18/01/2025 11:35

(Article begins on next page)

A Minimum-Jerk Approach to Handle Singularities in Virtual Fixtures

Giovanni Braglia[1], Sylvain Calinon[2], Luigi Biagiotti[1]

Abstract—Implementing virtual fixtures in guiding tasks constrains the movement of the robot's end effector to specific curves within its workspace. However, incorporating guiding frameworks may encounter discontinuities when optimizing the reference target position to the nearest point relative to the current robot position. This article aims to give a geometric interpretation of such discontinuities, with specific reference to the commonly adopted Gauss-Newton algorithm. The effect of such discontinuities, defined as Euclidean Distance Singularities, is experimentally proved. We then propose a solution that is based on a linear quadratic tracking problem with minimum jerk command, then compare and validate the performances of the proposed framework in two different human-robot interaction scenarios.

Index Terms—Human-Robot Collaboration, Motion and Path Planning, Optimization and Optimal Control, Physical Human-Robot Interaction.

I. INTRODUCTION

AS collaborative robotics research is making great strides towards safe Human-Robot Interaction (HRI), users are becoming more confident to utilize robots in co-manipulation tasks. In such scenarios, it is common to adopt Virtual Fixtures (VFs) to constraint the robot's motion to a specific manifold [1]. Especially in those situations where the human is required to manually execute precise tasks, it has been proven that the use of VFs facilitates adherence to the desired task while reducing the mental workload required to maintain accuracy [2], [3].

In many guiding applications, allowing compliance with respect to the VF curve $\boldsymbol{\mu}$ ensures a more natural and intuitive interaction with the user [4]–[6]. A common approach in such cases, is to adjust the reference position on $\boldsymbol{\mu}$ to the point that minimizes the displacement with the actual position of the robot. In this way, the robot is able to follow the user movements, whilst imposing the
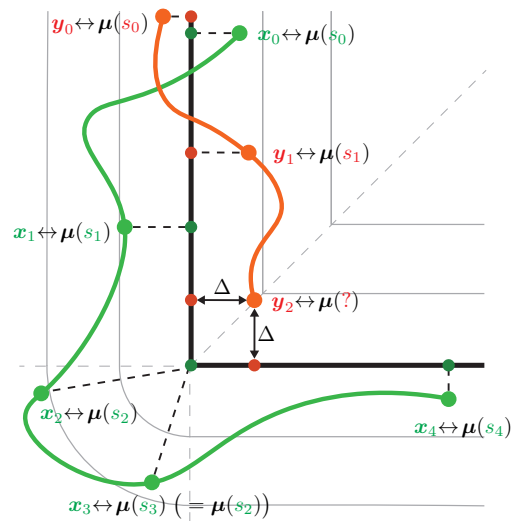
Figure 1: 2D visualization of an Euclidean distance singularity. Continuous gray lines represent the iso-lines with respect to the constraint path $\boldsymbol{\mu}(s)$ depicted in black. While a distance-based method could correctly update the reference position $\boldsymbol{\mu}(s_t)$ for the green trajectory, it fails to find a stable solution for the red trajectory, as $\boldsymbol{y}_2$ has the same distance Δ from 2 points in $\boldsymbol{\mu}(s)$.

virtual constraint [2]. As far as we know, we noticed that this procedure has not yet been extended to positions in the workspace that share the same distance to multiple points on the VF, here called *Euclidean distance singularities* (EDSs). This problem is illustrated in Fig. 1 and investigated throughout this article.

A. Related Works

The term virtual fixture (VF) is used when restricting the robot movements to a specified manifold, and it owes much of its popularity to guiding applications [1], [3]. Here, the human interacts with the environment while being enforced by the robot to adhere to a specific curve [4]. Usually, the definition of VFs restricts to the imposition of geometrical constraints not associated with any time-law. When focusing on path planning, it is common practice to define geometrical and time constraints in separates moment [6], [7]. Given a geometrical constraint $\boldsymbol{\mu}$, the time-law can be managed through the definition of the so-called phase variable s_t to further define the dynamics $\boldsymbol{\mu}(s_t)$. In this way, one can properly compute s_t to regulate the manipulator dynamics depending on geometrical, kinematic and external constraints [8], [9]. To the best of our knowledge, we believe that the methods for determining

the time-law can be categorized into three main groups: linear, user-defined and optimization-based.

In linear approaches, the time scaling is adjusted by proportionally modify the duration of the task execution T, typically varying a scalar value to regulate the velocity of the phase variable $\dot{s}_t$. A classic example of this category is Dynamic Movement Primitives (DMPs), where a nonlinear term controlling the system's dynamics is regulated by a tunable phase variable, provided by the canonical system [10]. Another example is provided by [9], where a scalar parameter adjusts online the velocity of the manipulator to slow down when approaching the human.

Depending on the type of application, one may want to influence the variation of s_t in a different way. We refer to user-defined approaches whenever the definition of the time-law is made explicit through an analytic formulation that is not scalar on $\dot{s}_t$. Again, some variations of DMPs transform the canonical system such that $\dot{s}_t$, for example, depends on the position error [11] or external force measurements [12], [13]. Another example can be found in [4], where authors utilize a dynamical system representation for the computation of $\dot{s}_t$ along the virtual constraint.

The last category, which will be the focus of this article, includes all the techniques where the time-law s_t is obtained as the solution of an optimization problem [7], [14]. In particular, we observed that many virtual fixture applications rely on distance-based methods, selecting the point on the curve $\boldsymbol{\mu}(s)$ that minimizes the Euclidean distance from the robot's end effector (EE) position $\boldsymbol{x}$ [2], [15]–[17]. Finding an analytical solution to calculate the minimum distance point $\boldsymbol{\mu}(s^*)$ is generally non-trivial. Nevertheless, the presence of techniques such as the Gauss-Newton (GN) algorithm, simplifies the problem to the computation of the phase s_t^* that minimizes the residual $\|\boldsymbol{x}_t - \boldsymbol{\mu}(s_t^*)\|$ [18]. The GN algorithm is known to be computationally efficient and was applied in various robotic applications [19], [20]. However, its use to update s_t may not lead to a convex formulation for the minimum distance problem [21]. In this article, we refer to this condition as *Euclidean-distance singularity* (EDS). We demonstrate analytically and experimentally that approaching an EDS might produce abrupt changes in the phase velocity $\dot{s}_t$.

Controlling the phase velocity is common in planning algorithms, and the outcome solution usually produce a time-optimal trajectory that exploits the maximum accelerations/torques given the constraints of the robot actuators [7]. This could potentially result in acceleration profiles with a high rate of change, which can stress the actuators leading to long term damage [22]. In these cases, enhancing smoothness is a good compromise between minimum time task execution, and acceleration noise reduction. To achieve so, it is a common practice to minimize the jerk of the planned trajectories [8], [23]. Minimum jerk trajectories have been widely investigated also in the reproduction of human arm movements [24], [25]. As demonstrated in [26], minimum jerk trajectories can faithfully reproduce the speed profile of arm movements in reaching tasks or curve tracking, making them suitable

for human-robot interaction scenarios [5]. This motivates our research to propose a novel methodology for the online phase update in guiding VFs, exploiting a jerk-controlled framework based on minimum distance, while ensuring robustness against EDSs.

B. Contribution

The main contributions of this paper are: (1) present a theoretical representation of the so-called Euclidean distance singularities (EDSs) with an experimental proof of their limitation; (2) provide a formulation for updating the phase variable using minimum jerk control with a linear quadratic tracking (LQT) algorithm and (3), a real-time application of the proposed framework in a human-robot interaction setup. The outline of this paper is as follows. Section II offers a theoretical background on the addressed EDSs problem. Section III presents a jerk-based LQT formulation for optimal phase update. Section IV validates and discusses the implementation of our proposed methodology in two experimental scenarios, and Section V concludes this work.

II. BACKGROUND

In this article, we aim to constrain the robot to follow a predefined reference path $\boldsymbol{\mu}$ in the task space. Once established, the operator can use the robot's guidance to navigate along this path. Details of this mechanism are provided in the following paragraphs.

A. Definition of the virtual constrain

To define the constraint $\boldsymbol{\mu}$, we first kinesthetically move the robot end effector to demonstrate the target curve. The recordings are then filtered using the spatial sampling (SS) algorithm from [13], to extract the geometrical path information regardless of the timing introduced during the demonstration. This associates the recordings $\boldsymbol{r}(t) = [r_x(t), , r_y(t), , r_z(t)]^\top$ with their filtered counterparts $\boldsymbol{r}_{\Delta,k} = [r_{x,\Delta,k}, , r_{y,\Delta,k}, , r_{z,\Delta,k}]^\top$ and curvilinear coordinates $s_k \in [0, L]$, with $k = 1, ..., M$ and $L = \Delta M$ representing the curve length $\boldsymbol{r}_\Delta$. The free parameter Δ defines the geometric distance between consecutive samples in $\boldsymbol{r}_{\Delta,k}$ and is such that, given an analytical approximation $\boldsymbol{\mu}(s) \approx \boldsymbol{r}_{\Delta,k}$, the following property holds:

$$\left\|\frac{d\boldsymbol{\mu}(s)}{ds}\right\|_{s=s_k} \approx \frac{\|\boldsymbol{r}_{\Delta,k+1} - \boldsymbol{r}_{\Delta,k}\|}{\|s_{k+1} - s_k\|} = \frac{\Delta}{\Delta} = 1. \quad (1)$$

We expressed $\boldsymbol{\mu}(s)$ using basis functions approximation, representing the signal $\boldsymbol{r}_{\Delta,k}$ as a weighted sum of nonlinear terms, namely [27]

$$\boldsymbol{\mu}(s_t) = \sum_{i=1}^{N} \omega_i \phi_i(s_t) = \boldsymbol{\omega}^\top \boldsymbol{\phi}(s_t). \quad (2)$$

In (2), the constraint curve is formed by summing N basis functions ϕ_i, each weighted by ω_i, typically found via a Least-Squares solution [10], i.e.

$$\boldsymbol{\omega} = \operatorname*{argmin}_{\boldsymbol{\omega}} \sum_{k=0}^{M} \|\boldsymbol{r}_{\Delta,k} - \boldsymbol{\omega}^\top \boldsymbol{\phi}(s_k)\|^2, \quad (3)$$

indicating with $|| \cdot ||$ the Euclidean norm of the residual term. The basis functions $\phi_i(s_t)$ can be chosen in different ways, in this case we used Bernstein polynomials [27], [28]. Again, the phase variable is denoted by s_t, but thanks to the SS algorithm it carries an additional meaning of being the curvilinear coordinate of the curve $\boldsymbol{\mu}$.

For instance, with the expression in (2) we can compute the velocity and acceleration of $\boldsymbol{\mu}(s_t)$ respectively as

$$
\dot{\boldsymbol{\mu}}(s_t) = \frac{\partial \boldsymbol{\mu}(s(t))}{\partial t}\bigg|_{t=\alpha\Delta t} = \boldsymbol{\mu}'(s_t)\dot{s}_t \text{ and}
$$
$$
\ddot{\boldsymbol{\mu}}(s_t) = \frac{\partial^2 \boldsymbol{\mu}(s(t))}{\partial t^2}\bigg|_{t=\alpha\Delta t} = \boldsymbol{\mu}''(s_t)\dot{s}_t^2 + \boldsymbol{\mu}'(s_t)\ddot{s}_t, \tag{4}
$$

with $\ddot{s}_t$ and $\dot{s}_t$ obtained from their respective continuous counter part at sampling time Δt. In (4) and throughout this article, the derivative with respect to s_t is denoted with the prime symbol, while the discrete time derivative uses the dot notation. Note that, because of (1), $\boldsymbol{\mu}'(s_t)$ will always be well-defined and different from zero [6], [28].

B. Tracking Problem Statement

Let $\boldsymbol{\mu}(s) \in \mathbb{R}^3$ be the path constraint parametrized with respect to the curvilinear coordinate s, and $\boldsymbol{x}_t \in \mathbb{R}^3$ be the Cartesian position of the robot's end effector (EE). The objective of limiting the EE movement along the curve $\boldsymbol{\mu}(s)$ can be formulated as a tracking problem, that is

$$
s_t^\star = \underset{s_t \in [0, M\Delta]}{\operatorname{argmin}} \|\boldsymbol{x}_t - \boldsymbol{\mu}(s_t)\|^2. \tag{5}
$$

The optimal problem in (5) equals to update s_t at every instant t in order to find the closest point $\boldsymbol{\mu}(s_t^\star)$ to $\boldsymbol{x}_t$. From [21], the existence of a solution is guaranteed if:

$$
(\boldsymbol{x}_t - \boldsymbol{\mu}(s_t^\star))^\top \boldsymbol{\mu}'(s_t^\star) = 0, \text{ and} \tag{6a}
$$

$$
\|\boldsymbol{\mu}'(s_t^\star)\|^2 - (\boldsymbol{x}_t - \boldsymbol{\mu}(s_t^\star))^\top \boldsymbol{\mu}''(s_t^\star) > 0. \tag{6b}
$$

The conditions in (6) leads to the following geometrical intuition. On one hand, the first order necessary condition in (6a) states that the optimal $s_t^\star$ lies on the point $\boldsymbol{\mu}(s_t^\star)$ whose tangential component $\boldsymbol{\mu}'(s_t^\star)$ is orthogonal to the EE position $\boldsymbol{x}_t$. From (1), $\boldsymbol{\mu}'(s_t)$ represents the tangential unit vector to the curve. On the other hand, the second-order sufficient condition in (6b) guarantees the convexity of (5) if the Hessian of the cost function $\|\boldsymbol{x}_t - \boldsymbol{\mu}(s_t)\|^2$ is positive definite. In particular, given the constraint $\boldsymbol{\mu}(s)$ for $s \in [0, \Delta M]$, at every instant t a feasible subset $\chi_t = \{\bar{\boldsymbol{x}}_t \in \mathbb{R}^3 : \text{(6b) holds}\}$ can be deduced [16]. Equation (6) implies $\boldsymbol{\mu} \in C^2$, which can be ensured by a proper choice of the basis functions in (2) [27].

C. Gauss-Newton algorithm

One common technique for solution of the optimization problem in (5) comes from the Gauss-Newton (GN) algorithm [18], [20]. In this case, the evolution of the curvilinear coordinate s_t is given by $s_{t+1} = s_t + \Delta s$, with the term Δs being the GN update. Defining the residual $\boldsymbol{f}(s_t) = \boldsymbol{x}_t - \boldsymbol{\mu}(s_t)$, the GN algorithm provides an analytical solution for the computation of Δs, that is

$$
\Delta s_t = -\boldsymbol{H}(s_t)^{-1}\boldsymbol{g}(s_t), \text{ with}
$$
$$
\boldsymbol{g}(s_t) = 2\boldsymbol{J}_{\boldsymbol{f}}^\top \boldsymbol{f}(s_t), \quad \boldsymbol{H}(s_t) \approx 2\boldsymbol{J}_{\boldsymbol{f}}^\top \boldsymbol{J}_{\boldsymbol{f}}. \tag{7}
$$

In (7), $\boldsymbol{J}_{\boldsymbol{f}} \in \mathbb{R}^{3 \times D}$ defines the Jacobian matrix for $\boldsymbol{f}(s_t) \in \mathbb{R}^3$, while $\boldsymbol{g} \in \mathbb{R}^D$ and $\boldsymbol{H} \in \mathbb{R}^{D \times D}$ are the gradient vector and the Hessian matrix of the cost function with respect to the residual $\boldsymbol{f}(s_t)$, respectively. For our tracking problem we can assume $\boldsymbol{J}_{\boldsymbol{f}}(s_t) = -\boldsymbol{\mu}'(s_t)$. The advantage of the GN algorithm is that, despite being a second order technique, it converges in few steps. This is possible as the Hessian can be easily estimated from the Jacobian in (7), which guarantees its positive definiteness [18], [20].

Note that the GN algorithm neglects the second-order derivative term of the Hessian in (6b). However, when this term cannot be ignored, the GN algorithm may experience slow convergence. From (1) one can write (6b) as

$$
1 - (\boldsymbol{x}_t - \boldsymbol{\mu}(s_t))^\top \boldsymbol{\mu}''(s_t) > 0. \tag{8}
$$

Given the geometry of curves, we have that $\boldsymbol{\mu}''(s_t) = \boldsymbol{n}(s_t)\kappa(s_t)$, where $\boldsymbol{n}$ and κ define the normal versor and the curvature, respectively, calculated at the curvilinear coordinate s_t [6], [29]. In particular, observing that $r(s_t) = 1/\kappa(s_t)$ expresses the radius of the osculating circle at the point $\boldsymbol{\mu}(s_t)$, the condition in (8) can be written as

$$
(\boldsymbol{x}_t - \boldsymbol{\mu}(s_t))^\top \boldsymbol{n}(s_t) < r(s_t). \tag{9}
$$

Equation (9) provides the following geometric intuition: given the reference $\boldsymbol{\mu}(s_t)$, the center of the osculating sphere defines a boundary for the deviation of the EE position $\boldsymbol{x}_t$, out of which the second order condition for the optimality in (8) is violated. Therefore, the optimal problem in (5) no longer admits a unique solution, as it looses convexity due to the fact that the Hessian matrix represented in (6b) ceases to be positive definite [20], [21].

Notably, in [16] authors demonstrate that the time derivative of the optimal $s_t^\star$ in (5) can be written as

$$
\dot{s}_t^\star = \frac{\boldsymbol{\mu}'(s_t^\star)^\top \dot{\boldsymbol{x}}_t}{1 - (\boldsymbol{x}_t - \boldsymbol{\mu}(s_t^\star))^\top \boldsymbol{\mu}''(s_t^\star)}. \tag{10}
$$

Given the condition in (8), it can be evinced from (10) that the limit case $(\boldsymbol{x}_t - \boldsymbol{\mu}(s_t))^\top \boldsymbol{\mu}''(s_t) \to 1$ implies $\dot{s}_t^\star \to \infty$. Hence, small values of $\dot{\boldsymbol{x}}_t$ results in large derivatives of $\dot{s}_t^\star$ when close to the center of the osculating circle calculated at $\boldsymbol{\mu}(s_t^\star)$. In our case, as we allow compliance in the directions outside the path constraint $\boldsymbol{\mu}(s)$, this can compromise the contact with the robot and the human, given the generation of the reference trajectories outlined in (4). These considerations, that will be further analyzed in Section IV-B, provide an analytical explanation of the constraint imposed in [17] and lead to the definition of Euclidean distance singularity.
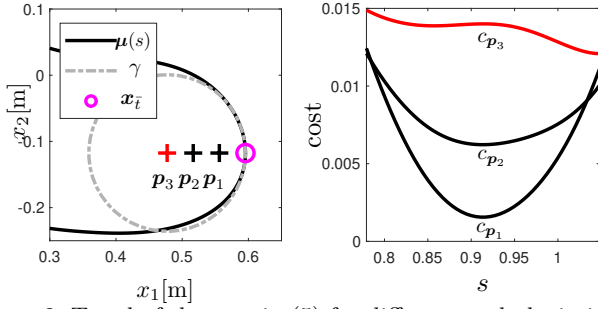
Figure 2: Trend of the cost in (5) for different path deviations.

Remark 1: *Given the optimal problem in* (5), *with $\bar{s}$ being curvilinear coordinate of the curve $\boldsymbol{\mu}(s) \in \mathbf{R}^n$, we denote with Euclidean distance singularity (EDS) the set:*

$$EDS = \left\{ \begin{array}{ll} \boldsymbol{x} \in \mathbf{R}^n: & (\boldsymbol{x} - \boldsymbol{\mu}(\bar{s}))^\top \boldsymbol{\mu}'(\bar{s}) = 0, \\ & (\boldsymbol{x} - \boldsymbol{\mu}(\bar{s}))^\top \boldsymbol{n}(\bar{s}) \geq r(\bar{s}) \end{array} \right\}. \quad (11)$$

The geometric intuition is that if we move along the normal direction $\boldsymbol{n}(\bar{s})$ by a distance greater than the radius of the osculating circle at $\boldsymbol{\mu}(\bar{s})$, the optimization problem in (5) degenerates into multiple local solutions. Though these conditions may seem restrictive, Sec. IV-B demonstrates that even near an EDS issues can arise.

Figure 2 provides a 2D example. The left subplot shows the osculating circle γ at $\boldsymbol{\mu}(s_{\bar{t}})$ in dashed gray. The right subplot depicts the cost functions $c_{\boldsymbol{p}_1}$, $c_{\boldsymbol{p}_2}$, and $c_{\boldsymbol{p}_3}$ in (5) concerning the EE position $\boldsymbol{x}_{\bar{t}}$ relative to points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$. Note that the cost function loses local convexity once the deviation reaches the center of the osculating circle $\boldsymbol{p}_3$. Note that, for curves with sharp corners, using a basis function approximation as in (2) ensures the continuity of the geometrical path and the Jacobian in (7) [28].

III. METHODOLOGY

With reference to equation (10), the proximity of the EE position $\boldsymbol{x}_t$ to EDSs could lead to high velocities $\dot{s}_t$. One way to mitigate such undesired behaviour, is to formulate the optimal problem in (5) such that we can control the damping on the velocity term. This is possible by considering minimum-jerk trajectories. Moreover, many results in literature indicate that this model is well-suited for human-robot co-manipulation tasks [5], [26], [30].

Minimum-jerk approaches highlight that the velocity profile of hand movements is established by minimizing its squared jerk [24]. Authors in [31] extend the minimum jerk formulation to the so-called *jerk-accuracy* (JA) model, where the tracking precision and the control action is modulated with the help of a Lagrange multiplier. We here formulate the JA model as a linear quadratic tracking (LQT) problem for discrete-time systems. In doing so, the control command is fed to a chain of three discrete-time integrators, i.e. $u_t = \dddot{s}_t$, such that the reference $\boldsymbol{\mu}(s_t)$ stays aligned with the EE position $\boldsymbol{x}_t$. The problem becomes

$$\min_{u_t} \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{s}_t)^\top \boldsymbol{Q} \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{s}_t) + u_t^\top R u_t, \quad (12)$$
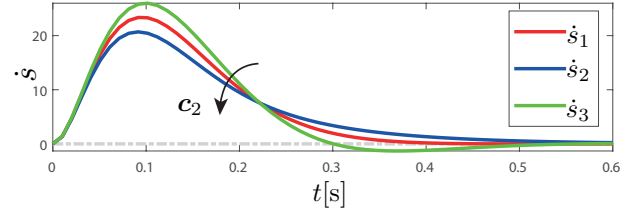
Figure 3: Behaviour of the curvilinear parameter velocity $\dot{s}$ with varying velocity weight $\boldsymbol{c}_2$.

with $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{s}_t) = \left[\boldsymbol{e}_t^\top, \dot{\boldsymbol{e}}_t^\top, \ddot{s}_t \right]^\top$, $\boldsymbol{e}_t = \boldsymbol{x}_t - \boldsymbol{\mu}(s_t)$, subject to

$$\underbrace{\begin{bmatrix} s_{t+1} \\ \dot{s}_{t+1} \\ \ddot{s}_{t+1} \end{bmatrix}}_{\boldsymbol{s}_{t+1}} = \underbrace{\begin{bmatrix} 1 & \Delta t & \Delta t^2/2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} s_t \\ \dot{s}_t \\ \ddot{s}_t \end{bmatrix}}_{\boldsymbol{s}_t} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \Delta t \end{bmatrix}}_{\boldsymbol{B}} u_t. \quad (13)$$

In (12), $\boldsymbol{s}_t = [s_t, \dot{s}_t, \ddot{s}_t]^\top$, $s_t \in [0, \Delta M]$, Δt is the sampling time, $\boldsymbol{Q} = \text{diag}(\boldsymbol{c}_1, \boldsymbol{c}_2, c_3)$ is called precision matrix with constant parameters $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathbb{R}^3$, $c_3 \in \mathbb{R}$, and the constant R is the weight control coefficient. The addition of $\ddot{s}_t$ to the non-linear term $\boldsymbol{f}$ guarantees that the computation of the Jacobian $\boldsymbol{J}_t = \partial \boldsymbol{f}_t / \partial \boldsymbol{s}_t$ is not rank deficient. Let the robot EE position be at $\bar{\boldsymbol{x}}_t$. We apply the Newton's method to minimize the cost $c(\boldsymbol{s}_t, u_t) = \boldsymbol{f}(\boldsymbol{s}_t)^\top \boldsymbol{Q} \boldsymbol{f}(\boldsymbol{s}_t) + u_t^\top R u_t$. This can be done by carrying out a second order Taylor expansion around the point $(\boldsymbol{s}_t, u_t)$, that is [20]:

$$\begin{aligned} \Delta c(\Delta \boldsymbol{s}_t, \Delta u_t) \approx \; & 2 \Delta \boldsymbol{s}_t^\top \boldsymbol{J}(\boldsymbol{s}_t)^\top \boldsymbol{Q} \boldsymbol{f}(\boldsymbol{s}_t) + 2 \Delta u_t^\top R u_t + \\ & + \Delta \boldsymbol{s}_t^\top \boldsymbol{J}(\boldsymbol{s}_t)^\top \boldsymbol{Q} \boldsymbol{J}(\boldsymbol{s}_t) \Delta \boldsymbol{s}_t + \Delta u_t^\top R \Delta u_t. \end{aligned} \quad (14)$$

Therefore, the optimization problem in (12) is linearized using the cost function in (14), i.e.

$$\min_{\Delta \boldsymbol{u}} \Delta c(\Delta \boldsymbol{s}, \Delta u) \quad \text{s.t.} \quad \Delta \boldsymbol{s} = \boldsymbol{S}_u \Delta \boldsymbol{u}, \quad (15)$$

with $\Delta \boldsymbol{s} = [\Delta \boldsymbol{s}_1, ..., \Delta \boldsymbol{s}_T]^\top$, $\Delta \boldsymbol{u} = [\Delta u_1, ..., \Delta u_{T-1}]^\top$, and $\boldsymbol{S}_u$ characterizing the control transfer matrix of the system's dynamics in (13) expressed at trajectory level for $t = 1, 2, ..., T$, namely $\boldsymbol{s} = \boldsymbol{S}_s \boldsymbol{s}_1 + \boldsymbol{S}_u \boldsymbol{u}$ [20]. By differentiating (15) with respect to $\Delta \boldsymbol{u}$ and equating to zero, one can compute at each iteration the optimal control command:

$$\Delta \boldsymbol{u}^\star = \left(\boldsymbol{S}_u^\top \boldsymbol{J}(\boldsymbol{s})^\top \boldsymbol{Q} \boldsymbol{J}(\boldsymbol{s}) \boldsymbol{S}_u + \boldsymbol{R} \right)^{-1} \left(-\boldsymbol{S}_u^\top \boldsymbol{J}(\boldsymbol{s})^\top \boldsymbol{Q} \boldsymbol{f}(\boldsymbol{s}) - \boldsymbol{R} \boldsymbol{u} \right). \quad (16)$$

With the proposed LQT formulation, the quadratic cost function in (12) offers intuitive and interpretable settings for parameters $\boldsymbol{Q}$ and R. Additionally, a second-order approximation reliably guides the optimization steps towards the minimum of the local estimation of the cost function (14) [20], [21]. While higher-order methods may provide a more accurate approximation, they require additional algorithms to find their minimum, thereby increasing computational effort.

The LQT algorithm described earlier was initially simulated in a reaching task. The velocity profiles $\dot{s}$ are displayed in Fig. 3. The figure demonstrates that the proposed framework achieves the characteristic bell-shaped

Algorithm 1 Iterative LQT algorithm

Input: $\boldsymbol{x}_{\bar{t}}$, $\boldsymbol{\mu}(s_{\bar{t}})$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{Q}$, R, in (12)- (13), I_{MAX} .

Output: $\boldsymbol{s}_{\bar{t}+1}^{\star}$.

   Initialization : for $t = \bar{t}, \bar{t}+1, ..., \bar{t}+T_W-1$ compute the transfer matrices $\boldsymbol{S}_{\boldsymbol{s}}$, $\boldsymbol{S}_{\boldsymbol{u}}$ and initialize $\boldsymbol{s}_{1,0} = \boldsymbol{s}_{\bar{t}}$, $\boldsymbol{u} = [0_{\bar{t}}, ..., 0_{\bar{t}+T_W-1}]$.

1: **for** $i = 1$ to I_{MAX} **do**
2:    Compute the dynamics $\boldsymbol{s} = \boldsymbol{S}_{\boldsymbol{s}} \boldsymbol{s}_{1,i-1} + \boldsymbol{S}_{\boldsymbol{u}} \boldsymbol{u}$
3:    Calculate the residual $\boldsymbol{f}(\boldsymbol{x}_{\bar{t}}, \boldsymbol{s})$ and the Jacobian $\boldsymbol{J}(\boldsymbol{s})$ in (12)- (15)
4:    Compute $\Delta\boldsymbol{u}^{\star}$ as in (16)
5:    **if** $(||\Delta\boldsymbol{u}^{\star}|| < \Delta_{min})$ **then**
6:       Local minimum reached, exit **for** loop
7:    **end if**
8:    Update control $\boldsymbol{u} = \boldsymbol{u} + \Delta\boldsymbol{u}^{\star}$
9:    Update initial state $\boldsymbol{s}_{1,i} = \boldsymbol{A}\boldsymbol{s}_{1,i-1} + \boldsymbol{B}u_1$
10:   Re-define the control as $\boldsymbol{u} = [u_2, u_3, ..., u_{T_W}, u_{T_W}]^{\top}$
11: **end for**
12: Save the state $\boldsymbol{s}_{\bar{t}+1}^{\star} = \boldsymbol{s}_{1,i} = [s_{1,i}, \dot{s}_{1,i}, \ddot{s}_{1,i}]^{\top}$
13: **return** $\boldsymbol{s}_{\bar{t}+1}^{\star}$

---

profile which characterizes human reaching tasks [26]. Moreover, by reducing $\boldsymbol{c}_2$ in the precision matrix $\boldsymbol{Q}$, the velocity error $\dot{\boldsymbol{e}} = \dot{\boldsymbol{x}}_t - \dot{\boldsymbol{\mu}}(s_t)$ becomes less damped, thus the controller more reactive. As a consequence, the velocity profile changes its sign before reaching zero, as for s_3 depicted in green. This behaviour is common in rapid arm movements, where a corrective movement is actuated to refine the reaching precision [31].

The implementation of the LQT algorithm is resumed in Algorithm 1. In its real-time implementation, the LQT algorithm is computed at each iteration with reduced time steps $t = 1, 2, ..., T_W$, using a Model Predictive Control approach. This allows for faster computation of the optimal control $\hat{u}t + 1$ by minimizing the time window T_W. Additionally, the control $\Delta\boldsymbol{u}^{\star}$ is iteratively computed until its norm reaches a lower bound Δmin, with a maximum number of epochs $I_{M!A!X}$ set to prevent unwanted latency. The initial conditions for each iteration are set to the last commanded pair $(\boldsymbol{s}_{t-1}, u_{t-1})$, providing a warm start for faster convergence.

IV. EXPERIMENTS AND RESULTS

We compare the proposed linear quadratic tracking (LQT) algorithm, together with the Gauss-Newton (GN), a virtual mechanism (VM) approach [4] and a simple control in gravity compensation (GC). The validation consists in two different experimental scenarios which are detailed in the following sections[1].

A. Experimental Setup

Experiments involved a Franka Emika Panda robot together with a Schunk FT-AXIA force/torque sensor.

---

[1]An illustrative video of the experiments can be found at: *https://youtu.be/FLrSDptwb8Q*
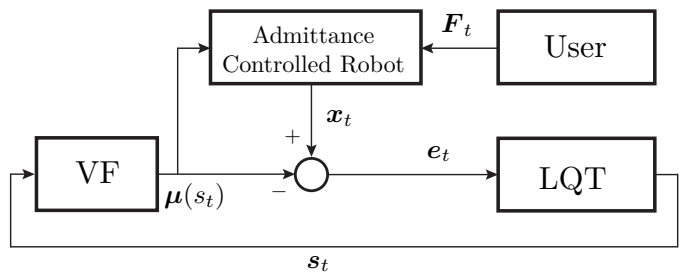

Figure 4: Controller framework. Here VF denotes the virtual fixture constraint.

The robot controller has been implemented in C++, while the codes for the optimization of s_t were developed in MATLAB; finally, each node was connected using Robot Operative System (ROS). The tested algorithms imposes that the generated reference point $\boldsymbol{\mu}(s_t)$ follows the end effector (EE) displacement caused by the user [6]. To minimize the kinesthetic effort while preventing the user from sensing the natural manipulator's inertia, the robot has been endowed with and admittance control. Specifically, the translation dynamics in the Cartesian space have been chosen as:

$$\boldsymbol{M}\ddot{\hat{\boldsymbol{\mu}}}_t + \boldsymbol{B}\dot{\hat{\boldsymbol{\mu}}}_t + \boldsymbol{K}\left(\hat{\boldsymbol{\mu}}_t - \boldsymbol{\mu}(s_t)\right) = \boldsymbol{F}_t, \qquad (17)$$

where $\boldsymbol{M} = \text{diag}(m, m, m)$, $\boldsymbol{B} = \text{diag}(b, b, b)$, $\boldsymbol{K} = \text{diag}(k, k, k)$ are the simulated inertia, damping and stiffness coefficient, respectively, while $\boldsymbol{F}_t$ represents the measured force at the EE. To achieve good tracking and low manipulator inertia, we empirically choose $m = 1.5[kg]$, $b = 15[N \cdot s/m]$ and $k = 200[N \cdot m]$. The position $\hat{\boldsymbol{\mu}}_t$ computed from the admittance model (17) is sent to the robot and used as a reference signal for inverse dynamics position control in Cartesian space [6]. During the experiments, the orientation of the EE was kept fixed. The adopted framework is schematized in Fig. 4.

Finally we consider the values in (12) for control weight R and the precision matrix $\boldsymbol{Q} = \text{diag}(\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3)$, given $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathbb{R}^3$ and $c_3 \in \mathbb{R}$. The initial values were selected based on the specific path $\boldsymbol{\mu}$ needed for the task. Specifically, we employed an inverse LQR approach as in [32] for computing first attempt values.

B. Center-reaching task

In Section III we made some theoretical considerations about the behaviour of the GN algorithm close to EDS. To validate so, an experiment consisting in moving the EE towards the center of the osculating circle $\boldsymbol{\gamma}(\hat{s})$ has been proposed. The performed task refers to Fig. 5a.

During the experiments three users were involved, two of them with no prior expertise in robotics. Each execution was recorded for approximately one minute. The numerical values chosen for the precision matrix $\boldsymbol{Q}$ were $\boldsymbol{c}_1 = 47.8 \cdot \mathbf{1}_3$, $\boldsymbol{c}_2 = 0.02 \cdot \mathbf{1}_3$, $c_3 = 0.01$ and the control weight was set to $R = 1e-5$. The constraint curve $\boldsymbol{\mu}$ was obtained by kinesthetic demonstration over the curve of Fig. 5a.

In Fig. 5 the red and blue colors refer to the GN and LQT algorithms, respectively. Figure 5b plots the recorded

EE position $\boldsymbol{x}_{GN}$ and $\boldsymbol{x}_{LQT}$ extrapolated from an interval of 6s; Fig. 5c plots the variations of the force modulus $|\boldsymbol{F}|$ and its argument $\angle\boldsymbol{F}$; finally, Fig. 5d- 5e summarise the statistics of all the task executions in terms of the rate of change of the force modulus and argument, denoted with $d|\boldsymbol{F}|/dt$ and $d\angle\boldsymbol{F}/dt$.

The experiments underlined that the proximity to the center of the osculating circle $\boldsymbol{\gamma}(\hat{s})$ induces abrupt directional changes when using the GN algorithm. Precisely, Fig. 5b demonstrates that reaching the center of $\boldsymbol{\gamma}(\hat{s})$ is not problematic in the LQT case, while it becomes complicated for the GN case. A major explanation of this phenomena is given in Fig. 5c. Despite the modulus of the force $|\boldsymbol{F}|$ being in the same range for both GN and LQT, the former displays sharp variations for the argument $\angle\boldsymbol{F}$, induced by the proximity with an EDS. This behaviour was observed throughout all the center-reaching task as revealed in the boxplots of Fig. 5d- 5e. On the left, one can observe that the rate of change of $|\boldsymbol{F}|$ is similar for both algorithms while, on the right, the GN case exhibits larger variations of the rate of change of $\angle\boldsymbol{F}$.

C. Target following task

In this experiment, the objective is to ensure that the EE's tip remains close to the moving point displayed on a second PC, as shown in Fig. 6a. The aim is to prove that the LQT algorithm proposed in 1 is applicable to a real-time human-robot interaction scenario, and compare it with the GN, VM and GC performances. The VM methodology for phase calculation was introduced in [4]. The authors consider a virtual mechanism (VM) connected to the end effector of the robot and characterized by a spring-damper system, namely:

$$\boldsymbol{F}_t = \boldsymbol{K}(\boldsymbol{x}_t - \boldsymbol{\mu}(s_t)) + \boldsymbol{B}(\dot{\boldsymbol{x}}_t - \dot{\boldsymbol{\mu}}(s_t)), \qquad (18)$$

where $\boldsymbol{K}$ and $\boldsymbol{B}$ are chosen to be the same as in (17). The force exterted in the VM is always orthogonal to its velocity, that is $\boldsymbol{\mu}'(s_t)^\top \boldsymbol{F}_t = 0$. By plugging (18) into this condition, one obtains:

$$\boldsymbol{\mu}'(s_t)^\top \big(\boldsymbol{K}(\boldsymbol{x}_t - \boldsymbol{\mu}(s_t)) + \boldsymbol{B}(\dot{\boldsymbol{x}}_t - \boldsymbol{\mu}'(s_t)\dot{s}_t)\big) = 0, \quad (19)$$

from which the following update law holds

$$\dot{s}_t = \big(\boldsymbol{\mu}'(s_t)^\top \boldsymbol{B}\boldsymbol{\mu}'(s_t)\big)^{-1}\boldsymbol{\mu}'(s_t)^\top \big(\boldsymbol{K}(\boldsymbol{x}_t - \boldsymbol{\mu}(s_t)) + \boldsymbol{B}\dot{\boldsymbol{x}}_t\big). \quad (20)$$

During the experiments ten users were involved, four of them with no prior expertise in robotics. Every user had to execute the target following task eight times, alternating the right and left hand for each of the three tested algorithms. Each task execution lasted 50 seconds. The numerical values chosen for the precision matrix $\boldsymbol{Q}$ were $\boldsymbol{c}_1 = 400.0\cdot\boldsymbol{1}_3$, $\boldsymbol{c}_2 = 0.14\cdot\boldsymbol{1}_3$, with $\boldsymbol{1}_3 = [1,1,1]^\top$, $c_3 = 0.01$ and the control weight was set to $R = 1e{-}5$. The constraint curve $\boldsymbol{\mu}$ was obtained by kinesthetic demonstration over the second PC monitor.

The results are summarized in Fig. 6. Figure 6b plots the recorded EE positions $\boldsymbol{x} = [x_t, y_t, z_t]^\top$ for the GN (red), LQT (blue), VM (green) and GC (violet) cases,

(a)

(b)

(c)

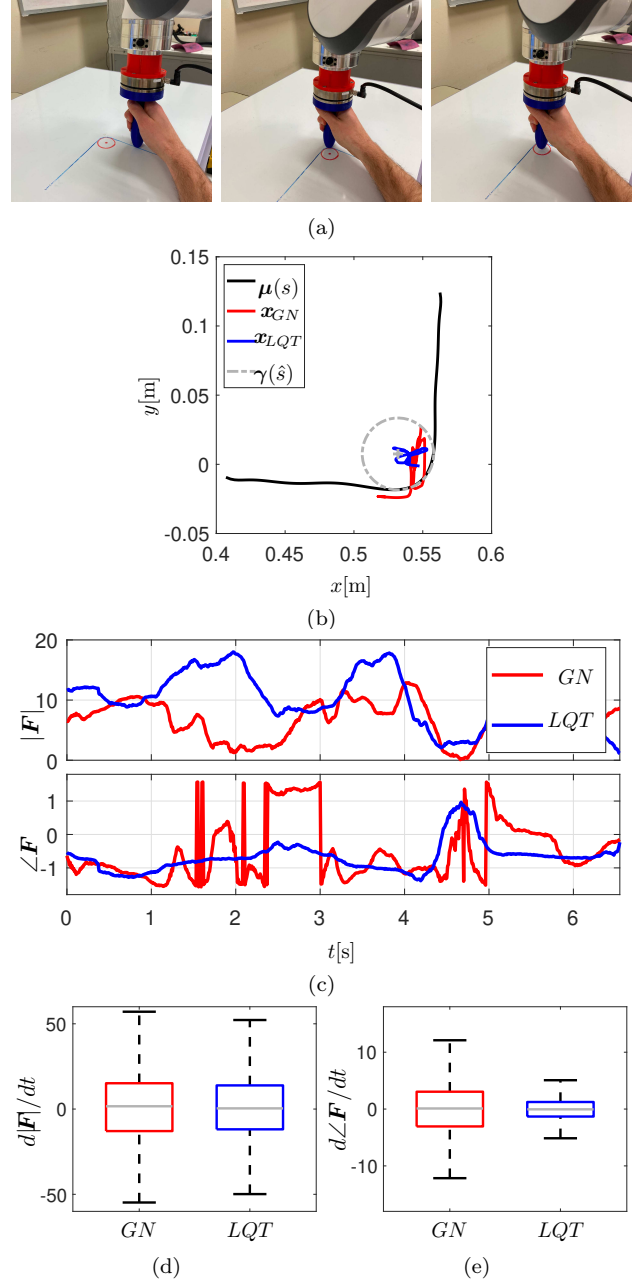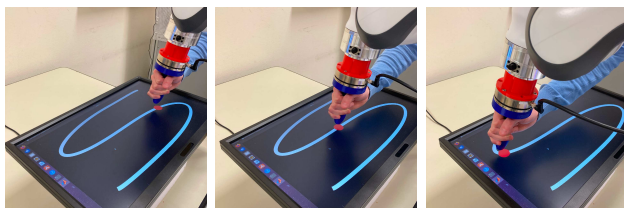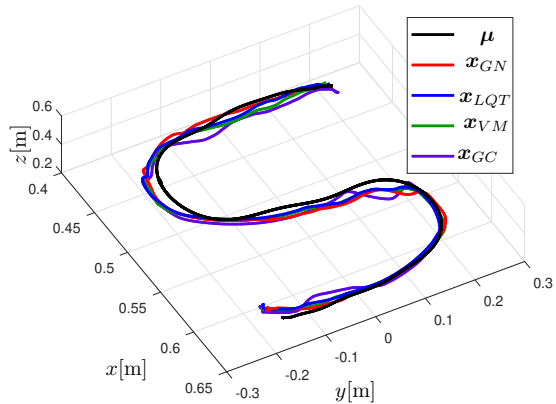(d)   (e)

Figure 5: Experimental results of the center-reaching task.

while the constraint curve $\boldsymbol{\mu} = \boldsymbol{\mu}(s)$ is depicted in black. Given the target position $\boldsymbol{\mu}_t = [\boldsymbol{\mu}_{p,t}, \boldsymbol{\mu}_{y,t}, \boldsymbol{\mu}_{z,t}]^\top$, it is straightforward to see that in the four cases the robot succeeded to follow $\boldsymbol{\mu}_t$ with a limited error $\boldsymbol{e}$. Numerical values for the term $||\boldsymbol{e}|| = ||\boldsymbol{x} - \boldsymbol{\mu}(s)||$ are resumed in Table I with their mean and standard deviation.
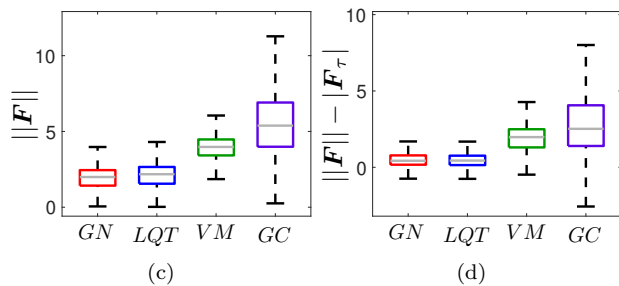
Interesting results come when analyzing the measurements of the forces applied to the end effector $\boldsymbol{F} = [F_x, F_y, F_z]^\top$. It is reasonable to assume that the virtual constraint $\boldsymbol{\mu}$ simplifies the target following task, as it guides the robot EE along with point movements. For the GC case, the user is not constrained to the curve $\boldsymbol{\mu}$, making it more difficult to perform the task. The outcome, on average, is a significant larger effort required for the GC case, described in Fig. 6c with the norm of the
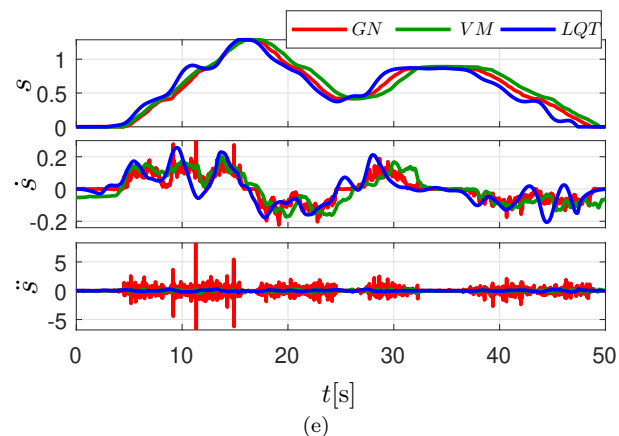
(a)

Table I
Errors between EE and target position and DSJ.

	$||e||[\text{cm}]$	$DSJ(s)$	$DSJ(\ddot{\boldsymbol{x}})$
GN	2.5 ± 1.16	$(1.02\pm3.446)e{+}14$	$(5.44\pm5.570)e{+}16$
LQT	2.3 ± 1.31	$\mathbf{(3.61\pm5.771)e{+}9}$	$\mathbf{(7.40\pm0.071)e{+}15}$
VM	2.3 ± 1.26	$(1.22\pm3.446)e{+}11$	$(1.53\pm1.129)e{+}16$
GC	2.2 ± 1.14	$\times$	$(1.18\pm1.347)e{+}17$

rable, differences emerged when comparing the generated trajectories. Indeed, the GN algorithm computes a velocity command with no smoothness cost on $\dot{s}$ and $\ddot{s}$, which affect the reference trajectories $\boldsymbol{\mu}(s_t)$, $\dot{\boldsymbol{\mu}}(s_t)$ and $\ddot{\boldsymbol{\mu}}(s_t)$ (see (4)). Conversely, the VM algorithm in (20) showed slight vibration, especially in regions with high curvature. This may be attributed to the choice of $\boldsymbol{B}$ and $\boldsymbol{K}$ parameters in (17), which is critical for human-in-the-loop applications [13], [33]. This is exemplified by the experimental measurements reported in Fig. 6e. Here we can appreciate that the LQT algorithm, by minimizing a jerk command, computes smoother derivatives $\dot{s}$ and $\ddot{s}$, demonstrating robustness against the difficulties encountered in the GN and VM cases. The smoothness has been quantitatively evaluated with the dimensionless squared jerk (DSJ) [30]:

$$DSJ(s) = \tau \sum_{t=t_0}^{T} \dddot{s}_t^2, \quad DSJ(\ddot{\boldsymbol{x}}) = \tau \sum_{t=t_0}^{T} ||\dddot{\boldsymbol{x}}_t||^2, \quad (21)$$

with $\tau = (T-t_0)^5/L^2$. Parameters $T = 50s$ and $L = 3.5m$ represent the task duration and the path length, respectively. The metric in 21 was applied both to the phase variable s computed by the algorithms (absent for the GC case), and the actual end effector position $\boldsymbol{x}$. To reduce noise from computing derivatives, data for $DSJ(\ddot{\boldsymbol{x}})$ were pre-processed using a moving average filter with a window size of $w = 20$. The results in Table I show an improvement in smoothness with the proposed LQT algorithm.

D. Discussion

Results from the target following task confirm the previous statement, with higher force demand when no virtual constraint is implemented. Indeed, this increases cognitive stress for the user, who must track the target point while ensuring the EE follows the specified curve. Results also highlights that incorporating the proposed LQT framework has the advantage, over a GN and VM implementations, of generating smoother trajectories as evinced by the results in Fig. 6e and Table I. By specifying the precision matrix $\boldsymbol{Q}$ and the control weight R, one can robustly contain unwanted vibrations by minimizing a jerk command and guaranteed a smooth execution of the proposed guiding tasks [25], [26].

Nevertheless, a high cost on the jerk command can reduce the algorithm tracking precision, as it penalizes high acceleration profiles [31]. In the target following task, this effect could alter the perceived inertia of the robot when moving along the task curve. Although lower accelerations may impact the robot's responsiveness [22], we demonstrate in the center-reaching task the importance of imposing a smoothness cost. Precisely, when reaching

Figure 6: Experimental results of the target following task.

acquired force measurements. If we define $F_{\tau,t} = \boldsymbol{\mu}'_t \cdot \boldsymbol{F}_t$ as the force component $\boldsymbol{F}_t$ projected to the tangential direction of the curve $\boldsymbol{\mu}'_t = \partial\boldsymbol{\mu}(s_t)/\partial s_t$, the difference $||\boldsymbol{F}_t|| - |\boldsymbol{F}_{\tau,t}|$ reported in Fig. 6d quantifies the amount of force which is spent in holding the EE close to the curve $\boldsymbol{\mu}$. Again, as the GN, LQT and VM algorithms impose a virtual constrain, they exhibit substantial lower effort requirements if compared to the GC case, which makes the GC less suitable for this kind of application.

While GN, LQT and VM performances appear compa-

EDSs, the solution of the optimization cost in (5) degenerates [21]. Using the GN algorithm, experiments underlined the presence of sharp changes in the force direction exerted to the EE. This is caused by the generation of abrupt velocity commands in (7), as opposed to the proposed algorithm which, instead, successfully prevents this condition.

V. Conclusion

We proposed an approach based on a linear quadratic tracking (LQT) algorithm to regulate the position of the end effector (EE) of the robot along a virtual constraint. Our approach allows to generate smooth trajectories, avoiding abrupt changes in the derivatives of the EE's reference position. Results demonstrate that our approach is robust against the proximity to Euclidean Distance Singularities (EDS), which is not the case for a solution based on a Gauss-Newton (GN) framework. In the experimental evaluation, the proposed LQT algorithm was compared with GN, a virtual mechanism (VM), and a gravity compensation (GC) guidance, displaying higher reliability in the LQT case. In future works, we plan to extend our methodology to a user study, incorporating subject-related metrics to better evaluate the user perspective.

References

[1] L. B. Rosenberg *et al.*, "The use of virtual fixtures to enhance operator performance in time delayed teleoperation," *Journal of Dynamic Systems Control*, vol. 49, pp. 29–36, 1993.

[2] M. Selvaggio, G. A. Fontanelli, F. Ficuciello, L. Villani, and B. Siciliano, "Passive virtual fixtures adaptation in minimally invasive robotic surgery," *IEEE Robot. Automat. Lett.*, vol. 3, pp. 3129–3136, 2018.

[3] S. A. Bowyer, B. L. Davies, and F. R. y Baena, "Active constraints/virtual fixtures: A survey," *IEEE Trans. Robot.*, vol. 30, pp. 138–157, 2013.

[4] G. Raiola, X. Lamy, and F. Stulp, "Co-manipulation with multiple probabilistic virtual guides," in *2015 IEEE/RSJ IROS*, 2015, pp. 7–13.

[5] F. Amirabdollahian, R. Loureiro, and W. Harwin, "Minimum jerk trajectory control for rehabilitation and haptic applications," in *Proceedings 2002 IEEE ICRA*, vol. 4, 2002, pp. 3380–3385.

[6] L. Sciavicco, B. Siciliano, L. Villani, and G. Oriolo, *Robotics: Modelling, planning and Control, ser. Advanced Textbooks in Control and Signal Processing.* Berlin, Germany: Springer-Verlag, 2011.

[7] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. on Automatic Control*, vol. 54, pp. 2318–2327, 2009.

[8] C. G. L. Bianco and F. Ghilardelli, "A scaling algorithm for the generation of jerk-limited trajectories in the operational space," *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 284–295, 2017.

[9] G. Braglia, M. Tagliavini, F. Pini, and L. Biagiotti, "Online motion planning for safe human–robot cooperation using b-splines and hidden markov models," *Robotics*, vol. 12, 2023.

[10] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, pp. 328–373, 2013.

[11] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings 2002 IEEE ICRA*, vol. 2, 2002, pp. 1398–1403.

[12] A. Sidiropoulos and Z. Doulgeri, "A reversible dynamic movement primitive formulation," in *2021 IEEE ICRA*, 2021, pp. 3147–3153.

[13] G. Braglia, D. Tebaldi, and L. Biagiotti, "Phase-free dynamic movement primitives applied to kinesthetic guidance in robotic co-manipulation tasks," *arXiv:2401.08238*, 2024.

[14] P. Reynoso-Mora, W. Chen, and M. Tomizuka, "On the time-optimal trajectory planning and control of robotic manipulators along predefined paths," in *2013 American Control Conference*, 2013, pp. 371–377.

[15] H. Girgin, T. Löw, T. Xue, and S. Calinon, "Projection-based first-order constrained optimization solver for robotics," *arXiv:2306.17611*, 2023.

[16] B. Bischof, T. Glück, and A. Kugi, "Combined path following and compliance control for fully actuated rigid body systems in 3-d space," *IEEE Trans. on Control Systems Technology*, vol. 25, pp. 1750–1760, 2016.

[17] L. Zhang, S. Guo, and Q. Sun, "An assist-as-needed controller for passive, assistant, active, and resistive robot-aided rehabilitation training of the upper extremity," *Applied Sciences*, vol. 11, p. 340, 2020.

[18] Y. Wang, "Gauss–newton method," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, pp. 415–420, 2012.

[19] O. Khatib, M. Jorda, J. Park, L. Sentis, and S.-Y. Chung, "Constraint-consistent task-oriented whole-body robot formulation: Task, posture, constraints, multiple contacts, and balance," *The International Journal of Robotics Research*, vol. 41, pp. 1079–1098, 2022.

[20] "Robotics codes from scratch (RCFS)," https://rcfs.ch/, Accessed: 2024.

[21] D. G. Luenberger, Y. Ye *et al.*, *Linear and nonlinear programming.* Springer, 1984, vol. 2.

[22] A. Palleschi, M. Garabini, D. Caporale, and L. Pallottino, "Time-optimal path tracking for jerk controlled robots," *IEEE Robot. Automat. Lett.*, vol. 4, pp. 3932–3939, 2019.

[23] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," *IEEE Trans. on industrial electronics*, vol. 47, pp. 140–149, 2000.

[24] A. Biess, D. G. Liebermann, and T. Flash, "A computational model for redundant human three-dimensional pointing movements: integration of independent spatial and temporal motor plans simplifies movement dynamics," *Journal of Neuroscience*, vol. 27, pp. 13 045–13 064, 2007.

[25] A.-N. Sharkawy, "Minimum jerk trajectory generation for straight and curved movements: Mathematical analysis," *arXiv:2102.07459*, 2021.

[26] E. Todorov and M. I. Jordan, "Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements," *Journal of Neurophysiology*, vol. 80, pp. 696–714, 1998.

[27] S. Calinon, "Mixture models for the analysis, edition, and synthesis of continuous time series," *Mixture Models and Applications*, pp. 39–57, 2020.

[28] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots.* Springer Science & Business Media, 2008.

[29] V. A. Toponogov, *Differential geometry of curves and surfaces.* Springer, 2006.

[30] H. Chen, W. Xu, W. Guo, and X. Sheng, "Variable admittance control using velocity-curvature patterns to enhance physical human-robot interaction," *IEEE Robot. Automat. Lett.*, 2024.

[31] Y. Meirovitch, D. Bennequin, and T. Flash, "Geometrical invariance and smoothness maximization for task-space movement generation," *IEEE Trans. Robot.*, vol. 32, pp. 837–853, 2016.

[32] M. C. Priess, R. Conway, J. Choi, J. M. Popovich, and C. Radcliffe, "Solutions to the inverse lqr problem with application to biological systems analysis," *IEEE Trans. on Control Systems Technology*, vol. 23, pp. 770–777, 2014.

[33] F. Ferraguti, C. Talignani Landi, L. Sabattini, M. Bonfe, C. Fantuzzi, and C. Secchi, "A variable admittance control strategy for stable physical human–robot interaction," *The International Journal of Robotics Research*, vol. 38, pp. 747–765, 2019.