World Electric Vehicle Journal

MDPI

Article

Ultra-Fast Nonlinear Model Predictive Control for Motion Control of Autonomous Light Motor Vehicles

Vaishali Patne [1,*,†], Pramod Ubare [1,†], Shreya Maggo [1,†], Manish Sahu [2,†], G. Srinivasa Rao [2,†], Deepak Ingole [3,†] and Dayaram Sonawane [1,†]

[1] Department of Instrumentation and Control, COEP Technological University, Shivajinagar, Pune 411005, India; upr18.instru@coeptech.ac.in (P.U.); maggosg19.instru@coeptech.ac.in (S.M.); dns.instru@coeptech.ac.in (D.S.)
[2] Vehicle R&D Establishment, Ahmednagar 414006, India
[3] KPIT Technologies, Pune 411057, India; deepak.ingole@kpit.com
* Correspondence: pva18.instru@coeptech.ac.in
† These authors contributed equally to this work.

Abstract: Advanced Driver Assistance System (ADAS) is the latest buzzword in the automotive industry aimed at reducing human errors and enhancing safety. In ADAS systems, the choice of control strategy is not straightforward due to the highly complex nonlinear dynamics, control objectives, and safety critical constraints. Nonlinear Model Predictive Control (NMPC) has evolved as a favorite option for optimal control due to its ability to handle such constrained, Multi-Input Multi-Output (MIMO) systems efficiently. However, NMPC suffers from a bottleneck of high computational complexity, making it unsuitable for fast real-time applications. This paper presents a generic framework using Successive Online Linearization-based NMPC (SOL-NMPC) for for the control in ADAS. The nonlinear system is linearized and solved using Linear Model Predictive Control every iteration. Furthermore, offset-free MPC is developed with the Extended Kalman Filter for reducing model mismatch. The developed SOL-NMPC is validated using the 14-Degrees-of-Freedom (DoF) model of a D-class light motor vehicle. The performance is simulated in MATLAB/Simulink and validated using the CarSim® software (Version 2016). The real-time implementation of the proposed strategy is tested in the Hardware-In-the-Loop (HIL) co-simulation using the STM32-Nucleo-144 development board. The detailed performance analysis is presented along with time profiling. It can be seen that the loss of accuracy can be counteracted by the fast response of the proposed framework.

Keywords: Advanced Driver Assistance System (ADAS); model predictive control; nonlinear systems; embedded implementation; real-time vehicle control

Citation: Patne, V.; Ubare, P.; Maggo, S.; Sahu, M.; Rao, G.S.; Ingole, D.; Sonawane, D. Ultra-Fast Nonlinear Model Predictive Control for Motion Control of Autonomous Light Motor Vehicles. *World Electr. Veh. J.* **2024**, *15*, 299. https://doi.org/10.3390/wevj15070299

Academic Editors: Yujie Shen, Ying Zhang, Junjie Chen and Yuan Li

Received: 4 May 2024
Revised: 7 June 2024
Accepted: 11 June 2024
Published: 4 July 2024

1. Introduction

With the invention of autonomous, intelligent vehicles, a lot of research is being made towards the enhancement of safety as well as performance of the system. The introduction of Advanced Driver Assistance Systems (ADAS) has definitely improved the quality of driving by reducing the human errors responsible for many accidents [1]. In order to move towards complete control, researchers are working on various individual control systems such as active chassis [2], drift control [3], etc. Many advanced systems have already become an integral part of vehicles like adaptive cruise control [4], automatic parking assist, Anti-lock Braking System (ABS), etc. Various technologies have been used for achieving control. Conventional Proportional-Integral-Derivative (PID) control was used for trajectory tracking [5]. The authors of [6] implemented steering control using Sliding Mode Control (SMC) while [7] used disturbance observer-based SMC for ABS control. Stochastic linear model predictive control was used by [8] for adaptive cruise control. Meanwhile, the authors of [9] proposed the use of algorithms based on Volterra Polynomial for fast control.

It is necessary to validate the accuracy of the control in real time using implementation on embedded platforms like ARM or Field-Programmable Gated Arrays (FPGA). The authors of [10] implemented the lateral control of a vehicle for a 2-DoF model using LMPC on FPGA (Xilinx's Zedboard). Raspberry-pi was used by the authors of [11] for implementing obstacle avoidance using LIDAR. Detailed survey of FPGA implementations of various solutions for autonomous vehicles is presented in [12].

The use of advanced control strategies is crucial for obtaining safe and promising results for ADAS applications. The individual control functions effectively when it operates independently without any dependencies on other vehicle components. Nevertheless, it can encounter significant challenges when tasked with managing multiple inputs and outputs. Model Predictive Control (MPC) is one such technology which has been in the industry in last few decades, mainly due to its ability to handle Multiple-Input Multiple-Output (MIMO) systems. Moreover, control strategies like PID and SMC do not take into account the constraints on the control inputs while formulating the problem explicitly [13]. This is another important property of MPC where the constraints are considered explicitly, delivering the inputs within the bounds specified. A detailed discussion on the application of MPC for ADAS is presented in [14].

MPC has already been used for various aspects of vehicle control, such as distributed MPC for formation control [15], MPC along with Particle Swarm Optimization (PSO) for path tracking [16], nonlinear MPC for active chassis control [2], economic MPC for vehicle platoon [17], etc. Successive linearization-based MPC was used for vehicle motion control while using a nonlinear model of tire only [18]. The effectiveness of the MPC is profound only when the mathematical model used for the control represents the accurate and actual vehicle dynamics. Majority of the applications found in the literature use models with varying Degrees-of-Freedom (DoF) such as 2-DoF [10], 4-DoF [19], 6-DoF [20], and even 8-DoF [21]. The complete behavior of the vehicle as a system is properly defined using a complete car model, which is essentially a 14-DoF system. Very few controls are listed in the literature using this complete car model. The authors of [22] controlled speed for a 14-DoF vehicle model; however, the prediction model used was only 8-DoF.

Worldwide, CarSim® is the preferred choice for analyzing vehicle dynamics when working with various controllers. The authors of [23] tested an 11-DoF Electric Vehicle (E) model, while the authors of [24] tested the electronic stability program of a vehicle system using CarSim for a 2-DoF vehicle model using fuzzy-PID. More implementations for vehicle control using CarSim can be found in [25].

Nonlinear MPC suffers from the disadvantage of highly complex computations, which makes it difficult for the control of systems with fast sampling time. Researchers are working extensively on reducing such. The authors of [26] proposed MPC with robust-to-early termination. In this approach, the optimization is executed only for a fixed time. The suboptimal solution obtained using the constraints ensures recursive feasibility in spite of early termination. Moreover, the authors of [27] used moving horizon estimation with MPC for feedback control of linear systems.

The primary contribution of the paper is the introduction of another ultra-fast control option for real-time control of complex highly nonlinear Multi-Input Multi-Output (MIMO) systems. It is especially crucial for applications like ADAS that traditionally require multiple controllers. These cases present extreme challenges for designers due to the interdependency of objectives and control parameters, for example, in the case of cruise control while managing the lane. In such cases, with traditional NMPC, it becomes very difficult to achieve the sampling time in microseconds or even in the sub-milliseconds range. Linear MPC fails to produce accurate results while explicit MPC becomes a non-viable option due to huge memory requirement. In order to achieve the desired sampling time while using resource-constrained embedded platforms, the use of Successive Online Linearization-based NMPC (SOL-NMPC) is proposed for such complicated applications.

This paper aims to employ a 14-DoF Light Motor Vehicle (LMV) model to regulate speed and yaw rate. Utilizing the entire car model enhances control accuracy while

concurrently escalating computational complexity. It is proposed to use SOL-NMPC for the control of this complex MIMO system. The behavior of the proposed approach in terms of results obtained is compared with the traditional Nonlinear MPC (NMPC). The system is implemented on an STM32F746ZG -based development board (Nucleo−144). To analyze the proper working of the vehicle, the co-simulation is carried out using Simulink and CarSim.

The structure of the paper as follows. Section 2 explains the proposed method for fast control. Here, traditional NMPC is explained along with proposed approach using successive linearization. Section 3 explains the 14-DoF mathematical model of Light Motor Vehicle (LMV). The steps for embedded implementation are described in Section 4. Section 5 presents the results obtained while Section 6 presents a detailed discussion of the performance along with time profiling. Section 7 ends the paper with concluding remarks.

2. Proposed Method for Ultra-Fast Control

The use of MPC is proposed here for the effective control of speed as well as yaw rate given the MIMO nature of the system. Traditional NMPC along with the challenges faced are explained, followed by the proposed approach using linearization of the nonlinear system at every iteration.

2.1. Nonlinear Model Predictive Control (NMPC)

Nonlinear MPC is an advanced optimization technique preferred due to its ability of systematically handling nonlinear constraints of nonlinear systems. The Optimal Control Problem (OCP) is solved in every iteration, generating a sequence of control outputs. In the traditional NMPC, the objective function for reference tracking is minimized to track the desired reference. The objective function is formulated as the Constrained Finite-Time Optimal Control (CFTOC) problem (1), which is then solved using Receding Horizon Controller (RHC) [28] (Chapter 13).

$$\min_{U} \sum_{k=0}^{N-1} \Big(||y_k - y_{\text{ref},k})||_Q^2 + ||u_k||_R^2 \Big), \tag{1}$$

subject to,

$$x_{k+T_s} = f(x_k, u_k), \quad k = 0, \ldots, N-1, \tag{2}$$

$$y_k = g(x_k, u_k), \quad k = 0, \ldots, N-1, \tag{3}$$

$$\Delta u_k = u_k - u_{k-T_s}, \quad k = 0, \ldots, N-1, \tag{4}$$

$$u_{\min} \leq u_k \leq u_{\max}, \quad k = 0, \ldots, N-1, \tag{5}$$

$$y_{\min} \leq y_k \leq y_{\max}, \quad k = 0, \ldots, N-1, \tag{6}$$

$$u_{-1} = u(t - T_s), \tag{7}$$

$$x_0 = \hat{x}(t), \tag{8}$$

where $u \in \mathbb{R}^{N_u}$, $x \in \mathbb{R}^{N_x}$, and $y \in \mathbb{R}^{N_y}$ are the vectors of inputs, states, and outputs, respectively. Functions $f : \mathbb{R}^{N_x}$ and $g : \mathbb{R}^{N_y}$, respectively, represent the state and output transition functions. Sampling time is denoted by T_s and current time is denoted by t. N_u, N_x, N_y are the number of control inputs, states, and outputs, respectively. u_{min} and u_{max} are bounds on u while y_{min} and y_{max} represent output constraints.

NMPC is designed to track reference (y_{ref}) using weighting matrices $Q \succeq 0$ and $R \succ 0$, and the prediction horizon, N. Initial values of states and control inputs are given by x_0 and u_{-1} [29]. The Nonlinear Problem (NLP) solver provided by MATLAB, *fmincon*, is used here for performance evaluation. It accepts the linear as well as nonlinear, equality, and inequality constraints along with the cost function. During each sampling instant, the OCP of (1) is solved using the Interior Point Method (IPM) or Sequential Quadratic Programming (SQP) method available with *fmincon*. This generates an optimal sequence of

control inputs $U = \{u_0^\star, \ldots, u_{N-1}^\star\}$. The system is then excited for the next sampling time using only the first control action ($u_0^\star$). The procedure is repeated every iteration with a new initial condition for states $\hat{x}$.

Though NMPC is preferred for its accuracy as well as its effective handling of nonlinear systems, high computational burden while solving nonlinear CFTOC problem still remains the bottleneck for the application of NMPC. This is mainly observed with highly nonlinear systems or systems with higher dimensions. The demand for fast algorithms is increasing every day for real-time applications. Even with the present day, high-performance CPUs, the solving of the nonlinear equations within the expected sampling time in the sub-milliseconds range is challenging enough. Additionally, direct implementation of these methods and solvers on embedded devices is challenging and not straightforward. One solution is to use a linearized system model. The use of LMPC reduces the complexity, but at the cost of accuracy. Hence, to reduce the computational complexity of the traditional NMPC, the use of NMPC based on successive model linearization is proposed here. This approach, explained in the next section, increases the speed of computation compared to NMPC while achieving better performance than LMPC.

2.2. Successive Online Linearization-Based NMPC (SOL-NMPC)

The proposed approach using successive linearization is explained in this section. In this, as the name suggests, linearization of the nonlinear system model is performed in every iteration (Figure 1) [30]. The current values of the system states are used as the initial value for next iteration. This linearized model is considered as Linear Time Invariant (LTI) since it remains the same during that particular sampling interval. The next step is discretization and then the use of the state space matrices thus generated in the online formulation of the LMPC. These steps are described below in detail.

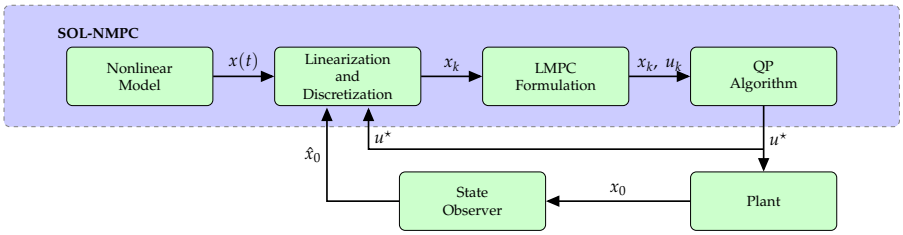

Figure 1. Block diagram showing the steps for the SOL−NMPC.

2.3. Nonlinear Model

It is reckoned that process variables, model parameters, and sampling time are known for the mathematical model of the nonlinear system under consideration ((2) and (3)).

2.4. Linearization

Once a nonlinear model is available, it is linearized at each sampling instant using the present values of states and inputs. This approximate linear model is obtained using the Taylor expansion. The state space matrices A_c, B_c, C_c, and D_c are calculated using partial differentiation considering the previous values of states and inputs (x_{k-T_s}, u_{k-T_s}) (9). This converts the nonlinear system into an approximate piece-wise linear system. Every iteration, the linearization point will keep changing.

$$A_c = \left.\frac{\partial f}{\partial x}\right|_{x_{k-T_s},\, u_{k-T_s}}, \; B_c = \left.\frac{\partial f}{\partial u}\right|_{x_{k-T_s},\, u_{k-T_s}}, \; C_c = \left.\frac{\partial g}{\partial x}\right|_{x_{k-T_s},\, u_{k-T_s}}, \; D_c = \left.\frac{\partial g}{\partial u}\right|_{x_{k-T_s},\, u_{k-T_s}}, \quad (9)$$

where g and f are given by Equations (3) and (2), respectively.

In case of online linearization, the finite difference method is applied at runtime to compute the derivatives, while in the offline case, the current states and controls are used in pre-calculated Jacobian matrices. The offline calculation will result in fast execution; however, knowledge of mathematics involved in the computation of the Jacobian for

highly complex nonlinear systems is necessary. MATLAB's *Jacobian* function can be used in such case.

The successive linearization approach used is different from the scheduling controllers where a bank of LTI models is prepared. Pre-formatted linear systems are obtained by linearizing the nonlinear system at various operating points to compensate for the variation in the model parameters. These are solved using some scheduling law. The authors of [31] used this strategy for control of the vehicle motion in lateral direction using the kinematic model of the vehicle. This is also specifically helpful when linearization fails. However, if the system is complex, choosing different operating points becomes tricky.

2.5. Discretization

The direct approach of first discretization and then optimization is used here for solving the OCP [32]. This approach works well with any initial guess while showing good convergence. A discrete-time system is now available for optimization.

$$\begin{aligned} A_d &= e^{A_c * T_s}, \\ B_d &= A_c^{-1}(A_d - I) * B_c. \end{aligned} \tag{10}$$

The computation of matrix exponentiation is tedious. Furthermore, if A_c is singular then A_c^{-1}, and hence, B_d does not exist. Therefore, keeping in mind the need for fast computation and possible singularity of matrix A_c, forward Euler approximation is used here, which is based on the Taylor series. This method preserves the stability for small sampling time. Moreover, the advantage of this method is more profound in the special situations when the Jacobian becomes singular since there is no need for matrix inversion. The 14-DoF LMV model considered here has a sampling time of just 1 ms and, hence, can be discretized using approximation without loss of accuracy as,

$$\begin{aligned} A_d &= I + A_c * T_s, \\ B_d &= Ts * B_c, \end{aligned} \tag{11}$$

where I is identity matrix ($N_x \times N_x$). The output Equation (13) remains the same; hence, $C_d = C_c$ and $D_d = D_c$ giving discrete-time matrices A_d, B_d, C_d, and D_d.

The exact linear equations, considering the linearization error, are given by,

$$x_k = A_d x_{k-T_s} + B_d u_{k-T_s} + \delta f, \tag{12}$$

$$y_k = C_d x_{k-T_s} + D_d u_{k-T_s} + \delta g, \tag{13}$$

where δf and δg present the differences between the values obtained while solving the exact nonlinear model (expected value) and approximate linear model (actual value) due to linearization, respectively, for states and outputs. However, for systems with fast sampling time, the newly obtained states are very close to the previous states, thus making δf and δg very small and, therefore, safely negligible.

The vectors x_{k-T_s}, u_{k-T_s} represent the latest operating point while x_k are the states at the next sampling instant and are given by,

$$x_k = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_{23}(k) \end{bmatrix}, \quad u_{k-1} = \begin{bmatrix} u_1(k - T_s) \\ u_2(k - T_s) \end{bmatrix}. \tag{14}$$

The states that cannot be measured are estimated using the Kalman filter. As the linearization is performed before optimization using the direct approach, the most current states available from previous sampling instant are used as the present operating point.

2.6. Formulation of Quadratic Programming (QP) Problem

The cost function to be minimized can be written as

$$\min_{U} \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k. \tag{15}$$

Equation (15) is further simplified to formulate the Quadratic Programming (QP) problem of the time varying linear model as,

$$\begin{aligned} \min_{U} \ & \frac{1}{2} U^T H U + c^T U, \\ \text{s.t.} \ & GU \leq w, \end{aligned} \tag{16}$$

where $H \in \mathbb{R}^{N_u N \times N_u N}$ denotes the Hessian matrix, $c \in \mathbb{R}^{N_x \times N_u N}$ is the gradient matrix with $G \in \mathbb{R}^{q \times N_u N}$, and $w \in \mathbb{R}^{\mathrm{q}}$, representing bounds, with q inequalities. Thus, for QP formulation, the nonlinear constraints of (1) are linearized using (12) and (13). These linearized constraints are further used in quadratic problem (16). The QP problems can be solved using methods like interior-point methods, active-set methods, or gradient-based methods [33]. These methods have fast convergence properties with good stability.

The proposed approach of SOL-NMPC uses the Fast Gradient Method (FGM) by [34] as the QP solver. This method achieves the desired accuracy in fewer iterations than other methods [35]. The authors also proved that, as compared to other methods, where there is a quadratic increase in the resource utilization and the time for a single iteration with the system dimensions and horizon length, FGM shows linear relationship. Hence, the use of FGM is advantageous for larger horizons. The method was coded in C and no ready-made solver was used.

Since linear systems are renowned for possessing a unique optimal solution, linearizing the nonlinear system during each iteration guarantees a singular optimal solution every sampling duration, thereby alleviating the issue of multiple optima that arises in nonlinear systems.

2.7. Singularity Issues

While converting the system from nonlinear to linear every iteration, there are chances that the Jacobian formed can be singular. This issue can be handled specific to a system where the situation causing the singularity can be avoided by carefully choosing different parameters. In some cases, singularities can be efficiently escaped by exploiting the system geometry [36]. Using approximate discretization can also help solve this problem up to a certain extent as long as the system sampling time is in sub-milliseconds. Moreover, the choice of penalty matrices and suitable prediction horizon prove to be important in this case.

The most commonly used methods to deal with singular matrices include singular value decomposition (SVD), eigenvalue decomposition, and decomposition using Cholesky. These methods change the structure of the matrix and a new one is formed with reduced dimensions [37]. However, this can cause the change in the matrix size each time and, hence, a change in the memory requirement. For real-time implementation, it is better to avoid dynamic memory allocation; hence, this approach finds relatively fewer takers.

In an alternative approach, a Cholesky decomposition of A_d matrix is carried in each iteration. Here, it is possible to identify non-positive definiteness of the matrices. In this case, a small perturbation can be added to make the matrix non-singular without disturbing the symmetry of the matrix. This is most commonly known as the Gershgorin circle theorem [38]. A diagonal matrix (K) containing small diagonal elements (formed by multiplying an identity matrix of size of number of states by a small factor (ϵ)) is added to the matrix iteratively until a positive definite matrix is formed (Algorithm 1). Though, with proper choice of ϵ, this approach may not cause instability, but there might be an issue of offset or suboptimal solution.

Algorithm 1 Treatment for singularity using the Gershgorin theorem.

while $(A_d \not\succ 0)$ **do**
  $K = I * \epsilon$
  $A_d = A_d + K$
end while

2.8. Disturbance Model

As explained in Section 2.5, the exact linearization is given by Equation (12). For the system under consideration, due to fast sampling, the δf and δg can be ignored. However, in cases where these linearization errors are considerably large, these need to be dealt with while designing the MPC. There are various ways of considering the same. Reformulating the cost function or modifying the Hessian and gradient matrices to include the linearization error are some such methods. The model mismatch caused by these errors can also be corrected using disturbance modeling where these are incorporated as an additional state vector of uncertainties. Disturbance observer along with Kalman estimator is used here to ensure the robustness of the proposed approach against the linearization errors as well as any unpredictable perturbations caused in system states. In this case, the state vector is augmented with the disturbance vector (17), generating a disturbance model [39]. This greatly reduces the steady state offset error.

$$\underbrace{\begin{bmatrix} x(t+T_s) \\ d(t+T_s) \end{bmatrix}}_{x_e(t+T_S)} = \underbrace{\begin{bmatrix} A_c & B_{dist} \\ 0 & I \end{bmatrix}}_{A_e} \underbrace{\begin{bmatrix} x(t) \\ d(t) \end{bmatrix}}_{x_e(t)} + \underbrace{\begin{bmatrix} B_c \\ 0 \end{bmatrix}}_{B_e} u(t), \tag{17}$$

$$y_e(t) = \underbrace{\begin{bmatrix} C_c & C_{dist} \end{bmatrix}}_{C_e} \begin{bmatrix} x(t) \\ d(t) \end{bmatrix} + D_e u(t), \tag{18}$$

where B_{dist} and C_{dist} are disturbance matrices of dimensions $(N_y \times N_u)$ and $(N_y \times N_x)$, respectively.

2.9. State Estimator and Disturbance Observer

For Multi-Input Multi-Output (MIMO) systems, it may not be always possible to measure all the states. Hence, estimators like the Kalman estimator can be used to guess the non-measurable states. The Kalman filter is also effective in rejection of sensor noise. A state observer based on the Kalman filter (KF) is used here to estimate the internal non-measurable states of the system, using available measurements of the system inputs and outputs. Kalman is also known as the recursive filter, which needs only the current measured states and the previous estimated states.

The KF is further extended for nonlinear dynamics. In the Iterative Extended Kalman Filter (IEKF), the nonlinear system and output functions are linearized at the most current estimate. The states, Kalman gain, as well as the error covariance matrix are computed in every iteration. Issues may arise, such as errors in linearization and convergence problems. The key to mitigating most of the challenges associated with the filter lies in appropriately tuning filter parameters and establishing accurate initial estimates.

The predict phase of the observer estimates the states at the current instant while the same are updated using Kalman gain in the update phase. The system model in (12) and (13) is considered for only state observer without consideration for disturbances, while the model in (17) is considered for reducing the mismatch between nonlinear and linearized version of the system under consideration. The following equations are used to estimate the extended state $\hat{x}$ using the available measurements of states of plant:

$$\hat{x_e}(t+T_s) = A_e \hat{x_e}(t) + B_e u(t) + L(y(t) - \hat{y_e}(t)), \tag{19}$$

$$\hat{y_e}(t) = C_e \hat{x_e}(t) + D_e u(t), \tag{20}$$

where L is the state filter gain matrix of dimensions $N_x \times N_y$. This can be obtained using Kalman filter gain or by pole placement. The extended matrices of (17), A_e, B_e, C_e, and D_e, are obtained after linearization (Section 2.4) and discretization (Section 2.5) in every iteration using the latest states and input vectors while x_e and y_e are the extended states and outputs, respectively, considering the difference between the plant and model output as the disturbance state.

The next section explains the 14-DoF vehicle model.

3. The 14-DoF Vehicle Model

The system of autonomous Light Motor Vehicle (LMV) is considered here as the case study. The 14-DoF vehicle model is used for the study of the behavior of LMV in all directions, viz., vertical, lateral, and longitudinal directions. It consists of four unsprung masses of the wheels and a sprung mass of vehicle body. The vehicle body has 6-DoF which includes the vertical, longitudinal, roll, lateral, yaw motion, and pitch. Each of the wheels has spin and vertical motion (2-DoF). A lumped mass model is presented in this configuration. Other DoFs of the tires are treated as an input variable or neglected, e.g., the toe angle of each wheel. For a detailed explanation of the mathematical model of the 14-DoF LMV used in this paper, please refer to [40]. The 23 states (x_1 to x_{23}) and two inputs (u_1, u_2) are given in Table 1.

Table 1. States and inputs of the 14−DoF vehicle model.

Variable	Parameter	Description	Unit
x_1	$\dot{z_s}$	Velocity of sprung mass	rad/s
x_2	z_s	Displacement of sprung mass	rad
x_3	$\dot{z_{u1}}$	Velocity of unsprung mass (front-left)	rad/s
x_4	$\dot{z_{u2}}$	Velocity of unsprung mass (front-right)	rad/s
x_5	$\dot{z_{u3}}$	Velocity of unsprung mass (rear-left)	rad/s
x_6	$\dot{z_{u4}}$	Velocity of unsprung mass (rear-right)	rad/s
x_7	z_{u1}	Displacement of unsprung mass (front-left)	rad
x_8	z_{u2}	Displacement of unsprung mass (front-right)	rad
x_9	z_{u3}	Displacement of unsprung mass (rear-left)	rad
x_{10}	z_{u4}	Displacement of unsprung mass (rear-right)	rad
x_{11}	V_x	Longitudinal velocity	m/s
x_{12}	d	Longitudinal displacement	m
x_{13}	V_y	Lateral Velocity	m/s^2
x_{14}	$\dot{\theta}$	Pitch velocity	rad/s
x_{15}	θ	Pitch angle	rad
x_{16}	$\dot{\phi}$	Roll velocity	rad/s
x_{17}	ϕ	Roll angle	rad
x_{18}	$\dot{\gamma}$	Yaw rate	rad/s
x_{19}	ω_1	Angular velocity of front-left wheel	rad/s
x_{20}	ω_2	Angular velocity of front-right wheel	rad/s
x_{21}	ω_3	Angular velocity of rear-left wheel	rad/s
x_{22}	ω_4	Angular velocity of rear-left wheel	rad/s
x_{23}	γ	Yaw	rad
u_1	δ	Steering angle	rad
u_2	T_{a1}	Torque to front wheels	N m

The aim is to control the motion, i.e., manage velocity while maintaining the lane by adjusting both steering angle (δ) and torque to front wheels (T_{a1}). The main objective here is to achieve partial automation in driving which is level 2 of vehicle autonomy where two tasks are being handled simultaneously [41]. The objective is tested for four different variations of velocity and yaw rate as given in Table 2.

Table 2. Various test scenarios considered used for testing of the 14−DoF vehicle model.

	Scenario−1	**Scenario−2**	**Scenario−3**	**Scenario−4**
Description	J−Turn	Lane Change	Constant velocity, varying yaw rate	Constant yaw rate, varying velocity
Speed (km/h)	50	50	50	$t = 0$ to 40 → Vx = 20 $t = 40$ to 60 → Vx = 35 $t = 60$ to 80 → Vx = 50 $t > 80$ → Vx = 75
Yaw rate (rad)	$t = 0$ to 50 → yaw rate = 0 $t > 50$ → yaw rate = 0.1	$t = 0$ to 25 → yaw rate = 0 $t = 25$ to 50 → yaw rate = 0.2 $t = 50$ to 75 → yaw rate = −0.2 $t > 75$ → yaw rate = 0	$t = 0$ to 20 → yaw rate = 0 $t = 20$ to 40 → yaw rate = 0.05 $t = 40$ to 60 → yaw rate = 0.1 $t = 60$ to 80 → yaw rate = 0.15 $t > 80$ → yaw rate = 0.2	$t = 0$ to 20, yaw rate = 0 $t > 20$, yaw rate = 0.2

MPC Parameters

Tuning of the penalty matrices of the SOL-NMPC as well as the observer covariance are utmost challenging due to the extreme complex nature of the system under consideration. Various design parameters used for SOL-NMPC are explained here. The model is solved for $N_x = 23$, $N_u = 2$, and $N_y = 2$. Longitudinal velocity—V_x (x_{11}) and the yaw rate—$\dot{\gamma}$ (x_{18}) are the two outputs considered here. Using the initial value of velocity $V_x(0) = 1$, the initial values of inputs and states used are: $u_0 = [0;0]$ and $x_0 = [0;0;0;0;0;0;0;0;0;0;Vx_0;0;0;0;0;0;0;0;Vx_0/0.325;Vx_0/0.325;Vx_0/0.325;Vx_0/0.325]$, respectively. The penalty matrices obtained by experimentation are $R = \text{diag}(100,\ 10^{-8})$ using prediction horizon $N = 3$, with all elements of Q matrix 0 except $Q_{11} = 10$, $Q_{18} = 1000$. Based on literature available [40], the sampling time considered is $Ts = 1$ ms.

The steps for hardware implementation are explained in the next section.

4. Embedded Implementation

It is of utmost importance for real-time applications that the control algorithm must be implementable on some portable, easy-to-use embedded platform. The most commonly used hardware for such implementations is the ARM cortex development board, one of the user-friendly as well as easily available, cost-effective options. With the latest versions of MATLAB (version: 2022*a*), it is possible to generate the C-code of the NMPC (*fmincon*) solver. The generated code is directly compatible with the ARM-based embedded platforms. However, the SOL-NMPC using FGM is our own code, developed directly using C, which is convenient for the embedded implementation.

This section presents the step-by-step implementation of SOL-NMPC on the ARM ($STM32$)-based low-cost microcontroller board (Nucleo-144), which is extensively used for research in academics as well as in the automotive industry.

The explanation of steps shown in Figure 2 for implementation of the proposed framework is as follows:

- Nonlinear model: The plant is nonlinear in nature. Every iteration, this model is linearized.
- Formulation for LMPC: After the nonlinear model is linearized and discretized as explained in Section 2.2, the necessary matrices (constraints, gradient, and hessian) are formulated every sampling instant, to solve the convex QP problem.
- QP solver for embedded implementation: *Fast Gradient Method (FGM)* is used to optimize the QP problem formed for the embedded implementation.
- SIL testing and verification: Software-In-the Loop (SIL) simulation is performed after the complete C code is ready, using Intel i7 2 GHz PC. At this stage, the penalty parameters are tuned properly to achieve the desired output as well as disturbance rejection.
- $STM32$Cube IDE: The code is compiled for $STM32F746ZG$ Microcontroller (Nucleo−144 development board) using $STM32$Cube IDE after the performance achieved is satisfactory. The resulting code is then analyzed for solver time as well as memory.

- HIL Testing and Verification: The implementability of the proposed approach is validated after implementation. A digital oscilloscope is used to measure the solver time (Figure 3).

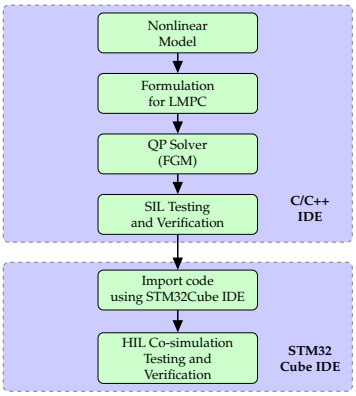

Figure 2. Flowchart showing the step-by-step embedded implementation of SOL-NMPC.

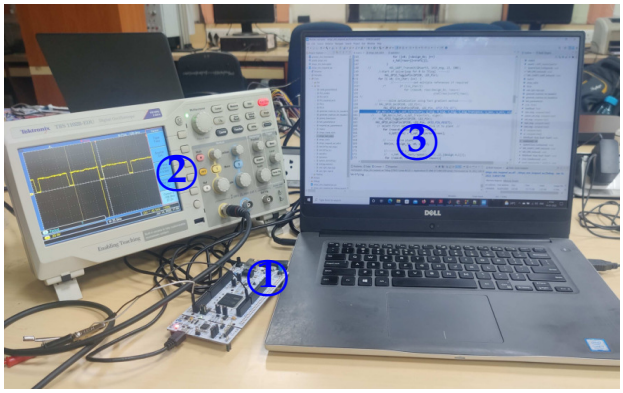

Figure 3. Lab setup to demonstration the implementation of SOL-NMPC using (1) *STM32F746ZG* microcontroller, (2) Digital Oscilloscope, and (3) PC.

The results obtained are discussed in detail in the next section.

5. Results

The SOL-NMPC controller for motion control of vehicle is first tested using MATLAB-Simulink. Once the desired performance is achieved by proper tuning of penalty matrices, the model file of Simulink is then ported in CarSim. CarSim is the preferred tool universally for analyzing vehicle dynamics and developing various controllers. The Class-D sedan vehicle is selected for testing. The validation diagram is as shown in Figure 4. The results obtained for various test scenarios considered are presented in this section.

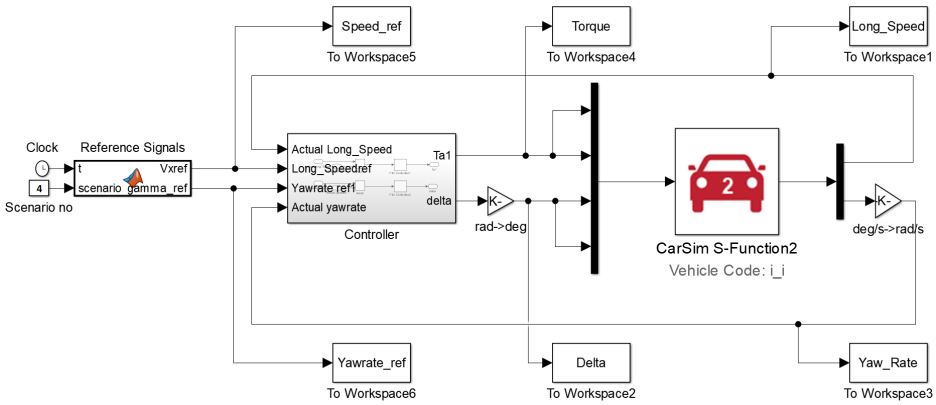

Figure 4. Simulink block diagram used for validation of SOL-NMPC using CarSim for the 14−DoF vehicle model.

5.1. Results of SIL Using MATLAB*-Simulink*

The successive linearization-based approach is tested for controlling the motion of the LMV. The objective is to maintain the speed while following the desired path. The path is set using various options and combinations of yaw rate and longitudinal speed. It can be observed that there exists a steady-state error in the simulation results. One of the reasons for such can be the mismatch between the actual plant and its mathematical model considered. This can be reduced using state and disturbance observer as explained in Section 2.8. This can be considered as the Offset-Free SOL-NMPC (OFSOL-NMPC).

5.2. Results of CarSim Validation

After the tuning of the parameters is perfected and desired results are achieved in SIL using MATLAB-Simulink for both standard SOL-NMPC as well as OFSOL-NMPC, the proposed approach is validated using CarSim. Class-D Sedan vehicle is selected. A new Simulink file is generated using the CarSim block to replace the vehicle system model (Figure 4). The speed and yaw rate are obtained from CarSim. For estimation of the remaining 21 states, Iterated Extended Kalman Filter (IEKF) is used along with disturbance observer as explained in Section 2.9.

As the time required to solve one iteration of *fmincon* is much higher than the desired sampling time, the control generated will be highly suboptimal. Hence, it is not suitable for the validation purpose. The proposed approach of SOL-NMPC as well as OFSOL-NMPC runs faster than traditional NMPC, thus making it suitable for the real-time validation. The results obtained are presented in comparison with those from Simulink.

6. Performance Discussion

The detailed analysis of the results achieved for the proposed approach is presented here. Performance analysis is presented in terms of the Key Performance Indicators (KPIs) such as Mean Square Error (MSE), Integral Square Error (ISE), Integral Absolute Error (IAE), Integral Time Square Error (ITSE), and Integral Time Absolute Error (ITAE).

After satisfactory simulation results are achieved, detailed time profiling is performed for the proposed approach. This helps in understanding the time-heavy component of the controller, which can be focused upon in case further reduction in time is necessary. The results are obtained for four test case scenarios as explained in Table 2.

6.1. Performance Analysis

Figure 5 present the outputs obtained using SOL-NMPC and OFSOL-NMPC as compared to traditional NMPC for the four test cases considered here. The Software-in-the-Loop testing is performed using MATLAB-Simulink. The solver time for SOL-NMPC followed by the OFSOL-NMPC is faster than that of NMPC, owing to a significant reduction in computational complexity.

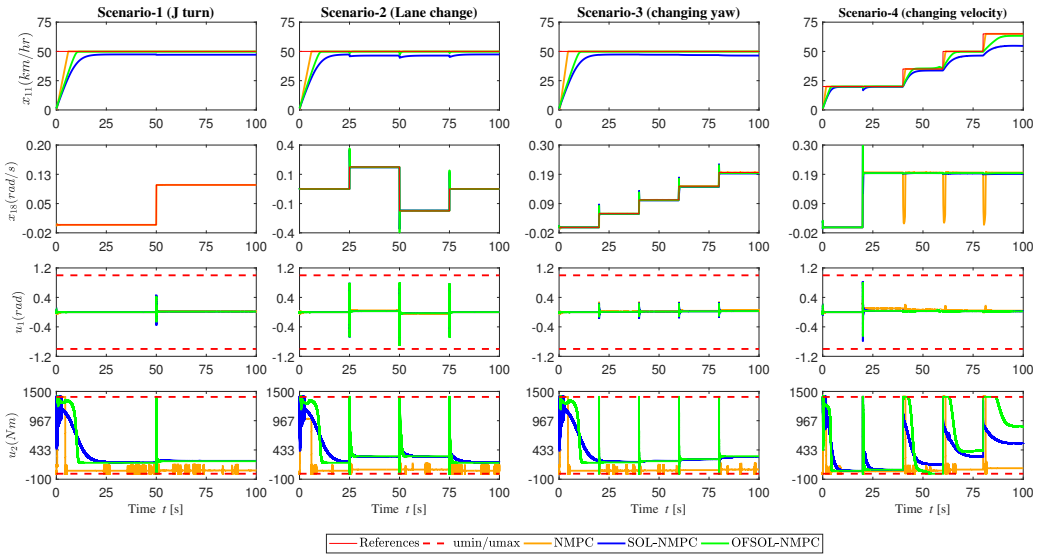

Figure 5. SIL results obtained for NMPC, SOL−NMPC, and OFSOL−NMPC using MATLAB−Simulink for the 14−DoF vehicle model.

Figure 6 present the outputs obtained for the four scenarios considered using SOL-NMPC and OFSOL-NMPC after CarSim validation. It can be seen that for SOL-NMPC, there is an offset between the reference and CarSim output; however, the output is seen to be extremely stable. The offset might be present due to differences in the way some of the internal parameters are handled by CarSim and MATLAB-Simulink. This offset is further reduced using Offset-Free NMPC (OFSOL-NMPC), at the cost of increased computational complexity and, hence, solver time.

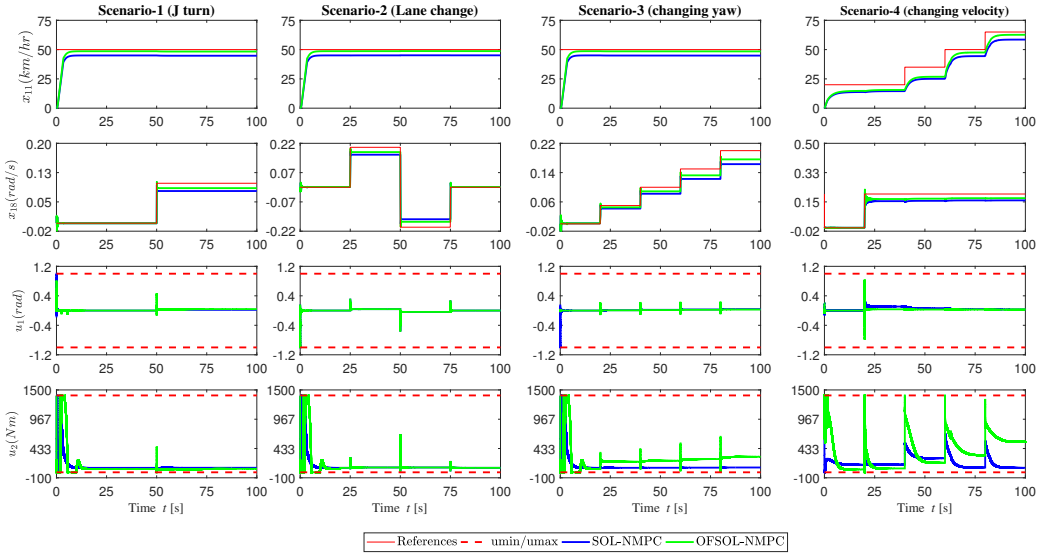

Figure 6. Output obtained for CarSim validation using SOL−NMPC and OFSOL−NMPC for the 14−DoF vehicle model.

Table 3 lists the KPIs for the four test scenarios considered. The controllers considered are traditional NMPC, standard SOL-NMPC, and OFSOL-NMPC. For ISE, square of error is integrated over time; thus, large errors are penalized more than smaller ones. Hence, smaller ISE means very small fluctuations about the reference. Integrated absolute error over time is expressed in IAE. Reducing IAE can make the system sluggish. In case of ITAE, the absolute error multiplied by the time is integrated while ITSE integrates the squared error multiplied by the time. These indicate weight errors existing after a long time after the execution starts compared to those at the start of the execution, and hence are larger

than MSE, ISE, and IAE. This analyzes the system behavior and indicates if it deviates after a reasonably long run-time. As all the values are quite lower, it can be safely inferred that the proposed approach is accurate and fast at the same time.

Table 3. Comparison of KPIs of NMPC, SOL−NMPC, and OFSOL−NMPC for 14−DoF vehicle control for all test cases.

	Scenario−1 (J−Turn)			Scenario−2 (Lane Change)		
KPIs	**NMPC**	***SOL−NMPC***	***OFSOL−NMPC***	**NMPC**	***SOL−NMPC***	***OFSOL−NMPC***
MSE	0.9892	1.8796	1.4408	0.9892	1.9435	1.4505
ISE	98.89	187.93	144.06	98.89	194.33	145.03
IAE	21.024	75.8775	35.6527	21.03	82.86	35.79
ITSE	147.5157	1291.252	352.36	147.54	1598.22	359.71
ITAE	42.508	2100.246	273.16	42.59	2428.7	273.6
	Scenario−3 (Changing Yaw)			**Scenario−4 (Changing Velocity)**		
KPIs	**NMPC**	***SOL−NMPC***	***OFSOL−NMPC***	**NMPC**	***SOL−NMPC***	***OFSOL−NMPC***
MSE	0.8158	1.906	1.45	0.107	0.9088	0.3591
ISE	81.557	190.553	144.96	10.68	90.88	35.90
IAE	16.58	78.99	35.61	6.65	66.011	30.23
ITSE	94.56	1500.47	355.68	362.94	6336.86	1865.05
ITAE	25.85	2343.55	256.15	254.33	4465.81	1759.27

It can be seen that NMPC as well as SOL-NMPC and OFSOL-NMPC are able to track the expected references quite accurately with OFSOL-NMPC showing marginally better performance than standard SOL-NMPC. It can be safely stated here that the fast speed of computation has helped in achieving the desired performance in spite of suboptimality.

6.2. Time Profile

Monte Carlo study was conducted with various initial values of the speed for both NMPC as well as OFSOL-NMPC. Table 4 presents the minimum, maximum, and average time taken by the OFSOL-NMPC (in blue color) in comparison with the traditional NMPC (in red color). It can be seen that it is not possible to solve the OCP within the desired solver time using the *fmincon* solver which, however, is possible using the SOL-NMPC as the solver time achieved is close to the desired sampling time. Moreover, it can be observed that a 95% decrease in the solver time (indicated in blue with down arrow) is achieved as a result of a significant reduction in complex computations due to linearization. The solver time is the total of the linearization time, discretization time, computation of matrices for QP formulation, and the QP solver time along with state and disturbance observer. The pie-chart in Figure 7 presents the detailed time profiling. It can be observed that the QP solver takes slightly less time than that required for the computation of matrices. This is dependent not only on the size of the system, but also on the prediction horizon. Reducing the prediction horizon will decrease the matrix computation time at the cost of accuracy, affecting the stability of the system.

Table 4. Computational complexity in terms of time required and resource utilization by NMPC and OFSOL−NMPC for the control of the 14−DoF vehicle model.

Solver	**Avg. Solver Time (ms)**	**Min Solver Time (ms)**	**Max Solver Time (ms)**	**RAM Used (kb)**	**Flash Used (kb)**
NMPC	18.10	14.68	18.75	1.66	86.79
OFSOL−NMPC	0.88 (95% ↓)	0.75	0.91	1.67	84.18

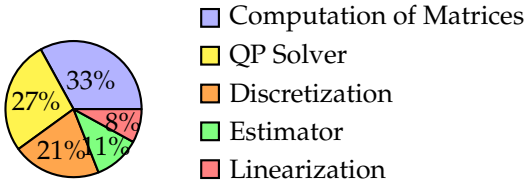

Figure 7. Time profile of OFSOL−NMPC for the 14−DoF vehicle model.

6.3. Stability and Feasibility

Stability refers to the property of a system to remain within a constrained region and to attain the desired state. Furthermore, the response of the system to disturbances or initial conditions remains predictable and controlled. Based on the results obtained after the addition of the disturbance observer, the system is observed to be able to track the desired reference even in the presence of uncertainties. In that sense, it can be said that the system is showing stable behavior. However, detailed mathematical analysis is not performed for the same here. The addition of terminal constraints is one way of ensuring stability, although the strict constraints may lead the system to infeasibility.

To ensure the stability, the terminal constraints need to be included [42]. Equation (15) thus changes to:

$$\min_{U} x_N^T P x_N + \sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k. \tag{21}$$

The terminal cost is given by matrix $P \in \mathbb{R}^{N_x \times N_x}$ with $x_k \in \mathbb{X}_f$. Here, X_f indicates the constraints on the final predicted state value. The matrix P is obtained by solving the discrete-time Riccati equation given by [28] (Chapter 9),

$$P = A_d^T P A_d - A_d^T P B_d (R + B_d^T P B_d)^{-1} B_d^T P A_d + Q. \tag{22}$$

The term $x_N^T P x_N$ is also called the cost associated with infinite horizon since practical problems use finite horizon for computational convenience. The terminal constraint set x_f is computed using a polyhedral set. The values of penalty matrices Q and R also need to be re-tuned for feasible P.

The inclusion of terminal constraints improves the accuracy of the results obtained (by almost 3 to 5%). However, though the terminal constraints lead to efficient long-term performance, they can make the optimal control problem harder. In general, the computation of terminal cost as well as constraints is performed offline to reduce the computational burden. However, when using the approach of successive linearization, the state-space matrices are computed every iteration and, hence, the P matrix also needs to be updated every time. This adds to the solver time to the extent that the maximum solver time obtained goes above the desired sampling time of 1 ms. It will be beneficial to include the same when implementing on embedded platforms with faster speed or with reconfigurable hardware like FPGA where it is possible to optimize memory to achieve the desired solver time.

Terminal constraints also impact the feasibility of the control. The X_f needs to be relaxed to some extent to obtain a feasible solution. Moreover, recursive feasibility refers to the fact that the feasible solution to the optimization problem at the current time step will generate a feasible solution at subsequent time steps as well. Here, it is assumed that there are no significant disturbances. Since the solution always lies within the desired boundary due to the addition of terminal cost, even if there is a change in reference, the control will always lie within the desired range. In such a case, slightly more time might be required to achieve the desired reference while keeping the system stable as well as feasible. The recursive feasibility usually ensures stability. The authors of [43] explored various conditions for ensuring asymptotic stability as well as feasibility.

6.4. HIL Co-Simulation Results

The $STM32F746ZG$ development board (Nucleo-144) is used for testing the implementability of the proposed approach. Refer to Section 4 for an explanation of the steps followed for the embedded implementation. The memory utilization is presented in Table 4 for both NMPC and OFSOL-NMPC. It can be observed that the memory required for the traditional NMPC solver is slightly more than OFSOL-NMPC, as in case of the successive linearization approach, even if a large matrices are formed, the computational resources required are much less. Moreover, the size of the matrices depends upon the number of variables as well as the prediction horizon.

7. Conclusions

For problems pertaining to ADAS systems, Model Predictive Control-based strategies provide interesting solutions since they mixes classical feedback with optimal control. In this paper, a part of the ADAS system, motion control of vehicle and Lane Keep Assist (LKA), is achieved using Successive Online Linearization-based Nonlinear MPC. The 14-DoF model of LMV is used as a case study. In case of SOL-NMPC, in every iteration, linearization of the system is performed. Hence, the computational complexity is reduced significantly due to the use of LMPC in every iteration. The LMPC is formulated as a standard QP problem. This QP problem can be solved using any available robust solvers within few microseconds to milliseconds. The QP solvers have been used in the industry for a few decades now. As a result, the algorithms used are quite stable and robust. In case of a sudden change in operating conditions or external disturbances, the correction in control based on the instantaneous states is executed immediately. The stable desired value is achieved as immediately as possible. This makes the system more robust to external disturbances while significantly reducing the computational complexity.

The performance was first simulated in the MATLAB-Simulink when steady-state error was observed. To remove the offset, disturbance observer was added and validated using the CarSim® software. It is seen that the performance achieved is at par with the traditional nonlinear MPC with drastically reduced solver time. Future work on this can include further reducing the offset observed in the output of CarSim. The feasibility of the proposed control for this model is also tested on the embedded platform. The ARM-based Nucleo−144 development board is used for verifying the implementability of the control algorithm in HIL. It has been noticed that the performance achieved with ARM-based embedded hardware closely resembles that of Software-In-the-Loop (SIL) testing. The impact of computational and resource limitations on the chosen hardware platform is minimal and can therefore be disregarded with confidence.

The approach needs to be further analyzed mathematically in respect of local as well as global stability. For the real-time implementation of ADAS for MIMO systems, the solver time achieved should be much less than the desired sampling time to incorporate various delays like sensor delay, communication delay, conversion times, etc. The ARM board being fixed architecture board, time as well as memory optimization are not possible. To further improve the solver time, high-speed hardware need to be used. Hence, it is necessary to explore options like reconfigurable hardware like FPGA. The solver speed can be improved due to inherent parallelism available in the FPGA. Further, the complete potential of MPC as applied to ADAS can be fully explored by adding few more control options. This outlines the path for future work in this area.

Author Contributions: Conceptualization, M.S., G.S.R. and D.S.; Data curation, V.P. and S.M.; Formal analysis, D.I. and D.S.; Investigation, V.P. and S.M.; Methodology, V.P., P.U. and D.S.; Project administration, V.P. and D.S.; Resources, M.S., G.S.R. and D.S.; Software, V.P. and S.M.; Supervision, D.I. and D.S.; Validation, V.P., P.U. and S.M.; Visualization, V.P., P.U. and D.I.; Writing—original draft, V.P.; Writing—review and editing, D.I. and D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: The authors would like to thank the "Vehicle Research and Development Establishment (VRDE)", Ahmednagar, Govt. of India (VRDE/21*CR*0002/UGS/CARS). Further, Vaishali Patne would like to acknowledge the contribution of the Department of Science and Technology, Govt. of India (WOS-A/ET-120/2018).

Conflicts of Interest: Manish Sahu and G. Srinivasa Rao are employees of Vehicle R&D Establishment. Deepak Ingole is an employee of KPIT Technologies Ltd., Pune, India. This paper reflects the views of the scientists, and not the company.

Abbreviations

ABS	Anti-lock Breaking System
ADAS	Advanced Driver Assistance System
CFTOC	Continuous Finite Time Optimal Control
DoF	Degrees of Freedom
EKF	Extended Kalman Filter
EV	Electric Vehicle
FGM	Fast Gradient Method
FPGA	Field Programmable Gated Array
HIL	Hardware-in-The-Loop
IAE	Integral Absolute Error
IEKF	Iterative Extended Kalman Filter
IPM	Interior Point Method
ISE	Integral Square Error
ITAE	Integral Time Absolute Error
ITSE	Integral Time Square Error
KF	Kalman Filter
KPI	Key Performance Indicators
LKA	Lane Keep Assist
LMPC	Linear Model Predictive Control
LMV	Light Motor Vehicle
LTI	Linear Time Invariant
MIMO	Multiple-Input Multiple-Output
MPC	Model Predictive Control
MSE	Mean Square Error
NLP	Nonlinear Problem
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
OFSOL-NMPC	Offset-Free SOL-NMPC
PID	Proportional-Integral-Derivative
PSO	Particle Swarm Optimization
QP	Quadratic Programming
RHC	Receding Horizon Control
SIL	Software-in-The-Loop
SMC	Sliding Mode Control
SOL-NMPC	Successive Online Linearization-based Nonlinear Model Predictive Control
SQP	Sequential Quadratic Programming

References

1. Ziebinski, A.; Cupek, R.; Grzechca, D.; Chruszczyk, L. Review of Advanced Driver Assistance Systems. *AIP Conf. Proc.* **2017**, *1906*, 120002.
2. Guo, H.; Liu, F.; Xu, F.; Chen, H.; Cao, D.; Ji, Y. Nonlinear Model Predictive Lateral Stability Control of Active Chassis for Intelligent Vehicles and Its FPGA Implementation. *IEEE Trans. Syst. Man, Cybern. Syst.* **2019**, *49*, 2–13. [CrossRef]
3. Zhang, F.; Gonzales, J.; Li, S.E.; Borrelli, F.; Li, K. Drift Control for Cornering Maneuver of Autonomous Vehicles. *Mechatronics* **2018**, *54*, 167–174. [CrossRef]
4. He, Y.; Ciuffo, B.; Zhou, Q.; Makridis, M.; Mattas, K.; Li, J.; Li, Z.; Yan, F.; Xu, H. Adaptive Cruise Control Strategies Implemented on Experimental Vehicles: A Review. *IFAC-PapersOnLine* **2019**, *52*, 21–27. [CrossRef]
5. Farag, W. Complex Trajectory Tracking Using PID Control for Autonomous Driving. *Int. J. Intell. Transp. Syst. Res.* **2020**, *18*, 356–366. [CrossRef]
6. Xu, T.; Liu, X.; Li, Z.; Feng, B.; Ji, X.; Wu, F. A Sliding Mode Control Scheme for Steering Flexibility and Stability in All-wheel-steering Multi-axle Vehicles. *Int. J. Control. Autom. Syst.* **2023**, *21*, 1926–1938. [CrossRef]
7. Wanaskar, V.; Shendge, P.; Phadke, S. A Disturbance Observer Based Sliding Mode Control for Anti-Lock Braking System. In Proceedings of the Conference on Advanced Computational and Communication Paradigms, Gangtok, India, 5–28 February 2019; pp. 1–6.
8. Moser, D.; Waschl, H.; Kirchsteiger, H.; Schmied, R.; Del Re, L. Cooperative Adaptive Cruise Control Applying Stochastic Linear Model Predictive Control Strategies. In Proceedings of the European Control Conference, Linz, Austria, 15–17 July 2015; pp. 3383–3388.

9. Suslov, K.; Gerasimov, D.; Solodusha, S. Smart Grid: Algorithms for Control of Active-Adaptive Network Components. In Proceedings of the IEEE Eindhoven PowerTech, Eindhoven, The Netherlands, 9 June–2 July 2015; pp. 1–6.
10. Li, Y.; Li, S.E.; Jia, X.; Zeng, S.; Wang, Y. FPGA Accelerated Model Predictive Control for Autonomous Driving. *J. Intell. Connect. Veh.* **2022**, *5*, 63–71. [CrossRef]
11. Baras, N.; Nantzios, G.; Ziouzios, D.; Dasygenis, M. Autonomous Obstacle Avoidance Vehicle Using LIDAR and an Embedded System. In Proceedings of the International Conference on Modern Circuits and Systems Technologies, Thessaloniki, Greece, 13–15 May 2019; pp. 1–4.
12. Kasem, A.; Reda, A.; Vásárhelyi, J.; Bouzid, A. A Survey About Intelligent Solutions for Autonomous Vehicles Based on FPGA. *Carpathian J. Electron. Comput. Eng.* **2021**, *13*, 7–11. [CrossRef]
13. Incremona, G.P.; Rubagotti, M.; Ferrara, A. Sliding Mode Control of Constrained Nonlinear Systems. *IEEE Trans. Autom. Control.* **2017**, *62*, 2965–2972. [CrossRef]
14. Musa, A.; Pipicelli, M.; Spano, M.; Tufano, F.; De Nola, F.; Di Blasio, G.; Gimelli, A.; Misul, D.A.; Toscano, G. A Review of Model Predictive Controls Applied to Advanced Driver-Assistance Systems. *Energies* **2021**, *14*, 7974. [CrossRef]
15. Van Parys, R.; Pipeleers, G. Distributed MPC for Multi-Vehicle Systems Moving in Formation. *Robot. Auton. Syst.* **2017**, *97*, 144–152. [CrossRef]
16. Tan, Q.; Dai, P.; Zhang, Z.; Katupitiya, J. MPC and PSO Based Control Methodology for Path Tracking of 4WS4WD Vehicles. *Appl. Sci.* **2018**, *8*, 1000. [CrossRef]
17. Hu, M.; Li, C.; Bian, Y.; Zhang, H.; Qin, Z.; Xu, B. Fuel Economy-Oriented Vehicle Platoon Control Using Economic Model Predictive Control. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 20836–20849. [CrossRef]
18. Czibere, S.; Domina, Á.; Bárdos, Á.; Szalay, Z. Model Predictive Controller Design for Vehicle Motion Control at Handling Limits in Multiple Equilibria on Varying Road Surfaces. *Energies* **2021**, *14*, 6667. [CrossRef]
19. Aulia, A.I.; Hindersah, H.; Rohman, A.S.; Hidayat, E. Design of MPC-based motion cueing for 4 DoF simulator platform. In Proceedings of the International Conference on System Engineering and Technology, Shah Alam, Malaysia, 7 October 2019; pp. 183–188.
20. Buchheit, B.; Schneider, E.N.; Alayan, M.; Dauth, F.; Strauss, D.J. Motion Sickness Prediction in Self-Driving Cars Using the 6DOF-SVC Model. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 13582–13591. [CrossRef]
21. Antehunegn, Y.; Belete, M. Control of 8-DOF Vehicle Model Suspension System by Designing Second Order SMC Controller. *GSJ* **2020**, *8*, 272–280.
22. Chen, S.p.; Xiong, G.m.; Chen, H.y.; Negrut, D. MPC-based path tracking with PID speed control for high-speed autonomous vehicles considering time-optimal travel. *J. Cent. South Univ.* **2020**, *27*, 3702–3720. [CrossRef]
23. Li, Y.; Deng, H.; Xu, X.; Wang, W. Modelling and Testing of In-Wheel Motor Drive Intelligent Electric Vehicles Based on Co-Simulation With Carsim/Simulink. *IET Intell. Transp. Syst.* **2019**, *13*, 115–123. [CrossRef]
24. Jiang, N.; Qiu, R. Modelling and Simulation of Vehicle ESP System Based on CarSim and Simulink. *J. Physics Conf. Ser.* **2022**, *2170*, 012032. [CrossRef]
25. CarSim. Using CarSim and TruckSim. Available online: https://www.carsim.com/publications/technical/index.php (accessed on 4 March 2024).
26. Hosseinzadeh, M.; Sinopoli, B.; Kolmanovsky, I.; Baruah, S. Robust-to-Early Termination Model Predictive Control. *IEEE Trans. Autom. Control* **2024**, *69*, 2507–2513. [CrossRef]
27. Gharbi, M.; Ebenbauer, C. Anytime MHE-based output feedback MPC. *IFAC-PapersOnLine* **2021**, *54*, 264–271. [CrossRef]
28. Borrelli, F.; Bemporad, A.; Morari, M. *Predictive Control for Linear and Hybrid Systems: Textbook*; Cambridge University Press: Cambridge, UK, 2017; pp. 251–287.
29. Allgöwer, F.; Findeisen, R.; Nagy, Z.K. Nonlinear Model Predictive Control: From Theory to Application. *J. Chin. Inst. Chem. Eng.* **2004**, *35*, 299–315.
30. Patne, V.; Ingole, D.; Sonawane, D. Towards Fast Nonlinear Model Predictive Control for Embedded Applications. *IFAC-PapersOnLine* **2022**, *55*, 304–309. [CrossRef]
31. Kang, C.M.; Lee, S.H.; Chung, C.C. Linear Parameter Varying Design for Lateral Control using Kinematics of Vehicle Motion. In Proceedings of the Annual American Control Conference, Milwaukee, WI, USA, 27–29 June 2018; pp. 3239–3244.
32. Sánchez, G.; Murillo, M.; Genzelis, L.; Deniz, N.; Giovanini, L. MPC for Nonlinear Systems: A Comparative Review of Discretization Methods. In Proceedings of the 017 XVII Workshop on Information Processing and Control, Mar del Plata, Argentina, 4–9 December 2017; pp. 1–6.
33. Nocedal, J.; Wright, S. Penalty and Augmented Lagrangian Methods. In *Numerical Optimization*; Springer: Berlin/Heidelberg, Germany, 2006; Chapter 17, pp. 497–528.
34. Nesterov, Y.E. A Method for Solving the Convex Programming Problem with Convergence Rate O($\frac{1}{k^2}$) . *Dokl. Akad. Nauk SSSR* **1983**, *269*, 543–547.
35. Markus, K.; Rolf, F. A Fast Gradient Method for Embedded Linear Predictive Control. *IFAC Proc. Vol.* **2011**, *44*, 1362–1367.
36. Berry, A.; Lemus, D.; Babuška, R.; Vallery, H. Directional Singularity-Robust Torque Control for Gyroscopic Actuators. *IEEE/ASME Trans. Mechatronics* **2016**, *21*, 2755–2763. [CrossRef]
37. Ayyildiz, E.; Gazi, V.P.; Wit, E. A Short Note on Resolving Singularity Problems in Covariance Matrices. *Int. J. Stat. Probab.* **2012**, *1*, 113–118. [CrossRef]

38. Varga, R.S. Geršgorin and His Circles. *Springer Ser. Comput. Math.* **2004**, *36*, 226.
39. Adhau, S.; Phalke, K.; Nalawade, A.; Patil, S.; Ingole, D.; Sonawane, D. Implementation and Analysis of Offset-Free Explicit Model Predictive Controller on FPGA. In Proceedings of the Indian Control Conference, New Delhi, India, 18–20 December 2019; pp. 231–236.
40. Sonawane, D.; Hanwate, S.; Ubare, P.; Marathe, R.; Rao, G.S.; Sahu, M. Development of Vehicle Dynamic Plant Model and Embedded Co-Simulation with ARM Platform. In Proceedings of the IEEE International Conference on Power Electronics, Drives and Energy Systems, Jaipur, India, 14–17 December 2022; Volume 1, pp. 1–6.
41. Mehdi Khosrow-Pour, D. Autonomous Vehicles. In *Encyclopedia of Information Science and Technology*; IGI Global: Hershey, PA, USA 2020; Chapter 1, pp. 1–11.
42. Fink, M. Implementation of Linear Model Predictive Control–Tutorial. *arXiv* **2021**, arXiv:2109.11986.
43. Liu, Z.; Stursberg, O. Recursive Feasibility and Stability of MPC with Time-Varying and Uncertain State Constraints. In Proceedings of the 18th European Control Conference, Naples, Italy, 25–28 June 2019; pp. 1766–1771.