Model Predictive Control for Self-Driving Cars

Alex Gisi

May 12, 2023

1 Introduction

In 2019 there were 36,500 traffic fatalities in the United States, and vehicle crashes cost the economy $340 billion [1]. In recent years, the number of traffic fatalities has been increasing, with almost 43,000 in 2021 [2]. Research in self-driving car technology offers a way to reduce the human and financial cost of crashes, as well as lower road congestion and ease the burden of drivers [3]. Toward this end, model predictive control has become a popular tool for enabling autonomous navigation. Model predictive control is well-suited to the self-driving car problem due to a number of factors, including ability to handle obstacle constraints and nonlinear vehicle models. The Society of Automotive Engineers (SAE) defined a classification system for the level of autonomy in a given vehicle, with level 0 indicating no driving automation and level 5 indicating that human intervention is never required [4]. In this report, we focus on control design, existing research, and examples for SAE level 4 autonomous vehicles using model predictive control, meaning driving without any human input in commonly occuring situations.

2 Overview

Model predictive control (MPC) is a general control methodology which has seen significant academic and industrial attention in the last 50 years. MPC aims to find control inputs which will drive a system toward a predefined reference state, given some constraints and a cost function which are functions of the control inputs and process state. To do so, the controller uses a model of the system. At each control step, the controller minimizes the cost with respect to a sequence of future inputs; via the model, the controller predicts how the future inputs will affect the cost function. The first control in the resulting sequence is physically applied to the process until the next control step, whereupon the optimization is repeated in a receding horizon fashion [5]. Accordingly, the defining characteristics of MPC are: the control law depending on predicted behavior; the outputs predictions are computed by a process model; the input is determined by optimising some measure of performance; and the receding horizon [6]. Significant benefits of model predictive control include conceptual simplicity, the ability to explicitly incorporate constraints in an "intellligent" fashion, easy extension

to the multivariable case, and ability to handle problems lacking a control law that can be computed off-line [5–7].

MPC has become popular in self-driving research as a way to provide online trajectory generation and tracking. Trajectory generation is the problem of finding a sequence of vehicle states which satisfy the car's kinematic and dynamic constraints as well as taking into consideration comfort and safety of passengers. A common application is replanning a reference trajectory in order to avoid a pop-up obstacle, as demonstrated in [8–10]. The open-loop trajectory generated by a controller with a cost function or constraints taking into account the obstacle and vehicle dynamics can be used to accomplish this sort of replanning. Trajectory tracking is the problem of computing from a sequence of states actuator (steering, throttle, brake) commands which result in the vehicle following the proposed trajectory in the closed-loop [11]. MPC is well-suited for both due to its ability to handle multiple inputs and outputs, consider time-dependent constraints like pop-up obstacles, use a nonlinear vehicle model, and track a potentially unfeasible reference trajectory [12].

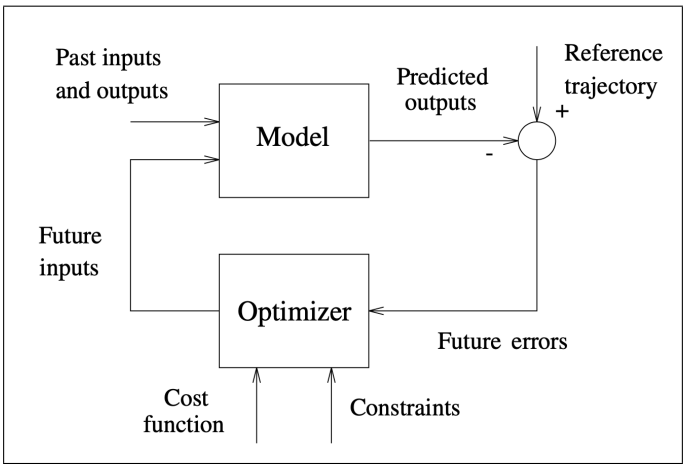


Figure 1: MPC Flowchart [5]

2.1 Examples

One approach for control designs addressing both problems is the use of a hierarchical architecture in which a top-level algorithm replans a reference trajectory to avoid obstacles, then sends the trajectory to a low-level algorithm which computes actuator commands to track the replanned trajectory. The top-level controller uses a lower-fidelity vehicle model to ease computational demands, whereas the low-level controller uses a higher-fidelity model to accurately track the reference. The high-level controller is sampled at a lower rate than the low-level. Also, the two may have different prediction and control horizons. For example, [8] compared a one-level controller using a

four-wheel vehicle model to a hierarchical controller using a point-mass model at the high (trajectory generation) level and a four-wheel vehicle model at the low (trajectory tracking) level. In a double lane change and obstacle avoidance manuveur on an icy road, the hierarchical controller could drive 15kph faster and took 39% less computation time.

A number of MPC designs for specific self-driving applications exist in the literature which assume a deterministic, observable environment [8–10, 13–17]. Naturally, such conditions do not hold for general driving. As Carvalho et. al. note, "driving requires forecasts, and forecasts can be highly uncertain in some driving scenarios." Such uncertainty can arise from measurement errors, friction coefficient estimation, driver behaviour and model mismatch [18].

One approach to handling uncertainty is robust MPC, which takes into account system disturbances to guarantee the control algorithm does not violate constraints. RMPC has been implemented in driving scenarios using a so-called tube-based approach which depends on offline computation of reachable vehicle states to perform constraint tightening. The offline computation is enabled by assuming deterministic, bounded uncertainties. The tube-based approach in [19] designed a robust controller based on a force-input nonlinear bicycle model by estimating the bounds of possible disturbances using experimental data. The controller was implemented in a full-size car to avoid obstacles on an icy road. It achieved collision-free path tracking up to 80kph with computational demands similar to standard MPC.

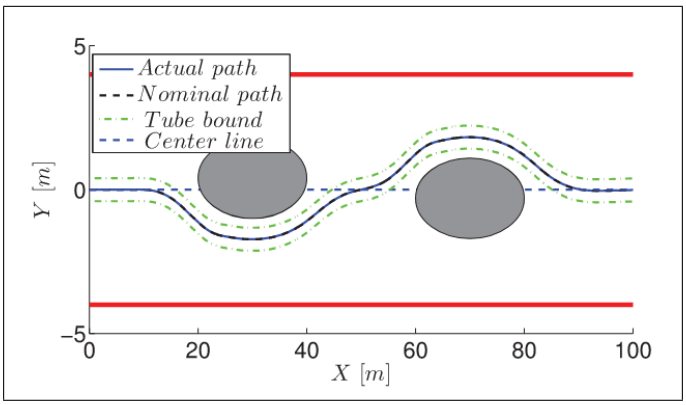


Figure 2: RMPC via a 'tube' [19]

A robust approach, however, can result in overly conservative behavior in common driving situations. Stochastic MPC reduces conservatism by allowing a small probability the constraints are violated, resulting in so-called chance constraints. This results in more natural, human-like manuevering. Furthermore, the driving aggressiveness is a tunable parameter. Instead of deterministic uncertainty like RMPC, the uncertainty is typically modelled with zero-mean Gaussian noise. In contrast to the development of

nonlinear MPC methods for self-driving applications [10, 17, 20], general SMPC research has mostly focused on linear systems [21]. In [22], a linear time varying vehicle model is used with chance constraints to implement SMPC for navigating in the presence of other (unpredictable) vehicles. An Interacting Multiple Model Kalman Filter is used to estimate the positions of the other vehicles. The authors experimentally examine the effect of changing the risk parameter, i.e. the chance of constraint violation, in urban driving, and concluded altering the parameter had a significant effect on the vehicle's navigational characteristics in urban environments.

3 Problem Formulation

The MPC problem described above is commonly formulated as the following constrained finite-time optimization problem.

$$\underset{U}{\operatorname{argmin}} \operatorname{Cost}(\cdot; T_p, T_c) \tag{1a}$$

subj. to

$$x_{k+1} = f(x_k, u_k) \qquad k = 0, \ldots, H_p - 1 \tag{1b}$$

$$u_k \in \mathcal{U} \qquad k = 0, \ldots, H_p - 1 \tag{1c}$$

$$x_k \in \mathcal{X} \qquad k = 0, \ldots, H_p - 1 \tag{1d}$$

$$u_k = u_{H_c - 1} \qquad k = H_c, \ldots, H_p - 1 \tag{1e}$$

$$x_0 = \hat{x}_0 \tag{1f}$$

The vectors $x \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ are the state and input vectors. Likewise, the sequences $X = \{x_0, \ldots, x_{H_p - 1}\}$ and $U = \{u_0, \ldots, u_{H_p - 1}\}$ are the generated sequences of state and input vectors.

Two time horizons are considered. The prediction horizon H_c is the length of time over which the response of the model to the inputs will be forecasted. The control horizon H_c is the length of time for which the controlled variables will be optimized. Naturally, $H_c \leq H_p$. The relation of the control and prediction horizons to the rest of the MPC process is shown in Figure 3.

The constraint (1b) represents using the system model to forecast future states. Constraints (1c) and (1d) enforce limitations on the input and process states, while (1e) represents that all controls after the control horizon are held constant at the value of the last control within the control horizon. Constraint (1f) initializes the system model to the the actual value of the system $\hat{x}$, which may have to be estimated.

The first control of the solution $U = \{u_0, \ldots, u_{H_p - 1}\}$ to (1a) is applied to the system for the next control period. The sampling rate of the control is typically assumed to be constant, but need not be in practice.

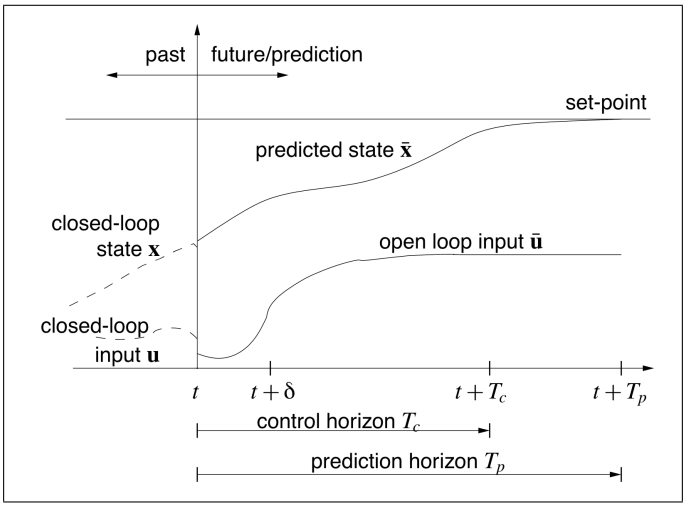


Figure 3: The MPC idea [23]

Note we assume the problem starts from the zero time for notational simplicity. At each control instance, the problem is repeated (with the same control and prediction horizons), so in general each problem starts from an offset.

Practically speaking, solving the optimization problem can be achieved with one of a number of freely available optimiization libraries. In particular, Ipopt is a library for nonlinear optimization with interfaces to general-purpose programming languages [24]. The MATLAB function fmincon is a powerful optimization tool and is called under the hood of MATLAB's high-level MPC abstractions [25, 26]. The author also found useful the CVXPY modeling language for convex optimization problems [27, 28].

3.1 Model

If the process model $f(x_k, u_k)$ is linear, the technique is known as linear model predictive control. The most common linear models are state space, transfer functions, impulse response, and step response models [6]. Of these, the state space model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned}$$

is especially popular, in particular because of easily handling multiple-input multiple-output processes.

Of course, many actual processess – including vehicle dynamics – are nonlinear. While linearization may be feasible if the process tends to operate around a set point, for processes with extreme nonlinearities or particular startup/shutdown modes, we may wish to treat them as nonlinear explicitly instead of inducing more model mismatch via

linearization. In this case, we use the general process model

$$x_{k+1} = f(x_t, u_t)$$
$$x_0 = \hat{x}_0$$

as in equation (1). The trade-off for more accurate process representation is more difficult system identification, increased computational complexity of the optimization problem, and the lack of nonlinear stability and robustness results [5].

3.2 Cost function

The cost function is usually implemented as a sum over "stage costs", where each stage is a point in the prediction horizon:

$$\text{Cost}(\cdot; T_p, T_c) = \sum_{k=0}^{H_p-1} \|x_k - x_k^{\text{ref}}\|_Q^2 + \sum_{k=0}^{H_c-1} \left(\|u_k\|_R^2 + \|\Delta u_k\|_S^2\right)$$

where the control increment is

$$\Delta u_k = u_k - u_{k-1} \;\forall\; k = 0, \ldots, H_c - 1;\; u_{-1} = u(-t_s)$$

which is initialized by taking the control input from the previous solution of the optimization problem $u(-t_s)$, i.e. t_s is the sampling time.

The norm $\|\cdot\|_A^2$ is the Euclidean norm weighted by the symmetric positive semidefinite matrix A. Accordingly, the matrices $Q \in \mathbb{R}^{n\times n}$ and $R, S \in \mathbb{R}^{m\times m}$ are the tracking, input, and input increment costs, respectively. The input increment cost may be dropped depending on the application, i.e. set $S = 0$.

An immediate concern is how to select the parameters of the cost function, since they will apparently have a significant impact on the performance of the closed-loop control. The selection of the prediction and control horizons and the weight matrices are distinct, but related, questions. In principle, best performance is given when control and prediction horizons are infinite, i.e. $T_c = T_p = \infty$ [23]. In reality, we have to solve the control problem in a reasonable time, which requires truncating the horizons. Reasoning from the ideal case, we should set the horizons as long as possible while maintaining acceptable real-time performance. A small value of the control horizon does not give the controlller enough degrees of freedom to incorporate information about the future trajectory, so $H_c \geq 3$ works well in practice. Furthermore, to take into account transient behavior of the process, the $H_p - H_c$ should be larger than the process settling time.

Adjustment of the weight matrices is a mostly ad-hoc process process. One should note that for a smaller control horizon, the same R, S matrices will have less effect on the cost than for a larger control horizon. Examples of how the prediction and control horizons and objective weights affect a SISO plant are given in chapter five of [6].

4 Autonomous Navigation

The adaption of model predictive control to autonomous navigation is quite natural; in fact, driving is a common analogy for explaining the principle ideas of MPC [6, 29]. Depending on the particular model, we derive state and input vectors representing physical attributes of the vehicle. Then we can use the MPC formulation outlined above, with some previously planned path serving as the nominal trajectory, to enable trajectory replanning and following.

4.1 Optimization Problem

The control problem for an autonomous vehicle without obstacle avoidance can be represented as the optimization problem

$$\underset{U}{\operatorname{argmin}} \sum_{k=0}^{H_p-1} \|\xi_k - \xi_k^{\text{ref}}\|_Q^2 + \sum_{k=0}^{H_c-1} \left(\|u_k\|_R^2 + \|\Delta u_k\|_S^2\right) \tag{2a}$$

subj. to

$$\xi_{k+1} = f(\xi_k, u_k), \qquad k = 0, \ldots, H_p - 1 \tag{2b}$$

$$u_{\min} \leq u_k \leq u_{\max} \qquad k = 0, \ldots, H_c - 1 \tag{2c}$$

$$\xi_{\min} \leq \xi_k \leq \xi_{\max} \qquad k = 0, \ldots, H_c - 1 \tag{2d}$$

$$u_k = u_{H_c-1}, \qquad k = H_c, \ldots, H_p - 1 \tag{2e}$$

$$\xi_0 = \hat{\xi}_0 \tag{2f}$$

$$\Delta u_k = u_k - u_{k-1}, \qquad k = 0, \ldots, H_c - 1 \tag{2g}$$

$$u_{-1} = u(-t_s) \tag{2h}$$

where ξ, Ξ have been substituted for the $x, \mathcal{X}$ in Section 3 to represent the system state to distinguish it from the vehicle's longitudinal position in the body frame. The state and inputs have box constraints due to the physical nature of the vehicle.

4.2 Car Models

We present two systems commonly used to model the vehicle in self-driving applications. Figure 4 describes notation used in both. We consider bicycle models, which lump together left and right front tires and left and right back tires into (imaginary) single front and rear tires.

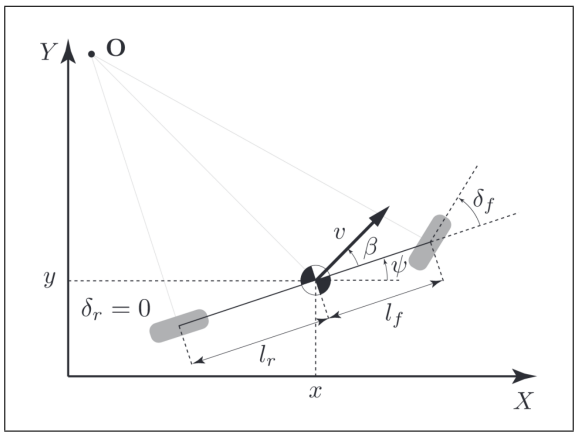


Figure 4: Bicycle model [12]

While the notion of a bicycle model may seems reductive, bicycle models are popular in control design [8, 19, 22] for their balance between an accurate representation and computational simplicity. Kong et. al. examined the effectiveness of the kinematic and dynamic models by comparing the models' open loop predictions to measurements obtained from a real car driven by an experienced driver [12]. They concluded the kinematic bicycle model provided satisfactory control performance in a range of experiments, and held several advantages over the dynamic model. Nevertheless, both models have their place in control design.

4.2.1 Kinematic Bicycle Model

The kinematic bicycle model assumes the vehicle movement is a function of its geometry.

$$\begin{aligned}
\dot{X} &= v\cos(\psi+\beta) \\
\dot{Y} &= v\sin(\psi+\beta) \\
\dot{\psi} &= \frac{v}{l_r}\sin\beta \\
\dot{v} &= a \\
\beta &= \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan\delta\right)
\end{aligned}$$

The model locates the vehicle in the global coordinates X, Y with yaw ψ radians counterclockwise from the X axis. The velocity is a direct function of the acceleration a. The slip angle β is obtained directly from geometric calculations. Note from Figure 4 that l_r and l_f represent the distance from the center of gravity to the rear and front wheelbase, respectively. The vehicle state vector is $\xi = [X, Y, \psi, v, \beta]^T$ and the input vector is $u = [a, \delta]^T$.

4.2.2 Dynamic Bicycle Model

The dynamic bicycle model incorporates the vehicle's interaction with the road via the lateral forces on the front and rear tires $F_{c,f}$ and $F_{c,r}$, respectively.

$$\begin{aligned}
\ddot{x} &= \dot{\psi}\dot{y} + a \\
\ddot{y} &= -\dot{\psi}\dot{x} + \frac{2}{m}\left(F_{c,f}\cos\delta + F_{c,r}\right) \\
\ddot{\psi} &= \frac{2}{I_z}\left(l_f F_{c,f} - l_r F_{c,r}\right) \\
\dot{X} &= \dot{x}\cos\psi - \dot{y}\sin\psi \\
\dot{Y} &= \dot{x}\sin\psi + \dot{y}\cos\psi
\end{aligned}$$

The dynamic bicycle model captures the longitudinal and lateral accelerations in the vehicle frame $\ddot{x}$ and $\ddot{y}$ respectively, as well the the angular acceleration $\ddot{\psi}$.

To calculate the tire forces, standard control designs use the linear tire model

$$F_{c,i} = -C_{\alpha_i}\alpha_i$$

where $i \in \{f, r\}$, α_i is the tire slip angle and C_{α_i} is the tire cornering stiffness. Most vehicle manuevers falls into the range of linear tire forces [8]. However, there are a number of tire models exist which capture the nonlinearities with various complexity, such as the uni-tire, magic formula, LuGre, and Fiala [18, 30]. The tradeoff for better tire representation in extreme maneuvers is more difficult parameter estimation.

4.3 Obstacle Avoidance

Naturally, we want to be able to replan in the presence of obstacles in the trajectory, whether those represent other cars or static things like debris in the road. We focus on two means of enabling obstacle avoidance. The first uses constraints to guarantee the trajectory can not pass over the obstacle. The second uses an additional term in the cost function to represent distance from the obstacle, thereby repelling the vehicle away from it. We introduce the idea in this section and examine their implementation in Section 5.

By adding a constraint on the position of the vehicle with respect to the obstacle, we can eliminate from contention those paths which would cause a collision.

$$\text{dist}(\text{car}, \text{obstacle}) + \text{safetyMargin} \geq 0$$

Since establishing the length of the shortest line between the car and the obstacle is an optimization problem in itself, finding $\text{dist}(\text{car}, \text{obstacle})$ is hard in general. In [31], a method for reformulating general collision avoidance constraints into smooth, differentiable contraints to improve optimization efficiency is given.

Accordingly, we seek a more computationally efficient method for obstacle avoidance. In [8], the obstacle distance cost $\text{Cost}_{\text{obs},t}$ for time t is added to the cost function

$$\text{Cost}_{\text{obs},t} = \frac{K \cdot v_t}{\text{dist}(\text{car}, \text{obstacle}) + \epsilon}$$

where K is a weighting term, v_t is the predicted velocity at time t, and ϵ a small value to avoid singularity.

5 Experiments

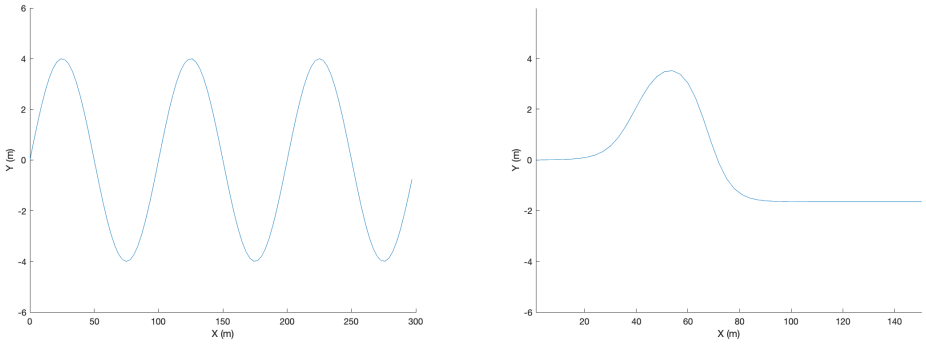
We implemented the control design presented in Section 4 in order to test performance in different driving situations. Two reference trajectories were used. The first, ref1, is the sinusoidal trajectory

$$Y^{\text{ref}}(X) = 4 \sin\left(\frac{2\pi}{100} X\right)$$
$$\psi^{\text{ref}}(X) = \tan^{-1}\left(\left(\frac{8\pi}{100}\right) \cos\left(\frac{2\pi}{100} X\right)\right)$$

The second, ref2, is the double lane change maneuver

$$Y^{\text{ref}}(X) = \frac{d_{y1}}{2}(1 + \tanh(z_1)) - \frac{d_{y2}}{2}(1 + \tanh(z_2))$$
$$\psi^{\text{ref}}(X) = \tan^{-1}\left(d_{y1}\left(\frac{1}{\cosh(z_1)}\right)^2 \left(\frac{1.2}{d_{x1}}\right) - d_{y2}\left(\frac{1}{\cosh(z_2)^2}\right)^2 \left(\frac{1.2}{d_{x2}}\right)\right)$$

defined in [8], where $z_1 = \frac{2.4}{25}(X - 27.19) - 1.2$, $z_2 = \frac{2.4}{21.95}(X - 56.46) - 1.2$, $d_{x1} = 25$, $d_{x2} = 21.95$, $d_{y1} = 4.05$, $d_{y2} = 5.7$.



(a) ref1 (b) ref2

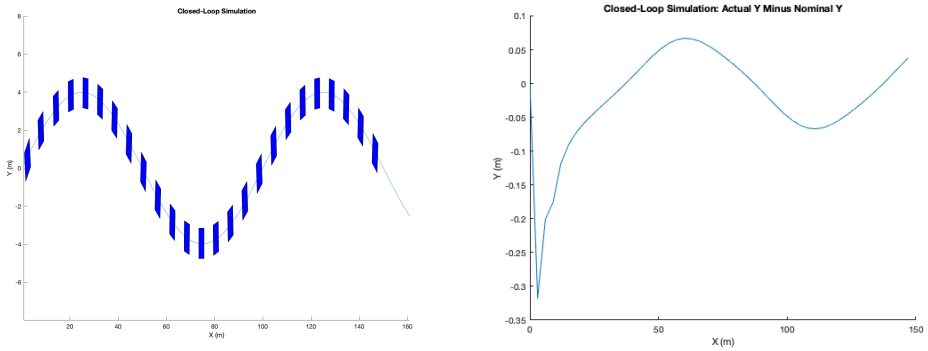
Figure 5: Reference trajectories

We implemented the input constraints

$$-1 \leq a \leq 1$$
$$-\frac{\pi}{4} \leq \delta \leq \frac{\pi}{4}$$

to reflect the physical limitations of a real car.

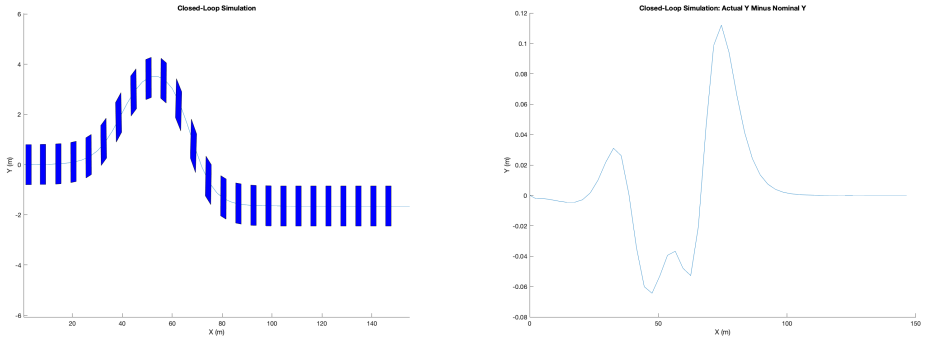
The initial state was $\xi = [0, 0, 5, 0]$, or an initial velocity of 5 m/s. Without the presence of obstacles, the closed loop result almost exactly matches the reference trajectory ($H_p = 5, H_c = 2$).



(a) ref1: closed loop (b) ref1: $Y^{\text{ref}} - Y$

Figure 6: ref1 closed-loop performance at $v^{\text{ref}} = 5$.

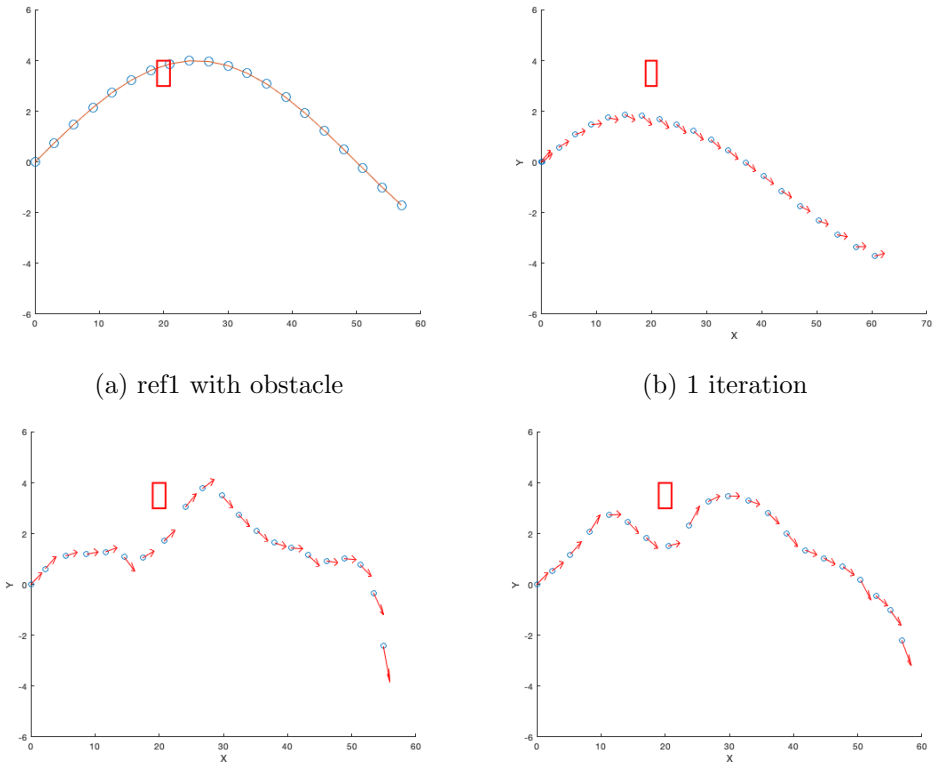
Note the small scale in Figure 7b, which shows the lateral trajectory almost exactly matched the reference. The same can be seen for ref2.



(a) ref2: closed loop (b) ref1: $Y^{\text{ref}} - Y$

Figure 7: ref2 closed-loop performance at $v^{\text{ref}} = 5$.

Now we turn our attention to the obstacle avoidance techniques introduced in Section 4.3. We define an obstacle as a rectangle along ref1. The vehicle must maneuver to avoid the obstacle to prevent a collision. We first examine the contraint-based technique in open-loop planning, in order to easily observe the properties of the generated trajectory. We restrict the number of the iterations the solver can take in order to reflect the potential real-time constraint when driving.



(a) ref1 with obstacle

(b) 1 iteration

(c) 5 iterations

(d) 10 iterations

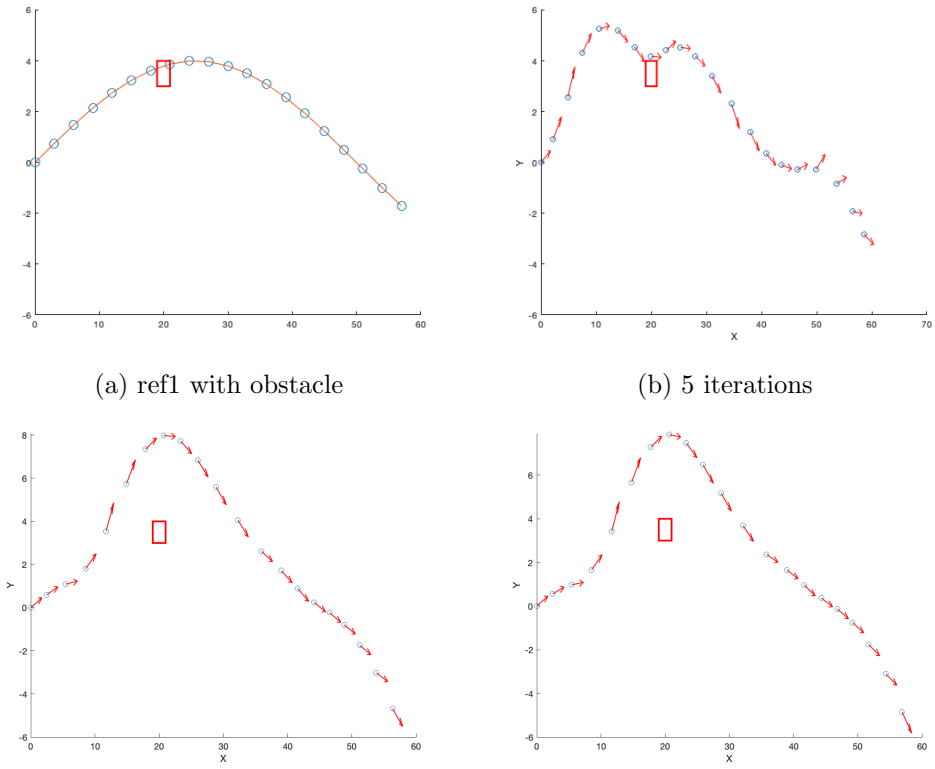
Figure 8: Reference and generated paths using contraint avoidance

As the iterations increase, the paths become progressively smoother and converge with respect to the reference trajectory. However, the fact that every iteration has to calculate the distance to the obstacle at each control instance over the prediction horizon results in a significant computational cost which increases linearly with the number of iterations (Figure 10b).

On the other hand, the cost-based avoidance method is much less computationally demanding (Figure 10a). We implemented the approach described in [8] with $K = 0.5$, $\epsilon = 0.01$ on ref1 with the same obstacle as before.

In contrast the constraint approach, as shown in the example restricted to five

iterations, there is no guarantee the controller will not plan a trajectory arbitrarily close to the obstacle. Furthermore, the obstacle cost term K must be determined on an ad-hoc basis and different values could give better or worse results depending on the exact scenario.



(a) ref1 with obstacle

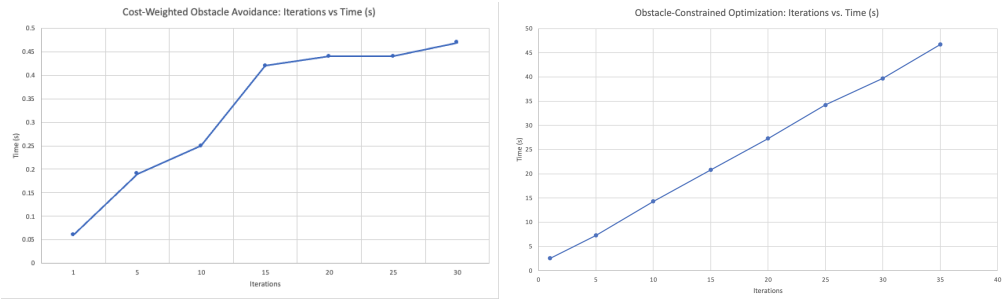
(b) 5 iterations

(c) 10 iterations

(d) 15 iterations

Figure 9: Reference and generated paths using cost-based avoidance

Note the scales in Figure 10 differ by an order of magnitude. Since real-time implementation is a central concern for MPC schemes, the computational benefit of a cost-term approach may be desirable, but depending on the application, the greater predictability of constraints could be necessary.



(a) Cost-weighted avoidance (b) Constraint avoidance

Figure 10: Computational cost of NMPC obstacle-avoidance schemes.

6 Conclusion

Model predictive control has been applied to many industrial processes with tremendous success. Its natural adaptation to autonomous navigation has resulted in many viable control designs, especially for linear vehicle models. The main challenge in MPC for autonomous vehicles has been the computational challenges associated with maintaining a high sampling rate while using a model detailed enough to capture nonlinearities. While advances in computing power have made such a trade-off feasible, mathematical advances in formulating the control problem continue to play their part. Much research now is focused on the concepts of robust and stochastic model predictive control, which is required to deal with the inherent environmental uncertainties encountered when driving. Possible extensions to the work described in this report include taking into account model parameter uncertainties or estimating model parameters online.

As noted in the introduction, the ultimate goal of researching autonomous navigation MPC schemes is to enable their implementation in consumer vehicles. To this end, the author believes a more general framework for choosing the vehicle's actions is required to supervise any MPC implementations in the vehicle. To this end, future work on autonomous navigation should focus on general artificial intelligence for decision-making under uncertainty.

References

[1] N. H. T. S. Administration, "The economic and societal impact of motor vehicle crashes, 2019 (revised)," Feb. 2023. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813403.

[2] N. H. T. S. Administration, "Early estimates of motor vehicle traffic fatalities and fatality rate by sub-categories in 2021," May 2022. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813298.

[3] S. D. Pendleton *et al.*, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[4] S. O.-R. A. V. S. Committee *et al.*, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *SAE International: Warrendale, PA, USA*, 2018.

[5] E. F. Camacho and C. Bordons, *Model predictive control* (Advanced textbooks in control and signal processing), eng, 2nd ed. London New York: Springer, 2004, ISBN: 978-0-85729-398-5.

[6] J. A. Rossiter, *Model-based predictive control: a practical approach*. CRC press, 2003.

[7] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," en, p. 26, 2000.

[8] Y. Gao, *Model predictive control for autonomous and semiautonomous vehicles*. University of California, Berkeley, 2014.

[9] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *Dynamic systems and control conference*, vol. 44175, 2010, pp. 265–272.

[10] M. A. Abbas, R. Milman, and J. M. Eklund, "Obstacle Avoidance in Real Time With Nonlinear Model Predictive Control of Autonomous Vehicles," *Canadian Journal of Electrical and Computer Engineering*, vol. 40, no. 1, pp. 12–22, 2017, Conference Name: Canadian Journal of Electrical and Computer Engineering, ISSN: 0840-8688. DOI: 10.1109/CJECE.2016.2609803.

[11] C. Badue *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113 816, 2021.

[12] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, ISSN: 1931-0587, 2015, pp. 1094–1099. DOI: 10.1109/IVS.2015.7225830.

[13] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli, "Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, ISSN: 2153-0017, Oct. 2013, pp. 378–383. DOI: 10.1109/ITSC.2013.6728261.

[14] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, Feb. 2017, Conference Name: IEEE Transactions on Vehicular Technology, ISSN: 1939-9359. DOI: 10.1109/TVT.2016.2555853.

[15] B.-C. Chen, B.-C. Luan, and K. Lee, "Design of lane keeping system using adaptive model predictive control," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, ISSN: 2161-8089, Aug. 2014, pp. 922–926. DOI: 10.1109/CoASE.2014.6899436.

[16] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 174–179. DOI: 10.1109/IVS.2017.7995716.

[17] P. Falcone *et al.*, "Nonlinear model predictive control for autonomous vehicles," 2007.

[18] A. Carvalho, S. Lefévre, G. Schildbach, J. Kong, and F. Borrelli, "Automated driving: The role of forecasts and uncertainty—A control perspective," en, *European Journal of Control*, SI: ECC15, vol. 24, pp. 14–32, Jul. 2015, ISSN: 0947-3580. DOI: 10.1016/j.ejcon.2015.04.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S094735801500062X (visited on 11/15/2022).

[19] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," en, *Vehicle System Dynamics*, vol. 52, no. 6, pp. 802–823, Jun. 2014, ISSN: 0042-3114, 1744-5159. DOI: 10.1080/00423114.2014.902537. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/00423114.2014.902537 (visited on 09/27/2022).

[20] B. Yi, P. Bender, F. Bonarens, and C. Stiller, "Model Predictive Trajectory Planning for Automated Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 24–38, Mar. 2019, Conference Name: IEEE Transactions on Intelligent Vehicles, ISSN: 2379-8904. DOI: 10.1109/TIV.2018.2886683.

[21] A. Mesbah, "Stochastic Model Predictive Control: An Overview and Perspectives for Future Research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, Dec. 2016, Conference Name: IEEE Control Systems Magazine, ISSN: 1941-000X. DOI: 10.1109/MCS.2016.2602087.

[22] A. Carvalho, Y. Gao, S. Lefevre, and F. Borrelli, *Stochastic predictive control of autonomous vehicles in uncertain environments*. Sep. 2014.

[23] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," vol. 11, 2002. [Online]. Available: https://pure.tue.nl/ws/files/3079152/555518.pdf#page=120.

[24] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[25] Mathworks, *Find minimum of constrained nonlinear multivariable function*. [Online]. Available: https://www.mathworks.com/help/optim/ug/fmincon.html#busp5fq-6.

[26] Mathworks, *Nonlinear MPC*. [Online]. Available: https://www.mathworks.com/help/mpc/ug/nonlinear-mpc.html.

[27] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[28] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[29] J. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Systems Magazine*, vol. 20, no. 3, pp. 38–52, Jun. 2000, Conference Name: IEEE Control Systems Magazine, ISSN: 1941-000X. DOI: 10.1109/37.845037.

[30] H. Guo, Z. Yin, D. Cao, H. Chen, and C. Lv, "A Review of Estimation for Vehicle Tire-Road Interactions Toward Automated Driving," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 1, pp. 14–30, Jan. 2019, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics: Systems, ISSN: 2168-2232. DOI: 10.1109/TSMC.2018.2819500.

[31] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-Based Collision Avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, May 2021, Conference Name: IEEE Transactions on Control Systems Technology, ISSN: 1558-0865. DOI: 10.1109/TCST.2019.2949540.