Embedded Optimization Algorithms for Steering in Autonomous Vehicles based on Nonlinear Model Predictive Control

Quirynen, R.; Berntorp, K.; Di Cairano, S.

TR2018-089   July 13, 2018

Abstract

Steering control for autonomous vehicles on slippery road conditions, such as on snow or ice, results in a highly nonlinear and therefore challenging online control problem, for which nonlinear model predictive control (NMPC) schemes have shown to be a promising approach. NMPC allows to deal with the nonlinear vehicle dynamics as well as the system limitations and geometric constraints in a rather natural way, given a desired trajectory that can be provided by a supervisory algorithm for path planning. Our aim is to study the real-time feasibility of NMPC-based steering control on an embedded computer and the importance of the appropriate vehicle model selection, the optimization solver choice and control horizon length. The presented computation times have been obtained on a Raspberry Pi 2 model, as a proof of concept for a future real-world implementation on an embedded microprocessor.

American Control Conference (ACC)

Embedded Optimization Algorithms for Steering in Autonomous Vehicles based on Nonlinear Model Predictive Control

Rien Quirynen[1], Karl Berntorp[1], Stefano Di Cairano[1]

Abstract— **Steering control for autonomous vehicles on slippery road conditions, such as on snow or ice, results in a highly nonlinear and therefore challenging online control problem, for which nonlinear model predictive control (NMPC) schemes have shown to be a promising approach. NMPC allows to deal with the nonlinear vehicle dynamics as well as the system limitations and geometric constraints in a rather natural way, given a desired trajectory that can be provided by a supervisory algorithm for path planning. Our aim is to study the real-time feasibility of NMPC-based steering control on an embedded computer and the importance of the appropriate vehicle model selection, the optimization solver choice and control horizon length. The presented computation times have been obtained on a Raspberry Pi 2 model, as a proof of concept for a future real-world implementation on an embedded microprocessor.**

I. INTRODUCTION

Many recent publications have shown the potential benefits of using Nonlinear Model Predictive Control (NMPC) for steering control in autonomous vehicle systems, e.g. [1–6] based on closed-loop simulation results and [7–10] based on real-world experimental results. Most of these prior works found the use of nonlinear optimization tools impractical because of the tight timing constraints in such a safety critical embedded application. A nonlinear and nonconvex Optimal Control Problem (OCP) needs to be solved at each sampling time instant under stringent timing requirements. For this purpose, tailored continuation-based online algorithms have been developed for solving these nonlinear optimal control problems in real-time [11], [12].

Depending on the targeted application, the dynamic vehicle model that is used in the NMPC scheme can exhibit different levels of complexity, ranging from a simplified single-track (ST) or bicycle model in [4], [5], [8], [10] to an ST model combined with Pacejka tire modeling in [1], [7] and a complete double-track (DT) vehicle model in [2], [6]. A detailed overview of these different vehicle models and of their use in an optimal control problem formulation, can be found in [13], [14]. The latter work presents model parameters that have been identified and validated based on real-world experiments and it presents time-optimal control solutions, in order to study different modeling choices for performing at-the-limit maneuvers.

A popular approach to efficiently implement nonlinear MPC is based on the Real-Time Iteration (RTI) scheme [15], which typically combines a direct multiple shooting type optimal control discretization [16] with an online variant of Sequential Quadratic Programming (SQP). By applying a continuation technique, one SQP iteration can be performed in order to update the state and control trajectories as a solution guess for the nonlinear OCP from one sampling instant to the next. An efficient numerical implementation of the RTI scheme requires a careful combination of differentiation techniques, integration schemes and convex solvers. An overview of explicit and implicit integration schemes with sensitivity analysis based on Algorithmic Differentiation (AD) can be found in [17], [18].

Within an SQP-based algorithm for NMPC, a tailored convex solver is needed to solve the optimal control structured quadratic programs (QP). In recent years, many such algorithms have been developed to directly deal with this particular sparsity structure, such as in the software implementations `FORCES` [19], `qpDUNES` [20] and `HPMPC` [21]. This paper will additionally include a new sparse solver, called `PRESAS`, more information on which can be found in [22]. Alternatively, one can use a condensing technique to numerically eliminate the state variables and preserve only the control inputs [16]. The resulting smaller but dense quadratic subproblem can be solved with any QP solver based on dense linear algebra such as `qpOASES` [23], `PQP` [24] or a particular variant of `ADMM` [25].

This article aims at providing a comparison of different available embedded optimization algorithms for autonomous vehicle control, using hardware with limited processing resources. Unlike prior publications such as in [2], [10], the presented computation times have been obtained using the Raspberry Pi 2 model for rapid prototyping. While they are not embedded processors by themselves, such Raspberry Pi models use ARM cores of the same type as those that are used by multiple high-end automotive microprocessors. Finally, simulation results are presented for variants of both single- and double-track vehicle models in NMPC, resulting in a trade-off between modeling accuracy and corresponding online computational efforts. Based on these results, one can observe that single-track modeling can be sufficiently accurate to be used in NMPC-based feedback control, even in the case of an aggressive double lane-change maneuver.

The paper is organized as follows. Section II summarizes the modeling choices for the nonlinear vehicle dynamics. Section III introduces the optimal control problem formulation and the online NMPC algorithm. A software implementation based on the `ACADO` code generation tool is presented in Section IV, including a discussion on embedded convex solvers for real-time optimal control. A comparison of different implementations of the NMPC scheme is carried out in Section V. Section VI finally concludes the paper.

[1]Mechatronics, Mitsubishi Electric Research Laboratories, Cambridge, MA, 02139, USA. `quirynen@merl.com`

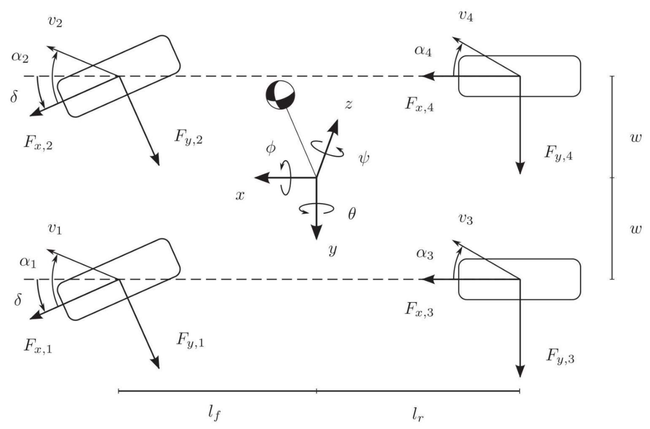

Fig. 1: Double-track vehicle model with roll and pitch dynamics to take load transfer into account [13].

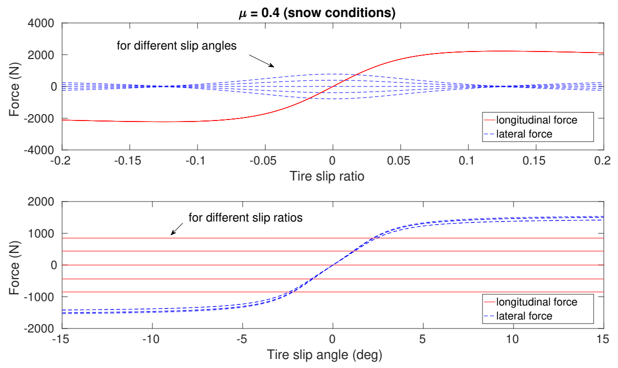

Fig. 2: Longitudinal and lateral tire forces with respect to slip angle and slip ratio according to the Pacejka model, where the coupling is based on the friction ellipse [27].

II. VEHICLE DYNAMICS

Let us briefly summarize some of the important modeling choices for the vehicle dynamics in order to further formulate the nonlinear MPC problem.

A. Chassis Modeling: Single- versus Double-track

One distinctive choice is whether to use a single- or double-track chassis model, as illustrated in Figure 1. The double-track or DT model includes roll (ϕ) and pitch (θ) dynamics such that it exhibits five degrees of freedom, namely two translational and three rotational. Based on these additional degrees of freedom in the dynamics, both the longitudinal and lateral load transfer between the four wheels of the vehicle can be accurately modeled. Note that load transfer modeling allows to capture differences in the tire forces that generate yaw moments when turning and differences in peak torques when accelerating or braking.

Alternatively, the single-track or ST model typically neglects roll and pitch dynamics and it can be obtained by lumping together the left and right wheel on each axle. Consequently, the standard ST model describes only one rotational degree of freedom. It is relatively straightforward to obtain alternative chassis models as a compromise between these two extreme cases. For example, one can formulate an ST model with pitch dynamics in order to represent the longitudinal load transfer as described in [13].

B. Tire Friction Forces and Wheel Dynamics

In order to represent the tire friction forces, one needs to define the slip angles α_i and slip ratios s_i

$$\begin{aligned} \alpha_i &= -\arctan\left(\frac{v_{y,i}}{v_{x,i}}\right), \\ s_i &= \frac{R_w\omega_i - v_{x,i}}{v_{x,i}}, \quad i \in \{f,r\} \text{ or } \{1,2,3,4\}, \end{aligned} \tag{1}$$

where R_w denotes the wheel radius, ω_i is the wheel angular velocity and $v_{x,i}, v_{y,i}$ are the longitudinal and lateral velocities for wheel i. Note that subscripts $\{f,r\}$ denote the front and rear wheels in the ST model, while the indices $\{1,2,3,4\}$ denote the four wheels in the DT model. As proposed by [26], we use the following first-order filter to describe the slip angles

$$\dot{\alpha}_i \frac{\sigma}{v_{x,i}} + \alpha_i = -\arctan\left(\frac{v_{y,i}}{v_{x,i}}\right), \tag{2}$$

where σ is the relaxation length, to account for the effect that tire forces do not develop instantaneously [27]. Finally, the wheel dynamics are given by

$$T_i - I_w\dot{\omega}_i - F_{x,i}R_w = 0, \quad i \in \{f,r\} \text{ or } \{1,2,3,4\}, \tag{3}$$

where I_w denotes the wheel inertia and T_i defines the driving/braking torque.

Using the above quantities, one can compute the nominal tire forces, i.e., under pure slip conditions, using Pacejka's magic formula [26], [28] in the following form

$$\begin{aligned} F^x_{0,i} &= \mu^x_i F^z_i \sin\left(C^x_i \arctan(B^x_i(1-E^x_i)s_i + E^x_i \arctan(B^x_i s_i))\right), \\ F^y_{0,i} &= \mu^y_i F^z_i \sin\left(C^y_i \arctan(B^y_i(1-E^y_i)\alpha_i + E^y_i \arctan(B^y_i \alpha_i))\right), \end{aligned} \tag{4}$$

where μ^x_i and μ^y_i are the friction coefficients and $B^\star_i$, $C^\star_i$ and $E^\star_i$ are model parameters. The magic formula (4) exhibits the typical saturation behavior in the tire forces as illustrated also in Figure 2. Under combined slip conditions, i.e., when both s and α are non-zero, one needs to model the coupling between longitudinal and lateral tire forces.

The simplest and therefore most straightforward modeling of these combined tire forces is based on the friction ellipse (FE) as follows

$$F^y_i = F^y_{0,i}\sqrt{1-\left(\frac{F^x_{0,i}}{\mu^x_i F^z_i}\right)^2}, \quad i \in \{f,r\} \text{ or } \{1,2,3,4\}. \tag{5}$$

The main limitation of the FE models is that the longitudinal force does not explicitly depend on the lateral slip (see Fig. 2). Therefore, more accurate models to represent the combined slip could be used, e.g., based on weighting functions [13], [26]. For this paper, the model parameters for both vehicle and tire friction are taken directly from [27] and they are based on experimental validation.

III. NONLINEAR MPC PROBLEM FORMULATION

We introduce the following tracking-type optimal control problem formulation in continuous time

$$
\begin{aligned}
\min_{x(\cdot),\, u(\cdot)} \quad & \int_0^T \|F(x(t),u(t)) - y_{\mathrm{ref}}(t)\|_W^2 \, \mathrm{d}t && \text{(6a)}\\
\text{s.t.} \quad & 0 = x(0) - \hat{x}_0, && \text{(6b)}\\
& 0 = f(\dot{x}(t), x(t), u(t)), \quad \forall t \in [0,T], && \text{(6c)}\\
& 0 \geq h(x(t), u(t)), \quad \forall t \in [0,T], && \text{(6d)}\\
& 0 \geq r(x(T)), && \text{(6e)}
\end{aligned}
$$

where $x(t) \in \mathbb{R}^{n_\mathrm{x}}$ denote the differential states and $u(t) \in \mathbb{R}^{n_\mathrm{u}}$ are the control inputs for $t \in [0,T]$. The objective in Eq. (6a) consists of a nonlinear least squares type Lagrange term. For simplicity, T denotes both the control and prediction horizon length and we do not consider a Mayer cost term. Note that the NMPC problem depends on the current state estimate $\hat{x}_0$ through the initial condition of Eq. (6b). The nonlinear dynamics in Eq. (6c) are described by an implicit system of Ordinary Differential Equations (ODE), which allows the formulation of both the single- and double-track vehicle dynamics [13]. Respectively, Eqs. (6d) and (6e) denote the path and terminal inequality constraints.

A. Objective Function and Inequality Constraints

The path constraints in the NMPC problem formulation consist of geometric and physical limitations of the system. Depending on the particular maneuver, one can include constraints on the longitudinal and lateral position of the vehicle. However, in practice, it is important to reformulate such requirements as soft constraints because of unknown disturbances. For simplicity, we define a quadratic penalization of the slack variable to ensure a feasible solution whenever possible. Depending on the convex solver that is used as discussed in Section IV-B, the latter can be replaced by an exact L1 penalty similar to [7]. In addition, bound constraints are taken into account on the steering angle, steering rate and the wheel torque values.

The equation in (6a) allows us to formulate any standard tracking-type objective. In the presented simulation results, the NMPC scheme is based on a direct tracking of a reference trajectory for the state and control variables

$$\|x(t) - x_{\mathrm{ref}}(t)\|_Q^2 + \|u(t) - u_{\mathrm{ref}}(t)\|_R^2, \tag{7}$$

where $Q \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ and $R \in \mathbb{R}^{n_\mathrm{u} \times n_\mathrm{u}}$ are the corresponding weighting matrices. Note that the optimal control problem consists of $n_\mathrm{x} = 11$ or $n_\mathrm{x} = 19$ state variables, respectively, when using the single- or double-track vehicle model. The number of control inputs is $n_\mathrm{u} = 4$, including the steering rate, the drive and brake torques and slack variable.

B. Direct Optimal Control Parameterization

In direct optimal control, the continuous time and therefore infinitely dimensional OCP in (6) is reformulated as a tractable nonlinear program (NLP) by a particular control and state parameterization. A popular approach is based on the direct multiple shooting method from [16]. For simplicity, let us formulate an equidistant grid over the control horizon consisting of the collection of time points t_i, where $t_{i+1} - t_i = \frac{T}{N} =: T_\mathrm{s}$ for $i = 0, \ldots, N-1$. Additionally, we consider a piecewise constant control parameterization $u(\tau) = u_i$ for $\tau \in [t_i, t_{i+1})$. The time discretization for the state variables can then be obtained by simulating the system dynamics using a numerical integration scheme. This corresponds to solving the following initial value problem

$$0 = f(\dot{x}(\tau), x(\tau), u_i), \quad \tau \in [t_i, t_{i+1}], \quad x(t_i) = x_i. \tag{8}$$

Note that the vehicle dynamics can be relatively stiff and the differential equations are implicitly defined for the DT model in [13], such that an implicit integration scheme should be used for NMPC to provide the required accuracy in a numerically efficient manner [17], [18], [29].

C. Online Tracking of a Geometric Path

A path planner or high level supervisory algorithm selecting, e.g., the target lane, would typically be implemented in addition to the proposed NMPC scheme in order to provide the time-varying reference trajectory that needs to be tracked online. By realizing the behavior that is requested by the path planner through the tracking controller, the autonomous vehicle can alternate between different common scenarios such as lane keeping and performing lane changes or obstacle avoidance. Similar to the work in [30] and based on [31], we model the reference path as a piecewise clothoidal trajectory such that the desired yaw angle ψ_des and yaw rate $\dot{\psi}_\mathrm{des}$ of the vehicle can be described as follows

$$\ddot{\psi}_\mathrm{des} = v_\mathrm{des}(t)\dot{\kappa}(t), \tag{9}$$

where $v_\mathrm{des}(t)$ denotes the reference velocity, and $\dot{\kappa}(t)$ denotes the change of desired yaw rate or the change of the curvature of the reference trajectory. For simplicity, the latter two quantities are typically provided as a piecewise constant trajectory, i.e., the velocity and the change of curvature are both constant over each sampling interval T_s. Based on this path representation, one can reformulate the vehicle dynamics using the position and orientation errors (e_X, e_Y, e_ψ) with respect to such reference trajectory.

IV. SOFTWARE IMPLEMENTATION FOR EMBEDDED NONLINEAR MPC

In order to assess the real-time feasibility of the different optimal control formulations for embedded applications, an efficient and tailored implementation is used in the open-source `ACADO` code generation tool [29], [32]. The nonlinear optimal control solver in this toolkit uses an online variant of SQP, known as the Real-Time Iteration (RTI) scheme [15]. The idea is to minimize the computational delay between obtaining the new state estimate $\hat{x}_0$ in Eq. (6b) and applying the next control input.

A. ACADO Code Generation Tool

Two algorithmic components are crucial to efficiently implement the RTI scheme for NMPC:

1) integrators with algorithmic differentiation for online problem linearization: an overview on sensitivity analysis for explicit and implicit integration schemes can be found in [17]. Note that stiff and/or implicit differential equations typically require implicit integrators, such as the lifted collocation methods in [18].
2) convex QP solvers tailored for optimal control structured problems and suitable for embedded hardware architectures: see [12] and the following subsection.

To obtain a real-time feasible NMPC implementation on the embedded control hardware, the `ACADO` code generation tool has been used as presented in [29], [32]. It is part of the open-source `ACADO` Toolkit which can be downloaded from `www.acadotoolkit.org`. The code generation tool exports highly efficient, standalone C-code implementing the RTI scheme for fast optimal control.

B. Embedded Optimal Control Solvers

Important properties to be taken into account when choosing or designing an embedded QP solver for real-time optimal control are the following:

- scaling of computational complexity and corresponding memory requirements with problem dimensions,
- warm starting capabilities for receding horizon control,
- software portability of solver code and dependencies,
- early termination of the solver in real-time applications to obtain a feasible but suboptimal solution,
- numerical performance on embedded control hardware with limited resources, e.g., limited or the absence of cache memory or the use of single-precision arithmetics.

Regarding the scaling of the algorithm, there is an important distinction between solvers that directly tackle the block structured QP and those that solve a dense QP instead, after eliminating the state variables in a condensing routine [16]. A popular example of the latter is the parametric active-set solver `qpOASES` [23], which provides a library-free implementation for embedded applications based on dense linear algebra routines. We also consider the projection-free `PQP` [24] algorithm based on dual optimization and a multiplicative update rule as well as a variant of the alternating direction method of multipliers (`ADMM`) in [25].

Additionally, there exist many sparsity exploiting convex solvers that are tailored to block structured optimal control problems. A popular approach is to use an interior-point method with block sparsity exploiting linear algebra, such as the block-tridiagonal Cholesky factorization of the Schur complement in [33] or a particular Riccati recursion for linear-quadratic control problems in [34]. Efficient implementations of such techniques for embedded optimal control can be found, for example, in the software tools `FORCES` [19] and `HPMPC` [21].

A solver called `PRESAS` has been proposed recently in [22], which applies the block structured factorization techniques with low-rank updates to preconditioning of an iterative solver within a primal active-set algorithm. For real-time applications, this primal active-set approach has the advantage that it can provide a feasible but suboptimal solution when being terminated early. In addition to interior-point and active-set based algorithms, other methods exist for real-time optimal control such as the dual Newton strategy in `qpDUNES` [20]. In the simulation results presented in this paper, we restrict to the tools that are mentioned above. An overview of algorithms can be found in [12].

V. NONLINEAR MPC SIMULATION RESULTS

This closed-loop simulation study is based on the standardized ISO 3888-2 double lane-change maneuver, developed for vehicle stability evaluation. Note that, in prior publications, computational results were often obtained on a modern PC featuring, for instance, an Intel i7 CPU at 2.7GHz in [2], [4]. Instead, this paper illustrates all computational timing results using the ARM-based Raspberry Pi 2 model[1], which is considerably closer to an embedded control hardware architecture that could be used when performing experiments. In fact, Raspberry Pi uses ARM cores of the same type as those used by high-end automotive embedded microprocessors, such as the TDAx family by Texas Instruments, the RZ family by Renesas and the TEGRA system by NVIDIA.

A. Vehicle Dynamics and Model Mismatch in NMPC

One of the most significant nonlinearities in the vehicle dynamics, when performing maneuvers on a low-friction road surface, comes from Pacejka's magic formula in Eq. (4) to model the tire forces under pure slip conditions. Figure 3 illustrates the typical behavior for the tire force trajectories, with respect to both the slip ratio and the slip angle, while performing a double lane-change on a snow-covered road. Note that in the latter figure, the resulting force, relative to the normal force, is defined as follows

$$F_i^{\text{res}} = \frac{\sqrt{F_i^{x^2} + F_i^{y^2}}}{F_i^z}, \quad i \in \{f, r\} \text{ or } \{1, 2, 3, 4\}. \tag{10}$$

We further perform closed-loop NMPC simulations for a front-wheel drive vehicle, using the DT model to simulate the vehicle dynamics and using either the ST or DT model within the NMPC problem formulation, with the model parameters from [27]. The DT model can generally be much more accurate than the ST model, but feedback control in the form of closed-loop based NMPC allows one to use a low-order (ST) model to control the vehicle dynamics in case of not too aggressive maneuvers, even on challenging road conditions [13], [14]. This can also be observed in Figures 4 and 5 that show the closed-loop steering control results using either the ST or DT model within NMPC. They show the closed-loop trajectories, when assuming either snow or asphalt conditions for the model used in the NMPC formulation, while actually performing the double lane-change maneuver on an asphalt (Figure 4) or snow road surface (Figure 5). Note that the slip angles for the ST based

[1]The Raspberry Pi 2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache.

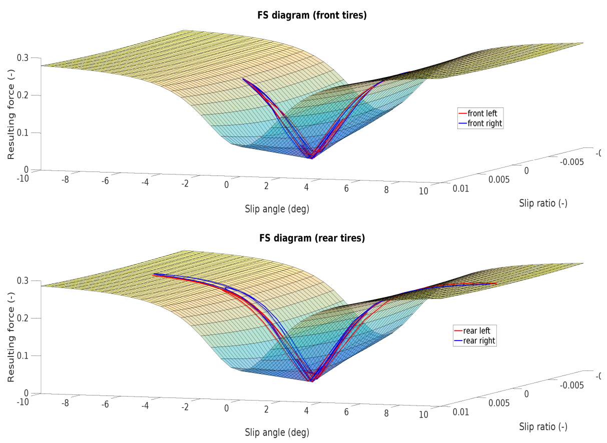
Fig. 3: NMPC closed-loop simulation results for double lane-change on snow: trajectories for resulting relative tire forces.

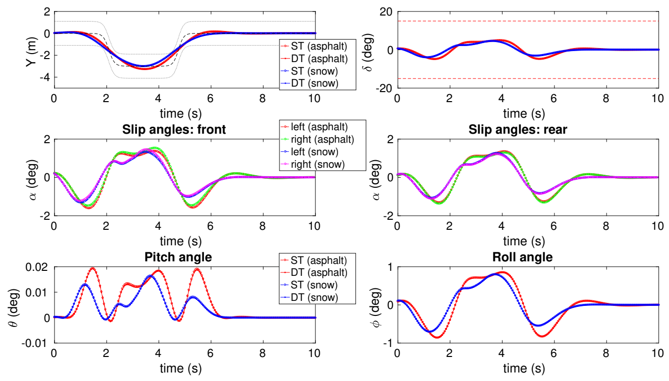
Fig. 4: NMPC closed-loop simulation results at 9 m/s for double lane-change on asphalt ($N = 50$ and $T_s = 50$ ms, legends refer to the type of road that is expected by NMPC).

NMPC scheme are shown in both figures, because they are indistinguishable from the results based on the DT model.

Fig. 4 illustrates that the double lane-change maneuver can be executed relatively easily on asphalt, even when the NMPC scheme expects the vehicle to be on a snow-covered road. On the other hand, the vehicle can quickly go unstable when the controller would overestimate the tire friction, as can be observed in Fig. 5 for the case where the NMPC scheme assumes asphalt instead of snow road conditions. More specifically, the unstable behavior can be seen in the violation of the soft constraints on the lateral position and in the oscillations of the slip, pitch and roll angles. It is interesting to point out that, even in such a case where the vehicle goes unstable, the closed-loop trajectories from the ST and DT based NMPC algorithm remain similar.

B. NMPC Parameters and Real-Time Feasibility

Table I and II show the average and worst-case computation times on a Raspberry Pi 2 embedded platform for the NMPC closed-loop simulation results, respectively, based on the ST and DT vehicle model. The sampling time for

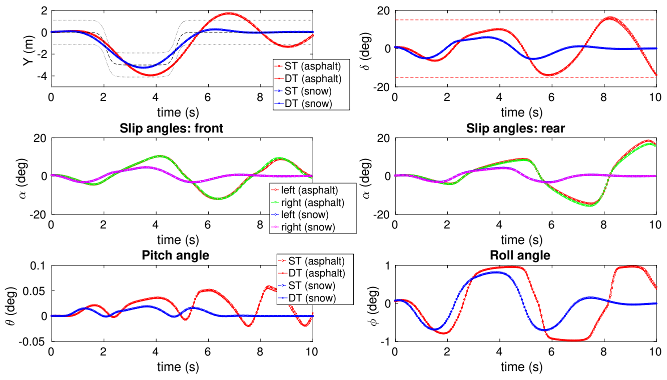
Fig. 5: NMPC closed-loop simulation results at 9 m/s for double lane-change on snow ($N = 50$ and $T_s = 50$ ms, legends refer to the type of road that is expected by NMPC).

these NMPC simulations is chosen to be equal to $T_s = 50$ ms. The corresponding computation times confirm that an efficient NMPC implementation based on the RTI scheme, in combination with a structure exploiting convex solver, allows one to obtain real-time feasible computations on an embedded control architecture.

Table I and II additionally illustrate how the NMPC computation times scale for an increasing control horizon length and this for different embedded QP solvers. The typical behavior of a quadratic computational complexity with the horizon length N for dense solvers, such as `PQP`, `ADMM` and `qpOASES`, versus the linear complexity of sparse solvers, such as `HPMPC` and `PRESAS`, can also be observed here. One can reduce the number of control intervals further by choosing the length of one shooting interval to become larger than the actual sampling time of the NMPC scheme. This is illustrated for the computation times corresponding to $N = 20$, for which a discretization time of 75 ms is used instead of the 50 ms for the other horizon lengths.

The computation times, reported in Table I and II, have been obtained using single-precision arithmetics. The use of single- versus double-precision arithmetics can have beneficial effects on the storage requirements and on the computational efficiency of running the solver code on the limited hardware resources. However, numerical linear algebra operations exhibit a problem-dependent conditioning such that an overall higher number of iterations can be required within the algorithm, when using a relatively lower accuracy in order to carry out the numerical computations.

VI. CONCLUSIONS

This paper investigates several aspects in the implementation of NMPC for steering control in autonomous vehicles, based on the `ACADO` code generation software. We presented a comparison for a range of embedded optimization algorithms tailored to real-time optimal control, based on detailed timing results on the Raspberry Pi 2 model. With the exception of aggressive, at-the-limit maneuvers, our closed-loop simulation results confirm that a low order, single-track

	$N = 20$	$N = 30$	$N = 40$	$N = 60$
	mean/max	mean/max	mean/max	mean/max
PQP	$\mathbf{23.0/25.7}$	$105/117$	$209/226$	$578/599$
ADMM	$\mathbf{14.7/27.5}$	$39.6/71.9$	$85.5/150$	$268/440$
qpOASES	$\mathbf{12.2/21.0}$	$26.1/50.1$	$48.5/81.7$	$132/240$
HPMPC	$\mathbf{11.9/16.0}$	$\mathbf{18.1/21.9}$	$\mathbf{24.5/29.2}$	$\mathbf{37.0/45.1}$
PRESAS	$\mathbf{6.44/9.61}$	$\mathbf{9.14/10.7}$	$\mathbf{12.4/13.7}$	$\mathbf{18.4/19.8}$

TABLE I: NMPC computation times (ms) for single-track vehicle model with varying horizon length ($T_s = 50$ ms), using Raspberry Pi 2 (ARM Cortex-A7, single-precision).[2]

	$N = 20$	$N = 30$	$N = 40$	$N = 60$
	mean/max	mean/max	mean/max	mean/max
PQP	$45.1/55.5$	$142/159$	$266/288$	$672/704$
ADMM	$33.4/50.2$	$69.7/101$	$129/190$	$344/506$
qpOASES	$\mathbf{34.5/40.4}$	$63.7/82.1$	$102/131$	$229/354$
HPMPC	$\mathbf{39.4/48.0}$	$58.7/77.5$	$82.6/126$	$122/136$
PRESAS	$\mathbf{26.0/30.2}$	$\mathbf{39.0/41.9}$	$52.5/65.3$	$78.7/82.5$

TABLE II: NMPC computation times (ms) for double-track vehicle model with varying horizon length ($T_s = 50$ ms), using Raspberry Pi 2 (ARM Cortex-A7, single-precision).[2]

vehicle model is sufficient for feedback control based on NMPC. This paper also illustrated the robustness of the NMPC scheme with respect to a model mismatch.

This work provides motivating results and a proof of concept for the NMPC scheme, which will lead to a real-world implementation on an embedded microprocessor.

References

[1] F. Borrelli, P. Falcone, T. Keviczky, and J. Asgari, "MPC-based approach to active steering for autonomous vehicle systems," *Intern. Journal of Vehicle Auton. Systems*, vol. 3, no. 2, pp. 265–291, 2005.

[2] J. V. Frasch, A. J. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Proc. Europ. Control Conf.*, 2013, pp. 4136–4141.

[3] R. C. Rafaila and G. Livint, "Nonlinear model predictive control of autonomous vehicle steering," in *2015 19th Intern. Conf. on System Theory, Control and Comp. (ICSTCC)*, Oct 2015, pp. 466–471.

[4] G. Schildbach, "A new nonlinear model predictive control algorithm for vehicle path tracking," in *Proc. 13th Intern. Symp. on Advanced Vehicle Control*, 2016.

[5] N. van Duijkeren, T. Keviczky, P. Nilsson, and L. Laine, "Real-time NMPC for semi-automated highway driving of long heavy vehicle combinations," in *Proc. 5th IFAC Conf. on Nonlinear Model Predictive Control (NMPC)*, 2015.

[6] M. Zanon, J. V. Frasch, M. Vukov, S. Sager, and M. Diehl, "Model predictive control of autonomous vehicles," in *Optimization and Optimal Control in Automotive Systems*. Springer, 2014, pp. 41–57.

[7] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. on Control Sys. Techn.*, vol. 15, no. 3, pp. 566–580, 2007.

[8] T. Kraus, H. J. Ferreau, E. Kayacan, H. Ramon, J. D. Baerdemaeker, M. Diehl, and W. Saeys, "Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles," *Comp. and Electr. in Agriculture*, vol. 98, pp. 25–33, October 2013.

[9] P. F. Lima, "Predictive control for autonomous driving with experimental evaluation on a heavy-duty construction truck," Ph.D. dissertation, KTH Royal Insitute of Technology, Stockholm, Sweden, 2016.

[2]The computation times that are highlighted in bold are theoretically real-time feasible, i.e., the maximum computation time is below 50 ms.

[10] R. Verschueren, S. D. Bruyne, M. Zanon, J. V. Frasch, and M. Diehl, "Towards time-optimal race car driving using nonlinear MPC in real-time," in *Proc. IEEE Conf. on Dec. and Control*, 2014, pp. 2505–2510.

[11] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear model predictive control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, M. Raimondo, and F. Allgöwer, Eds. Springer, 2009, vol. 384, pp. 391–417.

[12] H. J. Ferreau, S. Almer, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. L. Jerez, G. Stathopoulos, and C. Jones, "Embedded optimization methods for industrial automatic control," in *Proc. of the IFAC World Congress*, 2017.

[13] K. Berntorp, B. Olofsson, K. Lundahl, and L. Nielsen, "Models and methodology for optimal trajectory generation in safety-critical road-vehicle manoeuvres," *Vehicle System Dynamics*, vol. 52, no. 10, pp. 1304–1332, 2014.

[14] K. Lundahl, K. Berntorp, B. Olofsson, J. Åslund, and L. Nielsen, "Studying the Influence of Roll and Pitch Dynamics in Optimal Road-Vehicle Maneuvers," in *Proc. 23rd Int. Symp. on Dynamics of Vehicles on Roads and Tracks, Qingdao, China*, August 2013.

[15] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, "Nominal stability of the real-time iteration scheme for nonlinear model predictive control," *IEE Proc.-Control Theory Appl.*, vol. 152, no. 3, pp. 296–308, 2005.

[16] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *Proc. of the IFAC World Congress*. Pergamon Press, 1984, pp. 242–247.

[17] R. Quirynen, "Numerical simulation methods for embedded optimization," Ph.D. dissertation, KU Leuven and University of Freiburg, 2017.

[18] R. Quirynen, S. Gros, B. Houska, and M. Diehl, "Lifted collocation integrators for direct optimal control in ACADO toolkit," *Math. Prog. Computation*, vol. 9, no. 4, pp. 527–571, 2017.

[19] A. Domahidi and J. Perez, "FORCES professional," embotech GmbH (http://embotech.com/FORCES-Pro), July 2014.

[20] J. V. Frasch, S. Sager, and M. Diehl, "A parallel quadratic programming method for dynamic optimization problems," *Mathematical Programming Computations*, vol. 7, no. 3, pp. 289–329, 2015.

[21] G. Frison, H. B. Sorensen, B. Dammann, and J. B. Jørgensen, "High-performance small-scale solvers for linear model predictive control," in *Proc. Europ. Control Conf.*, June 2014, pp. 128–133.

[22] R. Quirynen, A. Knyazev, and S. Di Cairano, "Block structured preconditioning within an active-set method for real-time optimal control," in *Proc. Europ. Control Conf.*, vol. (accepted), 2018.

[23] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Math. Prog. Comp.*, vol. 6, no. 4, pp. 327–363, 2014.

[24] S. Di Cairano, M. Brand, and S. A. Bortoff, "Projection-free parallel quadratic programming for linear model predictive control," *International Journal of Control*, vol. 86, no. 8, pp. 1367–1385, 2013.

[25] A. U. Raghunathan and S. Di Cairano, "ADMM for convex quadratic programs: Q-linear convergence and infeasibility detection," *arXiv:1411.7288*, 2015.

[26] H. B. Pacejka, *Tyre and Vehicle Dynamics*. Elsevier, 2006.

[27] K. Berntorp, "Particle filtering and optimal control for vehicles and robots," Ph.D. dissertation, Department of Automatic Control, Lund University, 2014.

[28] E. Bakker, L. Nyborg, and H. B. Pacejka, "Tyre modelling for use in vehicle dynamics studies," in *SAE Technical Paper*, 1987.

[29] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators," *Opt. Control Appl. & Meth.*, vol. 36, pp. 685–704, 2014.

[30] S. Di Cairano, U. V. Kalabić, and K. Berntorp, "Vehicle tracking control on piecewise-clothoidal trajectories by MPC with guaranteed error bounds," in *IEEE 55th Conf. on Dec. and Control (CDC)*, Dec 2016, pp. 709–714.

[31] R. Rajamani, *Vehicle Dynamics and Control*. Springer US, 2012.

[32] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[33] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[34] G. Frison and J. B. Jørgensen, "Efficient implementation of the Riccati recursion for solving linear-quadratic control problems," in *Proc. of the IEEE Conf. on Control Applications*, Aug 2013, pp. 1117–1122.