21st IFAC World Congress 2020
Berlin, Germany, July 2020

Least Conservative Linearized Constraint Formulation for Real-Time Motion Generation

Bárbara Barros Carlos [*], **Tommaso Sartor** [**], **Andrea Zanelli** [***], **Moritz Diehl** [***], **Giuseppe Oriolo** [*]

[*] *Department of Computer, Control, and Management Engineering (DIAG), Sapienza Università di Roma, Italy (e-mail: barros, oriolo@diag.uniroma1.it).*
[**] *MECO Research Team, Department of Mechanical Engineering, KU Leuven, Belgium (e-mail: tommaso.sartor@kuleuven.be).*
[***] *Department of Microsystems Engineering (IMTEK), University of Freiburg, Germany (e-mail: andrea.zanelli, moritz.diehl@imtek.uni-freiburg.de).*

Abstract: Today robotics has shown many successful strategies to solve several navigation problems. However, moving into a dynamic environment is still a challenging task. This paper presents a novel method for motion generation in dynamic environments based on real-time nonlinear model predictive control (NMPC). At the core of our approach is a least conservative linearized constraint formulation built upon the real-time iteration (RTI) scheme with Gauss-Newton Hessian approximation. We demonstrate that the proposed constraint formulation is less conservative for planners based on Newton-type method than for those based on a fully converged NMPC method. Additionally, we show the performance of our approach in simulation, in a scenario where the Crazyflie nanoquadcopter avoids balls and reaches its desired goal in spite of the uncertainty about when the balls will be thrown. The numerical results validate our theoretical findings and illustrate the computational efficiency of the proposed scheme.

Keywords: nonlinear control, optimization problems, numerical methods, obstacle avoidance, real-time tasks.

1. INTRODUCTION

Autonomous navigation in dynamic environments still poses an important challenge for robotics research. In contrast to static scenarios, where global path planning strategies are well-suited, in dynamic environments the decision about motion must be based on the online perception of the world and on the fast reaction-time behavior. Traditional motion planning methods typically rely on graph-search methods (Dijkstra, 1959; Hart et al., 1968), combinatorial methods (Bhattacharya and Gavrilova, 2008), or sampling-based methods (LaValle, 2006; Kuffner and LaValle, 2000). Despite the effectiveness of those algorithms in finding a path between two given points, the quality of the path obtained may be far from optimal. Thus, the research to improve the path refinement led to a common subdivision of the problem into a *global* and *local* planner.

The local planner accounts for the dynamic constraints and generates sets of feasible local trajectories. Popular algorithms used in this layer are dynamic window approach (DWA) (Brock and Khatib, 1999) and timed elastic band (TEB) (Quinlan and Khatib, 1993). More recent motion planning methods focus, instead, on optimization-based strategies. Benjamin et al. (2019) develop the concept of linear interval programming functions, which is based on multi-objective optimization. In this case, the collision avoidance constraint is defined as a shrinking convex polygon around the obstacle. As the optimization problem seeks the set of so-called Pareto optimal solutions – which is rather demanding – its application in systems with tight runtime requirements is usually not possible. Another approach is risk-constrained model predictive control (Hakobyan et al., 2019), where expectation constraints are defined within a linearly constrained mixed-integer convex program. However, the efficiency of this method highly depends on the tightness of the continuous linear program relaxations. Other techniques involve manifold trajectory optimization (Watterson et al., 2020) and convex decomposition through interval allocation (Tordesillas et al., 2019), both exploiting the inflated ellipsoids concept for ensuring collision-free trajectories. These approaches have obtained promising results but the solutions achieved for local replanning often lead to optimal trajectories that are far too conservative.

Nonlinear model predictive control (NMPC) is the cornerstone of advanced process control and has gained increasing attention in the robotics community in recent years. It is a feedback control strategy based on the solution of finite-horizon optimal control problems (OCP) where nonlinear dynamics and constraints are taken into account. A great challenge associated with the employment

of NMPC-based trajectory optimizers as local planners has been the availability of fast and reliable algorithms for online implementations. The advances in optimization strategies have enabled the development of a mature class of optimization algorithms such as *sequential quadratic programming* (SQP) and *interior point* (IP) methods. However, the real-time dilemma lies in the fact that while the SQP/IP iterations are being performed, the physical system evolves, and the information used to compute the state estimate becomes outdated (Gros et al., 2020). To tackle this issue, the real-time iteration (RTI) scheme is proposed in Diehl et al. (2005) which trades speed for accuracy while incorporating "globalization features" (Sathya et al., 2018).

This paper presents a novel approach for real-time motion generation for robotic systems. More precisely, we exploit the algorithmic ideas of the RTI scheme with inexact Hessian to formulate a least conservative linearized collision avoidance constraint. We provide a theoretical analysis proving that the proposed constraint formulation is less conservative for planners based on Newton-type method than for those based on a *fully converged NMPC* approach. This analysis is further leveraged to overcome the numerical difficulties associated with the (local) feasibility of the optimization problem. Our framework uses the implementation of the RTI strategy by means of the high-performance software package `acados` (Verschueren et al., 2019). The quadratic program (QP) solver is `HPIPM`, which relies on the hardware-tailored linear algebra library `BLASFEO` (Frison et al., 2018). Furthermore, we use the Crazyflie nanoquadcopter as a relevant example in order to assess the performance and implementability of our real-time planner.

With respect to the challenges at hand, the main contributions of our work are twofold:

- An NMPC-based algorithmic framework for real-time motion generation that is reliable and guarantees less conservative optimal trajectories for robotic systems.
- A numerical validation that shows the efficiency of the proposed approach.

This paper is outlined as follows: in Section 2 preliminary concepts on NMPC and the RTI scheme are presented. In Section 3, the proposed real-time planner is introduced. The general planner problem formulation is tailored to the Crazyflie nanoquadcopter example, and the generation of the global trajectories are described in Section 4. Section 5 reports the results of the closed-loop simulations and the discussion of our approach. In Section 6, the conclusions are summarized.

2. PRELIMINARIES

2.1 Nonlinear Model Predictive Control

NMPC is an optimization-based control technique that has gradually become known in the robotics community over the past few years. In order to use NMPC to control a system, one has to solve a nonlinear nonconvex optimal control problem (OCP). Throughout this paper, let us consider the following general NMPC tracking formulation as the OCP that needs to be solved numerically in each time step:

$$\min_{\substack{\xi_0,\dots,\xi_N,\\ u_0,\dots,u_{N-1}}} \quad \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(\xi_i,u_i)-\tilde{\eta}\|_W^2 + \frac{1}{2}\|\eta_N(\xi_N)-\tilde{\eta}_N\|_{W_N}^2 \tag{1a}$$

s.t.

$$\xi_0 - \bar{\xi}_0 = 0, \tag{1b}$$
$$\xi_{i+1} - F(\xi_i,u_i) = 0, \qquad i=0,\dots,N-1, \tag{1c}$$
$$g(\xi_i,u_i) \le 0, \qquad i=0,\dots,N-1, \tag{1d}$$
$$g_N(\xi_N) \le 0, \tag{1e}$$

where $\xi \in \mathbb{R}^{n_\xi}$ and $u \in \mathbb{R}^{n_u}$ denote the state and input trajectories of the discrete-time system whose dynamics are described by $F : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_\xi}$. The functions in the Lagrange and Mayer least-squares terms are denoted by $\eta : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\eta_N : \mathbb{R}^{n_\xi} \to \mathbb{R}$, and will be weighted by the symmetric positive-definite matrices W and W_N, respectively. Variables $\tilde{\eta} : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and $\tilde{\eta}_N : \mathbb{R}^{n_\xi} \to \mathbb{R}$ denote the time-varying references. The functions $g : \mathbb{R}^{n_\xi} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ and $g_N : \mathbb{R}^{n_\xi} \to \mathbb{R}^{n_{c,N}}$ define the path and terminal constraints. Finally, N and $\bar{\xi}_0$ denote the horizon length and the current state of the system, respectively.

Problem (1) can be solved by a state-of-the-art optimization algorithm. Among others, two main classes of approaches stand out: SQP and IP methods. In our work, we use a variant of SQP for NMPC embedded applications, the RTI scheme. An explanation of the algorithm's main idea is presented in the next subsection.

2.2 The Real-Time Iteration Scheme

In real-time NMPC applications, the nonlinear program (NLP) (1) needs to be solved at every sampling instant under the available computational time. To that end, Diehl et al. (2005) propose an efficient Newton-type scheme that allows us to approximately solve the OCPs during the runtime of the real process, the so-called RTI scheme. In this strategy, only a single linearization and dense QP solve are carried out per sampling instant, leading to an approximate feedback control policy. Our approach uses the implementation of the RTI scheme by means of the high-performance software package `acados`. Thus, after linearization, the obtained QP for problem (1) used in `acados` reads as follows:

$$\begin{aligned}\min_{\substack{\xi_0,\dots,\xi_N,\\ u_0,\dots,u_{N-1},\\ s_0,\dots,s_N}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \begin{bmatrix}\xi_i\\ u_i\end{bmatrix}^T \overbrace{\begin{bmatrix}Q_i & S_i\\ S_i & R_i\end{bmatrix}}^{H_i} \begin{bmatrix}\xi_i\\ u_i\end{bmatrix} + \begin{bmatrix}q_i\\ r_i\end{bmatrix}^T \begin{bmatrix}\xi_i\\ u_i\end{bmatrix} \\ & + \frac{1}{2}(\xi_N^T Q_N \xi_N) + q_N^T \xi_N \\ & + \frac{1}{2}\sum_{i=0}^{N}(s_i^T P_i s_i) + p_i^T s_i \end{aligned} \tag{2a}$$

s.t.

$$\xi_0 - \bar{\xi}_0 = 0, \tag{2b}$$
$$\xi_{i+1} - A_i\xi_i - B_i u_i - c_i = 0, \qquad i=0,\dots,N-1, \tag{2c}$$
$$s_i + G_i^\xi \xi_i + G_i^u u_i \le 0, \qquad i=0,\dots,N-1, \tag{2d}$$
$$s_N + G_N^\xi \xi_N \le 0, \tag{2e}$$
$$0 \le s_i, \qquad i=0,\dots,N. \tag{2f}$$

The linearized dynamics and inequality constraints use the following matrices:

$$\begin{aligned} A_i &:= \nabla_\xi F(\xi_i^k, u_i^k)^T, \quad B_i := \nabla_u F(\xi_i^k, u_i^k)^T, \\ G_i^\xi &:= \nabla_\xi g(\xi_i^k, u_i^k)^T, \quad G_i^u := \nabla_u g(\xi_i^k, u_i^k)^T, \\ G_N^\xi &:= \nabla_\xi g_N(\xi_N^k)^T, \\ c_i &:= F(\xi_i^k, u_i^k) - A_i\xi_i^k - B_i u_i^k, \\ s_i &:= g(\xi_i^k, u_i^k) - G_i^\xi \xi_i^k - G_i^u u_i^k, \\ s_N &:= g_N(\xi_N^k) - G_N^\xi \xi_N^k, \end{aligned}$$

where the superscript k denotes the linearization point for the states and inputs obtained at the previous QP iteration. The quadratic cost uses the exact gradients q_i, r_i, q_N, the Gauss-Newton Hessian approximation H_i, both as defined in (Zanelli et al., 2018). The Hessian and the gradient with respect to the slack variables s_i are denoted by P_i and p_i, respectively. In `acados`, the slack variables are introduced and associated with the soft constraints on the outputs.

3. LEAST CONSERVATIVE REAL-TIME PLANNER

In this section, we detail and discuss the reformulations and parametrizations of QP (2) that are used in this paper.

3.1 Collision Avoidance Constraint and Physical Limits

To ensure kinodynamically feasible and collision-free trajectories, we consider a constrained OCP. Having the dynamic model of the robot as a constraint provides kinodynamically feasible state trajectories. Collision avoidance, on the other hand, is granted by constraining the robot's distance with respect to an obstacle $o \in \mathbb{R}^n$ in the *workspace* $\mathcal{W} \in \mathbb{R}^n$ to be larger than the *clearance* $d_\mathcal{C} \in \mathbb{R}^+$. The obstacles, here considered dynamic, are described as point-mass models. Furthermore, we assume a single-body robot, which can be well approximated by a point if one defines the clearance so as take into account: i) the actual size of the robot, ii) the actual size of the obstacle, iii) a safety margin between them. When extending our approach to a multi-body robot system, one needs to consider the enlargement of the clearance. Ultimately, in our method, for each obstacle present in the workspace, an additional collision avoidance constraint must be included in the OCP.

To formulate the collision avoidance constraint, we will consider the *distance function* to be the Euclidean distance between the robot and an obstacle in the workspace $d_\mathcal{O} : \mathcal{W} \to \mathbb{R}$. It can be formally defined as the following:

Definition 1. (Distance Function). Let us define the distance function for an arbitrary point $p \in \mathcal{W}$ with respect to an obstacle as

$$d_\mathcal{O}(p) = d_\mathcal{O}(p; o) = \|p - o\|_2,$$

and consider bounds, i.e. $d_\mathcal{C} \leq d_\mathcal{O}(p)$ and $\exists\, d_\mathcal{M}$ such that $d_\mathcal{O}(p) \leq d_\mathcal{M} \forall p \in \mathcal{W}$.

Based on the aforementioned definition, the obstacle avoidance constraint can be concisely written as

$$d_\mathcal{C} \leq \|p - o\|_2. \tag{3}$$

Note that (3) is not differentiable at $p = o$ because we can get different limits by choosing different paths to the origin. Hence, to tackle this problem, one can square both sides of the inequality,

$$d_\mathcal{C}^2 \leq \|p - o\|_2^2, \tag{4}$$

and recover the desirable property of being continuously differentiable everywhere. However, depending on the optimization algorithm behind the planner at hand, one constraint formulation can be more advantageous than the other.

For a planner whose optimization algorithm is based on a Newton-type method, e.g. with `acados`, note that only one system linearization and one QP (2) are performed per sampling time. Each QP corresponds to a linear approximation of the original NLP along a time-varying trajectory. In this configuration, squaring the Euclidean norm, as in (4), provides local trajectories that are over-conservative. In a case where the linearization points are far away from the boundaries of the linearized feasible set of the original NLP, the use of (3) leads to linear constraints that are "least" conservative. We will formalize this statement in Proposition 2. In contrast, for a planner whose optimization algorithm is based on a fully converged NMPC method, e.g. with `IPOPT` (Wächter and Biegler, 2006), in general, no constraint formulation has a distinct advantage over the other. As in this case, the iterations continue until convergence or a certain termination conditions are reached, the feasible sets for (3) and (4) are the same.

Thus, in view of the algorithmic ideas of the RTI scheme, our main theoretical result for Least Conservative Linearized (LCL) constraint formulation is the following:

Proposition 2. Let $H_1(\bar{p}) := \{p : c_1(\bar{p}) + \nabla_p c_1(\bar{p})^T(p - \bar{p}) \geq 0\}$ and $H_2(\bar{p}) := \{p : c_2(\bar{p}) + \nabla_p c_2(\bar{p})^T(p - \bar{p}) \geq 0\}$, where $c_1(p) := \|p - o\|_2 - d$ and $c_2(p) := \|p - o\|_2^2 - d^2$, denote the half-spaces defined by the linearization of the constraints $c_1(p) \geq 0$ and $c_2(p) \geq 0$ at any feasible point $\bar{p}$, respectively. Moreover, let $S := \{p : c_1(p) \geq 0\}$ denote the set defined by the original nonlinear constraints. Then, $H_1(p) \subset S$ and $H_2(p) \subset S$. Moreover, we have that $H_2(p) \subseteq H_1(p)$.

Proof. Define $c_{1,\text{lin}}(\bar{p}, p) := c_1(\bar{p}) + \frac{\partial c_1}{\partial p}(p - \bar{p})$ and $c_{2,\text{lin}}(\bar{p}, p) := c_2(\bar{p}) + \frac{\partial c_2}{\partial p}(p - \bar{p})$. Due to convexity of $c_1(p)$ and $c_2(p)$, we have that, for any $\bar{p}$ we have that

$$0 \leq c_{1,\text{lin}}(\bar{p}, p) \leq c_1(p) \tag{5}$$

and

$$0 \leq c_2, \text{lin}(\bar{p}, p) \leq c_2(p) \tag{6}$$

such that $p \in H_1(\bar{p}) \implies p \in S$ and $p \in H_2(\bar{p}) \implies p \in S$, which proves the first statement. In order to prove that $H_2(p) \subseteq H_1(p)$, we proceed as follows. Define the auxiliary functions $\hat{c}_2, \hat{c}_1 : \mathbb{R} \to \mathbb{R}$ as $\hat{c}_2(v) := v^2 - d^2$ and $\hat{c}_1(v) := v - d$. Moreover, define their linearizations $\hat{c}_{1,\text{lin}}(\bar{v}, v) = \hat{c}_1(\bar{v}) + \frac{\partial \hat{c}_1}{\partial v}(v - \bar{v})$ and $\hat{c}_{2,\text{lin}}(\bar{v}, v) = \hat{c}_2(\bar{v}) + \frac{\partial \hat{c}_2}{\partial v}(v - \bar{v})$. Due to convexity in v of $\hat{c}_{1,\text{lin}}$ and affinity in v of $\hat{c}_{2,\text{lin}}$ we have that, for any d, and any $\bar{v}$, the following holds:

$$\min_{v \geq d} \hat{c}_{1,\text{lin}}(\bar{v}, v) = \min_{v \geq d} \hat{c}_1(v) = 0 \tag{7}$$

and

$$\min_{v \geq d} \hat{c}_{2,\text{lin}}(\bar{v}, v) \leq \min_{v \geq d} \hat{c}_2(v) = 0. \tag{8}$$

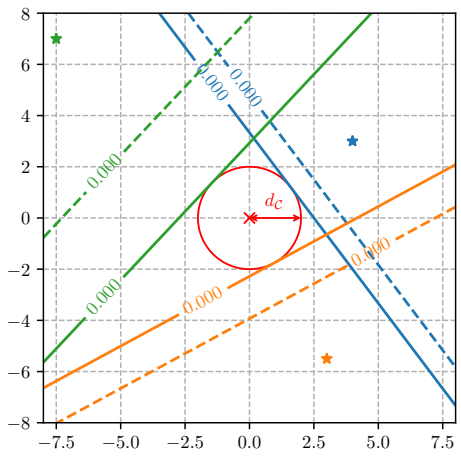


Fig. 1. Boundaries of the linearized feasible set – comparison between constraint formulations: linearization points $p^\star$ are represented as $\star$, in red the clearance around an obstacle marked by $\times$, $\|\cdot\|_2^2$ in dashed line and $\|\cdot\|_2$ in solid line. The contour lines at zero are shown for half-spaces $H_1(\bar{p})$ and $H_2(\bar{p})$.

Define $r(p) := \|p - o\|$. Then, we can state the following:

$$\begin{aligned}\min_{p\in H_1(p)} c_{2,\text{lin}}(\bar{p},p) &= \min_{p} \quad c_{2,\text{lin}}(\bar{p},p) \\ &\quad \text{s.t } c_{1,\text{lin}}(\bar{p},p)\geq 0 \\ &= \min_{p} \hat{c}_{2,\text{lin}}(r(\bar{p}), r(\bar{p}) + \frac{\partial r}{\partial p}(p-\bar{p})). \\ &\quad \text{s.t } \hat{c}_{1,\text{lin}}(r(\bar{p}), r(\bar{p}) + \frac{\partial r}{\partial p}(p-\bar{p}))\geq 0\end{aligned} \tag{9}$$

Introducing the change of variables $v = r(\bar{p}) + \frac{\partial r}{\partial p}(p - \bar{p})$, and using the fact that $\hat{c}_{1,\text{lin}}(r(\bar{p}), r(\bar{p}) + \frac{\partial r}{\partial p}(p - \bar{p})) \geq 0$ if and only if $v \geq d$, we obtain

$$\min_{p\in H_1(p)} c_{2,\text{lin}}(\bar{p},p) = \min_{v\geq d} \hat{c}_{2,\text{lin}}(r(\bar{p}), v) \leq \min_{v\geq d} \hat{c}_2(v) = 0. \tag{10}$$

This last inequality shows that $H_2(\bar{p}) \subseteq H_1(\bar{p})$ concluding the proof. □

The boundaries of the linearized feasible set, considering three distinct regular points $p^\star$, for constraint formulations (3) and (4) can be seen in Fig. 1. We notice that the linearized constraint formulation defined by (3) provides a solution that is closer to the lower bound $d_\mathcal{C}$.

Moreover, in order to account for the limits on the actuator, we additionally impose box constraints on the control input of the form $\mathcal{U} := \{u \in \mathbb{R}^{n_u} : u_{\min} \leq u \leq u_{\max}\}$.

3.2 Constraints Violation

The constraint presented in (3) guarantees collision avoidance instantaneously since its evaluation occurs only at t_i and t_{i+1}. This, however, does not ensure that there is no collision in-between shooting nodes, which may result in constraint violation. As a result, the predicted trajectory between time points may be seen penetrating the obstacle.

This issue can be addressed by expanding the obstacle, in all directions, by the maximum incursion distance (Kuwata, 2003), or by using the hyperplane separation theorem as in (Brossette and Wieber, 2017). In this paper, we do not use any special method to ensure collision-free trajectories in-between successive positions of the robot, as it comes with a higher computational burden. Therefore, we rely on a reasonable choice for the size of the discretization time step.

3.3 Feasibility and Soft Constraints

One major drawback regarding the use of hard constraints is that they may render the optimization problem infeasible: this is especially true in the case of state constraints (Frison and Jørgensen, 2015), such as the one in (3). In a case where the input constraints represent a physical limitation, they have to be enforced all the time. State constraints, instead, are only often desired – although are not always physically necessary – and hence should be satisfied whenever possible.

A more systematic approach for dealing with infeasibility is to modify the cost function to include a penalization on the violation of the collision avoidance constraint. This is usually achieved by introducing *slack variables* associated with the soft constraint, and heavily penalizing them. The optimization algorithm searches for a solution that minimizes the original cost function while keeping the slack variables equal to zero whenever possible.

Scokaert and Rawlings (1999) discuss the inclusion of a ℓ_1-norm $\mu||\epsilon||_1$ penalization term of the slack in order to obtain an exact penalty function – i.e. the controller violates the constraints only when it is necessary. A hazard with ℓ_1 penalty functions is their inherent non-smoothness at optimal points which may degrade the final solution accuracy. On the other hand, the ℓ_2^2-norm $\mu||\epsilon||_2^2$ benefits of being smooth and having "simple" derivatives. A major drawback to the ℓ_2^2 penalty function is the uneven way that it penalizes constraints, such that μ has to be very large to enforce asymptotic feasibility (Griffin and Kolda, 2010). In this work, the use of a ℓ_1-norm is preferred as a means to obtain an exact penalty function.

4. MOTION GENERATION BENCHMARK

4.1 System Model

Let $\{\mathcal{B}\}$ be the body-fixed frame located at the center of mass (CoM) of the nanoquadcopter which is aligned with the North-West-Up (NWU) inertial frame $\{\mathcal{I}\}$.

Table 1. Quadrotor state variables.

$p := (x, y, z)$	Position vector of the nanoquadrotor in $\{\mathcal{I}\}$ and described in $\mathbb{R}^3$.
$q := (q_w, q_x, q_y, q_z)$	Set of attitude unit quaternions described in $\mathbb{H}$.
$v_b := (v_x, v_y, v_z)$	Linear velocities in $\{\mathcal{B}\}$ and described in $\mathbb{R}^3$.
$\omega := (\omega_x, \omega_y, \omega_z)$	Angular rates described in $\mathbb{R}^3$.

Given a nonlinear dynamic model for a quadcopter in the form of

$$\dot{\xi} = f(\xi, u), \tag{11}$$

let us then define a state vector such as

$$\xi := (p, q, v_b, \omega)^T \in \mathbb{R}^{13}, \tag{12}$$

where the state variable vectors are presented in Table 1. The dynamics can be described using the Newton-Euler

formulation as in (Luis and Ny, 2016), leading to the dynamic system used in this paper. Furthermore, assuming that the rotor inertia is considerably small such that it is possible to change the propeller's rotational speed Ω_i instantaneously, we consider their collection as the control inputs of our system

$$u := (\Omega_1, \Omega_2, \Omega_3, \Omega_4)^T \in \mathbb{R}^4. \tag{13}$$

4.2 Tailored Problem Formulation

In this section, we tailor the NMPC formulation (1) into the following soft-constrained problem with LCL constraint formulation:

$$\begin{aligned}
\min_{\substack{\xi_0,\dots,\xi_N,\\ u_0,\dots,u_{N-1},\\ \epsilon_0,\dots,\epsilon_N}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \|\eta(\xi_i,u_i)\|^2_W + \frac{1}{2}\|\eta_N(\xi_N)\|^2_{W_N} + \\
& \frac{\mu}{2}\sum_{i=0}^{N} \|\epsilon_i\|_1 && \text{(14a)}\\
\text{s.t.} \quad & \\
\xi_0 - \bar{\xi}_0 &= 0, && \text{(14b)}\\
\xi_{i+1} - F(\xi_i, u_i) &= 0, \qquad i = 0,\dots,N-1, && \text{(14c)}\\
u_i &\in \mathcal{U}, \qquad i = 0,\dots,N-1, && \text{(14d)}\\
\epsilon_i + d_{\mathcal{O}}(p_i) &\geq d_{\mathcal{C}}, \qquad i = 0,\dots,N, && \text{(14e)}\\
\epsilon_i &\geq 0, \qquad i = 0,\dots,N, && \text{(14f)}
\end{aligned}$$

where $\epsilon \in \mathbb{R}^{n_\epsilon}$ is a vector that contains all slack variables over the horizon $\epsilon := (\epsilon_0, \dots, \epsilon_N)^T$, and μ is the penalty weight.

Remark 3. In our approach, for each obstacle in the workspace, we explicitly add new constraints (14e)-(14f) to the NLP. In this paper, two dynamic obstacles are considered, thus $n_\epsilon = 2 \times N$.

We consider a sampling time of $t_s = 15$ ms, and $N = 50$ shooting nodes. To discretize the dynamics (11), we use an explicit Runge Kutta of 4th order (ERK4) integration by means of the automatic differentiation and modeling framework `CasADi` (Andersson et al., 2012). Moreover, the residuals of the least-squares cost are described as

$$\eta(\xi, u) := \begin{bmatrix} \xi_i - \xi_i^r \\ u_i - u_i^r \end{bmatrix}, \quad \eta_N(\xi_N) := \xi_N - \xi_N^r, \tag{15}$$

where the quantities in (15) with the r superscript stand for the precomputed desired references. The gradient with respect to the lower (μ_l) and upper (μ_u) slack penalty values, and weighting matrices are

$$\begin{aligned}
\mu_l &:= (14.5 \cdot 10^3, 14.5 \cdot 10^3),\\
\mu_u &:= (14.2 \cdot 10^3, 14.2 \cdot 10^3),\\
W &:= \text{blkdiag}(30 \cdot \mathbf{I_2}, 60, 1 \cdot 10^{-1} \cdot \mathbf{I_4}, 2, 3, 5,\\
&\qquad 3 \cdot \mathbf{I_3}, 5 \cdot 10^{-2} \cdot \mathbf{I_4}),\\
W_N &:= \text{blkdiag}(30 \cdot \mathbf{I_2}, 60, 1 \cdot 10^{-1} \cdot \mathbf{I_4}, 2, 3, 5,\\
&\qquad 3 \cdot \mathbf{I_3}).
\end{aligned} \tag{16}$$

The model parameters of this benchmark correspond to the parameters of the real Crazyflie which were obtained from the study in (Luis and Ny, 2016). The Crazyflie is an open-source off-the-self platform for research and education in robotics developed by the Swedish company Bitcraze [1]. For this robotic system, the imposed input bounds on the speed of the propellers are $u_{\min} = 0$, $u_{\max} = 22$ krpm, while the bounds for the slack variables are $\epsilon_{\min} = 0$, $\epsilon_{\max} = 100 \times 10^3$. Finally, the bounds for the collision avoidance constraint are $d_{\mathcal{C}} = 0.2$ m, $d_{\mathcal{M}} = 100 \times 10^3$ m.

[1] `https://www.bitcraze.io`

4.3 Reference Trajectory Generation

Two different global reference trajectories for the Crazyflie are considered. They have been generated offline by solving the OCP (1) for system (11) considering constraint (1d) of the form $\mathcal{U}$ using `CasADi`. We assume a prediction horizon of $T = 3$ s, $N = 200$ shooting nodes, discretizing the dynamics via ERK4 integration method. The resulting NLPs provide the optimal trajectories that drive the nanoquadcopter from the hovering state at $p := (0, 0, 0.4)^T$ to the final desired positions: $p_1^d := (1, -1, 1)^T$ and $p_2^d := (0.5, -0.5, 1)^T$. We regulate for $q^d := (1, 0, 0, 0)^T$, so that the geodesic arc can be approximated by a straight line without major problems. As the obstacles are dynamic, the global references are unaware of them. Finally, we make use of the interior-point solver `IPOPT`, exploiting the *just-in-time* compiling function evaluations, to solve this optimization problem.

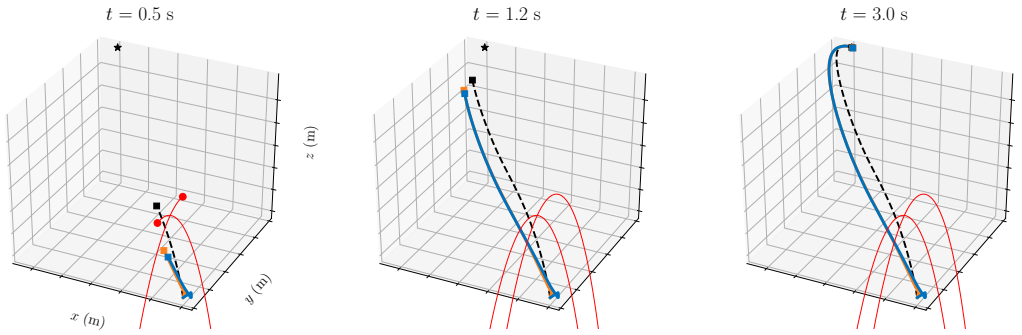
5. NUMERICAL SIMULATIONS AND DISCUSSION

5.1 Implementation Details

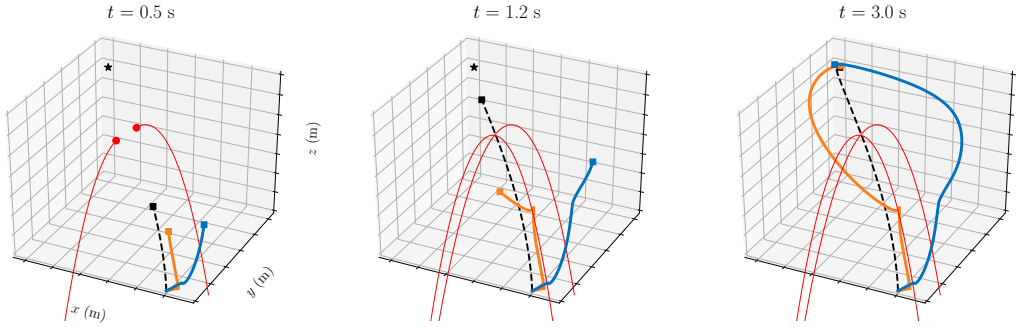
We make use of `acados` Python template-based interface to generate a library which implements our tailored problem formulation (14). The QPs arising in this NMPC formulation are solved using the high-performance interior-point method (`HPIPM`) solver that is built on top of the linear algebra package `BLASFEO`. The major difference between existing high-performance implementations of BLAS– and LAPACK–like routines for embedded applications and `BLASFEO` is that the last is optimized for small to medium scale matrices (Frison et al., 2018). The NLPs results were obtained on an Intel Core i5 @2.6 GHz running macOS Catalina. The `X64_INTEL_HASWELL` implementation of `BLASFEO` package has been used, which exploits a set of vectorized instructions for the target CPU. Additionally, for the NMPC solution, we use partial condensing `HPIPM`-based technique. This approach reformulates the large and sparse Hessian of problem (14) into a small and dense one which has a more suitable form for the QP solver.

5.2 Performance Evaluation

To evaluate the trajectory conservativeness introduced by the constraint approximation, we perform simulations comparing our real-time planner with LCL constraint formulation to its counterpart, a planner where constraint (14e) is defined by $\|\cdot\|_2^2$. The values of the slack penalties and weighting matrices used in both cases are those defined in (16). In simulation, the Crazyflie must travel from a hovering configuration to another while avoiding balls thrown at it. We assume that the obstacle's movement between two shooting nodes is ballistic. The results of our simulations are presented in Fig. 2 and Fig. 3.



(a) Final desired position p_1^d



(b) Final desired position p_2^d

Fig. 2. Generated trajectories among moving balls (red): real-time planner with LCL constraint formulation (orange), real-time planner with $\|\cdot\|_2^2$ constraint formulation (blue); reference trajectory is dashed. The $\star$ is the final goal.

As expected, the planner with LCL constraint formulation generates trajectories that are significantly closer to the minimum clearance $d_{\mathcal{C}}$, as compared to its counterpart. In addition, we note that there is no violation of the constraints. However, the effort required to tune the slack penalties directly impacts the amount of constraint violation. Much of the difficulty exists because, in this particular problem, the optimal solution will frequently lie on the boundary of the feasible set. Therefore, imposing very strict penalties – in the case of soft constraints – increases the difficulty in driving the solution towards the optimum while distancing the local trajectories from the lower bound. Conversely, if the penalty is not strict enough, then the search will tend to stall outside the feasible region and thereby violate the constraint. Although the tuning considered in this work has provided a remarkable closed-loop performance for both real-time planners – with no constraint violation – there is no guarantee that the violation will not occur. This is especially true in a real-world scenario, where the state estimation deals effectively with the uncertainty due to noisy sensor data and, to some extent, with random external factors, i.e. wind. One way to avoid this shortcoming and, therefore, recover feasibility is to retune the slack penalties of the soft-constrained problem.

To assess the computational complexity of the planner's algorithm, the average computational effort has been gathered. The values for final desired positions p_1^d and p_2^d are 4.76 ms and 7.97 ms, respectively. These results show the computational efficiency of the proposed scheme.

This paper is accompanied by a video showcasing the simulations: `https://youtu.be/ZRbGyikvsxw`.

6. CONCLUSION

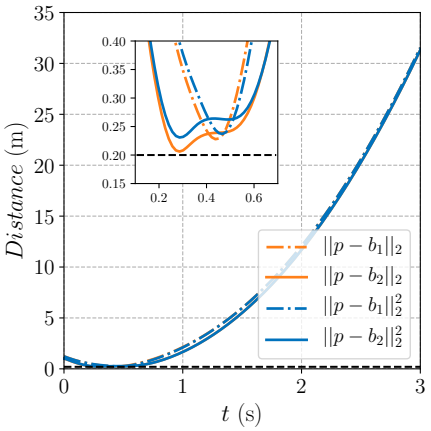
In this work, a new scheme for real-time motion generation for robotic systems has been presented. The key concepts that contribute to the effectiveness of our method stem from the algorithmic ideas tied to the RTI scheme to formulate a least conservative linearized constraint. We have shown a reformulation that guarantees less conservative trajectories for planners whose optimization algorithm is based on Newton-type method. Strategies to ensure both constraint satisfaction and feasibility of the optimization problem have also been discussed. Finally, our approach was tested in simulation, on a challenging example involving a high-dimensional quadcopter system, showing efficient computational performance. Future work will focus on real-world implementation.

REFERENCES

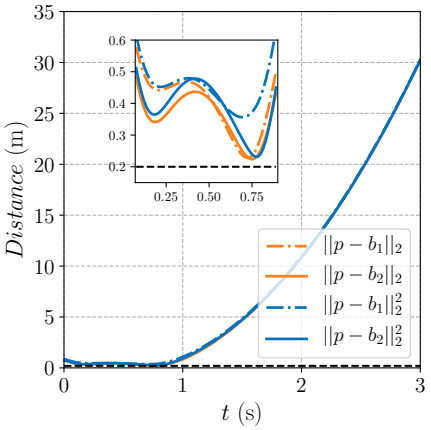
Andersson, J., Åkesson, J., and Diehl, M. (2012). CasADi: A symbolic package for automatic differentiation and optimal control. In *Recent advances in algorithmic differentiation*, 297–307. Springer.

Benjamin, M.R., Defilippo, M., Robinette, P., and Novitzky, M. (2019). Obstacle avoidance using multiobjective optimization and a dynamic obstacle manager. *IEEE Journal of Oceanic Engineering*, 44(2), 331–342.

Bhattacharya, P. and Gavrilova, M.L. (2008). Roadmap-based path planning-using the voronoi diagram for a



(a) Final desired position p_1^d



(b) Final desired position p_2^d

Fig. 3. Distance between the Crazyflie and the balls b_i with LCL constraint formulation (orange), and with $\|\cdot\|_2^2$ constraint formulation (blue). Inset shows the time instant of closest proximity to the minimum clearance.

clearance-based shortest path. *IEEE Robotics & Automation Magazine*, 15(2), 58–66.

Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *IEEE International Conference on Robotics and Automation*, volume 1, 341–346.

Brossette, S. and Wieber, P.B. (2017). Collision avoidance based on separating planes for feet trajectory generation. In *IEEE-RAS 17th International Conference on Humanoid Robotics*, 509–514.

Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5), 1714–1736.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1), 269–271.

Frison, G. and Jørgensen, J.B. (2015). Efficient solvers for soft-constrained MPC. In *19th Nordic Process Control Workshop*. Norwegian University of Science and Technology.

Frison, G., Kouzoupis, D., Sartor, T., Zanelli, A., and Diehl, M. (2018). BLASFEO: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software (TOMS)*, 44(4), 1–30.

Griffin, J.D. and Kolda, T.G. (2010). Nonlinearly constrained optimization using heuristic penalty methods and asynchronous parallel generating set search. *Applied Mathematics Research eXpress*, 2010(1), 36–62.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A., and Diehl, M. (2020). From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, 93(1), 62–80.

Hakobyan, A., Kim, G.C., and Yang, I. (2019). Risk-aware motion planning and control using CVaR-constrained optimization. *IEEE Robotics and Automation Letters*, 4(4), 3924–3931.

Hart, P.E., Nilsson, N.J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Kuffner, J.J. and LaValle, S.M. (2000). RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, volume 2, 995–1001.

Kuwata, Y. (2003). *Real-time trajectory design for unmanned aerial vehicles using receding horizontal control*. Master's thesis, Massachusetts Institute of Technology.

LaValle, S.M. (2006). *Planning algorithms*. Cambridge university press.

Luis, C. and Ny, J.L. (2016). Design of a trajectory tracking controller for a nanoquadcopter. *arXiv preprint arXiv:1608.05786*.

Quinlan, S. and Khatib, O. (1993). Elastic bands: Connecting path planning and control. In *IEEE International Conference on Robotics and Automation*, 802–807.

Sathya, A., Sopasakis, P., Van Parys, R., Themelis, A., Pipeleers, G., and Patrinos, P. (2018). Embedded nonlinear model predictive control for obstacle avoidance using PANOC. In *European Control Conference (ECC)*, 1523–1528.

Scokaert, P.O. and Rawlings, J.B. (1999). Feasibility issues in linear model predictive control. *AIChE Journal*, 45(8), 1649–1659.

Tordesillas, J., Lopez, B.T., Everett, M., and How, J.P. (2019). FASTER: Fast and safe trajectory planner for flights in unknown environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1934–1940.

Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Novoselnik, B., Frey, J., Albin, T., Quirynen, R., and Diehl, M. (2019). acados: a modular open-source framework for fast embedded optimal control. *arXiv preprint arXiv:1910.13753*.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1), 25–57.

Watterson, M., Liu, S., Sun, K., Smith, T., and Kumar, V. (2020). Trajectory optimization on manifolds with applications to quadrotor systems. *The International Journal of Robotics Research*, 39(2-3), 303–320.

Zanelli, A., Horn, G., Frison, G., and Diehl, M. (2018). Nonlinear model predictive control of a human-sized quadrotor. In *European Control Conference (ECC)*, 1542–1547.