Long-Horizon Vehicle Planning and Control Through Real-Time Iterations

Victor Fors
Department of Mechanical Engineering
Stanford University
Stanford, CA 94305 USA
fors@stanford.edu

J. Christian Gerdes
Department of Mechanical Engineering
Stanford University
Stanford, CA 94305 USA
cgerdes@stanford.edu

2023 IEEE Intelligent Vehicles Symposium (IV) | 979-8-3503-4691-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/IV55152.2023.10186610

***Abstract*—The length of the planning horizon when using nonlinear model predictive control is limited by computational resources. To achieve long-horizon vehicle planning and control, the practical feasibility of applying real-time iterations is demonstrated with an experiment on an automated vehicle. The proposed implementation is real-time capable with a planning horizon of over 10 s, requiring a fraction of the computational time compared to a solver for general nonlinear programming problems. The solutions obtained are close to the local optimum, even in the presence of real-world disturbances. Important details for the successful implementation in this domain are discussed and illustrated through simulation.**

***Index Terms*—Autonomous vehicles, NMPC, motion planning**

I. Introduction

According to the California Driver's Handbook, drivers should scan the road 10–15 seconds ahead of their vehicle so they can see hazards early and avoid having to make last minute moves. An autonomous vehicle should plan at least as far into the future to fulfill this requirement. Nonlinear Model Predictive Control (NMPC) is a method that can be used to plan vehicle motion such that the handling limits of the vehicle are respected. In NMPC, the continuous-time dynamics are discretized along an independent variable (commonly time) and the resulting finite-dimensional optimization problem is solved for a receding horizon with N prediction steps. Brown and Gerdes demonstrated an NMPC formulation to avoid pop-up obstacles [1], using 50 prediction steps for a planning horizon of 2.5 s. Li *et al.* used NMPC in a racing controller [2], using 30 prediction steps for a planning horizon of up to 2 s. In both cases, the available computational resources forced the planning horizon to be shorter than ideally desired.

One way to reduce the computational requirements of NMPC and enable a longer planning horizon is to reduce model complexity. Laurense and Gerdes proposed different model complexities for near- and long-term predictions in the NMPC formulation to achieve a longer planning horizon [3]. Another approach explored by Wohner *et al.* is to split the problem into a high-level NMPC for trajectory planning and a low-level NMPC for trajectory tracking [4]. The high-level NMPC has long solve times which is compensated by the low-level NMPC to achieve fast feedback.

NMPC based on the Real-Time Iteration (RTI) scheme [5] is a promising avenue to reduce the computational cost of NMPC.

Fig. 1. Automated 2018 Volkswagen Golf VII GTI Performance.

The underlying strategy of solving the nonlinear optimization problem in RTI stems from Sequential Quadratic Programming (SQP) [6], which is an iterative method in which the solution is gradually improved in each iteration by solving a Quadratic Programming (QP) subproblem. In contrast to SQP, RTI updates the next QP iterate with the current state information and sends feedback to the controlled system in between solving each QP. Properly implemented, RTI can provide the performance of NMPC at a computational cost close to that of linear model predictive control [7]. Svensson *et al.* demonstrated collision avoidance using an RTI style reformulation where the constraints are linearized around the solution of the previous planning iteration [8], using 40 prediction steps for a planning horizon of 4 s. Kloeser *et al.* used RTI to race around a track with RC cars [9], using 50 prediction steps for a planning horizon of 1 s. Even with some success using RTI and RTI style formulations for vehicle planning and control, RTI is not a popular choice in this domain and has not been used to significantly increase the number of prediction steps over that achievable with solvers for general nonlinear programming problems. Possible reasons are the challenges reported with RTI, in terms of issues with instability between iterations and infeasibility of the provided solutions [10]. A long planning horizon provides an additional challenge for RTI, because the longer sampling time needed results in fewer QP iterations in a given amount of time, which reduces the rate of convergence.

Contributions

A key contribution is a demonstration of the practical feasibility of using RTI for vehicle planning and control to extend the planning horizon to over 10 seconds. An implementation

to achieve this is proposed and is experimentally verified on an automated vehicle (see Fig.1) racing around a track, planning its own path without a prior reference and with the challenging goal of maximizing progress. Important details for the successful implementation of RTI in this setting are discussed and illustrated through simulation.

II. RTI Implementation

This section presents the form of the QP subproblem used to implement RTI and then provides an overview of the RTI algorithm.

A. The QP subproblem

Consider a nonlinear programming (NLP) problem of the form

$$\underset{y}{\text{minimize}} \;\; J(y) \tag{1a}$$
$$\text{subject to} \;\; h(y) = 0 \tag{1b}$$
$$g(y) \leq 0 \tag{1c}$$

where $y \in \mathbb{R}^{n_y}$, $J : \mathbb{R}^{n_y} \to \mathbb{R}$, $h : \mathbb{R}^{n_y} \to \mathbb{R}^{n_\lambda}$, and $g : \mathbb{R}^{n_y} \to \mathbb{R}^{n_\mu}$. A common way to solve an NLP is to introduce the Lagrange multipliers $\lambda \in \mathbb{R}^{n_\lambda}$ and $\mu \in \mathbb{R}^{n_\mu}$ with the Lagrangian [6]

$$L(y, \lambda, \mu) = J(y) + \lambda^{\mathrm{T}} h(y) + \mu^{\mathrm{T}} g(y) \tag{2}$$

For some optimal selection of $\lambda = \lambda^*$ and $\mu = \mu^*$, it is possible to replace the objective function (1a) with minimizing the Lagrangian $L(y, \lambda^*, \mu^*)$ [6]. The reason to use the Lagrangian is to account for nonlinearities when solving an approximated problem with linearized constraints. Let L, ∇L, and H denote the Lagrangian, the gradient of the Lagrangian, and the hessian of the Lagrangian, respectively, evaluated at the current guess $y = y^{\text{guess}}$, $\lambda = \lambda^{\text{guess}}$, and $\mu = \mu^{\text{guess}}$. A Taylor series approximation of the Lagrangian gives

$$L(\Delta y) \approx L + \nabla L \Delta y + \frac{1}{2} \Delta y^{\mathrm{T}} H \Delta y \tag{3}$$

where

$$\Delta y = y - y^{\text{guess}} \tag{4}$$

The NLP (1) can then be approximated as the QP problem

$$\underset{\Delta y}{\text{minimize}} \;\; \nabla L \Delta y + \frac{1}{2} \Delta y^{\mathrm{T}} H \Delta y \tag{5a}$$
$$\text{subject to} \;\; \nabla h(y^{\text{guess}}) \Delta y + h(y^{\text{guess}}) = 0 \tag{5b}$$
$$\nabla g(y^{\text{guess}}) \Delta y + g(y^{\text{guess}}) \leq 0 \tag{5c}$$

Given Δy and the Lagrange multipliers, λ_{QP} and μ_{QP}, obtained by solving (5), a better estimate of the locally optimal solution to (1) is obtained for a small enough $\alpha \in (0, 1]$ with the update [6]

$$y^{\text{guess}} \leftarrow y^{\text{guess}} + \alpha \Delta y \tag{6a}$$
$$\lambda^{\text{guess}} \leftarrow \lambda^{\text{guess}} + \alpha(\lambda_{QP} - \lambda^{\text{guess}}) \tag{6b}$$
$$\mu^{\text{guess}} \leftarrow \mu^{\text{guess}} + \alpha(\mu_{QP} - \mu^{\text{guess}}) \tag{6c}$$

Rather than solving (5), a commonly used simpler form is solved [6], where the search direction given by the linear term in the objective function is not dependent on the current guess of Lagrange multipliers.

$$\underset{y}{\text{minimize}} \;\; \nabla J(y^{\text{guess}}) \Delta y + \frac{1}{2} \Delta y^{\mathrm{T}} H \Delta y \tag{7a}$$
$$\text{subject to} \;\; \nabla h(y^{\text{guess}}) \Delta y + h(y^{\text{guess}}) = 0 \tag{7b}$$
$$\nabla g(y^{\text{guess}}) \Delta y + g(y^{\text{guess}}) \leq 0 \tag{7c}$$

This is equivalent to (5) in the equality constrained case [6], because $\nabla h(y^{\text{guess}}) \Delta y$ is constant as per (7b). To remove nonlinear inequality constraints from (1), slack variables z are introduced

$$\underset{y,z}{\text{minimize}} \;\; J(y) \tag{8a}$$
$$\text{subject to} \;\; h(y) = 0 \tag{8b}$$
$$g(y) + z = 0 \tag{8c}$$
$$z \geq 0 \tag{8d}$$

The exact hessian H is not necessarily positive semi-definite. To obtain a positive definite matrix similar to H, an eigenvalue decomposition of the hessian is performed where Λ is a diagonal matrix with the eigenvalues of H

$$H = V \Lambda V^{\mathrm{T}} \tag{9}$$

The so called mirrored version $\tilde{H}$ [11] of the hessian is then used in place of the exact hessian H in (7) to ensure the QP is convex

$$\tilde{H} = V \max(\text{abs}(\Lambda), \epsilon) V^{\mathrm{T}} \tag{10}$$

where ϵ is a small positive value.

B. The RTI Algorithm

The dynamics of the system are discretized in time into an N steps long planning horizon. Let y_i denote the vector of entries of y associated with the ith prediction step, $i = 0, 1, \ldots, N$. Similarly, let λ_i and μ_i denote the vector of entries of λ and μ, respectively, associated with the ith prediction step, $i = 0, 1, \ldots, N$. Further, let x_0 and u_0 be the vectors of entries of y_0 representing the state and inputs of the system, respectively.

The estimated state $\hat{x}_0$ of the real system can be incorporated as a linear constraint $\Delta x_0 = \hat{x}_0 - x_0^{\text{guess}}$ directly in the QP subproblem (7), because linear constraints will not appear in the hessian H. The RTI algorithm used is as follows:

1) Shift the previous NMPC solution, including the Lagrange multipliers.

$$y_i^{\text{guess}} \leftarrow y_{i+1}^{\text{guess}}, \quad i = 0, 1, \ldots, N-1 \tag{11a}$$
$$\lambda_i^{\text{guess}} \leftarrow \lambda_{i+1}^{\text{guess}}, \quad i = 0, 1, \ldots, N-1 \tag{11b}$$
$$\mu_i^{\text{guess}} \leftarrow \mu_{i+1}^{\text{guess}}, \quad i = 0, 1, \ldots, N-1 \tag{11c}$$

2) Evaluate the hessian $\tilde{H}$ (10), the sensitivities $\nabla J(y^{\text{guess}})$, $\nabla h(y^{\text{guess}})$, $\nabla g(y^{\text{guess}})$, and the constants $h(y^{\text{guess}})$, $g(y^{\text{guess}})$ in (7) using the current guess y^{guess}, λ^{guess}, μ^{guess}.

3) Predict the state $\hat{x}_0$ of the real system at the start of the next control horizon and update the constraint $\Delta x_0 = \hat{x}_0 - x_0^{\text{guess}}$ in the QP subproblem.
4) Solve the QP.
5) Give the control input $u_0 = u_0^{\text{guess}} + \Delta u_0$ to the real system.
6) Apply the Newton step (6a)–(6c).
7) Repeat from step 1.

Typically a full Newton step, i.e. $\alpha = 1$, is taken in step 6 of the algorithm. The exception is when initializing RTI where a gradually increasing value of α is used for the first 10 iterations. This initialization procedure is necessary when the initial guess is too far from the solution. Step 2 can be evaluated before updating the information of the actual system state to reduce feedback delay [5], [7], but in this implementation that part of the RTI algorithm is disregarded because solving the QP turns out to be the most computationally demanding step when the planning horizon is long.

III. Problem Formulation

This section presents the vehicle model and the Optimal Control Problem (OCP). For a more in-depth description and analysis of the model and the OCP, the reader is referred to [3], where a similar model and OCP are used.

A. Vehicle Model

A low-level controller on the vehicle takes two inputs, the steering rate $\dot{\delta}$ and the rate of change of the commanded longitudinal force $\dot{F}_x$. A single-track model in a road-aligned coordinate system is used to model the vehicle with eight states: the road-wheel steering angle δ, the longitudinal force command F_x, the yaw rate r, the lateral velocity v_y, the longitudinal velocity v_x, the yaw angle $\Delta\psi$ relative to the road centerline, the distance e to the road centerline, and the position s along the road.

1) Load Transfer: The vertical loads on the front and rear axles, F_{zf} and F_{zr}, are computed with

$$F_{zf} = \frac{mbg - hF_x}{a+b}, \quad F_{zr} = \frac{mag + hF_x}{a+b} \tag{12}$$

where m is the vehicle mass, b is the length from the center of mass to the rear axle, a is the length from the center of mass to the front axle, h is the height of the center of mass above the road, and g is the gravitational acceleration.

2) Longitudinal Tire Forces: The vehicle is front wheel drive and has fixed brake proportioning to mimic the available actuators for a human driver. To smoothly approximate the change in proportioning χ of F_x when transitioning between a driving and a braking force, the following function is used

$$\chi = \frac{\chi^{\text{drive}} + \chi^{\text{brake}}}{2} + \frac{\chi^{\text{drive}} - \chi^{\text{brake}}}{2} \tanh\left(\frac{2F_x}{k_F} + 1\right) \tag{13}$$

where χ^{drive} is the drive distribution, χ^{brake} is the brake distribution, and k_F is a parameter determining the slope in the approximation. The commanded front and rear longitudinal tire forces, $\bar{F}_{xf}$ and $\bar{F}_{xr}$, are then computed with

$$\bar{F}_{xf} = \chi F_x \tag{14}$$

$$\bar{F}_{xr} = (1-\chi) F_x \tag{15}$$

The maximum absolute value of longitudinal force that can be obtained at the axle $j \in \{f, r\}$ is limited by the tire–road friction, μ_j, and given by

$$F_{xj}^{\max} = \cos(\alpha_j)\mu_j F_{zj} \tag{16}$$

where α_f and α_r are the front and rear slip angles, respectively

$$\alpha_f = \operatorname{atan2}(v_y + ar, v_x) - \delta \tag{17}$$

$$\alpha_r = \operatorname{atan2}(v_y - br, v_x) \tag{18}$$

To maintain smoothness, the tire force obtained at the tire–road contact is approximated with

$$F_{xj} = \frac{\sqrt{(\bar{F}_{xj} + F_{xj}^{\max})^2 + \varepsilon_j^2} - \sqrt{(\bar{F}_{xj} - F_{xj}^{\max})^2 + \varepsilon_j^2}}{2} \tag{19}$$

where ε_j determines the smoothness and is computed with

$$\varepsilon_j = \varepsilon F_{zj} \tag{20}$$

3) Lateral Tire Forces: The lateral tire forces are computed using the Fiala brush tire model. The lateral tire stiffness $C_{\alpha j}$ is assumed to be a linear function of the vertical load, dependent on the parameter $C_{\alpha j}^N$

$$C_{\alpha j} = C_{\alpha j}^N F_{zj} \tag{21}$$

A braking or driving force on the same axle limits the maximum potential lateral force to

$$F_{yj}^{\max} = \sqrt{(\mu_j F_{zj})^2 - F_{xj}^2} \tag{22}$$

The lateral tire force obtained is then

$$F_{yj} = \begin{cases} -C_{\alpha j}\tan(\alpha_j) \\ \quad + \dfrac{C_{\alpha j}^2 \tan(\alpha_j)|\tan(\alpha_j)|}{3F_{yj}^{\max}} \\ \quad - \dfrac{C_{\alpha j}^3 \tan^3(\alpha_j)}{27(F_{yj}^{\max})^2}, & |\tan(\alpha_j)| \le \tan(\alpha_j^{\text{sl}}) \\ -F_{yj}^{\max}\operatorname{sign}(\alpha_j), & \text{otherwise} \end{cases} \tag{23}$$

where

$$\tan(\alpha_j^{\text{sl}}) = \frac{3F_{yj}^{\max}}{C_{\alpha j}} \tag{24}$$

4) Equations of Motion: The road description enters the model equations by the curvature $\kappa(s)$ of the centerline, which is obtained through a look-up table with linear interpolation. The vehicle is subject to rolling resistance and drag, described with the parameters C_{d0} and C_{d2}

$$F_x^{\text{drag}} = C_{d0} + C_{d2} v_x^2 \tag{25}$$

Finally, given the mass of the vehicle m and the moment of inertia I_{zz}, the equations of motion are

$$\dot{r} = \frac{aF_{yf}\cos(\delta) + aF_{xf}\sin(\delta) - bF_{yr}}{I_{zz}} \tag{26}$$

$$\dot{v}_y = \frac{F_{yf}\cos(\delta) + F_{xf}\sin(\delta) + F_{yr}}{m} - rv_x \tag{27}$$

$$\dot{v}_x = \frac{F_{xf}\cos(\delta) - F_{yf}\sin(\delta) + F_{xr} - F_x^{\mathrm{drag}}}{m} + rv_y \tag{28}$$

$$\dot{s} = \frac{v_x\cos(\Delta\psi) - v_y\sin(\Delta\psi)}{1-\kappa(s)e} \tag{29}$$

$$\Delta\dot{\psi} = r - \kappa(s)\dot{s} \tag{30}$$

$$\dot{e} = v_x\sin(\Delta\psi) + v_y\cos(\Delta\psi) \tag{31}$$

B. Optimal Control Problem

The OCP used in the NMPC is given below with the following terms described in the remainder of this section

$$\underset{x,u,z}{\text{minimize}} \quad -s_N + J_e + J_N + J_u + J_\alpha + J_\mu \tag{32a}$$

$$\text{subj. to} \quad x_0 = \hat{x}_0 \tag{32b}$$

$$-\dot{\delta}^{\max} \le \dot{\delta}_j \le \dot{\delta}^{\max}, \quad j = 0,1,\dots,N \tag{32c}$$

$$\dot{F}_{x,j} \le \dot{F}_x^{\max} \tag{32d}$$

$$-\delta^{\max} \le \delta_i \le \delta^{\max}, \quad i = 1,2,\dots,N \tag{32e}$$

$$x_i = f_d(x_{i-1}, x_i, u_{i-1}, u_i) \tag{32f}$$

$$\frac{F_{x,i}v_{x,i}}{P^{\lim}} \le 1 \tag{32g}$$

$$-1 \le \frac{\bar{F}_{xf,i}}{\mu_f F_{zf,i}\cos(\alpha_{f,i})} \le 1 \tag{32h}$$

$$-1 \le \frac{\bar{F}_{xr,i}}{\mu_r F_{zr,i}\cos(\alpha_{r,i})} \le 1 \tag{32i}$$

$$0 \le \frac{e_i - e_i^{\mathrm{intr.}} - e^{\min}(s_i)}{e^{\max}(s_i) - e^{\min}(s_i)} \le 1 \tag{32j}$$

$$-1 \le \frac{\tan(\alpha_{f,i})}{\tan(\alpha_{f,i}^{\mathrm{sl}})} - \tan(\hat{\alpha}_{f,i}^{\mathrm{excess}}) \le 1 \tag{32k}$$

$$-1 \le \frac{\tan(\alpha_{r,i})}{\tan(\alpha_{r,i}^{\mathrm{sl}})} - \tan(\hat{\alpha}_{r,i}^{\mathrm{excess}}) \le 1 \tag{32l}$$

$$\frac{F_{xf,i}^2 + F_{yf,i}^2 - (k_\mu\mu_f F_{zf,i})^2}{(\mu_f F_{zf,i})^2} - \hat{F}_{f,i}^2 \le 0 \tag{32m}$$

$$\frac{F_{xr,i}^2 + F_{yr,i}^2 - (k_\mu\mu_r F_{zr,i})^2}{(\mu_r F_{zr,i})^2} - \hat{F}_{r,i}^2 \le 0 \tag{32n}$$

1) Discretization: Given the inputs $u = [\dot{\delta}, \dot{F}_x]^{\mathrm{T}}$, the states $x = [\delta, F_x, r, v_y, v_x, \Delta\psi, e, s]^{\mathrm{T}}$, and the ODE $\dot{x} = f_c(x,u) = [\dot{\delta}, \dot{F}_x, \dot{r}, \dot{v}_y, \dot{v}_x, \Delta\dot{\psi}, \dot{e}, \dot{s}]^{\mathrm{T}}$, the vehicle model is discretized using trapezoidal integration with the step size Δt

$$\begin{aligned} x_{i+1} &= f_d(x_i, x_{i+1}, u_i, u_{i+1}) \\ &= \frac{f_c(x_i,u_i) + f_c(x_{i+1},u_{i+1})}{2}\Delta t \end{aligned} \tag{33}$$

The dynamics are then enforced in the OCP by (32f).

2) Objective Function: The primary goal of the objective function is to maximize distance s_N traveled along the road, analogous to minimizing the time to reach s_N. There are also costs related to staying on the road, improving the ability of the vehicle to achieve the commanded inputs, and to keep the vehicle in areas accurately capture by the model. The relative influence of each cost is controlled by the tuning parameters: Q_e, Q_ϕ, $R_{\dot{\delta}}$, $R_{\dot{F}}$, Q_α, and Q_F.

An intrusion past the road bounds by the distance $e_i^{\mathrm{intr.}}$ is penalized with a quadratic cost

$$J_e = \Delta t \sum_{i=1}^{N} Q_e (e_i^{\mathrm{intr.}})^2 \tag{34}$$

where $e_i^{\mathrm{intr.}}$ is implemented as a slack variable in the road constraint (32j). The expressions $e^{\min}(s_i)$, and $1/(e^{\max}(s_i) - e^{\min}(s_i))$ in (32j) are implemented as lookup tables using linear interpolation. To encourage a safe speed and heading at the end of the horizon, the vehicle is penalized if it does not drive parallel to the road center-line at the last point in the planning horizon. This is accomplished by penalizing the angle of the road-relative velocity vector $\phi = \Delta\psi + \mathrm{atan2}(v_y, v_x)$

$$J_N = Q_\phi \phi_N^2 \tag{35}$$

The slew rates, $\dot{\delta}$ and $\dot{F}_x$, are penalized to improve the ability of the vehicle to achieve the resulting steering, acceleration, and brake commands

$$J_u = \Delta t \sum_{i=0}^{N} u_i^{\mathrm{T}} R u_i = \Delta t \sum_{i=0}^{N} R_{\dot{\delta}}\dot{\delta}_i^2 + R_{\dot{F}}\dot{F}_{x,i}^2 \tag{36}$$

To avoid excessive slip angles, the slip angles larger than $\tan(\alpha_j^{\mathrm{sl}})$ are penalized. This is implemented as the constraints (32k)–(32l) with slack variables $\tan(\hat{\alpha}_{f,i}^{\mathrm{excess}})$ and $\tan(\hat{\alpha}_{r,i}^{\mathrm{excess}})$

$$J_\alpha = \Delta t \sum_{i=1}^{N} (Q_\alpha \tan(\hat{\alpha}_{f,i}^{\mathrm{excess}})^2 + Q_\alpha \tan(\hat{\alpha}_{r,i}^{\mathrm{excess}})^2) \tag{37}$$

Finally, to improve robustness with respect to the tire–road friction, tire utilization above a fraction k_μ of the maximum tire force is penalized. This is implemented with the constraints (32m)–(32n) with a quadratic cost on the slack variables $\hat{F}_{f,i}^2$ and $\hat{F}_{r,i}^2$

$$J_\mu = \Delta t \sum_{i=1}^{N} \left(Q_F \left(\hat{F}_{f,i}^2\right)^2 + Q_F \left(\hat{F}_{r,i}^2\right)^2 \right) \tag{38}$$

3) Slack Variables: The slack variables included to soften the constraints (32j)–(32n) are collected in z_i

$$z_i = [e_i^{\mathrm{intr.}}, \tan(\alpha_{f,i}^{\mathrm{excess}}), \tan(\alpha_{r,i}^{\mathrm{excess}}), \hat{F}_{f,i}^2, \hat{F}_{r,i}^2]^{\mathrm{T}} \tag{39}$$

Additionally, in the RTI implementation, the nonlinear inequality constraints (32g)–(32n) are reformulated to equality constraints like (8) by introducing additional slack variables.

4) Other Constraints: The constraints on $\dot{\delta}$ (32c), on $\dot{F}_x$ (32d), on δ (32e), and on the power (32g) model the limits of the vehicle actuation. The constraints (32h)–(32i) limit the commanded longitudinal tire forces to remain within the tire–road friction limit.

TABLE I
PARAMETERS

Notation	Value	Unit
a / b / h	1.194 / 1.436 / 0.55	m
m	1868	kg
I_{zz}	3049	kg·m^2
$C^N_{\alpha f}$ / $C^N_{\alpha r}$	15 / 25	1
μ_f / μ_r	0.9 / 1.03	1
C_{d0}	218	N
C_{d2}	0.4243	N/(m/s)^2
$\dot{\delta}^{\max}$	20	deg/s
$\delta^{\max}$	27	deg
$\dot{F}_x^{\max}$	10	kN/s
$P^{\lim}$	172	kW
χ^{drive} / χ^{brake}	1 / 0.75	1
k_F / ε	1000 / 0.025	N
N	149	1
Δt	70	ms
k_μ	0.9	1
Q_e	$1/0.1^2$	1/m^2
Q_ϕ	$1/5^2$	1/deg^2
$R_{\dot{\delta}}$	$1/5^2$	1/(deg/s)^2
$R_{\dot{F}}$	$1/10^2$	1/(kN/s)^2
Q_α	$1/(0.5)^2$	1
Q_F	$1/(1-k_\mu^2)^2$	1

TABLE II
AVERAGE COMPUTATIONAL TIME

Solver	Horizon	Prep. Time	Solve Time	Total Time
RTI exp.	$N = 149$	13.5 ms	25.8 ms	39.3 ms
RTI sim.	$N = 149$	13.8 ms	21.0 ms	34.8 ms
RTI sim.	$N = 99$	11.5 ms	13.2 ms	24.8 ms
RTI sim.	$N = 49$	8.9 ms	6.6 ms	15.5 ms
Ipopt sim.	$N = 149$	-	389.4 ms	389.4 ms
Ipopt sim.	$N = 99$	-	207.6 ms	207.6 ms
Ipopt sim.	$N = 49$	-	72.0 ms	72.0 ms

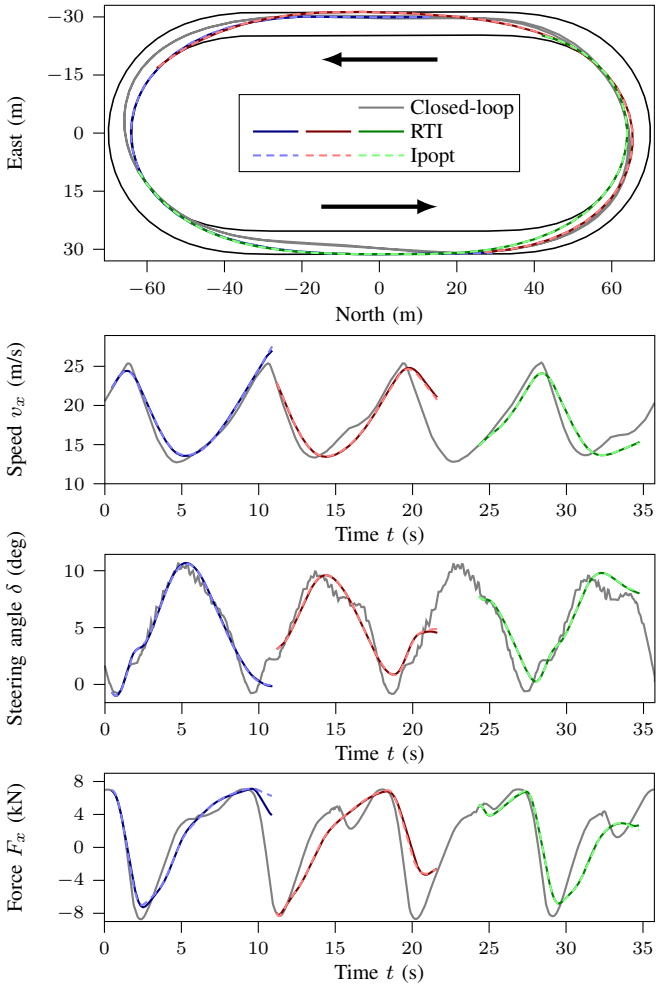

Fig. 2. Closed-loop trajectories obtained in the experiment with an automated vehicle using RTI together with three samples of the planned trajectories and the corresponding solutions obtained with Ipopt.

IV. EXPERIMENTAL RESULTS

An Experiment is conducted on an automated 2018 Volkswagen Golf VII GTI Performance (see Fig. 1) driving around an oval track. The RTI implementation used to control the vehicle is real-time capable, able to perform online planning and control with a planning horizon of 10.43 s, using $N = 149$ prediction steps. Parameters for the vehicle model and the OCP are listed in Table I. The QP subproblem (7) is symbolically computed using CasADi [12]. The evaluated QP subproblems are solved using the sparse QP solver OSQP [13].

A. Computational Requirements

The NMPC code is executed on a Intel Core i7-5700EQ CPU. Table II compares the computational time spent during the experiment with simulated results, and the nonlinear interior point solver Ipopt [14] used together with the linear solver MA57 [15]. The simulations waited for the solver to finish before continuing to enable comparisons with solvers not able to solve the OCP in real time. The preparation time spent by RTI (step 2 of the RTI algorithm in Sec. II-B) is slowly increasing with a longer planning horizon, while the time spent by the QP solver OSQP scales linearly with the planning horizon. Comparing the total time spent with the sampling time of 70 ms, Ipopt is not capable to solve the OCP online.

B. Convergence

The closed-loop path and trajectories for two laps, autonomously driven around an oval track by the experimental vehicle, are shown in Fig. 2. Additionally, three samples of the planned trajectories obtained with RTI during the execution are shown. For comparison, the corresponding solutions obtained with Ipopt for the same nonlinear optimization problems are shown, which have been obtained offline after the experiment. The RTI solutions are very similar to those obtained with the more computationally demanding Ipopt, which means that RTI has converged close to the optimum of the nonlinear problem.

The top plot in Fig. 2 shows that RTI originally plans a path close to the inside of the track at corner entry. The difference between the closed-loop trajectory and the initial plan on corner entry arises from a 1 deg tracking error of the low-level steering controller. As observed in the red planned trajectory in Fig. 2,

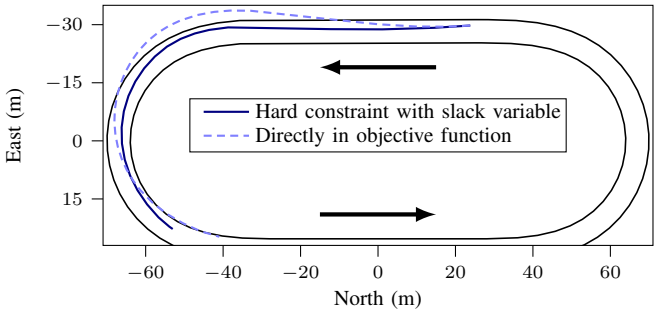

Fig. 3. Comparison of planned RTI paths with different formulations for the soft constraints on the road bounds.

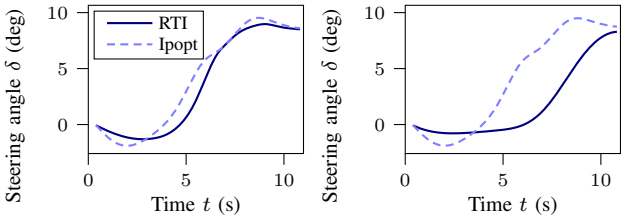

Fig. 4. Comparison of planned trajectories when reformulating the OCP like (8) (left) and without removing the nonlinear inequality constraints (right).

RTI is able to quickly converge to the new solution in the case of this model mismatch, changing to a strategy with a late apex, a rather typical racing line for a human driver.

V. Implementation Details

Gros *et al.* highlight a number of important details for the successful implementation of RTI [7]. This section highlights additional important components of the successful RTI implementation demonstrated in the previous section.

A. Soft Constraints

To avoid infeasibility issues when applying NMPC to real systems, a common method is to use soft constraints. One way to introduce a soft constraint is to include it directly in the objective function. For example, rather than introducing $e_i^{\text{intr.}}$ as a slack variable (32j), it can directly be implemented as $e_i^{\text{intr.}} = \max(e_i - e^{\max}(s_i), 0) + \min(e_i - e^{\min}(s_i), 0)$. Implementing a soft constraint directly in the objective function is computationally cheap as long as the constraint is not violated, but the constraint is not visible for the QP solver unless it was violated in the previous iteration and can thus result in instabilities. Comparing the path planned with RTI during the initialization phase after two partial Newton steps, with $\alpha = 0.25$ and $\alpha = 0.5$, demonstrates how implementing the road constraint directly in the objective function can lead to a large constraint violation (see Fig. 3). By introducing hard constraints like (32j)–(32n) and then softening them with slack variables, constraint violation is included as a part of the QP.

B. Inequality Constraints

It is common to use a QP subproblem of the form of (7) even though the NLP has inequality constraints. Keeping the nonlinear inequality constraints in the OCP (32) results in approximately 15% faster solve times of the QP subproblems than when removing them using slack variables like in (8), but results in slower convergence of the OCP (32). Figure 4 shows the difference in convergence during the initialization phase after five partial Newton steps, with $\alpha = 0.1, 0.2, \ldots, 0.5$.

VI. Conclusions

This paper has demonstrated that it is possible to use RTI for vehicle planning and control to achieve a planning horizon longer than 10 s, similar to the lookahead horizon expected of a human driver. Further, the solutions obtained with RTI are close to the local optimum obtained by Ipopt, while requiring a fraction of the computational time. Finally, experimental verification on an automated vehicle demonstrates that the implementation is robust to real-world disturbances, with the capability to rapidly replan in the case of a model mismatch.

References

[1] M. Brown and J. C. Gerdes, "Coordinating tire forces to avoid obstacles using nonlinear model predictive control," *IEEE Trans. Intell. Veh.*, vol. 5, no. 1, pp. 21–31, 2020.

[2] N. Li, E. Goubault, L. Pautet, and S. Putot, "A real-time NMPC controller for autonomous vehicle racing," in *2022 6th Int. Conf. Automation, Control and Robots (ICACR)*, 2022, pp. 148–155.

[3] V. A. Laurense and J. C. Gerdes, "Long-horizon vehicle motion planning and control through serially cascaded model complexity," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 1, pp. 166–179, 2022.

[4] B. Wohner, F. Sevilla, B. Grueter, J. Diepolder, R. Afonso, and F. Holzapfel, "Hierarchical nonlinear model predictive control for an autonomous racecar," in *2021 20th Int. Conf. Advanced Robotics (ICAR)*, 2021, pp. 113–120.

[5] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control and Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.

[6] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1–51, 1995.

[7] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, 2020.

[8] L. Svensson, M. Bujarbaruah, A. Karsolia, C. Berger, and M. Törngren, "Traction adaptive motion planning and control at the limits of handling," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 5, pp. 1888–1904, 2022.

[9] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "NMPC for racing using a singularity-free path-parametric model with obstacle avoidance," in *21st IFAC World Congr.*, vol. 53, no. 2, 2020, pp. 14 324–14 329.

[10] E. Siampis, E. Velenis, S. Gariuolo, and S. Longo, "A real-time nonlinear model predictive control strategy for stabilization of an electric vehicle at the limits of handling," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 6, pp. 1982–1994, 2018.

[11] R. Quirynen, B. Houska, M. Vallerio, D. Telen, F. Logist, J. Van Impe, and M. Diehl, "Symmetric algorithmic differentiation based exact hessian SQP method and software for Economic MPC," in *53rd IEEE Conf. Decision and Control*, 2014, pp. 2752–2757.

[12] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[13] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.

[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[15] HSL, "A collection of Fortran codes for large scale scientific computation," 2021, Accessed: 2021-09-08. [Online]. Available: http://www.hsl.rl.ac.uk