A Single-Step Optimization-Based Kinematic Controller for Real-Time Path Tracking of Legged Robots

by Deniz, N.N. and Cheein, F.A.A.

Deniz, N.N. and Cheein, F.A.A. (2025) 'A Single-Step Optimization-Based Kinematic Controller for Real-Time Path Tracking of Legged Robots' *IEEE Transactions on Industrial Informatics*, 22(2), pp. 729-739.

Deniz *et al.*: A Single-Step Optimization-Based Kinematic Controller for Real-Time Path Tracking of Legged Robots

A Single-Step Optimization-Based Kinematic Controller for Real-Time Path Tracking of Legged Robots

Nestor N. Deniz[1] and Fernando A. Auat Cheein[2], *IEEE Senior Member*

***Abstract*—Controlling the locomotion of autonomous legged robots requires sophisticated techniques to handle system constraints, often resulting in high computational demands that limit real-time implementation. This paper presents a computationally efficient path tracking algorithm for legged robots, leveraging their kinematic model and built-in low-level motor control. Unlike traditional Model Predictive Control (MPC), the proposed method simplifies the framework by using a single-step prediction window, achieving performance comparable to MPC with longer horizons while significantly reducing computational overhead. Comparative analysis against a Nonlinear MPC with and without integral mode, a Linear Time-Varying LQR, and a Reinforcement Learning agent demonstrates that the proposed algorithm achieves similar or superior tracking performance with much lower computational requirements and tuning. This makes the approach well-suited for real-time applications in resource-constrained robotic systems. Additionally, the method facilitates fast prototyping for industrial applications, as performance and stability can be established using simple kinematic and path parameters. Our controller simplifies design requiring almost no tuning obtaining a similar performance in comparison with an NMPC with a control horizon of 20 sampling instant but with much less computational effort. Implemented algorithms including trained RL agent can be found here at https://github.com/nahueldeniz/A-Single-Step-Optimization-Based-Kinematic-Controller-for-Real-Time-Path-Tracking-of-Legged-Robots.git.**

***Index Terms*—Legged robots; Path tracking; Optimization-based controller.**

I. Introduction

LOCOMOTION mechanisms, particularly legged robots, have witnessed a surge in popularity and accessibility, allowing legged robots to manoeuvre through intricate scenarios with greater ease compared to their wheeled counterparts. Contemporary legged robots can be controlled at a high level, leveraging pre-existing low-level functions within their software to translate speed commands into motor torques. This capability expedites the prototyping and implementation of path-following and trajectory-tracking for autonomous navigation, particularly in industrial applications.

Manuscript received: September, 12, 2024; Revised July, 22, 2025; Accepted August, 25, 2025.

This paper was recommended for publication by Co-Editor-in-Chief Prof. Shen Yin upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Advaced Center for Electrical and Electronic Engineering (AC3E) from Chile.

[1]Nestor N. Deniz is with the Instituto de Investigación en Senales, Sistemas e Inteligencia Computacional, sinc(i), UNL-CONICET, Santa Fe, Argentina ndeniz@sinc.unl.edu.ar

[2]Fernando A. Auat Cheein is with the Department of Electronic Engineering, Harper Adams University, Newport, Shropshire TF10 8NB, UK. FAuat@harper-adams.ac.uk

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

Fig. 1: Legged robot Unitree Go1 used for field experiments. The robot is equipped with the following sensors: ① real-time kinematics (RTK) GPS Navcom SF-3040; ② inertial measurement unit (IMU) Vectornav VN200S.

In this work, we introduce an optimisation-based quasi algebraic controller (OBQAC) tailored for the class of orthogonal nonlinear control-affine systems, such as the kinematic model of legged robots. Building upon the foundational work outlined in seminal studies by [1] and [2], which introduced an innovative algebraic method for path tracking controllers tailored to nonholonomic vehicles, our research extends these techniques to encompass constrained controls. We demonstrate its applicability for fast prototyping with legged robots by leveraging pre-existing built-in low-level controllers. Our reformulation leads to an optimisation problem that considers the limitations of the robot's actuators, constraining accelerations to achieve energy-efficient and smooth manoeuvres while maintaining a lower computational burden compared to optimisation-based controllers such as nonlinear model predictive control (NMPC), albeit with comparable performance.

The core ideas of this work are summarized as follows:
i) Section III presents the formulation of a single-step optimization-based controller designed for wheeled and legged robots. This approach leverages the control-affine structure of their kinematic models and their built-in low-level gait-generation controllers (in teh case of legged robots) to achieve the following:

- Low computational burden (suitable for real-time applications).
- No offline training requirements (enabling fast prototyping and industrial deployment).

ii) Moreover, in Section III, global stability of the proposed controller is rigorously guaranteed under mild assumptions on the kinematic model's velocity constraints. Additionally, the section provides:

- An analysis of convergence time to the target in path-tracking tasks.
- A quantitative evaluation of the controller's computational load.

iii) Performance comparisons are conducted through simu-

lations and field experiments across diverse scenarios. The proposed method is benchmarked against:

- Nonlinear Model Predictive Control (NMPC) with and without integral action.
- A Linear Time-Varying (LTV) Linear Quadratic Regulator (LQR).
- A Deep Reinforcement Learning (DRL) method for trajectory tracking.

The remainder of the work is structured as follows: Section II reviews related work on nonlinear control-affine systems and legged robots, including gait generation. In Section III, we specify the types of system to which our approach applies and develop the formulation of the proposed controller within the context of path tracking for quadruped robots. Section IV presents a comparative analysis of our controller and the NMPC with and without integral action and an LTV LQR controller under different scenarios of field experiments. Finally, Section V shows the conclusions of our work.

II. Related Works

This section reviews previous research on path tracking of control-affine systems and provides an overview of low-level control and gait generation for legged robots.

A. Control of legged robots through its kinematic model

Legged robots, including bipedal and quadrupedal systems, require precise foot placement to navigate safely. A major challenge in controlling these robots stems from intermittent foot-terrain contact, which introduces significant complexity into controller design [3], [4]. Nevertheless, considerable progress has been made, and commercially available legged robots now incorporate low-level controllers that generate necessary gaits for locomotion (for details on gait generation, see [5] and references therein). By leveraging these controllers, path tracking can be achieved using the control-affine kinematic model of legged robots. This approach enables rapid industrial solutions, prototyping, trajectory-tracking of non-admissible curves by exploiting the structure of driftless nonlinear control-affine systems while keeping low the computational burden [6]–[8]. The analysis and design of controllers for control-affine systems remain an active research topic, as these systems accurately capture non-linearities while reducing computational complexity compared to full nonlinear models in control problems involving online optimization [9], [10].

Several control strategies leveraging the control-affine structure of mobile robot kinematic models have been proposed for real-time path tracking. For instance, [11] addresses nonlinearities in wheel-terrain interactions for ground vehicles, reducing the computational burden of their predictive controller while ensuring stability via state-of-the-art methods. Subsequent work in [12] linearizes the kinematic model of a mobile robot along its trajectory to derive a linear time-varying system, expressed as a lattice piece-wise affine model. This method handles constraints effectively while maintaining trajectory-tracking performance. Similarly, [13] employs Taylor series expansions to approximate the kinematics of a wheel-legged robot, alleviating computational load and incorporating kinematic/dynamic constraints in a predictive control framework.

Recent publications propose hybrid approaches combining offline training with model-based online optimization, termed data-driven path-following MPC, to mitigate computational burdens [14]. These methods replace state-space models with hybrids of spatial kinematics and data-driven models. Additionally, [15] introduces an approximate online adaptive solution for infinite-horizon optimal tracking in control-affine systems with unknown drift dynamics, using Bellman equations to simulate unexplored state-space regions.

The kinematic model and an affine error structure are utilized in [16] for an interconnected three-element vehicle. The affine error system facilitates Adaptive Dynamic Programming (ADP), resulting in a partially unknown control method. An actor-critic reinforcement learning (RL) neural network (NN) structure provides control policies to achieve accurate trajectory tracking. Continuing with RL methods, [17] employs Deep RL (DRL) to train an agent controlling a six-degree-of-freedom ground vehicle, maintaining a trajectory within half the vehicle's width from reference paths. A similar approach in [18] proposes a DRL-based Adaptive Pure Pursuit (DRAPP) algorithm, combining geometric Pure-Pursuit with a DRL-learned policy and a terrain-adaptation mechanism. In [19], sliding-mode control replaces model predictive control, integrating deep learning models to estimate and compensate for skid-steering robot slippage in real time.

RL-based path planning for skid-steer robots in complex environments is explored in [20], where Q-Learning (QL), Deep Q Networks (DQN), and Deep Deterministic Policy Gradient (DDPG) address challenges such as large maps, high-dimensional states, obstacles, and wheel-terrain interactions under slip conditions.

While various strategies exist for controlling legged and mobile robots, model-based approaches like model predictive control remain challenging for real-time deployment on mobile platforms. Machine learning techniques reduce online computation but require extensive offline training, hyper-parameter tuning, and large datasets. This work leverages built-in low-level gait controllers and proposes a single-step-optimization-based kinematic controller for real-time path tracking. By exploiting the control-affine structure of mobile robots (like the one shown in Figure 1), the method facilitates rapid prototyping and industrial solutions with minimal computational load and parameter tuning while the stability is guaranteed under mild assumptions.

III. Problem Statement

This section introduces the foundational elements required to formulate the proposed controller. The optimisation-based quasi-algebraic controller (OBQAC) extends the earlier algebraic controller proposed in [1] for wheeled vehicles, which was subsequently adapted for aerial vehicles in [2]. In the following, the kinematic model of the legged robot shown in Figure 1, characterized by a control-affine structure, will be presented. Additionally, the path tracking problem will be briefly outlined. The formulation of the OBQAC, including

the design of the control set, its application to legged robots, and an analysis of error convergence, will then be detailed.

A. Kinematic model of the legged robot

In this subsection, we present the kinematic model of the legged robot depicted in Figure 1. The discrete-time approximation of this kinematic model is given as follows:

$$\begin{aligned} q_{k+1} = &\quad q_k + \mathcal{F}(q_k) u_k T_s. \\ q_{k+1} = &\quad q_k + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_k) & -\sin(\theta_k) & 0 \\ 0 & \sin(\theta_k) & \cos(\theta_k) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_{z,\,k} \\ v_{x,\,k} \\ v_{y,\,k} \\ v_{z,\,k} \end{pmatrix} T_s. \end{aligned} \tag{1}$$

The state is defined by the robot's pose as $q_k = (\theta_k,\, x_k,\, y_k,\, z_k)^T \in \Re^{n_q}$, where θ_k denotes the robot orientation in the global (navigation) frame, and x_k, y_k and z_k are the robot's coordinate also in the global frame. The control vector is denoted as $u_k = (\omega_{z,k},\, v_{x,k},\, v_{y,k},\, v_{z,k})^T$, with $\omega_{z,k}$ indicating the angular speed with respect to the local z-axe of the robot, such that $(\theta_{k+1} - \theta_k)/T_s = \omega_{z,k}$. The velocities $v_{x,k}$, $v_{y,k}$ and $v_{z,k}$ are measured in the global frame and are aligned with the local x, y and z robot axes.

B. path tracking problem

In this subsection, the trajectory tracking of mobile platforms will be briefly described. For more details, the reader is referred to [21] and references therein. In the following, it will be assumed that the trajectory to be tracked lies in the plane $x - y$, with the z coordinate representing the legged robot's height relative to the $x - y$ plane. Furthermore, the trajectory is assumed to be compatible with the vehicle's kinematics and is defined by a parametric equation, as follows:

$$\mathcal{P}_k := \quad (\mathbf{p}_{\theta,k}, \mathbf{p}_{x,k}, \mathbf{p}_{y,k}, \mathbf{p}_{z,k})^T. \tag{2}$$

where $\mathbf{p}_{\theta,k}$ is the reference attitude at time k, $\mathbf{p}_{x,k}$ and $\mathbf{p}_{y,k}$ are the reference coordinates in the $x - y$ plane at time k, and $\mathbf{p}_{z,k}$ is the reference for the robot's altitude with respect to the $x - y$ plane at time k. At each time $k \in \mathbb{Z}_{\geq 0}$, a reference pose $\mathcal{P}_k$ for the robot is defined. Note moreover that $\mathbf{p}_{\theta,k} = \tan\big((\mathbf{p}y, k - \mathbf{p}y, k-1)/(\mathbf{p}x, k - \mathbf{p}x, k-1)\big)^{-1}$ $\forall, k \in \mathbb{Z}_{\geq} 0$. The path tracking objective is to minimise the difference between $\mathcal{P}_k$ and the robot's pose q_k at every time $k \in \mathbb{Z}_{\geq 0}$. In the following, it will be assumed that the speed and accelerations of the $\mathcal{P}_k$ is bounded in the following sense:

$$\begin{aligned} \tfrac{|\mathbf{p}_{\theta,k+1} - \mathbf{p}_{\theta,k}|}{T_s} &= |\nu_{r_\theta,k+1}| \leq \quad |V_\theta| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|\mathbf{p}_{x,k+1} - \mathbf{p}_{x,k}|}{T_s} &= |\nu_{r_x,k+1}| \leq \quad |V_x| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|\mathbf{p}_{y,k+1} - \mathbf{p}_{y,k}|}{T_s} &= |\nu_{r_y,k+1}| \leq \quad |V_y| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|\mathbf{p}_{z,k+1} - \mathbf{p}_{z,k}|}{T_s} &= |\nu_{r_z,k+1}| \leq \quad |V_z| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \end{aligned} \tag{3}$$

$$\begin{aligned} \tfrac{|\nu_{r_\theta,k+1} - \nu_{r_\theta,k}|}{T_s} &= |a_{r_\theta,k+1}| \leq \quad |A_\theta| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|\nu_{r_x,k+1} - \nu_{r_x,k}|}{T_s} &= |a_{r_x,k+1}| \leq \quad |A_x| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|\nu_{r_y,k+1} - \nu_{r_y,k}|}{T_s} &= |a_{r_y,k+1}| \leq \quad |A_y| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|\nu_{r_z,k+1} - \nu_{r_z,k}|}{T_s} &= |a_{r_z,k+1}| \leq \quad |A_z| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \end{aligned} \tag{4}$$

where $\nu_{r_\theta,k+1}$, $\nu_{r_x,k+1}$, $\nu_{r_y,k+1}$ and $\nu_{r_z,k+1}$ is the rate of change of $\mathcal{P}_{k+1}$, and V_x, V_y, V_z and $V_\theta \in \Re$ are their maximum values. Moreover, $a_{r_\theta,k+1}$, $a_{r_x,k+1}$, $a_{r_y,k+1}$ and $a_{r_z,k+1}$ are the accelerations of every component of the reference $\mathcal{P}_{k+1}$, assumed to be bounded by A_θ, A_x, A_y and A_z, respectively. In the sequel, the robot velocities are assumed to be bounded as follows:

$$|\omega_{z,k}| \leq |\mathcal{V}_\theta|, \quad |v_{x,k}| \leq |\mathcal{V}_x|, \quad |v_{y,k}| \leq |\mathcal{V}_y|, \quad |v_{z,k}| \leq |\mathcal{V}_z|. \tag{5}$$

where $|\mathcal{V}_\theta| > |V_\theta|$, $|\mathcal{V}_x| > |V_x|$, $|\mathcal{V}_y| > |V_y|$, $|\mathcal{V}_z| > |V_z|$ and $\mathcal{V}_M = (\mathcal{V}_x^2 + \mathcal{V}_y^2 + \mathcal{V}_z^2)^{1/2}$. The robot accelerations are bounded as follows:

$$\begin{aligned} \tfrac{|\omega_{z,k+1} - \omega_{z,k}|}{T_s} &= |a_{\theta,k+1}| \leq |\mathcal{A}_\theta| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|v_{x,k+1} - v_{x,k}|}{T_s} &= |a_{x,k+1}| \leq |\mathcal{A}_x| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|v_{y,k+1} - v_{y,k}|}{T_s} &= |a_{y,k+1}| \leq |\mathcal{A}_y| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \\ \tfrac{|v_{z,k+1} - v_{z,k}|}{T_s} &= |a_{z,k+1}| \leq |\mathcal{A}_z| < \infty \;\forall\, k \in \mathbb{Z}_{\geq 0}. \end{aligned} \tag{6}$$

where $a_{\theta,k+1}$, $a_{x,k+1}$, $a_{y,k+1}$ and $a_{z,k+1}$ are the legged robot accelerations at time $k + 1$. Moreover, $|\mathcal{A}_\theta| > |A_\theta|$, $|\mathcal{A}_x| > |A_x|$, $|\mathcal{A}_y| > |A_y|$, $|\mathcal{A}_z| > |A_z|$ and $\mathcal{A}_M = (\mathcal{A}_x^2 + \mathcal{A}_y^2 + \mathcal{A}_z^2)^{1/2}$.

C. Single step optimisation based kinematic controller

Building on the control law previously reported in [1], in this section, we introduce a modification to that control law such that the controls u_k remain inside the bounded set $\mathcal{U}_c \subset \mathcal{U}_u \subseteq \Re^{n_u}$, where $\mathcal{U}_c$ is defined as a box set, such that $\underline{\omega} \leq \omega_{z,k} \leq \overline{\omega}$, $\underline{v}_x \leq v_{x,k} \leq \overline{v}_x$, $\underline{v}_y \leq v_{y,k} \leq \overline{v}_y$ and $\underline{v}_z \leq v_{z,k} \leq \overline{v}_z$, such that $\mathcal{U}_c := [\underline{v}_\omega, \overline{v}_\omega] \times [\underline{v}_x, \overline{v}_x] \times [\underline{v}_y, \overline{v}_y] \times [\underline{v}_z, \overline{v}_z]$, and $\mathcal{U}_u$ is the set of unconstrained controls. Note from Eq. (1) that replacing q_{k+1} with $\mathcal{P}_{k+1}$, the control u_k that steers the state q_k to $\mathcal{P}_{k+1}$ can be easily computed. Let us define $\Delta_{q,k} := \mathcal{P}_{k+1} - q_k = (\Delta_{\theta,k},\, \Delta_{x,k},\, \Delta_{y,k},\, \Delta_{z,k})^T$, with $\Delta_{\theta,k} = \mathbf{p}_{\theta,k+1} - \theta_k$, $\Delta_{x,k} = \mathbf{p}_{x,k+1} - x_k$, $\Delta_{y,k} = \mathbf{p}_{y,k+1} - y_k$, and $\Delta_{z,k} = \mathbf{p}_{z,k+1} - z_k$. Then, solving Eq. (1) for the control u_k gives the following:

$$\mathcal{F}(q_k)^T \tfrac{\Delta_{q,k}}{T_s} = \quad u_k.$$

where $\mathcal{F}^T$ is the inverse of $\mathcal{F}(q_k)$ since it is orthogonal, and $u_k \in \mathcal{U}_u$ is the unconstrained control that steers the state q_k to the reference $\mathcal{P}_{k+1}$. However, we aim to constrain u_k to $\mathcal{U}_c$. To achieve this, one could start from u_{k-1} and add the components $\Delta u = T_s, (a_{\theta,k},\, a_{x,k},\, a_{y,k},\, a_{z,k})^T$ (velocities of the control signal) as required in order to approach u_k while satisfying the constraints. To formalize this, let us define the optimization variable $w \in \Re^{n_u}$ and the objective function: $\mathbb{J} := w^T W w$, with $W \succ 0$. Then, the constrained controls u_k can be computed by solving the following optimisation problem:

$$\min_{\mathrm{w}} \mathbb{J}$$
$$\text{s.t.} \begin{cases} w = & \mathcal{F}(q_k)^T \tfrac{\Delta_{q,k}}{T_s} - u_k. \\ u_k = & u_{k-1} + \Delta u. \\ u \in & \mathcal{U}_c. \end{cases} \tag{7}$$

The control signal u_k obtained through the formulation given by Eq. (7) remains within the set $\mathcal{U}_c$, ensuring that the manoeuvres are compatible with the legged robot. Moreover, the control signal u_k minimises the distance to $\mathcal{F}(q_k)^T \Delta_{q,k}/T_s$ instead of computing a control based on minimising the distance $|\mathcal{P}_k - q_k|$. This is a key difference when compared to common MPC strategies, where the robot may not move when it starts from an ill initial condition, the controller is not well-tuned, and/or the control horizon is not large enough. Additionally, these controls computed using the formulation given by Eq. (7) adhere to acceleration limits, resulting in smoother and more energy-efficient movements, as will be shown later in Section IV, Experiments and Results.

D. Global stability

The global stability can be proven under the assumptions made in the former section. Let us define the error at time k as $e_k = \mathcal{P}_k - q_k$ and the difference of velocities between the reference and the legged robot as $s_k = \nu_{r,k} - u_k$, such that the error dynamics can be written as follows:

$$\begin{aligned} e_{k+1} &= e_k + T_s\, s_k. \\ &= e_k + T_s(\nu_{r,k} - u_k). \end{aligned} \tag{8}$$

Then, to achieve $e_{k+1} = 0$, the control u_k obtained by solving Eq. (8) needs to be applied:

$$u_k = \frac{e_k}{T_s} + \nu_{r,k}. \tag{9}$$

The constrained control can be as large as $\nu_{r,k}$ by assumption. However, the value of u_k obtained from Eq. (9) may not belong to the set $\mathcal{U}_c$ for large values of e_k. Therefore, let us define the constant $0 < \delta_e \leq 1$ such that $u_k = \delta_e e_k/T_s + \nu_{r,k}$ does belong to $\mathcal{U}_c$. Then, replacing this control back into Eq. (8):

$$\begin{aligned} e_{k+1} &= e_k + T_s\,(\nu_{r,k} - \tfrac{\delta_e e_k}{T_s} - \nu_{r,k}). \\ &= e_k\,(1 - \delta_e). \end{aligned} \tag{10}$$

Since $\delta_e \in (0, 1]$, the error is decreasing $\forall\ e_k \in \Re^{n_q}$. □

E. Convergence time

Due to the constrained nature of the control u_k, $e_k = 0$ cannot be reached in one step. Therefore, it becomes interesting to determine how long it takes for the legged robot to reach the target $\mathcal{P}_k$ given a certain initial condition, path (reference) dynamics, and control set $\mathcal{U}_c$. To achieve this, let us assume that $e_0 \neq 0$, where the error e_i at time i is given as follows:

$$e_i = \sqrt{d_{\theta,i}^2 + d_{s,i}^2}. \tag{11}$$

where:

$$\begin{aligned} d_{\theta,i}^2 &= (\mathbf{p}_{\theta,i} - \theta_i)^2. \\ d_{s,i}^2 &= (\mathbf{p}_{x,i} - x_i)^2 + (\mathbf{p}_{y,i} - y_i)^2 + (\mathbf{p}_{z,i} - z_i)^2 \end{aligned} \tag{12}$$

In addition, let us assume that the legged robot starts from rest while the reference $\mathcal{P}_0$ is already moving at its maximum speed, far away from the robot along a straight line. Moreover, let us assume that $|\theta_0 - \mathbf{p}_{\theta,0}| = \pi$. Assume the strategy for approaching the target is to first rotate π (rad) until the legged robot is pointing toward the target. The legged robot begins turning by accelerating from an initial angular velocity $\omega_0 = 0$. One sampling instant later, it reaches $\omega_1 = T_s\mathcal{A}_\theta$, and after n sampling instants, the angular velocity will be $\omega_n = n\,T_s\mathcal{A}_\theta$. The total angle rotated can be computed as follows:

$$\begin{aligned} \theta_0 &= 0. \\ \theta_1 &= \theta_0 + \omega_1\,T_s. \\ &= T_s^2\mathcal{A}_\theta. \\ \theta_2 &= \theta_1 + \omega_2\,T_s. \\ &= 3\,T_s^2\mathcal{A}_\theta. \\ &\vdots \\ \theta_n &= \tfrac{n(n+1)}{2} T_s^2\mathcal{A}_\theta. \end{aligned} \tag{13}$$

Then, the sampling instants N_U required to turn $\theta_n = \theta_{N_u} = \pi$ (rad) is:

$$N_U = \frac{\sqrt{1 + \frac{8\pi}{T_s^2\mathcal{A}_\theta}}}{2} - 1. \tag{14}$$

During the time the legged robot was turning, the Euclidean distance between the legged robot and the reference grows from $d_{s,0}$ to $s_{s,N_U} = d_{s,0} + N_U T_s V_r$, where $V_r = (V_x^2 + V_y^2 + V_z^2)^{1/2}$. Moreover, the legged robot starts to move linearly, accelerating at a rate $\mathcal{A} = (\mathcal{A}_x^2 + \mathcal{A}_y^2 + \mathcal{A}_z^2)^{1/2}$. After n' sampling times since the robot started to move linearly, the speed of the legged robot is $v_{n'} = n' T_s \mathcal{A}$. By assumption, the legged robot can move faster than the reference. In N_{V_r} sampling times, the legged robot can reach the reference speed such that $N_{V_r} = V_r/(T_s\mathcal{A})$. During this time, the distance between the reference and the legged robot increases at a decreasing rate given by $d_{s,i} = T_s(V_r - iT_s\mathcal{A})$. The total distance between the legged robot and the reference can be computed as follows:

$$d_{s,N_{V_r}} = d_{s,N_U} + T_s \sum_{i=1}^{N_{V_r}} (V_r - i\,T_s\mathcal{A}). \tag{15}$$

At this moment, since the legged robot moves at the same speed as the reference, the distance between the legged robot and the reference will no longer increase. Additionally, it can still increase its speed until it reaches its maximum speed $\mathcal{V}_M$ while reducing the distance $s_{s,N_{V_r}}$. The legged robot reaches its maximum speed after N_{V_M} sampling times, given by $N_{V_M} = (\mathcal{V}_M - V_r)/(T_s\mathcal{A})$, while the distance to the reference continues to decrease. The total distance can be computed as follows:

$$d_{s,N_{V_M}} = d_{s,N_{V_r}} - T_s \sum_{i=1}^{N_{V_M}} (V_r + i\,T_s\mathcal{A}). \tag{16}$$

Assuming that the reference is not reached while the legged robot is approaching its maximum speed, it will be reached

after N_t additional sampling instants while the robot is moving at its maximum speed $\mathcal{V}_M$, such that:

$$\begin{aligned}
0 = \quad & d_{s,N_{V_M}} - T_s \mathcal{V}_M N_t. \\
0 = \quad & d_{s,N_{V_r}} - T_s \sum_{i=1}^{N_{V_M}} (V_r + i\, T_s \mathcal{A}) - T_s \mathcal{V}_M N_t. \\
0 = \quad & d_{s,N_U} + T_s \sum_{i=1}^{N_{V_r}} (V_r - i\, T_s \mathcal{A}) - T_s \sum_{i=1}^{N_{V_M}} (V_r + i\, T_s \mathcal{A}) - \\
& T_s \mathcal{V}_M N_t. \\
0 = \quad & d_{s,0} + N_U T_S V_r + T_s \sum_{i=1}^{N_{V_r}} (V_r - i\, T_s \mathcal{A}) - \\
& T_s \sum_{i=1}^{N_{V_M}} (V_r + i\, T_s \mathcal{A}) - T_s \mathcal{V}_M N_t.
\end{aligned} \tag{17}$$

The value of N_t is given as follows:

$$N_t = \quad \frac{d_{s,0} + N_U T_s V_r + T_s \sum_{i=1}^{N_{V_r}} (V_r - i T_s \mathcal{A}) - T_s \sum_{i=1}^{N_{V_M}} (V_r + i T_s \mathcal{A})}{T_s \mathcal{V}_M} \tag{18}$$

Note that the value of N_t can be negative when the term $T_s \sum_{i=1}^{N_{V_M}} (V_r + i T_s \mathcal{A})$ is predominant. The total sampling times N_{reach} required to reach the reference is:

$$N_{reach} = \quad N_U + N_{V_r} + N_{V_M} + N_t. \tag{19}$$

which is a conservative value and should be taken as an upper bound.

F. Computational complexity analysis

To analyze the computational complexity of the proposed algorithm, let us rewrite the problem given by Eq. (7) as a Mixed Integer Quadratic Program (MIQP) with 4 variables and 8 linear constraints, as follows:

$$\min_{\mathrm{w}} \mathbb{J}$$
$$\text{s.t.} \begin{cases} w = & \mathcal{F}(q_k)^T \frac{\Delta_{q,k}}{T_s} - u_k. \\ u_k = & u_{k-1} + T_s(\delta_1 \mathcal{A}_\theta,\, \delta_2 \mathcal{A}_x,\, \delta_3 \mathcal{A}_y,\, \delta_4 \mathcal{A}_z)^T. \\ Au_k \leq & g. \end{cases} \tag{20}$$

where A and g are a matrix and vector of appropriate dimensions defining the linear constraints such that $u_k \in \mathcal{U}_c$, assuming $\mathcal{U}_c$ is a box-set. Moreover, δ_1, δ_2, δ_3, and δ_4 are discrete variables, each taking values from the set $-1, , 0, , 1$. Solving this problem involves evaluating 3^n possible combinations, where n is the number of variables. Each combination requires evaluating the cost function $w^T W w$, which, for a diagonal matrix W, involves 35 floating-point operations (20 multiplications and 15 additions).

Additionally, for each combination, 8 linear constraints must be checked, with each constraint requiring 8 floating-point operations (4 multiplications, 3 additions, and 1 comparison). Consequently, each combination requires $35 + 8m$ operations, where m is the number of linear constraints. Evaluating all 3^n possible combinations thus requires $3^n \times (35 + 8m) = 3^n \times 35 + 3^n \times 8 \times m$ floating-point operations, amounting to 8019 operations for $n = 4$ and $m = 8$.

This theoretical bare-metal analysis imposes an upper bound on the number of required operations when the problem given by Eq. (7) is solved by brute-forcing through the 81 possible points. Note that replacing the discrete variables δ_1, δ_2, δ_3, and δ_4 with continuous variables that can take values in the range $[-1, , 1]$ reduces the complexity of solving the problem in Eq. (7) to a standard Quadratic Program (QP) instead of a more complex MIQP.

In practice, the problem given by Eq. (7) is solved using CasADi in Matlab with *ipopt*, an open-source primal-dual interior point method. CasADi automatically computes the gradient of the objective function, the Jacobian of the constraints function, and the Hessian of the Lagrangian function using algorithmic differentiation. This approach avoids brute-force evaluation, reducing the problem from an MIQP to a QP with polynomial time complexity.

Algorithm 1 summarises how to apply the proposed controller for achieving path tracking with the legged robot shown in Figure 1.

Algorithm 1: Single-Step Kinematic Controller for Real-Time Path Tracking of Legged Robots

Data: $\mathcal{P}_k$, $\mathcal{U}_c$, e_0, T_s, V, A, $\mathcal{V}$, $\mathcal{A}$, $u_0 = \mathbf{0}$.
Result: N_{reach}
while *!(reach end of path)* **do**
  $\mathcal{P}_{k+1} \leftarrow (\mathbf{p}_{\theta,k+1}, \mathbf{p}_{x,k+1}, \mathbf{p}_{y,k+1}, \mathbf{p}_{z,k+1})^T$
  $q_k \leftarrow (\theta_k, x_k, y_k, z_k)^T$
  $\Delta_{q,k} \leftarrow \mathcal{P}_{k+1} - q_k$
  Solve opt. problem given by Eq. (7)
  $u_k \leftarrow u_{k-1} + \Delta u$
  Apply u_k to the legged robot
end

IV. Experiments and Results

In this section, simulated and field experiments are conducted to highlight the strengths and weaknesses of the proposed method. We will follow the methodology suggested in [22], [23] to evaluate the path tracking performance of outdoor mobile robots. Moreover, we will compare the proposed method against the Linear Time Varying (LTV) Linear Quadratic Regulator (LQR), the nonlinear model predictive controller with different configurations including the one with integral action (iNMPC) reported in [24], and a Deep Reinforcement Learning method reported in [17] and trained for the legged robot shown in Figure 1.

A. Simulated experiments

The simulated experiments are carried out on a PC with 12 Intel cores i7-8700 @ 3.2 (GHz) with 32 (GiB) of RAM running Ubuntu 22.04. These experiments involve following the Lemniscate of Bernoulli path using different control strategies

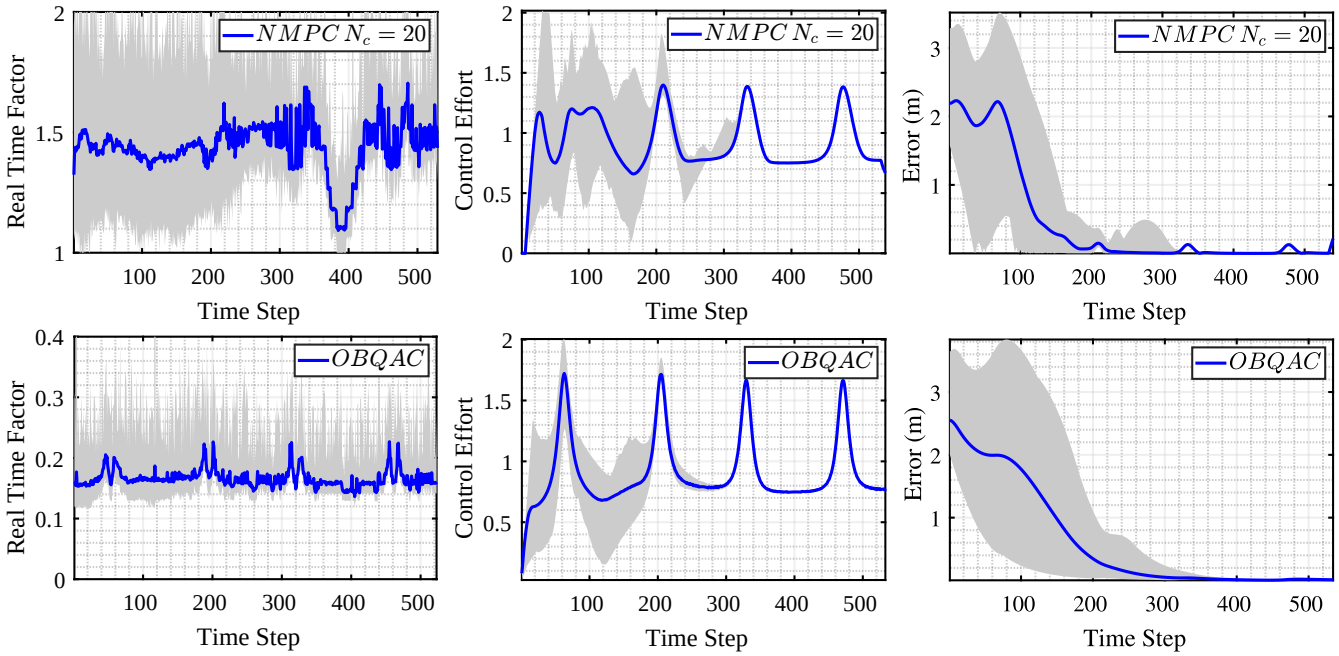

Fig. 2: Real-time factor ψ, control effort Π, and deviation error Ξ for the Nonlinear Model Predictive Controller (NMPC -first row-) with control horizon length $N_c = 20$ and the proposed controller (second row). Both controllers exhibit similar performance; however, the NMPC cannot be solved in real time as its RTF exceeds 1. The horizontal axis represents the duration of the experiment, expressed as the number of sampling instants T_s (s).

to highlight the weaknesses and strengths of each approach. The path $\mathcal{P}_k = (\mathbf{p}_{\theta,k}, \mathbf{p}_{x,k}, \mathbf{p}_{y,k}, \mathbf{p}_{z,k})$ is defined as follows:

$$\begin{cases} \mathbf{p}_{\theta,k} = & \tan((\mathbf{p}_{y,k+1} - \mathbf{p}_{y,k})/(\mathbf{p}_{x,k+1} - \mathbf{p}_{x,k}))^{-1} \\ \mathbf{p}_{x,k} = & \dfrac{1.75\sqrt{2}\cos(\Theta_k)}{\sin(\Theta_k)^2 + 1} \\ \mathbf{p}_{y,k} = & \dfrac{\sqrt{32}\cos(\Theta_k)\sin(\Theta_k)}{\sin(\Theta_k)^2 + 1} \\ \mathbf{p}_{z,k} = & 0.29 \end{cases} \tag{21}$$

where $\Theta_k \in [0, 2\pi]$ is designed such that $((\mathbf{p}_{x,k+1} - \mathbf{p}_{x,k})^2 + (\mathbf{p}_{y,k+1} - \mathbf{p}_{y,k})^2)^{1/2}/T_s = \nu_r$, where ν_r is the speed at which the reference moves. In these experiments, $\nu_r = 0.75$ (m/s). The height of the robot $\mathbf{p}_{z,k}$ is assumed to remain constant at 0.29 (m) throughout the entire path. In order to compare the performance of the different algorithms, we follow the methodology and metrics reported in [22]. These metrics are the control effort Π, the deviation error Ξ, and the real-time factor ψ, defined by the following equations:

$$\begin{aligned} \Pi &= \tfrac{1}{K}\sqrt{\textstyle\sum_{i=1}^{K} \omega_{z,i}^2 + v_{x,i}^2 + v_{y,i}^2 + v_{z,i}^2} \\ \Xi &= \tfrac{1}{K}\sqrt{\textstyle\sum_{i=1}^{K} d_{\theta,i}^2 + d_{s,i}^2} \\ \psi &= \tfrac{1}{T_s K}\textstyle\sum_{i=1}^{K} \xi_i \end{aligned} \tag{22}$$

where ξ_i is the time required for each controller to compute the velocities to be applied to the robot. The real-time factor evaluates the computational feasibility of the algorithm by comparing the time taken to compute control commands with the actual time available (T_s at most) for execution. A real-time factor less than or equal to 1 is essential for ensuring the algorithm can be deployed in real-world applications where delays can compromise safety and performance. The deviation Ξ measures the deviation of the robot's trajectory from the nominal or desired path. It is a direct indicator of the algorithm's accuracy and its ability to achieve precise tracking, which is critical in applications requiring high precision, such as navigation in cluttered environments or performing complex tasks. The values of $d_{\theta,i}^2$ and $d_{s,i}^2$ are computed according to Eq. (12). The control effort Π quantifies the energy or effort required to execute the control commands. Minimizing control effort is crucial in robotics to improve energy efficiency, reduce wear on actuators, and extend the operational life of the robot, especially for legged robots that often operate on limited power sources.

The set $\mathcal{U}_s$ is designed as a box set such that $\mathcal{U}_c = [-5\pi/4, 5\pi/4]$(rad/s) $\times$ $[-5/4, 5/4]$(m/s) $\times$ $[-5/4, 5/4]$(m/s) $\times$ $[-5/4, 5/4]$(m/s). The angular acceleration between sampling times is limited to $\pm\pi/2$ (rad/s^2) while the linear accelerations are limited to ± 0.25 (m/s^2). The sampling time is $T_s = 0.05$ (s). The initial position of the robot is generated randomly such that $d_{s,0} = 3$ (m) and $d_{\theta,0} \in [-\pi, \pi]$. With this configuration , the sampling instants N_{reach} required by the robot to converge to the reference is $N_{reach} = 1637$ or $N_{reach}\, T_s = 81.85$ (s), with $N_U = 178$, $N_{V_r} = 849$, $N_{V_M} = 1152$ and $N_t = -542$.

Figure 2 shows the metrics real-time factor (ψ), control effort Π, and deviation Ξ when the legged robot is controlled with an NMPC without integral action and $N_c = 20$ (Figure 2-top) and when it is controlled with the proposed controller (Figure 2-bottom). The deviation and control effort are similar for both controllers. However, the real-time factor ψ is

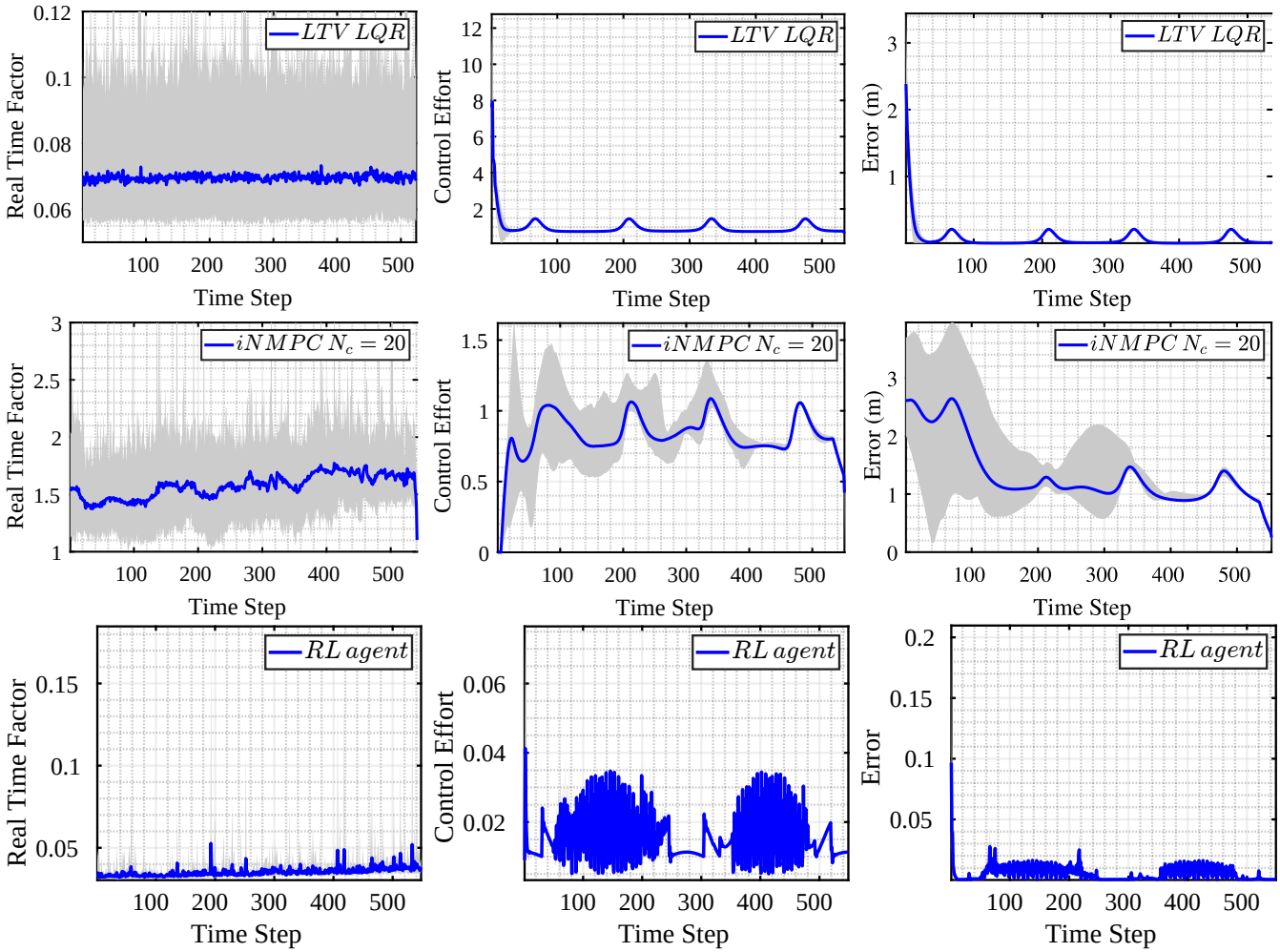

Fig. 3: Real-time factor ψ, control effort Π, and deviation error Ξ for the Linear Time-Varying Linear Quadratic Regulator (LTV LQR -first row-), the NMPC with integral mode with $N_c = 20$ (second row), and the Reinforcement Learning (RL) agent (third row). The LTV LQR outperforms all the implemented controllers; however, it does not satisfy the velocity and acceleration constraints. The RL agent fails when the initial position is not sufficiently close to the reference. The horizontal axis represents the duration of the experiment, expressed as the number of sampling instants T_s (s).

approximately 1.5 for the NMPC, indicating that real-time implementation is not feasible with the available computer.

Figure 3 depicts the results when the controllers are the LTV LQR (Figure 3-top), the iNMPC (NMPC with integral mode) with $N_c = 20$ (Figure 3-middle) and, the Reinforcement Learning (RL) agent (Figure 3-bottom). The iNMPC appears to reach the path too slowly, and the real-time factor remains above 1. The LTV LQR demonstrates outstanding performance; however, it is unable to handle the constraints imposed on accelerations, as shown in Figure 5-middle. Regarding the Reinforcement Learning (LR) agent, it follows the structure presented in [17] and it was implemented in Matlab. The agent was trained using the following reward function:

$$\text{reward} = \quad e^{-kd \times distance} + e^{-kc \times action\_penalty}$$

where $kd = 5$ and $kc = 0.001$ are constants, distance is the distance between the robot and the target, and action_penalty is the norm of the vector of controls. In addition, a bonus reward of $10 \times$ (EPISODE_STEPS $-$ NUM_STEPS)/EPISODE_STEPS is added if the robot reaches the vicinity of the target within a distance no greater than 0.05 (m). When the distance to the target increase from one step to the next, the reward is decreased by 1. The agent is trained with EPISODE_STEPS $= 500$ over a maximum of 5000 episodes. The RL agent achieves the lowest average real-time factor. The control effort is also low on average. However, the velocity profiles appear more abrupt compared to the other controllers. The tracking error remains low but does not converge to zero. Additionally, if the robot does not start close enough to the reference, the agent is unable to track the path. A similar issue arises when the reference speed is increased.

In this figure, the left y-axis corresponds to the accelerations of the LTV LQR controller, while the right y-axis corresponds to the accelerations obtained with the proposed approach. Figure 5-top shows the velocity profiles for both controllers.

Figure 4-top and 4-middle show the performance of the

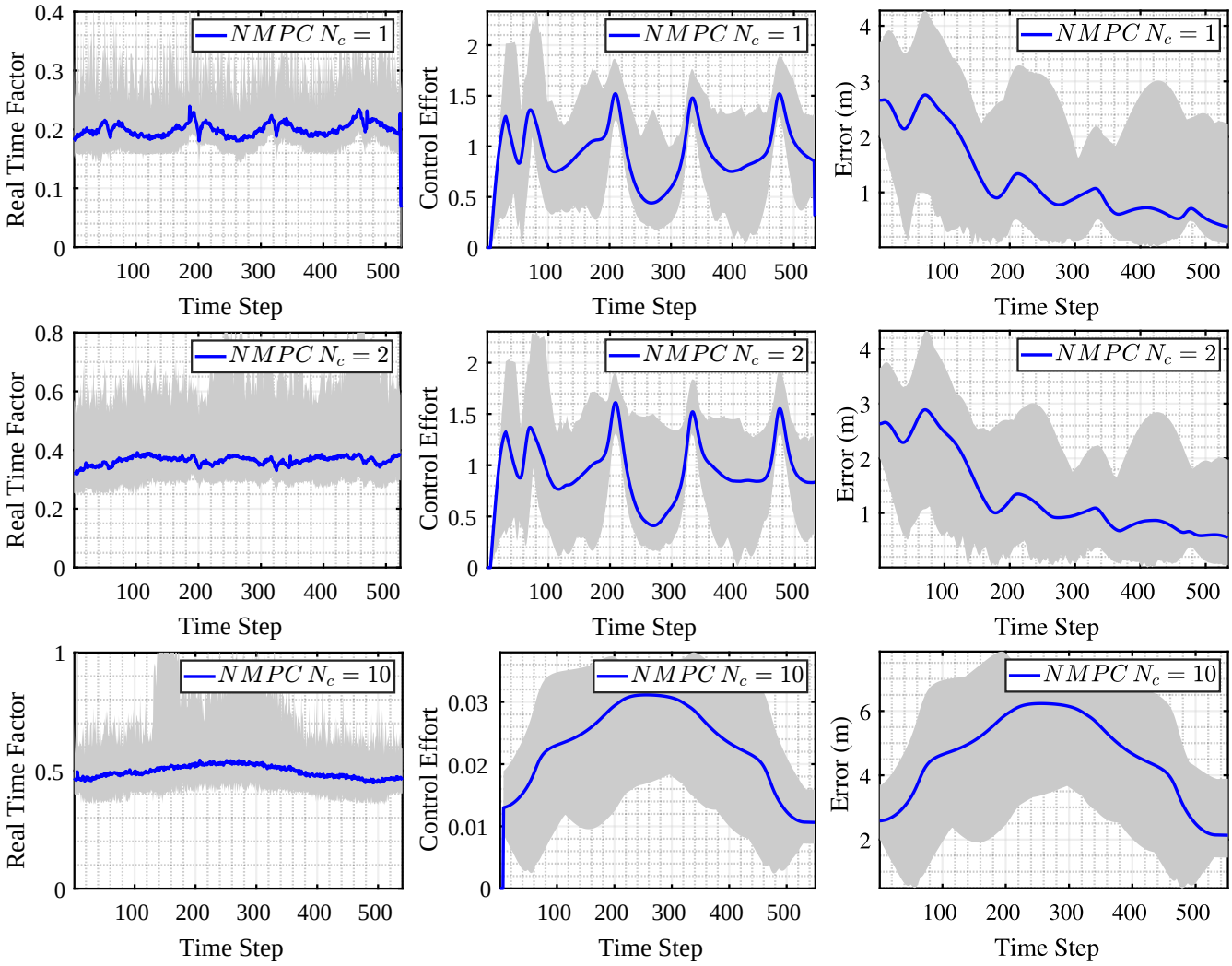

Fig. 4: Real-time factor ψ, control effort Π, and deviation error Ξ for NMPC with $N_c = 1$ (top), $N_c = 2$ (middle), and $N_c = 10$, where the controls are weighted 10^2 times higher than the deviation with respect to the path (bottom). The horizontal axis represents the duration of the experiment, expressed as the number of sampling instants T_s (s).

NMPC with $N_c = 1$ (Figure 4-top) and $N_c = 2$ (Figure 4-middle). The real-time factor is now below 1. The control effort Π behaves similarly to the NMPC with $N_c = 20$; however, the deviation Ξ decreases very slowly.

The main difference with the proposed approach is that the NMPC computes the controls that minimize $d^2_{\theta,k}$ and $d^2_{s,k}$, while the proposed approach computes the control u_k that minimizes the distance to the (unconstrained) control $\mathcal{F}(q_k)^T \frac{\Delta_{q,k}}{T_s}$. Taking this into account, poor tuning of the weighting matrices in the NMPC can lead to very poor performance, as shown in Figure 4-bottom, where the stage cost weights the controls 100 times higher than the deviation from the path.

B. Field experiments

The field experiments were conducted using the legged robot depicted in Figure 1. The robot is equipped with a real-time kinematics (RTK) GPS Navcom SF-3040 and an inertial measurement unit (IMU) Vectornav VN-200S. Leveraging the low-level primitives available on the vehicle, we utilize the nonlinear control-affine kinematic model of the vehicle for control purposes.

The experiments involve tracking the following trajectories: i) circular-shaped, ii) rectangular-shaped, and iii) infinity-shaped (Lemniscate). Three different controllers are implemented on the legged robot: a) OBQAC, b) iMPC$_1$, and c) iMPC$_2$, with integral mode and control horizon lengths N_c set to 10 and 20, respectively. The predictive controllers are configured with quadratic stage costs and share the same weighting matrices: $Q = \text{diag}(2,\ 2,\ 2,\ 0.1)$, $Q_f = \text{diag}(10,\ 10,\ 10,\ 5)$, and $R = \mathbf{I}_{4\times4}$.

The set $\mathcal{U}_c$ is a *box* set such that $[-\pi/2, \pi/2]$ (rad/s) $\times[-0.5,\ 0.5]$ (m/s) $\times[-0.5,\ 0.5]$ (m/s) $\times[-0.5,\ 0.5]$ (m/s).

The accelerations are limited to the following feasible values: $|(\omega_k - \omega_{k-1})|/T_s \leq \pi/2$ (rad/s^2), $|(v_{x,k} - v_{x,k-1})|/T_s \leq 0.25$ (m/s^2), $|(v_{y,k} - v_{y,k-1})|/T_s \leq 0.25$ (m/s^2), and $|(v_{z,k} - v_{z,k-1})|/T_s \leq 0.25$ (m/s^2).

For each trajectory, the function Θ_k is designed such that $\nu_r = 0.35$ (m/s). The metrics used to perform the comparisons

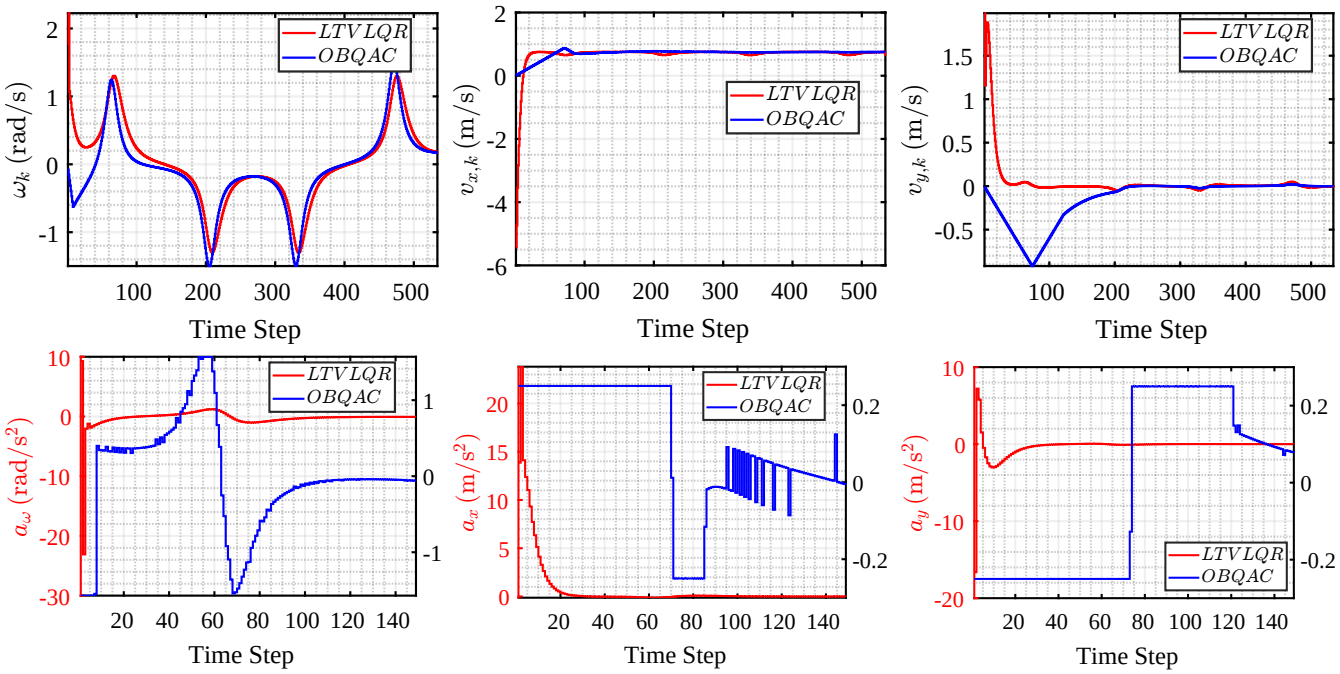

Fig. 5: Velocities (top) and accelerations (bottom) for 1 trial of the simulated experiments for the LTV LQR (red line, left y-axis) and the proposed OBQAC (blue line, right y-axis) methods. The first column corresponds to the angular velocity (top) and acceleration (bottom). The column in the middle corresponds to the linear velocity v_x (top) and linear acceleration a_x (bottom). The last column corresponds to the linear velocity v_y (top) and linear acceleration a_y (bottom). The angular acceleration is limited to $\pm\pi/2$ (rad/s^2), and the linear accelerations are limited to ± 0.25 (m/s^2). Velocities are shown for the entire path, while accelerations are depicted for only the first 150 sampling instants. The LTV LQR generates much more aggressive velocity profiles since it does not constrain the accelerations. The horizontal axis represents the duration of the experiment, expressed as the number of sampling instants T_s (s).

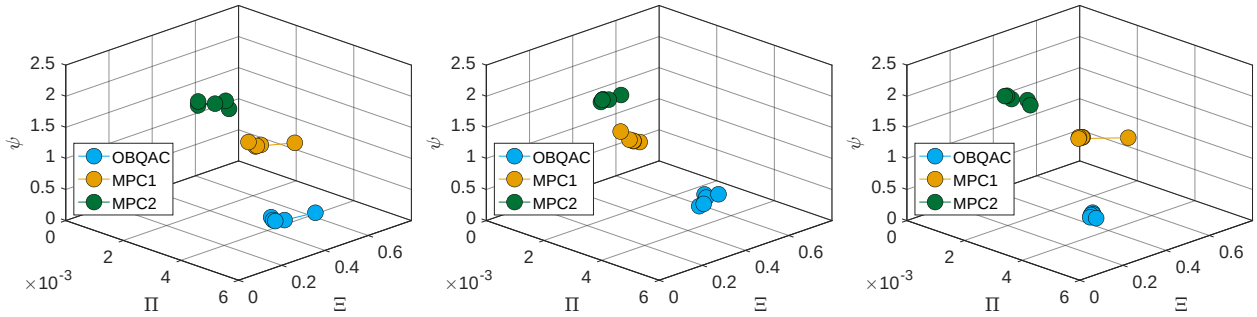

Fig. 6: Summary of the deviation Ξ with respect to the nominal path, control effort Π, and real-time factor ψ for the OBQAC, MPC1, and MPC2 controllers over 5 trials on three different trajectories: circular (left), rectangular (center), and infinity-shaped (right).

are the same as those used in the simulated experiments, given by Eqs. (22).

Figure 6 shows 3D plots summarising the metrics used for comparison given by Eqs. (22). The iMPC$_2$ achieves the best performance in terms of control effort and deviation, as expected from a predictive controller with a large control horizon. However, the mean RTF is above 1 almost all the time. However, this controller archives the best performance. This apparent lack of influence of an RTF as large as 2 is due to the velocity ν_r being set as low as 0.35 (m/s). Since the sampling time is $T_s = 0.05$ (s), then it entails an error of 0.35 (m/s) $\times 0.05$ (s) $\times RTF = 0.035$ (m).

The proposed approach (OBQAC) achieves a performance Ξ comparable to those of iMPC$_1$ and iMPC$_2$ with the lowest RTF. It achieves the higher control effort Π when compared to the iMPC controllers. Figure 7 illustrates the results obtained from the field experiments with the proposed OBQAC over three different trajectories. Each trajectory was tracked 5 times with each controller. The legged robot is depicted for Only 1 trial of the experiments.

Figure 8 shows the power consumption of the legged robot during 5 trials of field experiments when the trajectory is infinity-shaped. Only 2 controllers were implemented: a) OBQAC and b) NMPC without integral mode

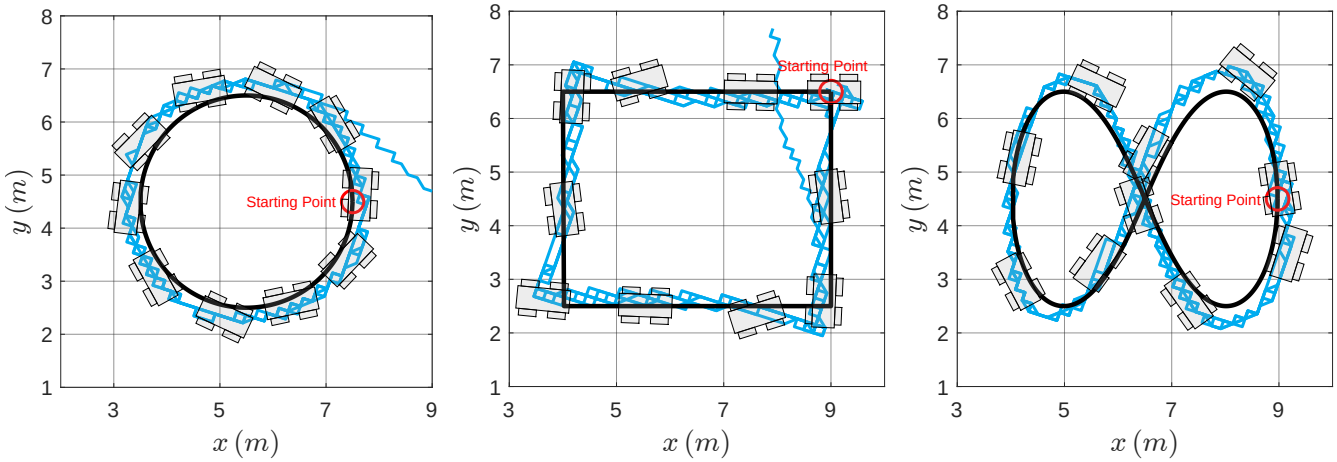

Fig. 7: Field experiments over 5 trials for each trajectory. The light-blue lines represent the trajectory tracked by the legged robot when controlled by the proposed approach. The red circle represents the path starting point, while the robot starts from an arbitrary point near the starting point. The robot's pose is depicted every 1 (s) for only 1 trial.

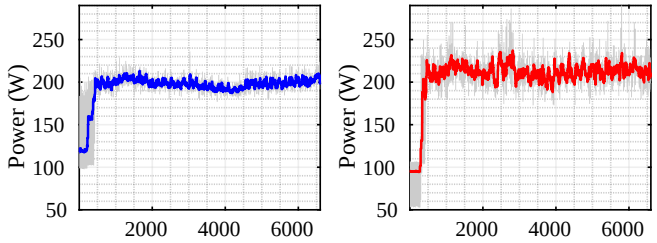

Fig. 8: Power consumption of the legged robot shown in Figure 1 during 5 trials of field experiments when the trajectory is infinity-shaped. The blue line (left) corresponds to the average power consumption when the legged robot is controlled with the proposed controller. The red line (right) corresponds to the average power consumption when the legged robot is controlled with an MPC without integral mode and $N_c = 10$. The horizontal axis represents the duration of the experiment, expressed as the number of sampling instants T_s (s).

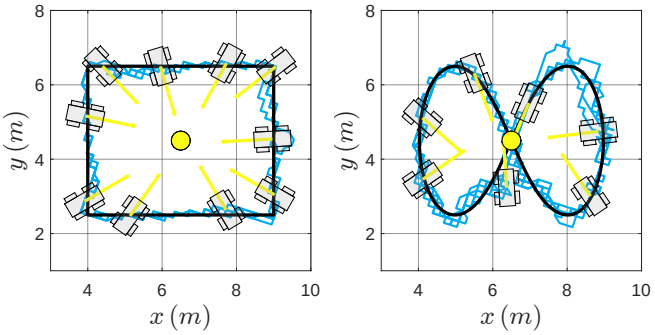

Fig. 9: Legged robot tracking a rectangular-shaped and an infinity-shaped trajectory, with its focal point being the center point of the trajectory (yellow circle). Light-blue lines represent the trajectories followed by the robot, while yellow arrows indicate its direction. The infinity-shaped trajectory poses the challenge of requiring a π (rad) turn to continuously face it.

and $N_c = 10$ and tuned with the following matrices: $Q = \text{diag}(10, 10, 0.001, 5)$, $R = \mathbf{I}_{4\times4}$, and $P = \text{diag}(50, 50, 0.005, 25)$. Figure 8-left corresponds to the average power consumption when the legged robot is controlled with the OBQAC, while Figure 8-right shows the average power consumption when the robot is controlled with the NMPC over 5 trials. The mean average power dissipated by the legged robot when it was controlled with the OBQAC was 189.19 (w), whilst when it was controlled with the NMPC the mean power was 209.9 (w). Moreover, as can be seen from Figure 8, the instantaneous power consumption seems to be more constant when the legged robot is controlled with the OBQAC.

1) Tracking a trajectory while facing at a fixed point: This field experiment involves tracking squared and infinity-shaped trajectories while the legged robot faces a fixed point located at the centre of the trajectories. The robot is controlled using the OBQAC, tuned with the same gains as in the previous field experiments. In Figure 9-left, the legged robot is depicted for a single trial as it travels along a rectangular-shaped trajectory. Yellow arrows indicate the direction in which the robot is pointing. The legged robot accurately maintains its orientation towards a fixed point while tracking the trajectory. Figure 9-right displays the results when the robot tracks an infinity-shaped trajectory. This trajectory poses a challenge for the robot as it requires executing a π (rad) turn when passing through the centre and maintaining its orientation towards that point.

V. Conclusions

This work introduced an optimization-based quasi-algebraic controller (OBQAC) designed for path tracking in orthogonal nonlinear control-affine systems, making it particularly suitable for mobile platforms such as legged robots. Simulated experiments demonstrated the effectiveness of the proposed controller, while field experiments validated its real-time applicability.

The OBQAC enables rapid prototyping and implementation of industrial solutions by leveraging the built-in low-level controllers available in commercial legged robots. The controller exhibited comparable performance to a nonlinear model predictive control (NMPC) with control horizon of 20 samples, but with significantly lower computational effort. Moreover, unlike reinforcement-learning agents, this approach requires no training procedures. Moreover, the stability of the controller was analytically established as a function of the control set $\mathcal{U}_c$.

Overall, the proposed OBQAC offers a straightforward yet effective strategy that capitalizes on the control-affine structure of autonomous navigation systems, particularly legged robots, to facilitate the implementation of path-tracking solutions. Its simplicity and effectiveness make it a valuable tool for educational, industrial, and research applications.

REFERENCES

[1] F. A. Cheein and G. Scaglia, "Trajectory tracking controller design for unmanned vehicles: A new methodology," *Journal of Field Robotics*, vol. 31, no. 6, pp. 861–887, 2014.

[2] M. E. Serrano, D. C. Gandolfo, and G. J. Scaglia, "Trajectory tracking controller for unmanned helicopter under environmental disturbances," *ISA Transactions*, vol. 106, pp. 171–180, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001905782030272X

[3] A. Aydinoglu and M. Posa, "Real-time multi-contact model predictive control via admm," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3414–3421.

[4] S. Le Cleac'h, T. A. Howell, S. Yang, C.-Y. Lee, J. Zhang, A. Bishop, M. Schwager, and Z. Manchester, "Fast contact-implicit model predictive control," *IEEE Transactions on Robotics*, vol. 40, pp. 1617–1629, 2024.

[5] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, 2023.

[6] V. Grushkovskaya and A. Zuyev, "Stabilization of non-admissible curves for a class of nonholonomic systems," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 656–661.

[7] J. Lee and S. B. Choi, "Integrated control of steering and braking for path tracking using multi-point linearized mpc," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 5, p. 3324 – 3335, 2023, cited by: 4. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85141567776&doi=10.1109%2fTIV.2022.3218734&partnerID=40&md5=f5a06a244c9f34f37cd486cb0e926cf7

[8] M. T. Peterson, T. Goel, and J. C. Gerdes, "Exploiting linear structure for precision control of highly nonlinear vehicle dynamics," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1852–1862, 2023.

[9] C. Tamekue and S. Ching, "Control analysis and synthesis for general control-affine systems," 2025. [Online]. Available: https://arxiv.org/abs/2503.05606

[10] Z. Zhang, G. Puthumanaillam, M. Vora, and M. Ornik, "Motion planning and control with unknown nonlinear dynamics through predicted reachability," 2025. [Online]. Available: https://arxiv.org/abs/2503.03633

[11] W. Liu, J. Zhang, D. Sun, and H. Hu, "Piecewise affine based model predictive trajectory tracking control with stability guarantees for autonomous ground vehicles*," in *2022 China Automation Congress (CAC)*, 2022, pp. 3664–3670.

[12] K. Wang, Z. Xu, K. Zhang, Y. Huang, and J. Xu, "Lattice piecewise affine approximation of explicit nonlinear model predictive control with application to trajectory tracking of mobile robot," *IET Control Theory & Applications*, vol. 18, no. 2, pp. 149–159, 2024. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cth2.12553

[13] K. Zhang, J. Wang, H. Peng, and Y. Dang, "Model predictive control based path following for a wheel-legged robot," in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 3925–3930.

[14] Z. Dorbetkhany, "Model predictive control of skid-steered mobile robot with deep learning system dynamics," *Nazarbayev University: Astana, Kazakhstan*, 2023.

[15] R. Kamalapurkar, L. Andrews, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for infinite-horizon approximate optimal tracking," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 5083–5088.

[16] A. Ghasemzadeh, R. Amjadifard, and A. Keymasi-Khalaji, "Adaptive dynamic programming for kinematic control of 3 interconnected wheeled mobile robots," in *2024 32nd International Conference on Electrical Engineering (ICEE)*, 2024, pp. 1–7.

[17] S. Srikonda, W. R. Norris, D. Nottage, and A. Soylemezoglu, "Deep reinforcement learning for autonomous dynamic skid steer vehicle trajectory tracking," *Robotics*, vol. 11, no. 5, 2022. [Online]. Available: https://www.mdpi.com/2218-6581/11/5/95

[18] A. Joglekar, S. Sathe, N. Misurati, S. Srinivasan, M. J. Schmid, and V. Krovi, "Deep reinforcement learning based adaptation of pure-pursuit path-tracking control for skid-steered vehicles," *IFAC-PapersOnLine*, vol. 55, no. 37, pp. 400–407, 2022, 2nd Modeling, Estimation and Control Conference MECC 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322028592

[19] P. Nourizadeh, F. J. S. McFadden, and W. N. Browne, "Trajectory tracking control of skid-steering mobile robots with slip and skid compensation using sliding-mode control and deep learning," 2023. [Online]. Available: https://arxiv.org/abs/2309.08863

[20] K. Dawson, O. Menendez, C. Camacho, M. Torres-Torriti, and A. Prado, "On the assessment of reinforcement learning techniques for path planning of skid-steer mobile robots subject to terrain constraints," in *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*, 2024, pp. 1–7.

[21] T. P. Nascimento, C. E. T. Dórea, and L. M. G. Gonçalves, "Nonholonomic mobile robots' trajectory tracking model predictive control: a survey," *Robotica*, vol. 36, no. 5, p. 676–696, 2018.

[22] S. A. Roth and P. Batavia, "Evaluating path tracker performance for outdoor mobile robots," in *Automation Technology for Off-Road Equipment Proceedings of the 2002 Conference*. American Society of Agricultural and Biological Engineers, 2002, p. 388.

[23] W. R. Norris and A. E. Patterson, "System-level testing and evaluation plan for field robots: A tutorial with test course layouts," *Robotics*, vol. 8, no. 4, 2019. [Online]. Available: https://www.mdpi.com/2218-6581/8/4/83

[24] S. You, M. Greiff, R. Quirynen, S. Ran, Y. Wang, K. Berntorp, R. Dai, and S. Di Cairano, "Integral action nmpc for tight maneuvers of articulated vehicles," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 3155–3161.

Nestor Nahuel Deniz received the Bachelor's degree in electronic engineering from the National Technological University in Parana, Entre Rios, Argentina. He obtained his PhD from the National University of the Litoral in Argentina. Currently, he serves as an Assistant Researcher at the National Scientific and Technical Research Council in Argentina. His current research interests centre on robotics, model-based estimation and control techniques, reinforcement learning, autonomous navigation of vehicles, data-driven control, neural networks, and Koopman operator's theory.

Fernando Auat Cheein (Senior Member, IEEE) received the Bachelor degree in electronic engineering from Universidad Nacional de Tucuman, Argentina, in 2002, and the M.Sc. and Ph.D. degrees in Engineering from Universidad Nacional de San Juan, Argentina, in 2005 and 2009, respectively. He is currently with the Department of Electronic Engineering, Harper Adams University, Newport, Shropshire TF10 8NB, UK. His research is focused on autonomous and intelligent vehicles, robotics and perception in agriculture, motion planning and control, and efficient navigation strategies. Prof. Auat Cheein is currently an Associate Editor of several journals.