Position and Orientation Tunnel-Following NMPC of Robot Manipulators Based on Symbolic Linearization in Sequential Convex Quadratic Programming

Alejandro Astudillo[1], Joris Gillis[1], Moritz Diehl[2], Wilm Decré[1], Goele Pipeleers[1], and Jan Swevers[1]

***Abstract*—The tunnel-following nonlinear model predictive control (NMPC) scheme allows to exploit acceptable deviations around a path reference. This is done by using convex-over-nonlinear functions as objective and constraints in the underlying optimal control problem (OCP). The convex-over-nonlinear structure is exploited by algorithms such as the generalized Gauss-Newton (GGN) method or the sequential convex quadratic programming (SCQP) method to reduce the computational complexity of the OCP solution. However, the modeling effort and engineering time required to implement these methods is high. We address the problem of reducing the modeling effort in the implementation of SCQP, focusing on a standard sequential quadratic programming (SQP) implementation where symbolic linearization is applied to the nonlinear part of the convex-over-nonlinear functions in the objective and constraints. The novelty of this paper is twofold. It introduces a novel operator that applies symbolic linearization in a transparent and easy way to solve nonconvex OCPs with the SCQP method, and introduces a meaningful representation of an orientation-tunnel for robotic applications by means of a convex-over-nonlinear constraint, which preserves the convexity exploitation by the SCQP method. The proposed technique is demonstrated in a tunnel-following task for a 7-degrees-of-freedom manipulator.**

***Index Terms*—Optimization and optimal control, motion control, tunnel-following, sequential convex quadratic programming, symbolic linearization.**

I. INTRODUCTION

PATH-FOLLOWING nonlinear model predictive control (NMPC) [1] is of significant importance in the execution of robotics applications defined by a path reference and subject to constraints that must be satisfied during such task execution. Certain applications, such as collaborative robots [2], welding [3], gluing [4] and bin-picking [5], allow deviations from a desired path within a defined neighborhood of the path, with applications like gluing or bin-picking with suction or magnetic grippers allowing for larger deviations than applications like spot welding, for instance. Such deviations introduce additional freedom to the optimization problem, which can be exploited, e.g., to improve the time-optimality of a task, to reduce the energy consumed by a machine while performing a task, or to reduce the likelihood of damage to a robot while working near its joint or actuation limits.

The tunnel-following NMPC scheme, introduced by van Duijkeren [6], allows us to include such deviations in nonconvex optimal control problems (OCP) in the form of an upper-bounded squared ℓ-2 norm of a nonlinear measure of deviation, e.g., a Euclidean distance from the path. This special *convex-over-nonlinear* structure can be exploited to solve such OCPs with variations of the Sequential Quadratic Programming (SQP) method such as the Generalized Gauss-Newton (GGN) method [7], [8] and the Sequential Convex Quadratic Programming (SCQP) method [9].

The GGN method replaces the exact Hessian (EH) of the Quadratic Programming (QP) subproblems in the SQP method with an approximation based on first-order derivatives of the objective function and second-order derivatives of some convex functions present also in the objective. Conversely, the SCQP method exploits the convexity in both the objective and the constraints to create a particular Hessian approximation that leads to better convergence properties when compared to the GGN method [9], [10]. The implementation of either the GGN or the SCQP method involves an additional modeling and implementation effort with respect to the SQP method. Such implementation requires the modeler, who translates the OCP into a nonlinear program (NLP), to identify and select the *convex-over-nonlinear* functions in the objective and constraints of the OCP, besides their corresponding Lagrangian multipliers, to build the Hessian approximation. This leaves opportunities for improvement in terms of (i) reducing the effort needed to implement the GGN and SCQP methods, and (ii) defining measures of deviation for additional variables describing robot motions, such as orientations.

The contribution of this paper is twofold. First, it extends the

Manuscript received: September 9, 2021; Revised November 27, 2021; Accepted January, 6, 2022. This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments. This work benefits from the FWO project G0A6917N of the Research Foundation - Flanders (FWO - Flanders) and Flanders Make SBO MULTIROB: "Rigorous approach for programming and optimal control of multi-robot systems". Flanders Make is the Flemish strategic research centre for the manufacturing industry. *(Corresponding author: Alejandro Astudillo).*

[1]Alejandro Astudillo, Joris Gillis, Wilm Decré, Goele Pipeleers and Jan Swevers are with the MECO Research Team, Department of Mechanical Engineering, KU Leuven, 3001 Leuven, Belgium, and also with the DMMS lab, Flanders Make, 3001 Leuven, Belgium {alejandro.astudillovigoya, joris.gillis, wilm.decre, goele.pipeleers, jan.swevers}@kuleuven.be.

[2]Moritz Diehl is with the Department of Microsystems Engineering (IMTEK) and the Department of Mathematics, University of Freiburg, 79110 Freiburg, Germany moritz.diehl@imtek.de.

Digital Object Identifier (DOI): see top of this page.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2022.3142396

tunnel-following scheme presented by van Duijkeren [6] to include freedom in the orientation of the end-effector of a robot manipulator, not only penalizing the orientation error in the objective but also adding a *convex-over-nonlinear* constraint to define a feasible region of the orientation error, while giving a meaningful representation to the concept of an orientation tunnel. Second, it explores an efficient implementation of the SCQP method, and consequently of the GGN method, by exploiting the idea of symbolic linearization of the nonlinear part of the *convex-over-nonlinear* functions in the objective and constraints of OCPs, so that the modeling effort required to implement the SCQP or GGN method is reduced. The exploitation of the convexity in the tunnel-following scheme by the SCQP method is preserved with the addition of the orientation tunnel.

Note that, even though the tunnel-following scheme is applied to robot manipulators in this paper, it can also be applied to other types of systems such as mobile robots and autonomous vehicles, for example to exploit the entire width of a lane to generate optimal motions.

A. Notation

Let us denote $\mathbf{0}$ as a zero matrix, $\langle z_1, \cdots, z_n \rangle := \begin{bmatrix} z_1^\top & \cdots & z_n^\top \end{bmatrix}^\top$ as the vertical concatenation of two or more column vectors z_i, $i \subset \{1, ..., n\}$, and $||z||_{\mathbf{W}} := \sqrt{z^\top \mathbf{W} z}$ as the weighted ℓ-2 norm of vector z. Throughout this paper we also denote $\frac{\partial f}{\partial w}(w) \in \mathbb{R}^{m \times n}$ as the Jacobian of a continuously differentiable function $f(w) : \mathbb{R}^n \to \mathbb{R}^m$, and $\nabla f(w) := \frac{\partial f}{\partial w}(w)^\top \in \mathbb{R}^{n \times m}$ as its gradient. If $f(w)$ is a scalar function, i.e., $m = 1$, the Hessian of $f(w)$ is defined as $\nabla^2 f(w) := \frac{\partial^2 f}{\partial w^2}(w) \in \mathbb{R}^{n \times n}$. A subscript in the gradient or Hessian operator indicates that the differentiation is performed with respect to such subscript, e.g., $\nabla_{w_i} f(w) := \frac{\partial f}{\partial w_i}(w)$.

B. Outline

This paper is organized as follows. The tunnel-following scheme and its extension are presented in Section II. Section III discusses the SCQP method. Next, symbolic linearization and its application in the GGN and SCQP methods are presented in Section IV. Experiments and results are discussed in Section V. We close the paper with concluding remarks.

II. TUNNEL-FOLLOWING NMPC SCHEME

The path-following scheme aims to make the output of a system follow a path reference, penalizing any deviation from this reference, i.e., penalizing tracking-errors. A natural extension of this scheme arising in applications with robot manipulators is the position tunnel-following NMPC scheme, presented in [6]. Here, the end-effector of a manipulator is allowed to deviate from a position reference up to a user-defined tolerance $\rho_\bullet \in \mathbb{R}_{>0}$, see Fig. 1, while any excursion beyond $\rho_\bullet$ is heavily penalized in the objective of an optimal control problem (OCP). This section introduces an extended tunnel-following scheme with orientation constraints and presents the OCP associated to it.

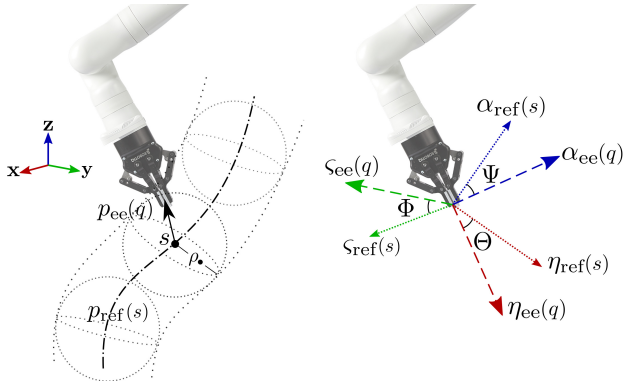

Fig. 1. Illustration of the $\rho_\bullet$-neighborhood around a reference path $p_{\mathrm{ref}}(s)$ that defines a position-tunnel when evaluated along the path-progress variable s (left) and the error angles $\{\Theta, \Phi, \Psi\}$ that define the magnitude of the error between an orientation reference $R_{\mathrm{ref}}(s) = [\eta_{\mathrm{ref}}(s)\ \varsigma_{\mathrm{ref}}(s)\ \alpha_{\mathrm{ref}}(s)]$ and the orientation of the end-effector $R_{\mathrm{ee}}(q) = [\eta_{\mathrm{ee}}(q)\ \varsigma_{\mathrm{ee}}(q)\ \alpha_{\mathrm{ee}}(q)]$ (right).

A. System dynamics

For a robot manipulator with n_{dof} degrees-of-freedom (dof), let us define the state vector $x := \langle q, \dot{q} \rangle \in \mathbb{R}^{2n_{\mathrm{dof}}}$, and the input vector $u := \tau \in \mathbb{R}^{n_{\mathrm{dof}}}$, where $q \in \mathbb{R}^{n_{\mathrm{dof}}}$, $\dot{q} \in \mathbb{R}^{n_{\mathrm{dof}}}$ and $\tau \in \mathbb{R}^{n_{\mathrm{dof}}}$ are the generalized joint position, velocity and torque, respectively. The robot dynamics can be defined by the ordinary differential equation $\dot{x} = \xi(x, u) = \langle \dot{q}, FD(q, \dot{q}, \tau) \rangle$, where FD is the forward dynamics function of the robot manipulator.

Following the approach presented in [6], we introduce the path-progress variable $s \in \mathcal{T} := [0, 1]$ subject to double integrator dynamics $\xi_s(\chi, \nu) := \langle \dot{s}, \ddot{s} \rangle$ with state vector $\chi := \langle s, \dot{s} \rangle \in \mathbb{R}^2$ and input $\nu := \ddot{s} \in \mathbb{R}$.

The path dynamics ξ_s are used to augment the system dynamics as $\hat{\xi}(\hat{x}, \hat{u}) := \langle \xi(x, u), \xi_s(\chi, \nu) \rangle$, where $\hat{x} := \langle x, \chi \rangle \in \mathbb{R}^{n_{\hat{x}}}$, $\hat{u} := \langle u, \nu \rangle \in \mathbb{R}^{n_{\hat{u}}}$, $n_{\hat{x}} = 2n_{\mathrm{dof}} + 2$ and $n_{\hat{u}} = n_{\mathrm{dof}} + 1$. Such augmented dynamics $\hat{\xi}$ are discretized by means of a Runge-Kutta integrator with sample time δ_t to obtain the discrete representation

$$\hat{x}_{k+1} = f_d(\hat{x}_k, \hat{u}_k). \tag{1}$$

B. Tracking Errors

Tracking errors are agnostic to the scheme used (path-following or tunnel-following). These errors may include time-tracking error, position-tracking error, and orientation-tracking error, among others, and depend on the definition of a path reference according to the following two assumptions.

Assumption 1 (Path reference)**.** *A path reference is given as input to the path- or tunnel-following scheme. This reference is composed by a path-progress-dependent reference $\dot{s}_{\mathrm{ref}}(s) : \mathcal{T} \to \mathbb{R}$, a position reference $p_{\mathrm{ref}}(s) : \mathcal{T} \to \mathbb{R}^3$ and an orientation reference $R_{\mathrm{ref}}(s) : \mathcal{T} \to SO(3)$. The path reference is assumed to be computed by an optimal motion planner, such as [11] or [12].*

Assumption 2 (Small orientation error)**.** *The reference $\{\dot{s}_{\mathrm{ref}}, p_{\mathrm{ref}}, R_{\mathrm{ref}}\}$ is computed by an optimal motion planner, as in Assumption 1, so that $|\Theta| \ll \pi/2$, $|\Phi| \ll \pi/2$, $|\Psi| \ll \pi/2$*

$\forall s \in \mathcal{T}$, *i.e., the end-effector is able to follow* R_{ref} *with small orientation errors, which are represented by the error angles* $\{\Theta, \Phi, \Psi\}$ *(see Fig. 1).*

Furthermore, the definition of the tracking errors requires functions that describe the position $p_{\text{ee}}(q) : \mathbb{R}^{n_{\text{dof}}} \to \mathbb{R}^3$ and the orientation $R_{\text{ee}}(q) : \mathbb{R}^{n_{\text{dof}}} \to SO(3)$ of the end-effector. Such functions are retrieved from the forward kinematics function of the robot.

1) Time-tracking error: Based on Assumption 1, let us define the time-tracking error as

$$e_{\dot{s}}(\hat{x}) = \dot{s} - \dot{s}_{\text{ref}}(s) \in \mathbb{R}. \tag{2}$$

The norm of $e_{\dot{s}}$ is heavily penalized in the objective function of the OCP with a penalty $\mathbf{w}_{\dot{s}}$ to make it converge to zero with high priority.

2) Position-tracking error: Similarly, we can define the position-tracking error as

$$e_{\mathcal{P}}(\hat{x}) = p_{\text{ee}}(q) - p_{\text{ref}}(s) \in \mathbb{R}^3, \tag{3}$$

whose ℓ-2 norm represents the Euclidean distance between p_{ee} and p_{ref}. In the tunnel-following scheme, the squared ℓ-2 norm of $e_{\mathcal{P}}$ is included with a small penalty $\mathbf{w}_{\mathcal{P}}$ as a regularization term in the objective function and is added within a soft-constraint to the OCP. Such constraint allows the squared ℓ-2 norm[1] of $e_{\mathcal{P}}$ to be greater than zero but lower than certain user-defined position tolerance $\rho_{\mathcal{P}} \in \mathbb{R}_{\geq 0}$ squared. By setting such upper-bound to $||e_{\mathcal{P}}(\hat{x})||^2$, the end-effector is allowed to stay within a sphere of radius $\rho_{\mathcal{P}}$ and centered at $p_{\text{ref}}(s)$. The tunnel around $p_{\text{ref}}(s)$ is formed from the union of all such spheres along s, as shown in Fig. 1.

Definition 1 (Position-tunnel constraints)**.** *Consider a slack variable* $l_{\mathcal{P}} \in \mathbb{R}_{\geq 0}$ *that is heavily penalized in the objective function as a linear cost with weight* $\mathbf{w}_l$*. The position-tunnel is defined by the following constraint*

$$||e_{\mathcal{P}}(\hat{x})||^2 \leq \rho_{\mathcal{P}}^2 + l_{\mathcal{P}}. \tag{4}$$

This slack variable is introduced to guarantee feasibility of the solution of the OCP, even when it is not feasible to remain inside the tunnel, and its penalization enforces convergence of $p_{\text{ee}}(q)$ to the tunnel of radius $\rho_{\mathcal{P}}$.

Even though the presence of nonconvex functions in the objective or constraints of an OCP is usually undesirable, the position-tunnel constraint (4) has a special *convex-over-nonlinear* structure that allows to exploit convexity of its outermost part by using the SCQP method, as detailed in Section III.

Definition 2 (Convex-over-nonlinear function)**.** *The composition of a convex function* $\phi(w)$ *and a nonlinear function* $c(w)$*, so that* $\psi(w) \coloneqq (\phi \circ c)(w) = \phi(c(w))$ *is known as a convex-over-nonlinear function.*

Based on this definition, the left-hand side of the position-tunnel constraint (4) is a *convex-over-nonlinear* function where $\phi(c) = c^2$ and $c(\hat{x}) = e_{\mathcal{P}}(\hat{x})$.

[1]The squared ℓ-2 norm is used instead of the ℓ-2 norm to avoid the computation of square roots and to add curvature to the structure of the functions in the objective and constraints, which is exploited by the solution method presented in Section III.

3) Orientation-tracking error: The computation of the orientation-tracking error is not as straightforward as the position-tracking error, since it depends on the way orientations are being represented, e.g., axis-angle representation, rotation matrices, unit quaternions, or Euler angles. Although a rotation matrix is not a minimal representation of orientation, in this paper we use rotation matrices as they have no singularities and are bijective.

The orientation reference is described by the rotation matrix $R_{\text{ref}}(s) = [\eta_{\text{ref}}(s)\ \varsigma_{\text{ref}}(s)\ \alpha_{\text{ref}}(s)]$, where η_{ref}, ς_{ref}, and α_{ref} are the first, second and third column of R_{ref}, respectively. Similarly, the orientation of the end-effector is given by $R_{\text{ee}}(q) = [\eta_{\text{ee}}(q)\ \varsigma_{\text{ee}}(q)\ \alpha_{\text{ee}}(q)]$. Let us now define the rotation needed to align $R_{\text{ref}}(s)$ and $R_{\text{ee}}(q)$ as

$$R_{\mathcal{O}}(q, s) = R_{\text{ref}}(s) R_{\text{ee}}^{\top}(q). \tag{5}$$

Following the approach shown in [13, Section 3.7], and based on Assumption 2, $R_{\mathcal{O}}(q, s)$ can be transformed into an axis-angle representation, with vector $\mathbf{r}$ representing the direction of the axis of rotation and angle ϑ representing the magnitude of the rotation around $\mathbf{r}$, without losing bijection since $|\vartheta| < \pi/2$ (Assumption 2). Based on such representation, the orientation error can be defined as

$$e_{\mathcal{O}} = \mathbf{r} \sin(\vartheta) \in \mathbb{R}^3. \tag{6}$$

The axis of rotation $\mathbf{r}$ is defined by means of the off-diagonal elements of $R_{\mathcal{O}}$ as

$$\mathbf{r} = \frac{1}{2\sin(\vartheta)} \langle r_{\mathcal{O}_{32}} - r_{\mathcal{O}_{23}}, r_{\mathcal{O}_{13}} - r_{\mathcal{O}_{31}}, r_{\mathcal{O}_{21}} - r_{\mathcal{O}_{12}} \rangle, \tag{7}$$

where $r_{\mathcal{O}_{ij}}$ corresponds to the element of the i-th row and j-th column of $R_{\mathcal{O}}$. Replacing (7) in (6), we get an expression for $e_{\mathcal{O}}$ that depends on the columns of the rotation matrices R_{ref} and R_{ee} as follows

$$e_{\mathcal{O}}(\hat{x}) = \frac{1}{2} \langle r_{\mathcal{O}_{32}} - r_{\mathcal{O}_{23}}, r_{\mathcal{O}_{13}} - r_{\mathcal{O}_{31}}, r_{\mathcal{O}_{21}} - r_{\mathcal{O}_{12}} \rangle \tag{8}$$

$$= \frac{1}{2} \left(\eta_{\text{ee}} \times \eta_{\text{ref}} + \varsigma_{\text{ee}} \times \varsigma_{\text{ref}} + \alpha_{\text{ee}} \times \alpha_{\text{ref}} \right). \tag{9}$$

where the dependencies on q and s have been dropped to increase readability. Note that the restriction of $|\vartheta| < \pi/2$ can be defined by means of the columns of the rotation matrices as $\eta_{\text{ee}}^{\top} \eta_{\text{ref}} > 0$, $\varsigma_{\text{ee}}^{\top} \varsigma_{\text{ref}} > 0$, $\alpha_{\text{ee}}^{\top} \alpha_{\text{ref}} > 0$ [13], or by means of the error angles $\{\Theta, \Phi, \Psi\}$ between the pairs $(\eta_{\text{ee}}, \eta_{\text{ref}})$, $(\varsigma_{\text{ee}}, \varsigma_{\text{ref}})$, and $(\alpha_{\text{ee}}, \alpha_{\text{ref}})$, respectively, as $|\Theta| < \pi/2$, $|\Phi| < \pi/2$ and $|\Psi| < \pi/2$ (see Fig. 1).

Remark 1. *The representation of the orientation error* (9) *is not unique, and can depend on other representations of orientation, e.g., angle-axis or unit quaternions.*

Let us now address the existence of an orientation tunnel of radius $\rho_{\mathcal{O}}$ and the definition of the orientation-tunnel constraint. Given the orthonormality property of rotation matrices and the definition of the cross product, the squared ℓ-2 norm of $e_{\mathcal{O}}$ can be computed as

$$||e_{\mathcal{O}}(\hat{x})||^2 = \frac{1}{4} \left(\sin^2(\Theta) + \sin^2(\Phi) + \sin^2(\Psi) \right). \tag{10}$$

Based on Assumption 2 and the small angle approximation $\sin(w) \approx w$, $||e_{\mathcal{O}}(\hat{x})||^2$ can be approximated as

$$||e_{\mathcal{O}}(\hat{x})||^2 \approx \frac{1}{4}\left(\Theta^2 + \Phi^2 + \Psi^2\right) = \left\| \frac{1}{2}[\Theta \quad \Phi \quad \Psi]^\top \right\|^2 \tag{11}$$

which resembles the computation of a Euclidean distance in the three-dimensional space defined by Θ, Φ and Ψ. By imposing an upper bound $\rho_{\mathcal{O}}^2$ to $||e_{\mathcal{O}}(\hat{x})||^2$, a feasible sphere of radius $\rho_{\mathcal{O}}$ is defined in such three-dimensional space. This creates a tunnel when considering the progress of s.

Definition 3 (Orientation-tunnel constraints)**.** *Analogous to constraint* (4)*, consider a user-defined orientation tolerance $\rho_{\mathcal{O}} \in \mathbb{R}_{\geq 0}$ and a slack variable $l_{\mathcal{O}} \in \mathbb{R}_{\geq 0}$ that is heavily penalized as a linear cost with weight $\mathbf{w}_l$. The constraint that defines the orientation-tunnel is*

$$||e_{\mathcal{O}}(\hat{x})||^2 \leq \rho_{\mathcal{O}}^2 + l_{\mathcal{O}}. \tag{12}$$

Similar to the position-tracking error, the squared ℓ-2 norm of $e_{\mathcal{O}}$ is included as a regularization term in the objective function of the OCP with a small penalty $\mathbf{w}_{\mathcal{O}}$. Please note that the orientation-tunnel constraint (12) has the same *convex-over-nonlinear* structure as the position-tunnel constraint (4), with $\phi(c) = c^2$ and $c(\hat{x}) = e_{\mathcal{O}}(\hat{x})$.

C. Optimal Control Problem

Having discussed the system dynamics and tracking errors, it is now necessary to discuss the underlying OCP that is solved within the tunnel-following NMPC scheme. Such OCP has the following structure

$$\begin{aligned}
\min_{w} \quad & \hat{\mathcal{V}}(w) && && \text{(13a)}\\
\text{s.t.} \quad & \hat{x}_0 - \mathbf{x}_0 = \mathbf{0}, && && \text{(13b)}\\
& \hat{x}_{k+1} - f_d(\hat{x}_k, \hat{u}_k) = \mathbf{0} && k = 0, ..., N-1, && \text{(13c)}\\
& g_k(\hat{x}_k, \hat{u}_k) \leq \mathbf{0} && k = 0, ..., N-1, && \text{(13d)}\\
& \|\zeta_\bullet(\hat{x}_k)\|^2 \leq \rho_\bullet^2 + \hat{l}_{\bullet,k} && k = 0, ..., N, && \text{(13e)}\\
& g_N(\hat{x}_N) \leq \mathbf{0} && && \text{(13f)}
\end{aligned}$$

where $w = \left\langle \hat{x}_0, \hat{u}_0, \hat{l}_0, \ldots, \hat{u}_{N-1}, \hat{l}_{N-1}, \hat{x}_N \right\rangle$ is the vector of decision variables, $N \in \mathbb{Z}_{>0}$ is the prediction horizon, $\hat{l}_k = \langle l_{\mathcal{P},k}, l_{\mathcal{O},k} \rangle \in \mathbb{R}^2$ is the vector of slack variables, $\mathbf{x}_0 \in \mathbb{R}^{n_{\hat{x}}}$ is the estimate of the current state of the system,

$$\hat{\mathcal{V}}(w) = \|\mathcal{V}_N(\hat{x}_N)\|_{\mathbf{P}}^2 + \sum_{k=0}^{N-1} \left[\|\mathcal{V}(\hat{x}_k, \hat{u}_k)\|_{\mathbf{Q}}^2 + \mathbf{w}_l \hat{l}_k \right] \tag{14}$$

is the objective function, while $\mathbf{P} \succeq 0$ and $\mathbf{Q} \succeq 0$ are weighting matrices of appropriate dimensions. General inequality constraints are defined in g_k and g_N. We are specially interested in the structure of the objective function (14) and the constraints (13e). The nonlinear functions $\mathcal{V}_N$ and $\mathcal{V}$ in the objective define least-squares costs and are subjected to a squared, weighted ℓ-2 norm. Furthermore, constraint (13e) sets an upper-bound to the squared ℓ-2 norm of the nonlinear function $\zeta_\bullet$, which defines a tracking error.

Remark 2. *The OCP* (13) *could contain multiple constraints of the type* (13e)*.*

The specific functions used within the OCP (13) are described below. Functions $\mathcal{V}(\hat{x}_k, \hat{u}_k)$ and $\mathcal{V}_N(\hat{x}_N)$ in the objective function (14) are defined as follows

$$\mathcal{V}(\hat{x}_k, \hat{u}_k) = \langle e_{\dot{s}}(\hat{x}_k), e_{\mathcal{P}}(\hat{x}_k), e_{\mathcal{O}}(\hat{x}_k), \hat{x}_k, \hat{u}_k \rangle, \tag{15}$$

$$\mathcal{V}_N(\hat{x}_N) = \langle e_{\mathcal{P}}(\hat{x}_N), e_{\mathcal{O}}(\hat{x}_N), s_N - 1, \hat{x}_N \rangle, \tag{16}$$

where $\langle \hat{x}_k, \hat{u}_k \rangle$ and $\hat{x}_N$ are included in $\mathcal{V}$ and $\mathcal{V}_N$, respectively, as regularization terms with small penalties. The term $s_N - 1$ is included in $\mathcal{V}_N$ to attract the state s towards $s_{\max} = 1$. Equality constraints (13c) correspond to the evaluation of the dynamics (1) along the horizon according to the multiple-shooting approach [14]. Inequality constraints, such as boundaries of states, inputs and slack variables, are shown below.

$$\begin{aligned}
& q_{\min} \leq q_k \leq q_{\max} && \\
& 0 \leq s_k \leq 1 && q_{\min} \leq q_N \leq q_{\max} \\
& \tau_{\min} \leq \tau_k \leq \tau_{\max} && \mathbf{0} \leq \dot{q}_N \leq \mathbf{0} \\
& -\dot{s}_k \leq 0 && \mathbf{0} \leq e_{\mathcal{P}}(\hat{x}_N) \leq \mathbf{0} \\
& -\hat{l}_k \leq \mathbf{0} && \mathbf{0} \leq e_{\mathcal{O}}(\hat{x}_N) \leq \mathbf{0} \\
& \underbrace{\qquad\qquad}_{g_k(\hat{x}_k, \hat{u}_k) \leq \mathbf{0}} && \underbrace{\qquad\qquad}_{g_N(\hat{x}_N) \leq \mathbf{0}}
\end{aligned} \tag{17}$$

These inequality constraints are included in (13d) and (13f). Please note that terminal equality constraints on $\langle \dot{q}_N, e_{\mathcal{P}}(\hat{x}_N), e_{\mathcal{O}}(\hat{x}_N) \rangle$ can be set as terminal inequality constraints with $\mathbf{0}$ as both lower and upper bound. Finally, the position-tunnel constraint (4) and the orientation-tunnel constraint (12) are added as two instances of constraint (13e).

III. SEQUENTIAL CONVEX QUADRATIC PROGRAMMING (SCQP)

So far this paper has focused on extending the tunnel-following NMPC scheme by adding orientation-tunnel constraints. We now move on to discuss the sequential convex quadratic programming (SCQP) method [9], an algorithm that exploits convexity in the objective and constraints of an OCP to improve the optimality of the solution, compared to the generalized Gauss-Newton (GGN) method, while reducing the computational complexity of the solution process, compared to the sequential quadratic programming (SQP) method.

The definition of *convex-over-nonlinear* functions, i.e., Definition 2, allows us to introduce a general version of the OCP (13), including *convex-over-nonlinear* functions in its objective and constraints, as the following NLP

$$\begin{aligned}
\min_{w} \quad & \psi_0(w) && && \text{(18a)}\\
\text{s.t.} \quad & h_i(w) = 0, && i = 1, ..., n_h, && \text{(18b)}\\
& \psi_i(w) \leq 0, && i = 1, ..., n_g. && \text{(18c)}
\end{aligned}$$

where (18b) includes the equality constraints (13b–13c), while (18c) includes the inequality constraints (13d–13f).

Remark 3. *The outer convex function in a convex-over-nonlinear function can be an identity or linear mapping.*

A. Hessian approximations

The NLP (18) can be solved by means of the SQP method, where the QP subproblem

$$\min_{d_k} \quad \frac{1}{2} d_k^\top B_k^{\mathrm{SQP}} d_k + \nabla \psi_0(w_k)^\top d_k \tag{19a}$$

$$\text{s.t.} \quad \nabla h_i(w_k)^\top d_k + h_i(w_k) = 0, \quad i = 1, ..., n_h, \tag{19b}$$

$$\nabla \psi_i(w_k)^\top d_k + \psi_i(w_k) \leq 0, \quad i = 1, ..., n_g, \tag{19c}$$

is solved to obtain a step d_k so that w_k is iterated as $w_{k+1} = w_k + d_k$ towards the local minimizer w^*. In the SQP method, the Hessian matrix B_k^{SQP} in (19) is defined as

$$B_k^{\mathrm{SQP}} = B_k^{\mathrm{EH}} \coloneqq \nabla^2 \mathcal{L}(w_k, \lambda_k, \mu_k). \tag{20}$$

which is the EH of the Lagrangian of (18) $\mathcal{L}(w, \lambda, \mu) := \psi_0(w) + \lambda^\top h(w) + \mu^\top \psi(w)$, where λ and μ are the Lagrange multipliers of the equality constraints h_i and the inequality constraints ψ_i, respectively. The EH is not guaranteed to be positive semidefinite and convergence of the SQP method cannot be guaranteed without B_k^{SQP} being positive semidefinite. Moreover, the evaluation of B_k^{EH} is computationally expensive, as it considers the information of all (non)linear functions in the NLP.

To overcome these drawbacks, the structure of $\psi_0(w) = \phi_0(c_0(w))$ can be exploited to approximate B_k^{EH} using the GGN method [8]. This approximation ignores all contributions of the constraints and the second-order derivatives of $c_0(w)$, and is defined as

$$B_k^{\mathrm{GGN}}(w) \coloneqq \frac{\partial c_0(w)}{\partial w}^\top \nabla_{c_0}^2 \phi_0(c_0(w)) \frac{\partial c_0(w)}{\partial w}, \tag{21}$$

which is positive semidefinite ($B_k^{\mathrm{GGN}}(w) \succeq 0$). This leads to convex QP subproblems but neglects curvature information from the constraints, which may cause instability in the SQP iterations due to large steps d_k.

Let us now turn to the SCQP method [9]. This method is a generalization of the GGN method where the second-order derivatives of the convex outer part ϕ_i in ψ_i (18c) are considered within the Hessian approximation, so that there is no neglection of the curvature information provided by such functions to the Hessian B_k^{SQP}. Thus, the SCQP Hessian approximation is defined as

$$B_k^{\mathrm{SCQP}}(w, \mu) \coloneqq \frac{\partial c_0}{\partial w}(w)^\top \nabla_{c_0}^2 \phi_0(c_0(w)) \frac{\partial c_0}{\partial w}(w) + \sum_{i=1}^{n_g} \mu_i \frac{\partial c_i}{\partial w}(w)^\top \nabla_{c_i}^2 \phi_i(c_i(w)) \frac{\partial c_i}{\partial w}(w). \tag{22}$$

Comparing (22) with (21), and since the outer functions ϕ_i are convex and the optimality condition of dual feasibility is satisfied ($\mu_i \geq 0$), it holds that $B_k^{\mathrm{SCQP}}(w, \mu) \succeq B_k^{\mathrm{GGN}}(w) \succeq 0$, which leads to the fact that $B_k^{\mathrm{SCQP}}(w, \mu)$ has a smaller approximation error than $B_k^{\mathrm{GGN}}(w)$ as proven in [9].

Even though B_k^{SCQP} is more expensive to compute than B_k^{GGN}, the additional complexity is almost negligible when SCQP is applied in the context of NMPC of highly nonlinear systems such as robot manipulators, as shown in Section V.

IV. SYMBOLIC LINEARIZATION

Some numerical optimization frameworks, such as CasADi [15], solve optimization problems in three steps: (i) construct a symbolic representation of the problem, (ii) instantiate a solver object, and (iii) evaluate such solver object. During the first step, functions in the objective and constraints are defined by means of symbolic expressions which depend on symbolic primitives. Such expressions are automatically differentiated by the framework, where needed, by using algorithmic differentiation (AD).

To solve (18) using the SCQP method with a numerical optimization framework, the modeler must explicitly define B_k^{SQP} in (19) to be equal to the symbolic expression (22) during the first step. This means that the modeler must (i) find the index i of the Lagrange multiplier μ_i of the corresponding *convex-over-nonlinear* constraint ψ_i, (ii) compute the Jacobians of c_i and the Hessians of ϕ_i, possibly with AD, (iii) build the symbolic expression of B_k^{SCQP} as in (22), and (iv) replace the EH of the SQP method with B_k^{SCQP}. These are exactly the kind of steps that a robotics engineer would hope to find abstracted away in a modeling framework, since such modeling effort increases the engineering time needed to prepare and solve an OCP. Changes to such a framework or cluttering of application code with hacks appear inevitable. They are not.

To reduce the modeling effort, we propose the use of symbolic linearization, i.e., linearization of symbolic expressions around a symbolic equilibrium point, by means of a novel operator to automatically generate the SCQP or GGN Hessian approximations within the SQP method.

Let us define the lin operator, which applies symbolic linearization to a symbolic expression $f(w_1, \cdots, w_{n_p}) : \mathbb{R}^{n_p} \to \mathbb{R}^{n_f}$ with symbolic primitives $w = \left(w_1, \cdots, w_{n_p}\right)$. We begin with the definition of a first-order Taylor expansion of a expression $f(\hat{w})$ around an equilibrium point $\overline{w}$

$$f_{\mathrm{lin}}(\hat{w}, \overline{w}) \coloneqq f(\overline{w}) + \frac{\partial f}{\partial \hat{w}}(\overline{w})(\hat{w} - \overline{w}). \tag{23}$$

Definition 4 (lin operator)**.** *Given a symbolic expression $f(w_1, \cdots, w_{n_p})$, the lin operator returns the expression $\tilde{f}(w) : \mathbb{R}^{n_p} \to \mathbb{R}^{n_f}$, whose first- and second-order derivatives correspond to the Jacobian of $f(w)$ and a zero matrix, respectively, as described below.*

$$\mathrm{lin}(f(w)) \coloneqq \begin{cases} \tilde{f}(w) \coloneqq f_{\mathrm{lin}}(w, w) = f(w) \\ \frac{\partial \tilde{f}}{\partial w}(w) = \frac{\partial f_{\mathrm{lin}}}{\partial \hat{w}}(w, w) = \frac{\partial f}{\partial w}(w) \\ \frac{\partial^2 \tilde{f}}{\partial w^2}(w) = \frac{\partial^2 f_{\mathrm{lin}}}{\partial \hat{w}^2}(w, w) = \mathbf{0} \end{cases} \tag{24}$$

Algorithm 1 describes a software implementation of the lin operator using existing CasADi constructs[2]. Here, (23) is applied using forward mode AD to compute the forward sensitivities of $f(w)$ as a Jacobian-times-vector product. The crucial programming trick in the lin operator is that the internal variable $\overline{w}$ is treated as a constant parameter when the expression $\tilde{f}(w)$ is called in CasADi. Thus, the differentiation of f_{lin} with respect to $\overline{w}$ is prohibited, allowing a correct computation of $\partial \tilde{f} / \partial w$ and $\partial^2 \tilde{f} / \partial w^2$.

[2]The source code of the lin operator is available at https://git.io/Ju0T9

Algorithm 1 - Software implementation of lin operator

Input: Symbolic expression $f(w_1, \cdots, w_{n_p})$

Output: Symbolic expression $\tilde{f}(w_1, \cdots, w_{n_p})$ with first and second-order derivatives as in (24)

1: $w \leftarrow$ Retrieve and concatenate all symbolic primitives.
2: $\overline{w} \leftarrow$ Instantiate new symbolic primitive with the same shape as w.
3: $Df(w, \hat{w}) \leftarrow$ Apply one sweep of forward mode AD to compute the forward sensitivities of $f(w)$ with seed $\hat{w}$.
4: $f_{\mathrm{lin}}(w, \overline{w}) \leftarrow f(\overline{w}) + Df(\overline{w}, w - \overline{w})$ as in (23).
5: $\tilde{f}(w_1, \cdots, w_{n_p}) \leftarrow$ Define function $\tilde{f}(w) := f_{\mathrm{lin}}(w, w)$ prohibiting the differentiation of f_{lin} with respect to the symbolic equilibrium point $\overline{w}$.

Having discussed symbolic linearization and the lin operator, we now show an alternative view of both the SCQP and the GGN method based on the symbolic linearization of elements in the objective and constraints of an NLP.

Definition 5 (Alternative NLP for SCQP)**.** *Applying the* lin *operator to (i) the nonlinear part of convex-over-nonlinear functions* $\psi_0(w) = \phi_0(c_0(w))$ *and* $\psi_i(w) = \phi_i(c_i(w))$ *in the objective and the inequality constraints and (ii) the equality constraints* $h_i(w)$ *in* (18)*, we define the following NLP*

$$\min_{w} \quad \phi_0(\mathrm{lin}(c_0(w))) \tag{25a}$$
$$\text{s.t.} \quad \mathrm{lin}(h_i(w)) = 0, \qquad i = 1, ..., n_h, \tag{25b}$$
$$\phi_i(\mathrm{lin}(c_i(w))) \leq 0, \quad i = 1, ..., n_g. \tag{25c}$$

This definition presents an alternative NLP whose exact Hessian of the Lagrangian is equal to B_k^{SCQP}. This can be proven relying on the definition of the lin operator. Such proof is straightforward and therefore omitted for the sake of simplicity.

Based on the structure of the QP (19) and its dependency on up to first-order derivatives, except for B_k^{SQP}, the equality between the Hessian of the Lagrangian of (25) and the SCQP Hessian approximation B_k^{SCQP} (22) proves the following proposition.

Proposition 1. *The application of the SQP method to NLP* (25) *is equivalent to the application of the SQP method to NLP* (18) *using the SCQP Hessian approximation* B_k^{SCQP} (22)*, i.e., the SCQP method.*

The proposition 1 implies that the use of the lin operator as in NLP (25) allows an implementation of the SCQP method where the modeling effort is reduced due to the abstraction of the complex steps in the construction of a symbolic representation of the problem.

Similarly, the GGN method can be used to solve NLP (18) by applying the SQP method to the following NLP.

Definition 6 (Alternative NLP for GGN)**.** *Unlike* (25)*, the implementation of the GGN method applies the* lin *operator to all the components of the constraints, including the convex part* ϕ_i *in* (26c)*, leading to the alternative NLP*

$$\min_{w} \quad \phi_0(\mathrm{lin}(c_0(w))) \tag{26a}$$
$$\text{s.t.} \quad \mathrm{lin}(g_i(w)) = 0, \qquad i = 1, ..., n_h, \tag{26b}$$
$$\mathrm{lin}(\phi_i(c_i(w))) \leq 0, \quad i = 1, ..., n_g. \tag{26c}$$

It is straightforward to prove that applying the SQP method to (26) is equivalent to applying the SQP method to (18) using the GGN Hessian approximation B_k^{GGN} (21), i.e., the GGN method.

V. RESULTS AND DISCUSSION

To evaluate the extended tunnel-following scheme implemented with the alternative NLP (25) for SCQP, we consider a 7-dof *Kinova Gen3* robot following a lemniscate-shaped path, see Fig. 2. The lin operator is applied to (i) the inner nonlinear elements in (14), (ii) the equality constraints (13b–13c), (iii) the inequality constraints (13d) and (13f), and (iv) the inner nonlinear part of constraints (13e), i.e., $e_{\mathcal{P}}(\hat{x}_k)$ and $e_{\mathcal{O}}(\hat{x}_k)$. The solution of OCP (13) (solved with SCQP, GGN and EH) employs the real-time iteration (RTI) scheme [16], with one QP subproblem being solved at every MPC iteration by using the QP solver QRQP [17], and is evaluated against an interior point (IP) method implemented in the widely used nonlinear optimization solver IPOPT [18]. Additionally, the proposed extended tunnel-following scheme is compared against the original position tunnel-following scheme presented by van Duijkeren [6], also solved with SCQP, to verify the implications of adding the orientation-tunnel constraint (12).

The tests were executed on a laptop with an Intel Core i7-8850H CPU running Ubuntu 18.04. All functions needed to solve the OCP (13), including the SQP solver and the Hessian approximations, were defined and code-generated using CasADi [15], and compiled using GCC 9.1.0 with compilation flags `-O3` and `-march=native`. The IP solver IPOPT does not allow code-generation.

The expressions for forward dynamics and kinematics of the robot, besides the definition of $q_{\min}$, $q_{\max}$, $\tau_{\min}$, and $\tau_{\max}$, were generated using the rigid-body dynamics library Pinocchio [19] and the interface presented in [20].

The parameters used to set the OCP (13) are shown in Table I. All other weights in $\mathbf{P}$ and $\mathbf{Q}$ are set to 10^{-3}.

TABLE I
PARAMETERS USED IN THE DEFINITION OF OCP (13)

Parameter	Value	Parameter	Value
N	16	$\mathbf{w}_{\dot{s}}$	20 s^2
δ_t	0.005 s	$\mathbf{w}_{\mathcal{P}}$	0.1 m^{-2}
$\rho_{\mathcal{P}}$	0.002 m	$\mathbf{w}_{\mathcal{O}}$	0.1 $\mathrm{sin(rad)}^{-2}$
$\rho_{\mathcal{O}}$	0.02 sin(rad)	$\mathbf{w}_l$	100 m^{-1}

The low-modeling-effort implementations of SCQP and GGN, i.e., applying the SQP method to (25) and (26), showed the same results in terms of tracking-error and optimality as the original implementation of these methods, i.e., applying the SCQP and GGN methods to (18).

The lemniscate-shaped path defined in the task specification is shown in Fig. 2. Here, the position-tunnel is shown in light yellow for $\rho_{\mathcal{P}} = 0.01$ m (for illustration purposes).

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2022.3142396

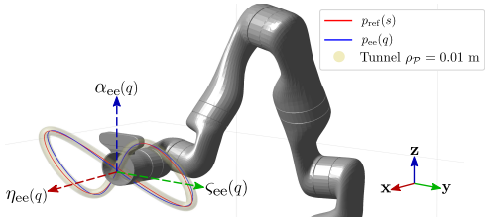

Fig. 2. Representation in simulation of the 7-dof *Kinova Gen3* robot following a lemniscate-shaped path with $\rho_{\mathcal{P}} = 0.01$ m.

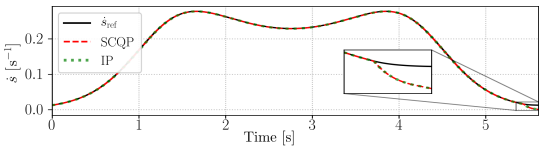

Fig. 3. Profile of the path velocity reference $\dot{s}_{\text{ref}}(s)$ specified for the tunnel-following NMPC scheme and the actual path velocity $\dot{s}$ achieved during the task execution with SCQP and IP. The path velocity $\dot{s}$ achieved with EH, GGN and the original position-tunnel implementation by van Duijkeren [6] is not shown for the sake of simplicity since the maximum difference with respect to SCQP is negligible.

Figure 3 shows the profile of the path velocity reference $\dot{s}_{\text{ref}}(s)$ specified for the task. The enlarged area shows that the trajectory of $\dot{s}$ deviates from its reference near the end of the task, since s was near to its final value $s_{\max} = 1$ where the robot should come to a standstill. The maximum error between $\dot{s}$ and $\dot{s}_{\text{ref}}$ is 1.22×10^{-2} s^{-1} for all methods (SCQP, GNN, EH, IP, van Duijkeren [6]) and occurs at $s = 1$.

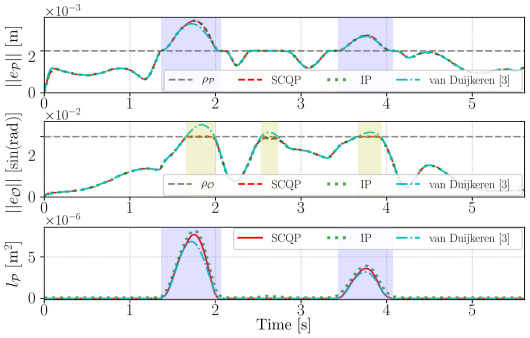

Fig. 4. Excursion of $e_{\mathcal{P}}$, $e_{\mathcal{O}}$ and $l_{\mathcal{P}}$ along the execution of the tunnel-following task. The light-purple areas highlight the time when $||e_{\mathcal{P}}||^2 > \rho_{\mathcal{P}}^2$, leading to $l_{\mathcal{P}} > 0$ to satisfy the position-tunnel constraint (4), while the light-yellow areas highlight the excursions of $||e_{\mathcal{O}}||$ beyond $\rho_{\mathcal{O}}$ for the position-only tunnel-following scheme [6]. The excursion of $l_{\mathcal{O}}$ is not shown since, for this task, $l_{\mathcal{O}} = \{0 \mid ||e_{\mathcal{O}}||^2 \leq \rho_{\mathcal{O}}^2 \ \forall\, s \in \mathcal{T}\}$ while the orientation-tunnel is not considered in [6]. Errors obtained with EH and GGN are not included in the figure for the sake of simplicity since the maximum difference with respect to SCQP is negligible.

The excursion of the position and orientation errors along the execution of the task is shown in Fig. 4. Although $||e_{\mathcal{P}}||^2$ exceeds the value $\rho_{\mathcal{P}}^2$ along some parts of the trajectory, the excursion of the corresponding slack variable $l_{\mathcal{P}}$ allows the satisfaction of constraint (4) to preserve the possibility of finding a feasible solution. In addition, $||e_{\mathcal{O}}||$ goes beyond $\rho_{\mathcal{O}}$ in the position-only tunnel-following scheme [6] due to the lack of an orientation constraint, while $||e_{\mathcal{P}}||$ has a similar excursion than in the extended tunnel-following scheme.

Let us now compare the SCQP and GNN Hessian approximations with the EH of the SQP method for the solution of the tunnel-following task. A comparison of the Hessians in terms of number of atomic operations, number of nonzeros and evaluation time, is detailed in Table II.

TABLE II
COMPARISON OF HESSIAN APPROXIMATIONS WITH RESPECT TO THE EXACT HESSIAN IN SQP

Hessian (approximation)	Number of atomic operations	Number of nonzeros	**Evaluation time**
EH	29513469	7428	3598.710 μs
GGN	342508	1380	37.218 μs
SCQP	635967	1380	77.732 μs

It is shown that, both the GGN and SCQP Hessian approximations are cheaper to evaluate with at least 46.41 times lower number of atomic operations, 5.38 times lower number of nonzeros, and more than 46.29 times lower evaluation time with respect to EH. Nevertheless, the evaluation time of the SCQP approximation is 2.09 times larger than the GGN Hessian as 1.86 times more atomic operations are performed. This difference, however, does not represent a drawback of the SCQP method, as shown below.

A measure of how much disturbance can be handled successfully in a one-step RTI scheme is the Karush-Kuhn-Tucker (KKT) residual, which indicates optimality and feasibility violations. Figure 5 shows a comparison of the methods to solve the extended tunnel-following scheme in terms of the evolution of the KKT residual. A comparison with van Duijkeren [6] is not included since it lacks the orientation-tunnel constraint (12), and consequently, solves a different OCP. It is shown that the satisfaction of the KKT conditions is better for the SQP method with EH and the IP method as expected, while the SCQP method achieves a better performance than the GGN method for almost the whole trajectory, only having similar performances at the beginning and at the end of the trajectory where the motion was slow.

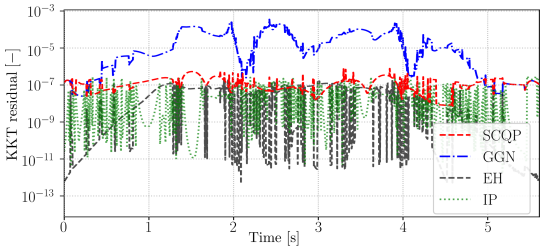

Fig. 5. Comparison of the performance of the position and orientation tunnel-following scheme with SCQP, GGN, EH and IP by means of the evolution of the KKT residuals.

Finally, in Fig. 6 we plot the solution times of the OCP from the proposed extended tunnel-following scheme along the evolution of the task with the three Hessian (approximations) and with IP, and compare them with the solution times of the original position tunnel-following scheme by [6]. The solution times include the time needed to evaluate the functions of the NLP and their derivatives, in addition to the time needed to solve the NLP with the evaluated methods. While the solution time with IP takes in average 827.25 ms, the solution time

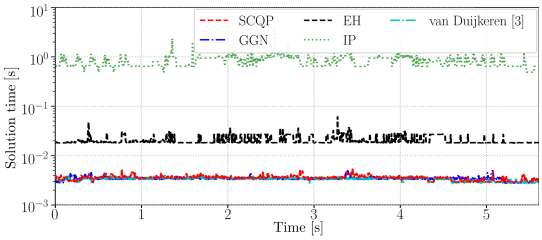

Fig. 6. Comparison of the solution time of the underlying OCP in the position and orientation tunnel-following NMPC scheme using SCQP, GGN, EH or IP, and the position tunnel-following scheme by van Duijkeren [6].

of the OCP (13) using the SCQP method is (with 3.53 ms) only 3.82% larger than the solution time with the position-only tunnel-following scheme by van Duijkeren [6] (3.40 ms), 3.21% larger than the solution time with GGN (3.42 ms), but 82.83% lower than the solution time with EH (20.57 ms).

The task of following a position tunnel with a 7-dof robot is kinematically redundant. Our orientation error extension allows to naturally exploit the remaining freedom to obey the orientation constraints with no adverse effect on position error and little effect on computation time. These results also imply that the use of the SCQP method in the (extended) position and orientation tunnel-following NMPC scheme allows generating more optimal solutions than GGN, while having negligible additional complexity and achieving solution times that allow real-time implementations, despite the fact that the SCQP Hessian is slower to evaluate and has more atomic operations than the GGN Hessian approximation.

VI. CONCLUSIONS

We presented a novel operator that allows the automatic application of symbolic linearization to *convex-over-nonlinear* functions to reduce the modeling effort required to implement the SCQP and GGN methods. We also presented a direct application of the SCQP method: the tunnel-following NMPC scheme, originally described in [6]. An extension of this scheme was proposed, in which an orientation-tunnel constraint was added to the underlying OCP to allow the exploitation of user-defined tolerances in the orientation of the end-effector of a robot manipulator. The proposed method was demonstrated in simulation on a task defined within the extended tunnel-following NMPC scheme. Both the SCQP and GGN methods are shown to be effectively implemented by using symbolic linearization. However, the SCQP method is able to better exploit the convexity present in *convex-over-nonlinear* functions in the constraints of the OCP in the tunnel-following NMPC scheme, i.e., the position and orientation tunnel constraints, achieving better performance with an almost negligible increase in the solution time with respect to the GGN method, but with a considerable decrease in solution time compared to the SQP and IP methods. With the definition of the symbolic lin operator, any software stack that relies on CasADi expressions for modeling (dynamic) optimization problems can be trivially extended to support GGN and SCQP. Future work will consider the validation of the proposed contributions through experiments on an actual robot, the comparison against other optimization-based controllers, the use of other types of variables that describe motion of a robot manipulator and applications where freedom on such variables can be exploited, in addition to the exploration of a systematic way to automatically apply the lin operator in an OCP when *convex-over-nonlinear* functions are recognized by a solver.

REFERENCES

[1] N. R. Patel and J. Rawlings, *Handbook of Model Predictive Control*, ser. Control Engineering, S. V. Raković and W. S. Levine, Eds. Cham: Springer International Publishing, 2019.

[2] M. Polic, M. Car, F. Petric, and M. Orsag, "Compliant Plant Exploration for Agricultural Procedures with a Collaborative Robot," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2768–2774, 2021.

[3] C. A. My, D. X. Bien, B. H. Tung, L. C. Hieu, N. V. Cong, and T. V. Hieu, "Inverse kinematic control algorithm for a welding robot - positioner system to trace a 3D complex curve," in *2019 Int. Conf. Adv. Technol. Commun.*, vol. 2019-Octob. IEEE, oct 2019, pp. 319–323.

[4] K. Castelli, A. M. A. Zaki, Y. Dmytriyev, M. Carnevale, and H. Giberti, "A Feasibility Study of a Robotic Approach for the Gluing Process in the Footwear Industry," *Robotics*, vol. 10, no. 1, dec 2020.

[5] B. Joffe, T. Walker, R. Gourdon, and K. Ahlin, "Pose estimation and bin picking for deformable products," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 361–366, 2019.

[6] N. Van Duijkeren, "Online Motion Control in Virtual Corridors - For Fast Robotic Systems," Ph.D. dissertation, KU Leuven, 2019.

[7] H. G. Bock, "Recent Advances in Parameteridentification Techniques for O.D.E." in *Numer. Treat. Inverse Probl. Differ. Integr. Equations*. Boston, MA: Birkhäuser Boston, 1983, pp. 95–121.

[8] N. N. Schraudolph, "Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent," *Neural Comput.*, vol. 14, no. 7, pp. 1723–1738, jul 2002.

[9] R. Verschueren, N. van Duijkeren, R. Quirynen, and M. Diehl, "Exploiting convexity in direct Optimal Control: a sequential convex quadratic programming method," in *2016 IEEE 55th Conf. Decis. Control*, dec 2016, pp. 1099–1104.

[10] F. Messerer and M. Diehl, "Determining the Exact Local Convergence Rate of Sequential Convex Programming," in *2020 Eur. Control Conf.* IEEE, may 2020, pp. 1280–1285.

[11] R. Verschueren, N. Van Duijkeren, J. Swevers, and M. Diehl, "Time-optimal motion planning for n-DOF robot manipulators using a path-parametric system reformulation," *Proc. Am. Control Conf.*, vol. 2016-July, pp. 2092–2097, 2016.

[12] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Automat. Contr.*, vol. 54, no. 10, pp. 2318–2327, 2009.

[13] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer London, 2009.

[14] H. Bock and K. Plitt, "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems," *IFAC Proc. Vol.*, vol. 17, no. 2, pp. 1603–1608, jul 1984.

[15] 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, mar 2019.

[16] M. Diehl, H. G. Bock, and J. P. Schlöder, "Real-time iterations for nonlinear optimal feedback control," *Proc. 44th IEEE Conf. Decis. Control. Eur. Control Conf. CDC-ECC '05*, pp. 5871–5876, 2005.

[17] J. A. Andersson and J. B. Rawlings, "Sensitivity Analysis for Nonlinear Programming in CasADi," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 331–336, 2018.

[18] 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.

[19] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *2019 IEEE/SICE Int. Symp. Syst. Integr.* IEEE, jan 2019, pp. 614–619.

[20] A. Astudillo, J. Carpentier, J. Gillis, G. Pipeleers, and J. Swevers, "Mixed Use of Analytical Derivatives and Algorithmic Differentiation for NMPC of Robot Manipulators," in *IFAC-PapersOnLine*, vol. 54, no. 20, Austin, Texas, 2021, pp. 78–83.