d$) respectively; u_d is the desired control command whilst y_d is the desired output from the regulated variable $y(k)$. Additionally, a cost function executed on each time step (k), over a prediction horizon ($[k, k+N]$), can be described as follows:

$$J(k) = \frac{1}{2}\tilde{\mathbf{u}}^T(k)\mathbf{H}\tilde{\mathbf{u}}(k) + \mathbf{F}^T(k)\tilde{\mathbf{u}}(k) + Cte \quad (18)$$

Where:

$$\begin{aligned} \mathbf{H} &:= 2\sum_{i=1}^{N}[\Psi_i^T C_r^T Q_y C_r \Psi_i + (\Pi_i^{(n_u,N)})^T Q_u (\Pi_i^{(n_u,N)})] \\ \mathbf{F(k)} &:= F_1 x(k) + F_2 y_d + F_3 u_d \\ \mathbf{F_1} &:= 2\sum_{i=1}^{N}[\Psi_i^T C_r^T Q_y C_r \Phi_i] \\ \mathbf{F_2} &:= -2\sum_{i=1}^{N}[\Psi_i^T C_r^T Q_y \Pi_i^{(n_r,N)}] \\ \mathbf{F_3} &:= 2\sum_{i=1}^{N}[(\Pi_i^{(n_u,N)})^T Q_u] \end{aligned} \quad (19)$$

$\mathbf{F_1}$, $\mathbf{F_2}$, and $\mathbf{F_3}$ are offline matrices obtained from system modelling and selected prediction horizons. $\mathbf{F}$ may change in accordance with state vector $x(k)$ and the desired output reference (y_d) and must be updated on each time step (k). Additionally, $\mathbf{H} \in \mathbb{R}^{Nnu.Nnu}$ is the positive definite Hessian of the quadratic function which guarantees a well-posed optimization problem [31]. The minimum value found over the sequence $\tilde{\mathbf{u}}(k)$ is not affected by Cte term.

B. CONSTRAINTS

One of the most significant advantages of MPC-based solutions is their ability to handle control limits. Control limits are restrictions imposed by physical systems and functions that must be considered by control laws. For lane-keeping applications, the most robust restriction involves keeping the vehicle within its road lane. Any LKAS MPC-based solution must include the maximum lateral displacement allowed while tracking the desired vehicle trajectory. Lateral displacement can be bounded as follows:

$$\mathbf{y_c^{min}} < \mathbf{y_c} < \mathbf{y_c^{max}} \quad (20)$$

$$\mathbf{y}_c = \mathbf{C}_c\mathbf{x} \quad ; \quad \mathbf{C}_c = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \quad (21)$$

Where $\mathbf{y_c^{min}}$ and $\mathbf{y_c^{max}}$ represent lateral displacement boundaries, $\mathbf{y}_c$ and $\mathbf{C}_c$ are the output vector and the matrix constraints applied to lateral vehicle displacement respectively. In LKAS, the target path (i.e., the path to be followed) is determined by using camera technologies. Through pre-processed and filtered road images, a target path (set-point) is generated in real time and must be followed by the LKAS MPC-based controller. Hence, the lateral displacement limits, including the target path, can be expressed as:

$$\begin{aligned} \mathbf{y}_c^{min} &= -(D_r/2) + (D_v/2) \\ \mathbf{y}_c^{max} &= (D_r/2) - (D_v/2) \end{aligned} \tag{22}$$

D_r and D_v are the road and vehicle width respectively. Another crucial constraint is related to the driver's steering command $u(k)$. The steering command, which serves as the control input for the LKAS MPC-based controller, must fall within a range of pre-defined maximum turning angles. Thus, the driver steering command must be restricted by the following limits:

$$\mathbf{u^{min}} < \mathbf{u} < \mathbf{u^{max}} \tag{23}$$

Where $\mathbf{u^{min}}$ and $\mathbf{u^{max}}$ are the minimal and maximal allowed inputs for the steering system. The rate of change of control variables plays an important role in vehicle dynamics. It is worth mentioning that abrupt modification in control variables, such as the steering angle, must be avoided for comfort and safety reasons. The restriction applied to the rate of change of control variables (Δu) can be formalised by the following:

$$\mathbf{\Delta u^{min}} < \mathbf{u(k+i)} - \mathbf{u(k+i-1)} < \mathbf{\Delta u^{max}} \tag{24}$$

Where $\mathbf{\Delta u^{min}}$ and $\mathbf{\Delta u^{max}}$ are the input rate of change constraints. Combining the equations 20-24, the LKAS vehicle trajectory restraint can be written in a compact matrix form as follows:

$$\begin{pmatrix} \mathbf{y_c^{min}} \\ \mathbf{u^{min}} \\ \mathbf{\Delta u^{min}} \end{pmatrix} < \begin{pmatrix} \mathbf{y_c(k+i)} \\ \mathbf{u(k+i-1)} \\ \mathbf{u(k+i)} - \mathbf{u(k+i-1)} \end{pmatrix} < \begin{pmatrix} \mathbf{y_c^{max}} \\ \mathbf{u^{max}} \\ \mathbf{\Delta u^{max}} \end{pmatrix} \tag{25}$$

Where $i \in \{1, \ldots, N\}$ represents the intermediate values within the entire prediction horizon (N). Once the vehicle trajectory restraint matrix has been determined, an LKAS cost function can be written by combining the equations 17 and 25 as follows:

$$\begin{aligned} \mathbf{\tilde{u}^{opt}}(k): \quad = \quad & \arg\min_{\mathbf{u}} \left[\frac{1}{2} \mathbf{\tilde{u}}^T \mathbf{H} \mathbf{\tilde{u}} + \mathbf{F}^T(k) \mathbf{\tilde{u}} \right] \\ & \text{subject to:} \\ & \mathbf{A_{ineq}} \mathbf{\tilde{u}} \leq \mathbf{B_{ineq}}(k) \\ & \mathbf{\tilde{u}}^{min} \leq \mathbf{\tilde{u}} \leq \mathbf{\tilde{u}}^{max} \end{aligned} \tag{26}$$

Where $\mathbf{A_{ineq}}$ and $\mathbf{B_{ineq}}(k)$ are vehicle states and rate of change constraint matrices included in the cost function. A complete description of the formulation of $\mathbf{A_{ineq}}$ and $\mathbf{B_{ineq}}(k)$ can be found in [31].

C. MPC PARAMETERIZATION

One of the most significant challenges for MPC control applications is the computing time needed to obtain the sequence of output commands. Since MPC calculates an optimized control output for a constrained dynamic system at each time step, it imposes a significant overhead, sometimes leading to computational burden [33] [34]. Parameterization methods arise as a potential solution to make MPC feasible in real-time applications [31]. This method reduces the number of degrees of freedom ($N.n_u$) from the optimization process by keeping a constant control profile over intermediate periods from the prediction horizon. Figure 2 highlights trivial and exponential methods for MPC parameterization.

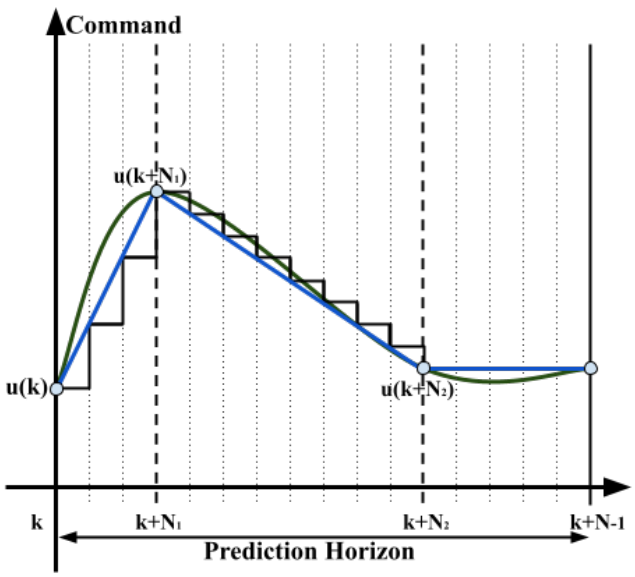

FIGURE 2: Trivial (blue line) and exponential (green line) parameterization over the entire prediction horizon for three-time intervals ($n_r = 3$). Adapted from [31].

In the trivial parameterization (TP), control variables are interpolated and maintained constant in some specific samples to diminish the size of n_p. Furthermore, intermediate values are obtained through linear interpolation. The search space from the optimization process is reduced from $N \times n_u$ to $n_r \times n_u$ (where n_r is the number of intervals regarding the trivial parameterization). With trivial parameterization, the decision variable dimension can be written as follows:

$$p = (u(k) \quad u(k+N_1) \cdots u(k+N_{n_r-1}))^T \tag{27}$$

Exponential parameterization (EP) is another strategy used to improve computational efficiency [32] by reducing the complexity of optimization problem [31]. Unlike TP, this technique represents a control sequence (u) using an exponential term [35]. The main benefit of this parameterization approach lies in its capability to produce a mathematical function through the combination of exponentials, allowing for the direct adjustment of coefficients.

Considering n_u actuators in the automotive system, a settling time vector ($\tau_r \in \mathbb{R}^{n_u}$) defines the time required for each actuator to reach the reference within a given tolerance band. This restriction avoids a sudden modification

in control commands or even exceeding the actuator's speed limits. Thus, the exponential control profile can be defined as follows:

$$u_j(k+i) = \sum_{l=1}^{n_e^{(j)}} \underbrace{\left[e^{\frac{-\lambda_j (i\tau)}{(l-1)\alpha+1}} \right]}_{m_{j,l}(i)} \cdot p_l^{(j)}; \quad \alpha > 1 \tag{28}$$

In the provided equation, $\alpha > 1$ and λ are the tuning parameters that can be defined based on the settling time of the actuators. To avoid undesired oscillations, one can set λ as $\frac{3}{\tau_r}$, where τ_r represents the desirable settling time. This choice ensures that the control signals exhibit a 95% response time with respect to the actuators' limits. Thus, the resulting control profile is defined with the vector p given by:

$$p = \begin{pmatrix} p^{(1)} \in \mathbb{R}^{n_e^{(1)}} \\ \vdots \\ p^{(n_u)} \in \mathbb{R}^{n_e^{(n_u)}} \end{pmatrix} \in \mathbb{R}^{n_p} \tag{29}$$

Where n_e is the number of exponentials in the control sequence with the dimension of vector p is defined such as:

$$n_p = \sum_{j=1}^{n_u} n_e^{(j)} \tag{30}$$

The new set of n_p is now determined solely by the number of actuators n_u and exponentials $n_e^{(j)}$ selected, independent of the prediction horizon that has been used. Consequently, the control sequence can be expressed in a compact form as follows:

$$u_j(k+i) = \sum_{l=1}^{n_e^{(j)}} [m_{j,l}(i)] \cdot p_l^{(j)} \tag{31}$$

which can be formalized as a product matrix of the form:

$$u_j(k+i) = [M_j(i)] \cdot p_l^{(j)}; \quad p^{(j)} \in \mathbb{R}^{n_e^{(j)}} \tag{32}$$

where $M_j(i) \in \mathbb{R}^{n_e^{(j)}}$ is defined by:

$$M_j(i) = (m_{j,1}(i) \cdots m_{j,n_e^{(j)}}(i)) \tag{33}$$

Once $M_j(i)$ can be written as a pre-computed offline vector, the decision variable p can be obtained through a straightforward transformation of Equation 33, for $j = 1, ..., n_u$. The expression of the input vector $u(k+i)$ at a future time instant $k+i$ can be represented as follows:

$$u(k+i) = \underbrace{BlockDiag\left(M_j(i)_{j=1}^{n_u}\right)}_{M(i)} \begin{pmatrix} p^{(1)} \\ \vdots \\ p^{(n_u)} \end{pmatrix}$$
$$u(k+i) = [M(i)] \tag{34}$$

By concatenating the above equation for $i = 0, \ldots, N-1$, the expression for the exponential parameterization Π_e can be formalized as follows:

$$\tilde{u} = \Pi_e \cdot p; \tag{35}$$

Where Π_e is the exponential selection matrix defined as:

$$\Pi_e := \begin{pmatrix} M(0) \\ \vdots \\ M(N-1) \end{pmatrix} \tag{36}$$

To reduce the computational burden, a new cost function (J) is redefined by incorporating the new control variable (p) and the matrix (Π_e), leading to the following formulation:

$$J(p) = \frac{1}{2} p^T \left(\Pi_e^T \mathbf{H} \Pi_e\right) p + \left(\Pi_e^T \mathbf{F}\right)^T p \tag{37}$$

subject to a new set of constraints defined as follows:

$$\mathbf{A_r} p \leq \mathbf{B_r}(k); \quad \tilde{\mathbf{u}}^{\mathbf{min}} \leq \Pi_e . p \leq \tilde{\mathbf{u}}^{\mathbf{max}} \tag{38}$$

Where $\mathbf{A_r}$ and $\mathbf{B_r}(k)$ are the reduced inequalities matrices of the exponential parameterization which represents the constraints definition of the optimization problem (26) being formalized as:

$$\mathbf{A_r} = \begin{pmatrix} \mathbf{A_{ineq}} . \Pi_e \\ -\Pi_e \\ +\Pi_e \end{pmatrix} \quad ; \quad \mathbf{B_r}(k) = \begin{pmatrix} \mathbf{B_{ineq}}(k) \\ -\tilde{\mathbf{u}}^{\mathbf{min}} \\ +\tilde{\mathbf{u}}^{\mathbf{max}} \end{pmatrix} \tag{39}$$

D. OPTIMIZATION ALGORITHMS

Optimization algorithms often combine the optimization problem and the cost function to find a feasible solution at each control interval. These algorithms enable real-time execution of MPC, which is crucial for automotive control applications. Four optimization algorithms were investigated to design and implement the LKAS MPC-based controllers:

- Interior Point Methods
- Active Set Methods
- Alternating Direction Methods of Multipliers
- Newton Method

Interior Point Methods (IPMs) are widely used to solve convex optimization problems with inequality constraints. IPMs typically model the problem constraints using parameterized penalty functions, also known as barrier functions [37]. These methods promote the convergence of the optimization algorithm, regardless of the type of conditions, data, and size. Using IPMs, unconstrained optimization problems with different barrier functions can be solved at each interaction until a stable point is reached. IPMs can be subdivided into primal barriers and primal-dual barriers. The former replaces the inequality constraints of the quadratic programming (QP) problem, including an objective-weighted barrier function. The latter combines an inner and outer loop of primal-barrier by reducing the barrier weight at each iteration of Newton's method [36]. Besides their strengths, IPMs are

easily warm-started and do not have an appropriate scale for large problems.

Active Set Methods (ASMs) establish a working set to solve the resulting QP problem with equality constraints. The selected working set, a linearly independent subset of active constraints, is continuously updated until optimal values are found. ASMs are classified as primal, dual, and parametric methods. Primal methods produce a sequence of feasible primary interactions until dual feasibility with an optimal solution is reached. Similarly, dual methods generate a sequence of dual feasible interactions until primal feasibility is reached with an optimal solution. In the strict convex case, it is equivalent to solving the dual of the QP problem with a primal active set method [39]. On the other hand, parametric methods involve tracking one solution to linear homotopy. The linear homotopy, parameterized by $t_a \in [0, 1]$, is often used to transition between a Quadratic Programming (QP) problem with a known solution $(t_a = 0)$ to one that needs to be solved $(t_a = 1)$ [36]. Although ASMs are recognized for fast convergence and can be hot-started in small QP problems, they are quite complex to implement, may not be robust for early termination, and can be difficult to translate from theory to practice. Current applications are commonly found in small and medium-sized problems [40].

Alternating Direction Methods of Multipliers (ADMMs), also known as First-Order Methods (FOMs), compute an optimal solution using the first-order information about the cost function. Since FOMs solely use subgradient information, they can easily be hot-started, leading to achieving fast run rates. When constraints are simple, projected gradient methods become attractive due to their simplicity and the availability of tight complexity bounds. Several FO methods solve the dual problem by using gradient ascent or multiplicative updates, for instance, [40]. Despite the advances from FOMs, they fail to detect primal or dual infeasibility. Furthermore, convergence in FOMs is highly dependent on the data and the input from the algorithm step size parameter [38]. Moreover, FOMs provide slower asymptotic convergence compared to IPMs and ASMs.

Newton's Method (NM) performs an approximation of two differentiable functions using a second-order Taylor expansion. NM searches for a minimal value of the approximate function interactively [41]. This method is based on successive objective function approximations (local quadratic function) and solving the resultant quadratic problem. Some benefits of NM include quadratic convergence and error reduction on each algorithm interaction. Nonetheless, NM often requires the calculation of a Hessian matrix, becoming expensive and difficult to obtain in some cases. Yet, this method could not achieve global convergence if the initial point is not close enough to the optimal value [42].

The abovementioned optimisation algorithms have been used for decades in many types of applications. In addition, some variants have also been proposed to implement real-time systems. The Quadratic Problem (QP) solvers investigated in this research are listed in Table 1:

TABLE 1: List of Quadratic Problem Solvers

Quadratic Programming Solver	Optmization Algorithm IPM	ASM	ADMM	NM	Ref.
QPSWIFT	X				[37]
MpcActive SetSolver		X			[43]
QuadProg		X			[44]
QPOASES		X			[36]
OSQP			X		[38]
FBstab				X	[40]

Legend: IPM - Interior Point Method; **ASM** - Active Set Method; **ADMM** - Alternative Direction Methods of Multipliers; **NM** - Newton's Method

V. EXPERIMENTATION

Experiments were conducted to evaluate MPC-based controllers designed for the LKAS system. The selected scenarios focused on assessing computing efficiency and control performance through the evaluation of classical, trivial and exponential parameterized MPC. The performance of various QP algorithms was also considered. Additionally, a comparative analysis involving MPC and LQR-based controllers was performed to examine LKAS operating restrictions. Simulation experiments were carried out using Matlab/Simulink 2022a on a PC with an Intel Core i7-9700 3.0GHz processor and 16 GB of RAM. The Creative Driving Scenario from Matlab was utilized to generate the target path. A sensor camera mounted in front of a virtual vehicle captured traffic scene data, including pavement marks, speed limit boards, etc. Traffic scene data is then used to compute the vehicle lateral displacement limits. Simulation input parameters are defined in Table 2:

TABLE 2: LKAS System and MPC controller parameters

Description	Parameter	Value	Unit
Lane Keeping Assist System	m	1,573	Kg
	I_z	2,873	Kg.m^2
	a	1.1	m
	b	1.58	m
	$C_{\alpha f}$	80,000	N/m
	$C_{\alpha r}$	80,000	N/m
	D	1,795	m
	V_x	166,667	m/s
MPC-based controller	N	10	deg
	u^{max}	25	deg
	u^{min}	-25	m
	y_c^{max}	-$(D_r/2)$ + $(D_v/2)$	m
	y_c^{max}	$(D_r/2)$ - $(D_v/2)$	deg/s
	Δu^{max}	5	deg/s
	Δu^{min}	5	deg/s

A. CLASSICAL MPC

Reference tracking for the LKAS MPC-based controller using the sideslip bicycle model is shown in Figure 3. Experimental results highlight that the vehicle's lateral displacement follows the targeted path, keeping the vehicle within the road lane. Similarly, the steering angle and its rate of change were held within predefined intervals, except at the beginning due to the initial state variable conditions $x_0 = [0.5; 0; 0; 0]$.

A similar experiment was carried out using the lateral speed bicycle model (see Equation 6). Again, the LKAS MPC-based controller follows the target reference throughout the entire experiment (see Figure 3). It is worth mentioning that the steering wheel angle profile shows a greater oscillation amplitude when compared to the output obtained from the previous experiment. A large variation in the steering angle rate of change could potentially diminish the lifetime of the steering actuator and the driving comfort of the LKAS system. Unlike the previous experiment outcomes, the steering angle rate of change was reduced, suggesting more comfortable driving conditions.

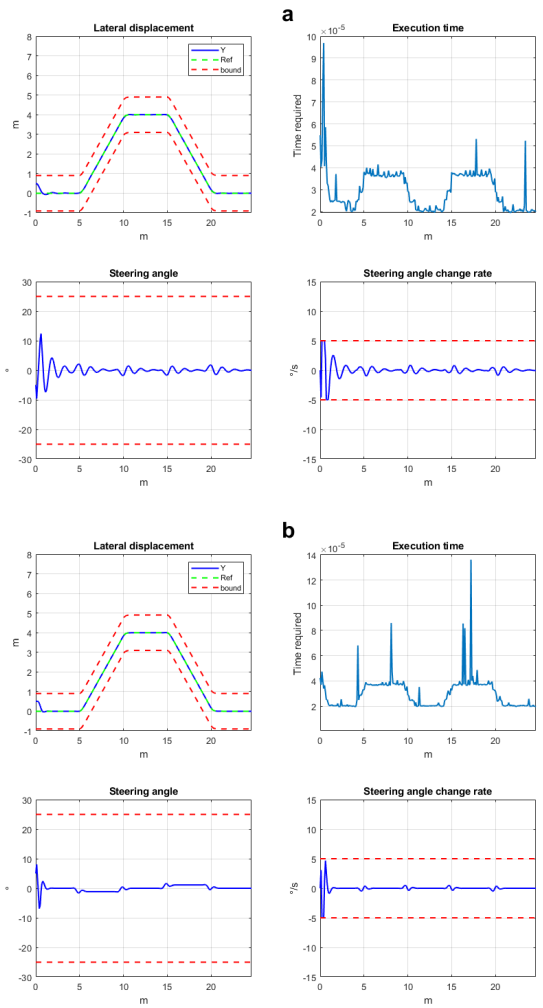

FIGURE 3: LKAS MPC-based controller performance with QPSWIFT and a) sideslip and b) lateral speed bicycle model.

An LQR controller was also considered to compare MPC and LQR performances. Additionally, the lateral speed bicycle model, using the same conditions and inputs, was employed to conduct another experiment. A tracking control test was executed with continuous monitoring of lateral displacement, steering angle, and its rate of change, as well as MPC execution time. Figure 4 shows the results obtained from this experiment. The LQR controller successfully tracks the reference with an average computing time of about $4.66\mu s$. This suggests that the LQR controller can be used in real-time applications and commercial ECUs. From the LKAS operating restraint perspective, the steering angle achieved a minimum and maximum value of [-180 to 120 deg], exceeding the operating restraint. Moreover, the rate of change of the steering angle achieved 150/sec, surpassing the physical and comfort limits imposed by the LKAS system. Whenever the output command exceeds physical limits, the LQR controller command variable saturates, degrading the reference tracking performance, as well as the lifetime of actuators.

Another experiment was conducted to determine whether distinct QP solvers may influence MPC performance. The lateral speed bicycle model was utilized with the same input data. The MPC prediction horizon (N) was set to 10 steps ahead. Prior analysis revealed that greater values severely degraded MPC computing performance. Steering command (Q_u) and reference tracking (Q_y) penalties were used to balance steering command amplitude against target path tracking accuracy. Q_u and Q_y values were selected to achieve a desired command profile. The maximum number of iterations (Max Iter) was determined through trial and error until each QP found an optimal solution. The tolerance value allowed slight changes in QP objective functions between successive iterations. QP solver input parameters are detailed in Table 3. Steering angle, lateral displacement error, and computation time were recorded throughout the entire experiment, with the simulation results presented in Figure 5.

TABLE 3: QPs input data for classical MPC.

Parameter	QP[1]	QP[2]	QP[3]	QP[4]	QP[5]	QP[6]
Qy	1E+03	1E+02	1E+03	1E+03	1E+02	1E+03
Qu	1E+03	1E+03	1E+03	1E+03	1E+02	1E+03
N	10	10	10	10	10	10
Max Iter	50	100	100	60	150	80
Tolerance	1.E-08	1.E-06	1.E-09	1.E-08	1.E-06	1.E-08
Avg time	3.E-05	2.E-04	8.E-05	5.E-04	7.E-05	2.E-04

Legend: **QP[1]** - QPSWIFT; **QP[2]** - MpcActive SetSolver; **QP[3]** - QPOASES; **P[4]** - QUADPROG; **QP[5]** - OSQP and **QP[6]** - Fbstab

The experiment output highlights satisfactory performance for both QP solvers, with no significant difference in reference tracking. Despite similar tracking responses, the computing time to calculate the u command varies, suggesting a potential violation of real-time requirements. QPswift provides the minimal average computational time ($29.55\mu s$), followed by OSQP ($67.14\mu s$). On the other hand, FBstab and Quadprog showed the worst performance ($2.29ms$ and $5.30ms$), respectively. Moreover, MPCActive SetSolver, OSQP, and QPswift computing time may increase whenever the tracking reference undergoes continuous or abrupt changes. Simulation experiments demonstrate that QP solver computing time is sensitive to the weighting terms. Experiments also revealed that excessive steering command penalization could lead to performance loss when tracking lateral displacement reference.

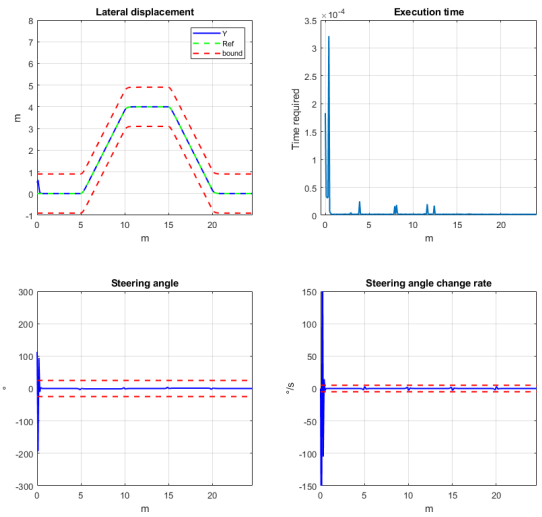

FIGURE 4: LKAS LQR-based controller performance with lateral speed bicycle model.

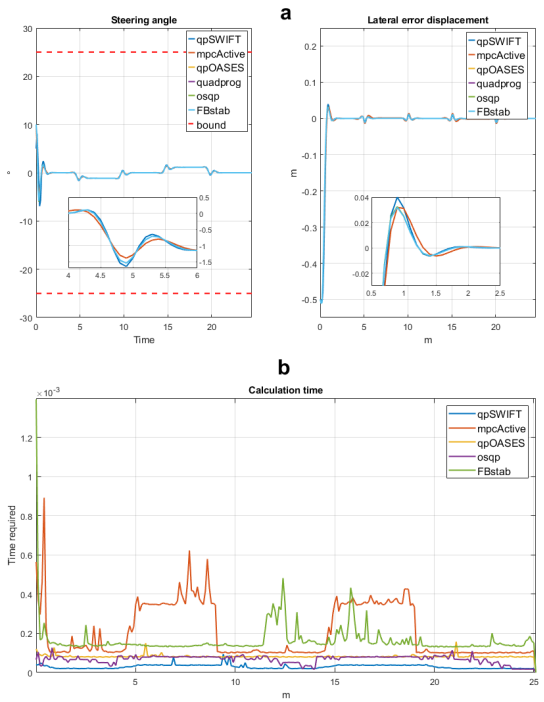

FIGURE 5: Experiment results with classical MPC controller a) Steering angle and lateral displacement error and b) QP solvers computational time.

B. TRIVIAL PARAMETERIZED MPC

Aiming to reduce the classical MPC computational time, a trivial parameterization technique was considered. In this approach, a control sequence is interpolated and kept constant in specific samples, reducing the degree of freedom and computational time required. To determine the initial settings, the QP configuration data for classical MPC were used as a baseline and then tuned interactively until a similar steering angle profile was verified. Table 4 highlights input data for simulation experiments.

TABLE 4: QPs input data for trivial MPC.

Parameter	QP[1]	QP[2]	QP[3]	QP[4]	QP[5]
Qy	1E+03	1E+02	1E+02	1E+02	1E-02
Qu	1E+01	1E+02	1E+02	1E+01	5E-05
DOF	[5;10]	[2;5;9]	[3;5;7]	[5;8]	[2;9]
N	10	10	10	10	10
Max Iter	50	100	100	150	80
Tolerance	1E-08	1E-06	1.E-09	1E-06	1.8E-08
Avg time	1.2E-05	1.E-04	6.2-05	5.5E-05	1.8E-04

Legend: **QP[1]** - QPSWIFT; **QP[2]** - MpcActive SetSolver; **QP[3]** - QPOASES; **QP[4]** - OSQP and **QP[5]** - Fbstab

Figure 6 shows the tracking performance capability, including LKAS restraint violation analysis. The simulation results illustrate a steering angle profile similar to the classical MPC response (see Figure 5). Once again, the magnitude of the steering angle and its rate of change were maintained within predefined limits. Furthermore, there is no significant deviation in lateral displacement error, suggesting satisfactory QP performance in tracking the targeted reference.

Figure 7 shows the QPs computational time required by trivial parameterized MPC. For better visualization, the data output for *quadprog* and *FBstab* were omitted. The time required for *FBstab* to compute a solution reduced from 2.0×10^{-4} to 1.0×10^{-8} s with no significant tracking performance variation. Similarly, MPCActive SetSolver was also reduced from 2.0×10^{-4} to 1.0×10^{-4} s with no significant tracking performance variation. QPswift was the best QP solution with the lowest average computational time of $17.08\mu s$.

C. EXPONENTIAL PARAMETERIZED MPC

Exponential parameterized MPC was also considered for computational performance analysis. Similarly, the QPs baseline data from classical MPC were tuned interactively to obtain a smooth input command profile (u) within the desired response time (t_r). The number of exponential (NoE) and alpha (α) parameters were used to tune the exponential parameterized technique. Appropriate selection of these parameters may reduce the complexity of the optimization problem with no significant modification to the control-loop performance [31]. Once again, the prediction horizon (N) received the same value used in previous experiments. QPs input data are shown in Table 5. Simulation results obtained with the lateral speed bicycle model are shown in Figure 8.

Experiment outcomes revealed an appropriate command profile where the steering angle and its derivate accomplished predefined command restraints. Although lateral displacement error increased with *QPOASES* and *Fbstab*, no relevant tracking performance deviation was observed. Thus, the exponential parameterization scheme can, in turn, replace classical MPC without closed-loop performance degradation. Computational time obtained with exponential MPC is

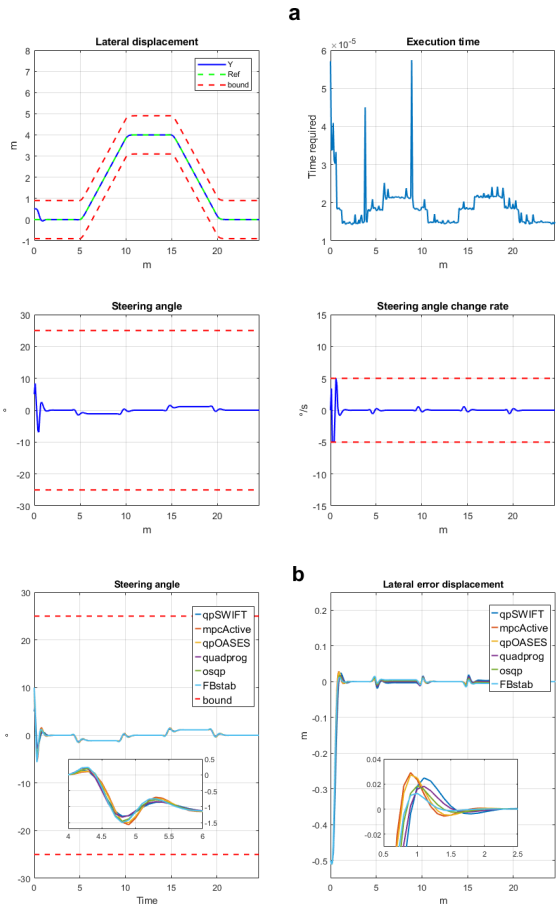

FIGURE 6: Simulation results for trivial parameterized MPC including a) Reference tracking performance test and b) Input command with lateral displacement error.

TABLE 5: QPs input data for exponential MPC.

Parameter	QP[1]	QP[2]	QP[3]	QP[4]	QP[5]
Qy	1E+02	1E+01	1E+02	1E+02	2.E-01
Qu	1E+02	1E+01	1E+02	1E+01	1.E+1
NoE	2	2	2	2	2
α	25	150	15	15	125
t_r	0.01	0.01	0.05	0.05	0.05
N	10	10	10	10	10
Max Iter	50	100	100	150	80
Tolerance	1E-08	1E-06	1.E-09	1E-06	1.2E-04
Avg time	1.4E-05	9.1E-05	4.0-05	4.3E-05	1E-08

Legend: **QP[1]** - QPSWIFT; **QP[2]** - MpcActive SetSolver; **QP[3]** - QPOASES; **QP[4]** - OSQP and **QP[5]** - Fbstab

shown in Figure 9. QP solvers with exponential MPC showed the lowest execution time with no significant difference in input command (u). Additionally, peak values from computational time occur less frequently when compared to previous solutions. Simulation output illustrates a remarkable average computational time reduction from those aforementioned strategies.

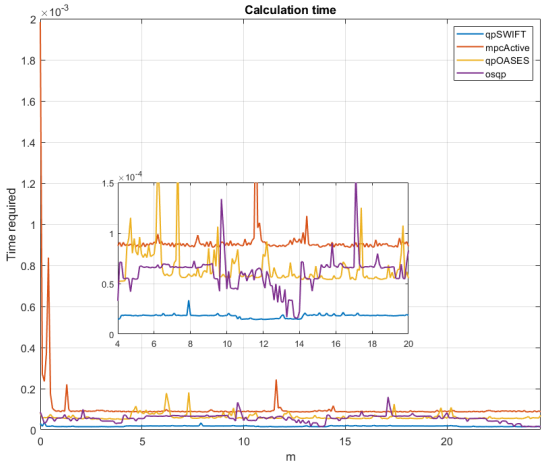

FIGURE 7: Computational time required by trivial parameterized MPC.

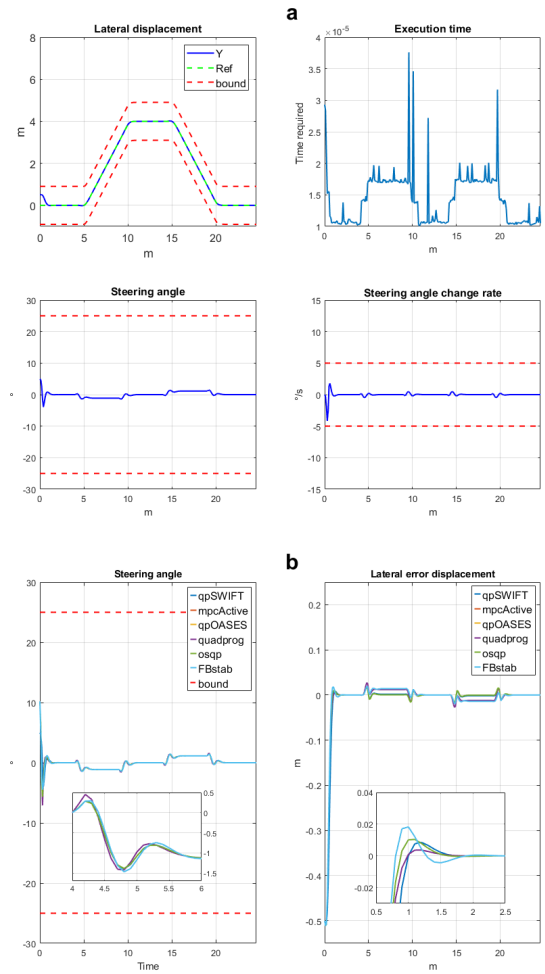

FIGURE 8: Simulation results obtained with exponential MPC including a) Tracking performance test and b) Input command with lateral displacement error.

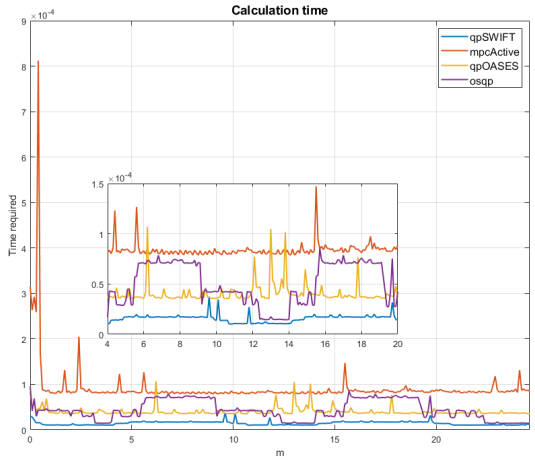

FIGURE 9: Computational time required by exponential parameterized MPC.

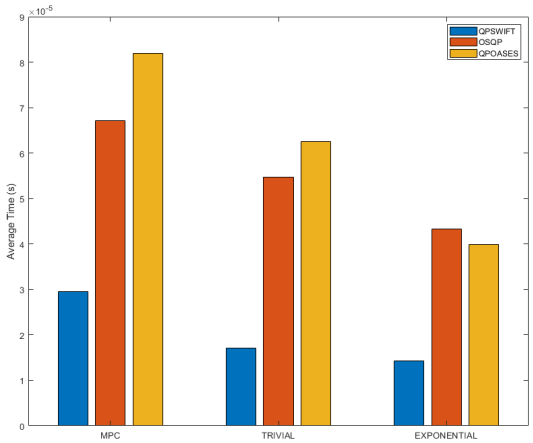

FIGURE 11: Greatest alternatives candidates for QPs.

VI. COMPARATIVE ANALYSIS

A comparative analysis was carried out to evaluate the LKAS MPC-based solutions including classical, trivial and exponential MPC with distinct QP solvers. Since classical and parameterized MPC provided analogous input command profiles, this section mainly considers real-time application issues. Figure 10 shows the average computational time (ACT) across various MPC-based solutions and their corresponding QPs. For classical, trivial, and exponential MPC, the QPSWIFT achieved the lowest execution time once compared to other QPs. The best combination between MPC and QPs is the exponential parameterized MPC with QPSWIFT which achieved the lowest execution time of $(14.26\mu s)$.

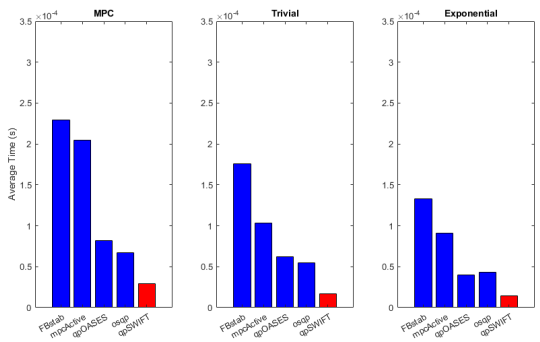

FIGURE 10: Comparing execution time for all QPsolvers

Figure 11 highlights the top three QPs in terms of computational performance. Once again, exponential MPC provides the best solution to implement an LKAS MPC-based controller. Exponential MPC using QPSWIFT achieved the lowest average computational time $(14.26\mu s)$. Likewise, QPOASES is the next alternative $(39.90\mu s)$ whilst OSQP achieved the third position $(43.24\mu s)$. Despite exponential MPC with QPSWIFT becoming the greatest solution to implement LKAS MPC-based controllers, there is no guarantee that QPSWIFT implementation will be available for a particular ECU. For this reason, it is crucial to combine alternative solutions to overcome application issues. Table 6 shows a detailed comparison of classical, trivial, and exponential MPC and QPs for real-time application.

TABLE 6: Comparison of the different strategies in the reduction of average computational time.

QP Solver	Classical vs Trivial	Trivial vs Exponential	Classical vs Exponential
QPSWIFT	42%	17%	52%
MpcActive SetSolver	49%	12%	56%
QPOASES	24%	36%	51%
OSQP	19%	21%	36%
FBstab	23%	30%	46%

According to the data presented in the aforementioned table, the QPs with the highest computational time reduction (56%) was the MPCActive SetSolver when exponential MPC replaced classical MPC. On the other hand, the QPs with the lowest reduction (19%) was the OSQP from classical to trivial MPC. Across all experiment scenarios, a significant reduction in computation time was verified whenever trivial or exponential parameterization techniques were employed. Experiment results showed that exponential MPC with QPswift appears to be the best candidate to implement an LKAS MPC-based solution. Furthermore, exponential parameterization reduces the computation burden significantly for all QPs. Moreover, this reduction was achieved without restraint violation and closed-loop control performance loss. This research demonstrates that the reduction of computation time in MPC-based strategies can be achieved through a strategic combination of an efficient solver and a well-designed theoretical approach.

VII. CONCLUDING REMARKS AND FUTURE WORK

This paper investigated model predictive control techniques to address the LKAS control problem. Three distinct MPC strategies were formulated and subsequently evaluated, with each strategy being employed with six different quadratic optimization problem-solving algorithms for a comprehensive analysis. This study significantly contributes to the current literature by systematically addressing the issue of minimizing computational costs for the practical implementation of MPC-like strategies in real ECUs. The effectiveness of the MPC parameterization technique was demonstrated through the reduction of computational burden, addressing the real-time and LKAS constraints. Additionally, the performance evaluation of the parameterized MPC strategy was carried out using widely recognized QP-solvers designed for the linear MPC formulation.

As a result, the average computation time experienced a substantial reduction for all simulation scenarios. In the MpcActive SetSolver algorithm, a reduction of 49% for the trivial parameterization technique was observed, and a 56% reduction when the exponential parameterization technique was employed. The QPSWIFT QP solver showcased the most favourable average computational time for the MPC controller with $(14.26\mu s)$, and by implementing the exponential parameterization technique, a 52% decrease in computation time was achieved compared to the standard MPC scheme. The time reduction strategies effectively preserved the intended steering angle command profile, and constraints were correctly handled. Furthermore, the significance of selecting an appropriate model for the controller was underscored, as it may influence controller performance and the desired vehicle response. The ability to keep the trajectories within imposed constraints was also successfully demonstrated. As a recommendation for future work, the next steps suggest integrating the QP solver into real-time simulations using an embedded system. This could involve testing on a real prototype to further validate the parameterized MPC scheme and enhance the practical applicability of the proposed approaches.

REFERENCES

[1] Chen, B., Luan, B. & Lee, K. Design of lane keeping system using adaptive model predictive control. 2014 IEEE International Conference On Automation Science And Engineering (CASE). pp. 922-926 (2014)

[2] Rosen, H., Bari, I., Paichadze, N., Peden, M., Khayesi, M., Monclús, J. & Hyder, A. Global road safety 2010–18: an analysis of global status reports. Injury. (2022)

[3] Xu, X., Grizzle, J., Tabuada, P. & Ames, A. Correctness guarantees for the composition of lane keeping and adaptive cruise control. IEEE Transactions On Automation Science And Engineering. **15**, 1216-1229 (2017)

[4] Dai, S. & Koutsoukos, X. Safety analysis of integrated adaptive cruise and lane keeping control using multi-modal port-Hamiltonian systems. Nonlinear Analysis: Hybrid Systems. **35** pp. 100816 (2020)

[5] Liang, J., Yang, Y., Zhu, X. & Sheng, K. Realization of Emergency Lane Keeping System by Adaptive Control based on the Finite State Machine. 2021 5th CAA International Conference On Vehicular Control And Intelligence (CVCI). pp. 1-6 (2021)

[6] Xu, S., Peng, H., Lu, P., Zhu, M. & Tang, Y. Design and experiments of safeguard protected preview lane keeping control for autonomous vehicles. IEEE Access. **8** pp. 29944-29953 (2020)

[7] Sentouh, C., Nguyen, A., Benloucif, M. & Popieul, J. Driver-automation cooperation oriented approach for shared control of lane keeping assist systems. IEEE Transactions On Control Systems Technology. **27**, 1962-1978 (2018)

[8] Nguyen, A., Sentouh, C. & Popieul, J. Driver-automation cooperative approach for shared steering control under multiple system constraints: Design and experiments. IEEE Transactions On Industrial Electronics. **64**, 3819-3830 (2016)

[9] Parkash, A. & Swarup, A. Control of autonomous vehicle for lateral dynamics using sliding mode and input-to state stability methods. 2020 International Conference On Emerging Trends In Information Technology And Engineering (ic-ETITE). pp. 1-7 (2020)

[10] Reddy, N., Farah, H., Huang, Y., Dekker, T. & Van Arem, B. Operational design domain requirements for improved performance of lane assistance systems: A field test study in The Netherlands. IEEE Open Journal Of Intelligent Transportation Systems. **1** pp. 237-252 (2020)

[11] Guo, L., Wei, L., Ge, P. & Qin, Z. Lane Keeping Controller based on LQR Optimized by Genetic Algorithm. 2022 6th CAA International Conference On Vehicular Control And Intelligence (CVCI). pp. 1-6 (2022)

[12] Saraçoğlu, K., Üleş, B. & Schmidt, K. A lane keeping system with a weighted preview measurement. 2018 26th Signal Processing And Communications Applications Conference (SIU). pp. 1-4 (2018)

[13] Nguyen, A., Sentouh, C. & Popieul, J. Fuzzy steering control for autonomous vehicles under actuator saturation: Design and experiments. Journal Of The Franklin Institute. **355**, 9374-9395 (2018)

[14] Chen, Z., Li, L. & Huang, X. Building an autonomous lane keeping simulator using real-world data and end-to-end learning. IEEE Intelligent Transportation Systems Magazine. **12**, 47-59 (2018)

[15] Yan, N., Liu, C. & Sun, Q. A Novel Incremental Control Method Based on H-infinity Control for Lane Keeping Assist. 2021 5th CAA International Conference On Vehicular Control And Intelligence (CVCI). pp. 1-6 (2021)

[16] Salt Ducajú, J., Salt Llobregat, J., Cuenca, Á. & Tomizuka, M. Autonomous ground vehicle lane-keeping LPV model-based control: Dual-rate state estimation and comparison of different real-time control strategies. Sensors. **21**, 1531 (2021)

[17] An, Q., Cheng, S., Li, L. & Peng, H. Novel dual-layer-oriented strategy for fully automated vehicles' lane-keeping system. IET Intelligent Transport Systems. **14**, 1778-1787 (2020)

[18] Farag, W. Real-time NMPC path tracker for autonomous vehicles. Asian Journal Of Control. **23**, 1952-1965 (2021)

[19] Rathai, K., Amirthalingam, J. & Jayaraman, B. Robust tube-MPC based lane keeping system for autonomous driving vehicles. Proceedings Of The Advances In Robotics. pp. 1-6 (2017)

[20] Liu, C., Carvalho, A., Schildbach, G. & Hedrick, J. Stochastic predictive control for lane keeping assistance systems using a linear time-varying model. 2015 American Control Conference (ACC). pp. 3355-3360 (2015)

[21] Lee, J., Choi, J., Yi, K., Shin, M. & Ko, B. Lane-keeping assistance control algorithm using differential braking to prevent unintended lane departures. Control Engineering Practice. **23** pp. 1-13 (2014)

[22] Kim, W., Son, Y. & Chung, C. Torque-overlay-based robust steering wheel angle control of electrical power steering for a lane-keeping system of automated vehicles. IEEE Transactions On Vehicular Technology. **65**, 4379-4392 (2015)

[23] Nguyen, A., Chevrel, P. & Claveau, F. On the effective use of vehicle sensors for automatic lane keeping via LPV static output feedback control. IFAC-PapersOnLine. **50**, 13808-13815 (2017)

[24] Murilo, A., Rodrigues, R., Teixeira, E. & Santos, M. Design of a Parameterized Model Predictive Control for Electric Power Assisted Steering. Control Engineering Practice. **90** pp. 331-341 (2019), https://www.sciencedirect.com/science/article/pii/S0967066119301091

[25] Schwenzer, M., Ay, M., Bergs, T. & Abel, D. Review on model predictive control: An engineering perspective. The International Journal Of Advanced Manufacturing Technology. **117**, 1327-1349 (2021)

[26] Rajamani, R. Vehicle dynamics and control. (Springer Science & Business Media,2011)

[27] Shu, P., Sagara, S., Wang, Q. & Oya, M. Improved adaptive lane-keeping control for four-wheel steering vehicles without lateral velocity measurements. International Journal Of Robust And Nonlinear Control. **27**, 4154-4168 (2017)

[28] Satouri, M., Razminia, A., Mobayen, S. & Skruch, P. Disturbance decoupling and tracking controller design for lateral vehicle dynamics. IEEE Access. **9** pp. 40706-40715 (2021)

[29] Chen, J., Sun, D., Zhao, M., Li, Y. & Liu, Z. A new lane keeping method based on human-simulated intelligent control. IEEE Transactions On Intelligent Transportation Systems. **23**, 7058-7069 (2021)
[30] Ulsoy, A., Peng, H. & Çakmakci, M. Automotive control systems. (Cambridge University Press,2012)
[31] Alamir, M. A Pragmatic Story of Model Predictive Control: Self Contained Algorithms and Case-studies. (CreateSpace Independent Publishing Platform,2013)
[32] Júnior, Z., Murilo, A. & Lopes, R. Vehicle stability upper-level-controller based on parameterized model predictive control. IEEE Access. **10** pp. 21048-21065 (2022)
[33] Faroni, M., Beschi, M., Berenguel, M. & Visioli, A. Fast MPC with staircase parametrization of the inputs: Continuous input blocking. 2017 22nd IEEE International Conference On Emerging Technologies And Factory Automation (ETFA). pp. 1-8 (2017)
[34] Chen, Y., Scarabottolo, N., Bruschetta, M. & Beghi, A. Efficient move blocking strategy for multiple shooting-based non-linear model predictive control. IET Control Theory & Applications. **14**, 343-351 (2020)
[35] Rodrigues, R., Murilo, A., Lopes, R. & De Souza, L. Hardware in the loop simulation for model predictive control applied to satellite attitude control. IEEE Access. **7** pp. 157401-157416 (2019)
[36] Ferreau, H., Kirches, C., Potschka, A., Bock, H. & Diehl, M. qpOASES: A parametric active-set algorithm for quadratic programming. Mathematical Programming Computation. **6** pp. 327-363 (2014)
[37] Pandala, A., Ding, Y. & Park, H. qpSWIFT: A real-time sparse quadratic program solver for robotic applications. IEEE Robotics And Automation Letters. **4**, 3355-3362 (2019)
[38] Stellato, B., Banjac, G., Goulart, P., Bemporad, A. & Boyd, S. OSQP: An operator splitting solver for quadratic programs. Mathematical Programming Computation. **12**, 637-672 (2020)
[39] Zhu, Q., Onori, S. & Prucka, R. Pattern recognition technique based active set QP strategy applied to MPC for a driving cycle test. 2015 American Control Conference (ACC). pp. 4935-4940 (2015)
[40] Liao-McPherson, D. & Kolmanovsky, I. FBstab: A proximally stabilized semismooth algorithm for convex quadratic programming. Automatica. **113** pp. 108801 (2020)
[41] Han, Y. & Lee, H. Comparison and Analysis of Convex Optimization Methods for Potential Field-based Path Planning. 2021 21st International Conference On Control, Automation And Systems (ICCAS). pp. 1447-1450 (2021)
[42] Adler, I., Hu, Z. & Lin, T. New proximal newton-type methods for convex optimization. 2020 59th IEEE Conference On Decision And Control (CDC). pp. 4828-4835 (2020)
[43] MatWorks - mpcActiveSetSolver [Online]. Available: https://www.mathworks.com/help/mpc/ref/mpcactivesetsolver.html, Accessed on: Ago. 3, 2023.
[44] MatWorks - quadprog [Online]. Available: https://www.mathworks.com/help/optim/ug/quadprog.html, Accessed on: Ago. 3, 2023.

JAMES DUVAN GARCIA MONTOYA received his degree in Mechatronic Engineering from Unicomfacauca, Colombia (2021), he is currently pursuing a Master's degree in Mechatronic Systems at the University of Brasilia, Brazil. His research interests include Automotive Control Systems, Model Predictive Control, Embedded Systems, Modeling, Real-time Simulation and Advanced Driver Assistance Systems.

EVANDRO LEONARDO SILVA TEIXEIRA received his B.S. degree in Electromechanical (2004) and Mechanical Engineering (2018) from University Centre UDF, a Master's degree (2006) and PhD (2012) in Mechatronic System from University of Brasília, Brazil. He is currently an Associate professor at the University of Brasilia. His research interest includes Automotive Electronics, Embedded Systems, Automotive Control, Modelling and Simulation.

ANDRÉ MURILO holds a degree in Mechatronic Engineering from the Pontifical Catholic University of Minas Gerais (2001), a Master's degree (2006) and a Ph.D. (2009) in Automatic Control from Grenoble INP, France. He is currently an Associate Professor at the Federal University of Lavras. He has experience in the areas of Model Predictive Control, Automotive Control Systems, Nonlinear Systems, Robotics, Mechatronics and Industrial Automation.

RAFAEL RODRIGUES DA SILVA received from the University of Brasilia his B.S.degree in Automotive Enginer(2015) and his Master's degree in Mechatronics System (2017). He worked as an Engineer during 5 years in the automotive industry (OEM). He is currently an Assistant professor at the University of Brasilia. He has experience in the areas of ADAS, Automotive Electronics, Automotive Networks, Automotive Systems and Automotive Driving Simulators.