IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 May 2025, accepted 20 June 2025, date of publication 26 June 2025, date of current version 8 July 2025.

*Digital Object Identifier 10.1109/ACCESS.2025.3583596*

RESEARCH ARTICLE

# Nonlinear Model Predictive Control for Trajectory Tracking of Omnidirectional Robot Using Resilient Propagation

**MAHMOUD EL-SAYYAH[1], MOHAMAD R. SAAD[1], (Senior Member, IEEE), AND MAAROUF SAAD[2], (Senior Member, IEEE)**

[1]School of Engineering, Université du Québec en Abitibi Témiscamingue, Rouyn-Noranda, QC J9X 5E4, Canada
[2]École de Technologie Supérieure (ETS), Montreal, QC H3C 1K3, Canada

Corresponding author: Mahmoud El-Sayyah (mahmoud.sayyah@uqat.ca)

This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC) under Grant RGPIN-2018-04916.

**ABSTRACT** This paper proposes an enhanced Nonlinear Model Predictive Control (NMPC) framework that incorporates a robust, convergent variant of the resilient propagation (RPROP) algorithm to efficiently solve the Nonlinear Optimization Problem (NOP) in real time. The controller is developed for both constrained and unconstrained trajectory tracking of Wheeled Mobile Robots (WMRs), with operational constraints handled via the external penalty method. The proposed method introduces adaptive step sizes and a backtracking mechanism, significantly improving convergence speed without compromising accuracy. Simulation results show that, even under constraints, the proposed method reduces computational time by a factor of 6 to 11 compared to the Interior Point method and 2 to 4 compared to the Active Set method. In addition, it achieves superior tracking accuracy, with root mean square (RMS) position tracking errors reduced by approximately 50% relative to the benchmark methods. Real-time experiments conducted on the Robotino Festo Omnidirectional Mobile Robot (OMR) validate the method's practical effectiveness, demonstrating faster convergence and improved velocity tracking performance, while maintaining comparable or better position tracking. These findings establish the proposed controller as a computationally efficient and accurate solution for real-time WMR trajectory tracking.

**INDEX TERMS** Nonlinear model predictive control (NMPC), omnidirectional mobile robot, resilient propagation, trajectory-following.

## I. INTRODUCTION

Over the past two decades, the rapid evolution of hardware and software technologies has revolutionized the capabilities of automated machinery, including mobile robots, across different sectors of society [1], [2]. Nowadays Wheeled Mobile Robots (WMRs) are entrusted to join operations in many fields of application such as planet exploration [3], agriculture [4], underground mines' inspection [5], [6], surveillance missions [7], healthcare support [8] and household assistance [9]. Due to their design's simplicity, low cost and high efficiency, WMRs are fully developed and commercialized for applications in constructed environments such as factories and warehouses [10]. However, when it comes to unconstructed environments such as underground mining operations, these applications are still limited. Engaging WMR in this field can improve productivity and increase miners' safety by performing tasks like automated transportation and inspection, monitoring hazardous contagious areas, and replacing the human operator on dangerous missions [11], [12]. Underground mine galleries are characterized by extreme conditions, including uneven terrains, the risk of falling rocks, and poor lighting [13], [14]. To operate effectively in these hazardous conditions, WMRs must demonstrate advanced operational capabilities, including robust path generation to identify feasible routes, obstacle

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

avoidance for safe navigation, and trajectory tracking to ensure precise adherence to predefined paths. Among these, trajectory tracking is pivotal for achieving the autonomy and stability required for reliable performance in complex environments.

Trajectory tracking aims to drive the robot towards a reference path, created online by path-generation methods, or predefined and uploaded offline [15]. Achieving accurate trajectory tracking in WMRs involves addressing the system's nonlinear, multivariable characteristics while adhering to operational constraints. Due to these challenges, trajectory tracking has been a widely addressed area in the field of robotic control. For instance, bioinspired backstepping [16], super-twisting sliding mode [17] and observer-based methods [18], [19], have been widely used for WMR trajectory tracking. However, these methods are designed primarily for unconstrained scenarios and do not explicitly consider operational constraints. To manage constraints, barrier functions have been incorporated, but this often introduces significant design complexity [20], [21]. Input saturation management methods [22], [23] have been proposed as another approach to handle constraints, but these techniques commonly suffer from overshooting, limiting their practical effectiveness. Model-free methods, such as those based on fuzzy logic or bio-inspired neural networks [24], [25], have been explored for constrained trajectory tracking. However, bypassing the system model may exclude valuable information critical for control. Additionally, these methods often demand significant expertise and extensive training, which may not always be accessible.

Model Predictive Control (MPC) has emerged as a robust solution to address these limitations [26]. MPC's ability to manage multivariable nonlinear systems while respecting operational constraints makes it particularly suited for trajectory tracking. For nonlinear systems, MPC is implemented as Nonlinear MPC (NMPC) [27]. Despite its strengths, NMPC faces substantial challenges in real-time applications due to the high computational burden associated with solving the underlying Nonlinear Optimization Problem (NOP). Several approaches have been proposed to alleviate this issue, including system linearization [28], [29], [30], adaptive horizon adjustments [31], and hardware optimizations [32]. While these techniques reduce computational demands, they often compromise tracking accuracy or are problem-specific, limiting their general applicability.

The optimization algorithm is one of the most extensively addressed components of NMPC for improving computational efficiency. Numerous strategies have been developed to solve the associated Nonlinear Optimization Problem (NOP). For instance, in [33], [34], the continuation method was combined with the Generalized Minimal Residual (GMRES) method, transforming the nonlinear problem into a linear one and solving it accordingly. However, this approach relies on specific structural properties of the system and cost function, limiting its general applicability. Moreover, its implementation becomes increasingly challenging when dealing with complex system dynamics or elaborate performance indices. Neural optimization methods are another popular class of techniques for solving NOPs [35], [36]. For example, [35] introduced a one-layer projection neural network for quadratic optimization, achieving faster convergence while maintaining moderate computational demand. Nevertheless, these methods require significant training data, which may not always be available, and their generalization to other problems is often constrained without retraining. Metaheuristic strategies, such as particle swarm optimization, ant colony optimization, and the gravitational search algorithm, were evaluated in [37] for NMPC applications. These approaches demonstrated good tracking performance with relatively low computational cost. However, they inherently provide only approximate solutions, and as such, convergence and optimality cannot be guaranteed, making them less suited for safety-critical applications. In contrast, conventional exact methods, such as the Interior Point Method (IPM) [38], [39], Active-Set Method (ASM) [40], [41], Sequential Programing (SQ), offer robust convergence and guaranteed optimality. Despite their strengths, their application in fast dynamic systems like WMRs remains limited to problems with linear structure or a small number of decision variables [42], thereby constraining their practical utility in high-speed real-time control settings.

Resilient propagation (RPROP), originally developed for training neural networks, is a gradient-based optimization technique that uses only the sign of the error gradient, substituting its magnitude with an adaptive step size [43]. This design enables faster convergence, but the classical RPROP lacks guaranteed convergence properties [44], [45]. To address this, a Robust Convergent variant of RPROP (RCPROP) was introduced [44], incorporating a backtracking mechanism to ensure convergence by reassessing step quality based on the global error function. While RPROP has previously been applied in NMPC for quadrotor control [46], its integration into constrained NMPC frameworks, particularly for mobile robotics, remains unexplored, an area this work aims to investigate and advance.

Motivated by the above-mentioned concepts, we introduce an improved NMPC approach that utilizes RCPROP algorithm to address and solve the trajectory tracking challenge of a WMR. The computational advantage of the proposed approach facilitates achieving faster convergence. The study considers both unconstrained and constrained NMPC cases to comprehensively evaluate the performance of the proposed algorithm across different practical scenarios. The unconstrained case provides a baseline to assess the algorithm's core efficiency in terms of tracking accuracy and convergence speed. Conversely, the constrained case evaluates the algorithm's ability to handle operational constraints, which are essential in real-world applications. By studying both cases together, we gain a deeper understanding of how the presence of constraints affects the performance

of the trajectory tracking algorithm and the optimization method. We thoroughly investigate and validate the integration of RCPROP into the NMPC framework, comparing it to benchmark methods and evaluating its effectiveness. For the constrained NMPC, we employ the external penalty method to effectively manage the constraints. The superiority of our proposed method is demonstrated through a comparative study against the well-known interior point method (IPM) and active-set method (ASM). The main contributions of this study are summarized as follows:

1. Implementation of RCPROP to efficiently tackle the Nonlinear Optimization Problem (NOP) leading to a significant reduction in computational cost and ensuring fast convergence. Unlike existing methods, our approach introduces a guaranteed convergent variant of RPROP.
2. Exploration of RCPROP's capability to handle operational constraints within the framework of NMPC. To the best of our knowledge, this aspect has remained unexplored until now.
3. Assessing the real-time suitability of the proposed method using the Robotino-Festo OMR.

The rest of the paper is organized as follows: Section II presents the NMPC setup including the kinematic modeling, objective function, and constraints. Next, the optimization algorithm is presented in Section III. Simulation and experimental results are illustrated in Section IV, and Section V concludes the paper.

## II. NMPC SETUP

### A. VEHICLE MODELING

As shown by its name, NMPC is a model-based method, therefore, an accurate model describing the system under control is necessary for the development of the controller. In this study, we consider an OMR with three omni-wheels located at 120° from each other (Figure 1). This type of robot has full manoeuvrability, which means it can move instantly in any direction without any reorientation. Define $(x,\ y)$, the position of the WMR in the global frame, $(\theta)$, its orientation, $(v_x,\ v_y)$, its translational velocities in the local frame, and $(\omega)$, its rotational velocity. The state vector of the WMR is chosen as:

$$q = \begin{bmatrix} x & y & \theta & v_x & v_y & \omega \end{bmatrix}^T, \tag{1}$$

and the vector of the control variables is $u = (a_x,\ a_y,\ a_\theta)$, where $(a_x,\ a_y)$ are the translational accelerations, and $(a_\theta)$ is the angular acceleration. By mapping the global coordinates into the local frame, we obtain the robot's kinematic equations as follows [30]:

$$\begin{aligned} \dot{x} &= v_x \cos\theta - v_y \sin\theta, \\ \dot{y} &= v_x \sin\theta + v_y \cos\theta, \\ \dot{\theta} &= \omega, \\ \dot{v}_x &= a_x, \\ \dot{v}_y &= a_y, \\ \dot{\omega} &= a_\theta. \end{aligned} \tag{2}$$

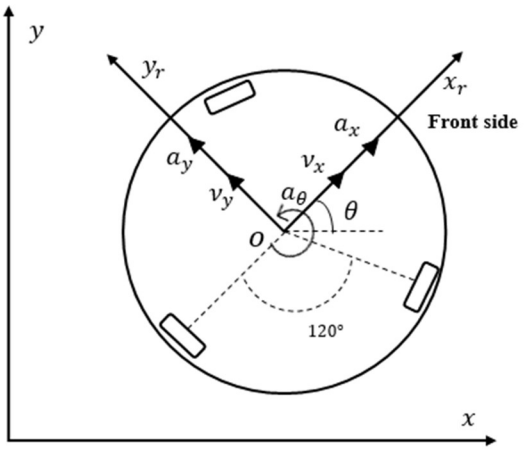



**FIGURE 1.** Local and global coordinates.

To discretize the kinematic model, the forward differences method is applied. The resulted discrete-time representation is as follows [30]:

$$q_{k+1} = f(q_k, u_k) = \begin{bmatrix} x_k + v_{xk} T \cos\theta_k - v_{y_k} T \sin\theta_k \\ y_k + v_{xk} T \sin\theta_k + v_{y_k} T \cos\theta_k \\ \theta_k + \omega_k T \\ v_{xk} + a_x T \\ v_{yk} + a_y T \\ \omega_k + a_\theta T \end{bmatrix}, \tag{3}$$

where $T$ is the sampling time. The robot used in this study accepts the linear and angular velocities of its center of gravity as inputs, hence there is no need to further develop the kinematic model to include the wheels velocities, and the robot geometry. The reader can refer to [30] for kinematic representation including the robot's geometry and wheels' velocities.

### B. UNCONSTRAINED NMPC

The NMPC method is one of the optimal control strategies, which aims to achieve the control's objectives by formulating them into one function called the objective function and then finding the control inputs that minimize this function. In this scenario, the objective is to minimize the errors between the reference state variables and their predicted counterparts. Hence, we opt for a quadratic cost function, which penalizes both state variables errors and control inputs, thus effectively steering the system towards the desired trajectory. First, we consider the unconstrained case where the NMPC control problem of the robot motion is written as [27]:

$$\begin{aligned} &\min J = \frac{1}{2} e_N^T Q_0 e_N + \sum_{k=0}^{N-1} \frac{1}{2} (e_k^T Q e_k + u_k^T R u_k) \\ &\text{s.t. } q_{k+1} = f(q_k, u_k), \end{aligned} \tag{4}$$

where $J$ is the objective function, $e_N = q_N - q_{dN}$ is the terminal state error, $e_k = q_k - q_{dk}$ is the tracking error, $q_{dk}$ is the reference trajectory, $Q$,$Q_0$(both $6 \times 6$) and $R$ ($3 \times 3$) are diagonal penalty matrices. $f(q_k, u_k)$ represents the kinematic equations that must still be adhered to, even in the unconstrained case. Solving (4) determines the control inputs ensuring that the state errors are decreasing along the control horizon. To incorporate the system's kinematics into the objective function, Lagrange multipliers vectors $\lambda_k$ ($6\times1$) with ($k = 1, \ldots, N$) are introduced in (4) and the NMPC control problem is transformed to:

$$\min J_a = \phi(e_N) + \sum_{k=0}^{N-1} \left( \frac{1}{2}(e_k^T Q e_k + u_k^T R u_k) + \lambda_{k+1}^T (f_k - q_{k+1}) \right), \tag{5}$$

where $\phi(e_N) = \frac{1}{2} e_N^T Q_0 e_N$ is the terminal cost, and $f_k \equiv f(q_k, u_k)$. The Lagrange multipliers $\lambda_k$ are associated with kinematic terms that are identically zero. Therefore, they can be chosen arbitrarily to simplify the derivation.

Define the Hamiltonian function of problem (5) as:

$$H_k \equiv H(q_k, u_k) = \frac{1}{2}(e_k^T Q e_k + u_k^T R u_k) + \lambda_{k+1}^T f_k. \tag{6}$$

Substituting (6) into the cost function in (5), we obtain:

$$J_a = \phi(e_N) + H_0 - \lambda_N^T q_N + \sum_{k=1}^{N-1} \left( H_k - \lambda_k^T q_k \right). \tag{7}$$

To minimize (7), we differentiate $J_a$ as:

$$dJ_a = \left( \frac{\partial \phi(e_N)}{\partial q_N} - \lambda_N^T \right) dq_N + \frac{\partial H_0}{\partial q_0} dq_0 + \frac{\partial H_0}{\partial u_0} du_0 + \sum_{k=1}^{N-1} \left( \frac{\partial H_k}{\partial q_k} - \lambda_k^T \right) dq_k + \frac{\partial H_k}{\partial u_k} du_k. \tag{8}$$

To simplify (8), we chose:

$$\lambda_N^T = \frac{\partial \phi(e_N)}{\partial q_N} = e_N^T Q_0 \frac{\partial e_N}{\partial q_N},$$
$$\lambda_k^T = \frac{\partial H_k}{\partial q_k} = e_k^T Q_k \frac{\partial e_k}{\partial q_k} + \lambda_{k+1}^T \frac{\partial f_k}{\partial q_k}. \tag{9}$$

These expressions define the values of $\lambda_k$ based on a backward propagation approach, starting from the terminal condition and recursively computing the multipliers at each previous time step. This procedure ensures the optimality conditions required for minimizing the Hamiltonian and the overall cost function. With these choices, (8) becomes:

$$dJ_k = \sum_{k=0}^{N-1} \frac{\partial H_k}{\partial u_k} du_k, \tag{10}$$

with

$$\frac{\partial H_k}{\partial u_k} = u_k^T R_k + \lambda_{k+1}^T \frac{\partial f_k}{\partial u_k}. \tag{11}$$

Here we used the fact that $q_0$, the state vector at the current sample instant of the controller, remains constant during the optimization process, which means that $dq_0 = 0$. Equation (10) shows that minimizing the Hamiltonian automatically leads to minimizing the objective function.

### C. CONSTRAINED NMPC

In real-time applications, there are always limitations and restrictions that need to be taken into consideration when designing the control algorithm. One of the main advantages of NMPC is the capability to handle control constraints explicitly. When constraints are imposed, the NMPC control problem becomes as follows [27]:

$$\begin{aligned} \min J &= \frac{1}{2} e_N^T Q_0 e_N + \sum_{k=0}^{N-1} \frac{1}{2}(e_k^T Q e_k + u_k^T R u_k) \\ \text{s.t. } & q_{k+1} = f(q_k, u_k), \\ & h(q_k, u_k) \leq 0, \end{aligned} \tag{12}$$

Here the cost function $J$ is designed to minimize tracking errors while balancing control effort. The penalization matrices have similar characteristics as in equation (4). In addition to the system dynamics, the cost function is also subject to a second constraint $h(q_k, u_k)$ that enforces the state and input limits, where $h(q_k, u_k)$ are inequality equations set as:

$$\begin{cases} -v_{\max} \leq v_{xk}; v_{yk} \leq v_{\max}, \\ -\omega_{\max} \leq \omega_k \leq \omega_{\max}, \\ -a_{t\max} \leq a_{xk}; a_{yk} \leq a_{t\max}, \\ -a_{n\max} \leq a_{\theta k} \leq a_{n\max}. \end{cases} \tag{13}$$

Here $v_{\max}$ and $\omega_{\max}$ are the maximum allowable translational and rotational velocities respectively, and $a_{t\max}$ and $a_{n\max}$ are the maximum translational and rotational accelerations respectively. These inequality constraints can be handled using different approaches such as the auxiliary variable method or the external penalty method. In [33], the auxiliary variable method was used to transform the inequality constraints to equality ones by adding a dummy variable which was later considered an additional variable in the optimization problem. However, using a dummy variable for each inequality constraint will increase the dimension of the optimization problem and consequently increase the computational burden. The external penalty method aims to transform the constrained problem to an unconstrained one by adding a penalty term to the objective function which penalizes the violation of constraints. It is a straightforward method proven to be effective [47], and it will be adopted in this work. The additional penalty term is chosen as [47]:

$$\mathrm{M}_i(q_k, u_k) = \begin{cases} 0\,, & h_i(q_k, u_k) \leq 0, \\ \mu_i \times h_i(q_k, u_k)^2\,, & h_i(q_k, u_k) > 0, \end{cases} \tag{14}$$

where $\mathrm{M}_i(q_k, u_k)$ and $\mu_i$ are the penalty cost and the weight factor of the $i$th inequality constraint $h_i(q_k, u_k)$. In this work,

$h_i(q_k, u_k)$ is:

$$\begin{cases} h_1(q_k, u_k) = v_{xk}^2 - v_{\max}^2, \\ h_2(q_k, u_k) = v_{yk}^2 - v_{\max}^2, \\ h_3(q_k, u_k) = \omega_k^2 - \omega_{\max}^2, \\ h_4(q_k, u_k) = a_{xk}^2 - a_{t\max}^2, \\ h_5(q_k, u_k) = a_{yk}^2 - a_{t\max}^2, \\ h_6(q_k, u_k) = a_{\theta k}^2 - a_{n\max}^2. \end{cases} \tag{15}$$

*Remark 1:* Here we took advantage of the constraints structure in (13), where the absolute values of the upper and lower limits are equal. However, in the general case, each limit should be treated as a distinct constraint. For example, if $v_{\min} \leq v_{xk} \leq v_{\max}$, this condition should be divided into two separate constraints, $h_a(q_k, u_k) = v_{xk} - v_{\max}$, and $h_b(q_k, u_k) = -v_{xk} + v_{\min}$.

The obtained unconstrained optimization problem can be written as follows:

$$\begin{aligned} \min J_c &= \frac{1}{2} e_N^T Q_0 e_N \\ &\quad + \sum_{k=0}^{N-1} \left[ \frac{1}{2} (e_k^T Q e_k + u_k^T R u_k) + \sum_{i=1}^{6} \mathrm{M}_i(q_k, u_k) \right], \\ \text{s.t. } & q_{k+1} = f(q_k, u_k). \end{aligned} \tag{16}$$

Using Lagrange multipliers to include system's kinematics into the objective function, and following similar approach as in section B, the costate equations can be found as:

$$\begin{aligned} \lambda_N^T &= \frac{\partial \phi(e_N)}{\partial q_N} = e_N^T Q_0 \frac{\partial e_N}{\partial q_N}, \\ \lambda_k^T &= \frac{\partial H_k}{\partial q_k} = e_k^T Q_k \frac{\partial e_k}{\partial q_k} + \lambda_{k+1}^T \frac{\partial f_k}{\partial q_k} + \sum_{i=1}^{6} \frac{\partial \mathrm{M}_i(q_k, u_k)}{\partial q_k}, \end{aligned} \tag{17}$$

and the gradients of the Hamiltonian as

$$\frac{\partial H_k}{\partial u_k} = u_k^T R_k + \lambda_{k+1}^T \frac{\partial f_k}{\partial u_k} + \sum_{i=1}^{6} \frac{\partial \mathrm{M}_i(q_k, u_k)}{\partial u_k}. \tag{18}$$

These equations represent the foundation of solving optimal control problems in NMPC. The costate equations propagate backward, linking the terminal and intermediate costs to system dynamics and constraints, while the Hamiltonian gradients guide the forward determination of optimal control inputs.

## III. OPTIMIZATION ALGORITHM

In this work, we employ an RPROP-based algorithm to address the nonlinear optimization challenges outlined in the previous section. Unlike traditional gradient-based methods, RPROP updates the step sizes of individual optimization variables using only the sign of their gradients, disregarding the magnitude. This approach introduces dynamic step variations, which adapt based on local knowledge of the objective function. The key feature of RPROP is that the step sizes are updated independently of the system's inputs or absolute gradient values, making it particularly effective for noisy or approximate gradient cases. The method was first introduced in [43], where the step size update mechanism is described as follows:

$$\Delta_{i,k} = \begin{cases} \min\left(\eta^+ \Delta_{i,k-1}, \Delta_{\max}\right) & \text{if } \frac{\partial H_{k-1}}{\partial u_i} \frac{\partial H_k}{\partial u_i} > 0 \\ \max\left(\eta^- \Delta_{i,k-1}, \Delta_{\min}\right) & \text{if } \frac{\partial H_{k-1}}{\partial u_i} \frac{\partial H_k}{\partial u_i} < 0 \\ \Delta_{i,k-1} & \text{otherwise} \end{cases}, \tag{19}$$

where $\Delta_{i,k}$ is the step size, $\Delta_{\min}$ and $\Delta_{\max}$ are the minimum and the maximum limits respectively, $\eta^+$ and $\eta^-$ are the factors for increasing and decreasing the step size, respectively, with $0 < \eta^- < 1 < \eta^+$. A consistent gradient sign triggers an increase in the step size to accelerate convergence, while a sign change indicates passing a local minimum, prompting a reduction in step size. The second stage involves updating the optimization variables as:

$$u_{i,k+1} = \begin{cases} u_{i,k} - \operatorname{sign}\left(\frac{\partial H_k}{\partial u_i}\right) . \Delta_{i,k} & \text{if } \frac{\partial H_{k-1}}{\partial u_i} \frac{\partial H_k}{\partial u_i} \geqslant 0, \\ u_{i,k-1} & \text{if } \frac{\partial H_{k-1}}{\partial u_i} \frac{\partial H_k}{\partial u_i} < 0. \end{cases} \tag{20}$$

When the partial derivative changes signs, this means the previous update was too large. Therefore, the algorithm will backtrack to reduce the step size. In this scenario, the current partial derivative is set to zero to avoid double penalties on the updates.

Despite its advantages, traditional RPROP does not guarantee convergence. As shown in [44], [45], independently updating step sizes for each variable can occasionally increase the objective function or result in negligible error reductions, causing the optimization process to stall. To overcome these limitations, a Robust Convergent variant (RCPROP) was proposed in [44].

RCPROP integrates an additional verification mechanism that evaluates the effect of step size updates on the overall objective function. If a proposed update does not decrease the objective function by a defined threshold, $\delta \min(\Delta_{i,k-1})$, where $\delta$ is the threshold of the optimization, the algorithm performs a global backtracking step to ensure convergence. Algorithm 1 provides a detailed outline of this approach.

This robust mechanism ensures the algorithm converges to a local minimum within a finite number of iterations. Further details and proof of convergence are available in [44].

*Remark 2:* The term "Robust" in this context is attributed specifically to the optimization algorithm rather than the tracking controller itself. It denotes the mechanism of global backtracking to prevent the algorithm from pursuing either increases in error or vanishingly small improvements in error, that could persist indefinitely. While this paper focuses on the optimization framework, the modeling and handling of uncertainties and external disturbances, particularly in real-time applications, are beyond the scope of this study. These aspects will be addressed and explored in future work.

The overall NMPC process begins with a known reference path for the robot. After initializing the input vector with an initial guess based on prior control inputs, the process uses Algorithm 1 to iteratively adjusts the control inputs by solving a nonlinear optimization problem to minimize the cost function while respecting system dynamics and constraints. Once optimized, the control inputs are applied to the robot, steering it along the desired trajectory while continuously updating the process in a receding-horizon manner.

## IV. CASE STUDIES AND ANALYSES

In this section, the performance of the proposed method is evaluated. Comparison with other benchmark approaches is done to demonstrate the capabilities of the RCPROP optimization algorithm to perform online optimization, handle constraints and reduce the computational burden.

### A. SIMULATION SETUP

Three methods will be compared:

1) NMPCPROP: This approach solves the NMPC problem using RCPROP algorithm.
2) NMPCIP: This method uses the IPM to solve the optimization problem of NMPC. Its implementation is based on [27].
3) NMPCAS: This algorithm uses the ASM for the optimization problem. It is implemented using MATLAB "fmincon" function.

The goal is to drive the OMR to follow a given trajectory by minimizing an objective function. Constrained and unconstrained cases will be considered. All methods minimize the same objective function: (4) for the unconstrained case and (16) for the constrained case. In our implementation, we employ a prediction horizon $N_P = 6$, and control horizon $N_c = 3$. The state and input weighting matrices are chosen as $R_k = diag(0.01, 0.01, 0.01)$ and $Q_k = Q_0 = diag(75, 75, 75, 5, 5, 5)$ to balance tracking accuracy and actuation effort. We note that the term control horizon $N_c$ does not appear in the objective functions. It is the number of variables optimized at each iteration which is less or equal to $N_p$.

For the Resilient Propagation algorithm, the update parameters are set to $\eta^+ = 1.6$, $\eta^- = 0.5$, with initial step sizes bounded by $\Delta_{\max} = 15$ and $\Delta_{\min} = 10^{-10}$. The convergence threshold and maximum iterations are $\delta = 10^{-6}$ and $I_{\max} = 20$ respectively.

*Remark 3:* All NMPC and RCPROP parameters were determined via a systematic trial-and-error procedure: starting from literature-informed initial guesses, we adjusted each parameter individually while monitoring closed-loop tracking error and solver convergence speed. Once a satisfactory operating point was found, we performed a sensitivity analysis by perturbing parameters by $\pm 10\%$ and observed negligible performance degradation, confirming robustness of the selected values.

**Algorithm 1** RCPROP

**while** $iter \leq I_{\max}$ **(Initialize iteration loop)**
**(check convergence)**
**if** $\max\left(\frac{\partial H_k}{\partial u_k}\right) \leq \delta$ **then return** $u_k$
**(evaluate the effect of last update)**
**if** $H_k > H_{k-1} - \delta \min(\Delta_{i,k-1})$ **then**
  **if** $\max(\Delta_{i,k-1} \leq \Delta_{\min})$ **then return** $u_{k-1}$
  **For** each $u_{i,k}$ **do**
    **(global backtracking)**
    $u_{i,k} = u_{i,k-1}$ ; $\frac{\partial H_k}{\partial u_i} = \frac{\partial H_{k-1}}{\partial u_i}$
    $\Delta_{i,k} = \max\left(\eta^- \Delta_{i,k-1}, \Delta_{\min}\right)$
**else**
**(update step sizes)**
  **for** each $u_{i,k}$ **do**
$$\Delta_{i,k} = \begin{cases} \min\left(\eta^+ \Delta_{i,k-1}, \Delta_{\max}\right) & \text{if } \frac{\partial H_{k-1}}{\partial u_i}\frac{\partial H_k}{\partial u_i} > 0 \\ \max\left(\eta^- \Delta_{i,k-1}, \Delta_{\min}\right) & \text{if } \frac{\partial H_{k-1}}{\partial u_i}\frac{\partial H_k}{\partial u_i} < 0 \\ \Delta_{i,k-1} & \text{otherwise} \end{cases}$$
  **(update optimization variables)**
  **for** each $u_{i,k}$ **do**
    $u_{i,k+1} = u_{i,k} - \text{sign}\left(\frac{\partial H_k}{\partial u_i}\right) . \Delta_{i,k}$

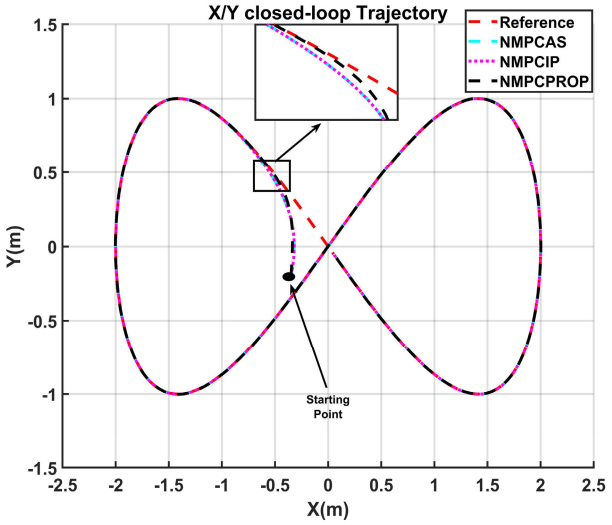



**FIGURE 2.** Tracking the eight-shaped trajectory: Unconstrained case.

An eight-shaped trajectory was selected as a benchmark test, offering a challenging scenario that incorporates continuous curvature and direction changes. It's described by the following equations:

$$\begin{aligned} x_d &= -2\sin((2\pi/P)t), \\ y_d &= 1\sin(2(2\pi/P)t), \\ v_{xd} &= \dot{x}_d,\ v_{yd} = \dot{y}_d, \end{aligned} \tag{21}$$

where $P = 24$ is the trajectory period. The desired orientation and rotational speed are selected to be zero and the initial position is set to $q_0 = [-0.35, -0.2, 0, 0, 0, 0]$. In the constrained case, constraints are imposed on both speed

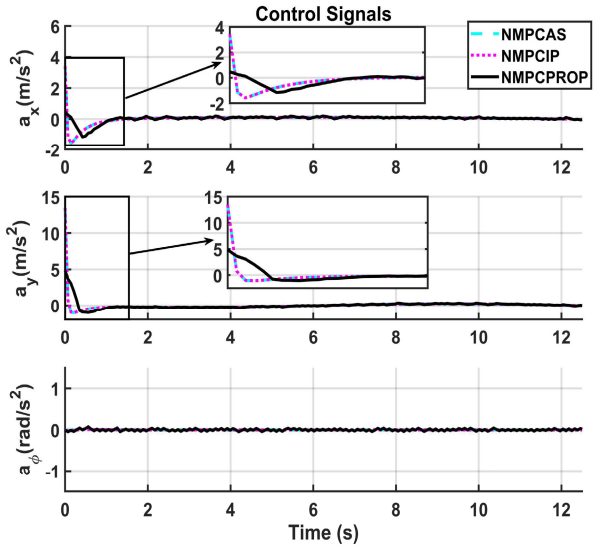



**FIGURE 3.** Optimized control signals: Unconstrained case.

**TABLE 1.** Maximum time per iteration, average time per iteration, time ratios and root mean square for different strategies: Unconstrained case.

| | | NMPCPROP | NMPCIP | NMPCAS |
|---|---|---|---|---|
| Max time/iteration | | 0.0278 | 0.3352 | 0.1992 |
| Max time ratio | | 1 | 12 | 7 |
| Average time/iteration | | 0.0014 | 0.0244 | 0.0121 |
| Average time ratio | | 1 | 17 | 8 |
| $M_x$ (m) | | 0.0356 | 0.0349 | 0.0349 |
| $M_y$ (m) | | 0.0314 | 0.0251 | 0.0251 |
| $M_\theta$ (rad) | | 0.0013 | 0.0006 | 0.0006 |

**TABLE 2.** Maximum time per iteration, average time per iteration, time ratios and root mean square for different strategies: Constrained case.

| | NMPCPROP | NMPCIP | NMPCAS |
|---|---|---|---|
| Max time/iteration | 0.0492 | 0.5452 | 0.2023 |
| Max time ratio | 1 | 11 | 4 |
| Average time/iteration | 0.0047 | 0.0311 | 0.0122 |
| Average time ratio | 1 | 6 | 2 |
| $M_x$ (m) | 0.0361 | 0.0373 | 0.0373 |
| $M_y$ (m) | 0.0609 | 0.0636 | 0.0636 |
| $M_\theta$ (rad) | 0.0018 | 0.0013 | 0.0013 |

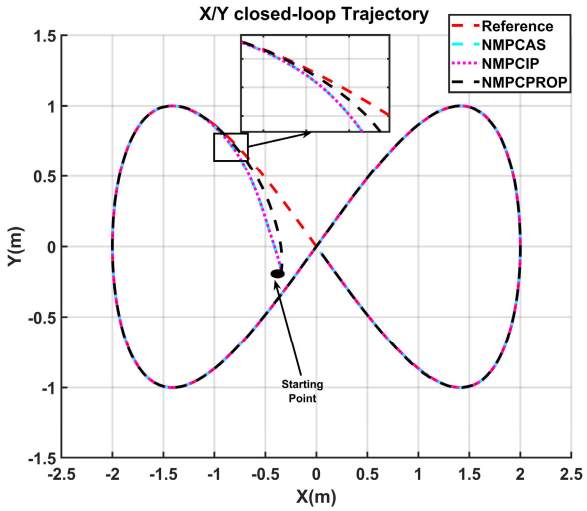



**FIGURE 4.** Tracking the eight-shaped trajectory: Constrained case.

and acceleration. The maximum speeds are set to $v_{\max} = 2$ m/s and $\omega_{\max} = 3$ rad/s, and the maximum accelerations to $a_{t\max} = 1$ m/s$^2$ and $a_{n\max} = 2$ rad/s$^2$, which reflect the limitations on the real robot. The tracking performance of the compared strategies is evaluated using the Root Mean Square (RMS) performance index given as follows:

$$M_i = \sqrt{\frac{\sum_{j=1}^{T_{sim}/T} \left(y_{ref}(j) - y_{sys}(j)\right)^2}{T_{sim}/T}}; \; i \in \{x, y, \theta\}. \tag{22}$$

### B. TRACKING ANALYSIS AND COMPUTATIONAL RESOURCES

The simulation results of the tracking problem are illustrated in the figures and tables below.

Figure 2 shows the performance of the three methods when tracking the eight-shaped trajectory without constraints. It is evident that NMPCPROP achieved faster convergence compared to the other two methods. The optimized control signals are shown in Figure 3. It can be appreciated that the initial control effort resulted using NMPCPROP is notably lower compared to the initial efforts of NMPCIP and NMPCAS.

Table 1 presents the average and maximum time required to complete one single iteration for each of the compared strategies, along with the RMS values computed from equation for (22) the unconstrained case.

It is notable that NMPCPROP exhibits a significant advantage in terms of computational cost, completing iterations 12 to 17 times faster than NMPCIP and 7 to 8 times faster than NMPCAS, making it highly suitable for real-time applications. Despite this substantial reduction in computational burden, NMPCPROP achieves a tracking performance remarkably close to NMPCIP and NMPCAS, with its RMS tracking error only slightly higher—by an order of $10^{-3}$. This slight discrepancy is acceptable and well justified given the considerable gain in computational efficiency.

Figure 4 shows the tracking performance of the compared methods when following the eight-shaped trajectory in the presence of constraints. We can clearly see that all three methods exhibit a decrease in convergence speed compared to the unconstrained scenario. This deceleration is expected,

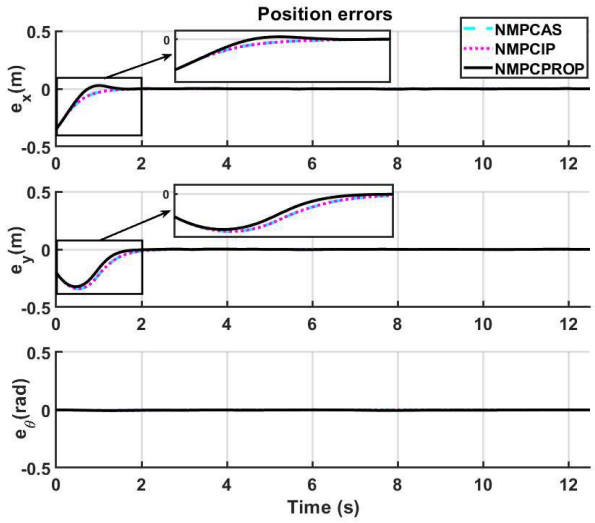



**FIGURE 5.** Position tracking errors: Constrained case.

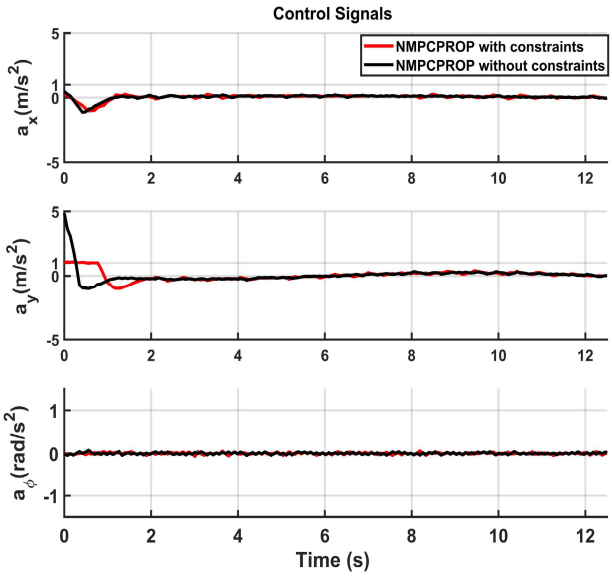



**FIGURE 6.** Comparison of control signals in constrained and unconstrained cases.

given the restrictions imposed on control inputs, resulting in slower convergence. Nonetheless, NMPCPROP maintains faster convergence than the other two methods in the constrained case.

Table 2 presents the performance under constraints, where NMPCPROP maintained its efficiency, operating 6 to 11 times faster than NMPCIP and 2 to 4 times faster than NMPCAS. Furthermore, NMPCPROP outperformed the other two methods in tracking accuracy for both x and y positions, confirming its effectiveness in constrained trajectory tracking scenarios. Figure 5 visualises the tracking errors for the constrained case, where the faster convergence of NMPCPROP can be seen.

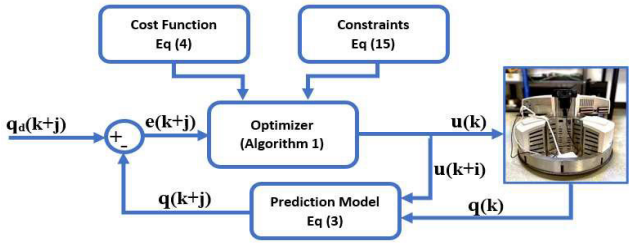



**FIGURE 7.** Overview of the trajectory tracking experiment.

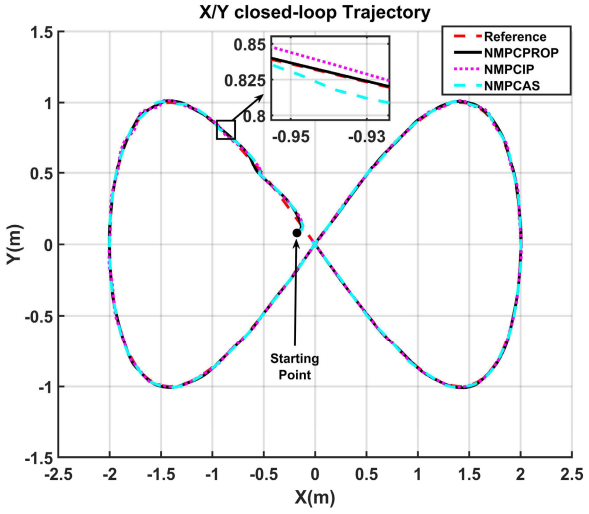



**FIGURE 8.** Real-time tracking of the eight-shaped trajectory.

The capability of RPROP to handle constrained optimization problem online has not been tested before. Therefore, Figure 6 shoes the performance of NMPCPROP under constraints. It managed to bring the control amplitudes under the maximum allowed values, after they were violated in the unconstrained case, while keeping fast convergence and good tracking performance.

### C. EXPERIMENTAL RESULTS

This section provides the experimental results of testing the three methods on the Robotino-Festo robot (Figure 7). It is a three-wheeled OMR. Each wheel has its individual motor which is controlled by a 32-bit microcontroller that uses PWM signals to actuate these DC motors using a PID. The entire system is controlled by a higher embedded PC with Intel i5, 2.4 GHz dual core, 8 GB RAM and 23 GB SSD.

The transition ratio between the drive shaft and omni-wheels is 32:1 which is achieved using a planetary gear unit [48]. The maximum translational and rotational speeds of the robot are 2m/s and 2rad/s respectively. A MATLAB-Simulink toolbox is used to implement the algorithms. The robot expects the translational and rotational speeds as inputs; therefore, integrators were included in the algorithms. The robot requires updates every 70ms, a timeframe deemed adequate for the application of all three methods, ensuring a

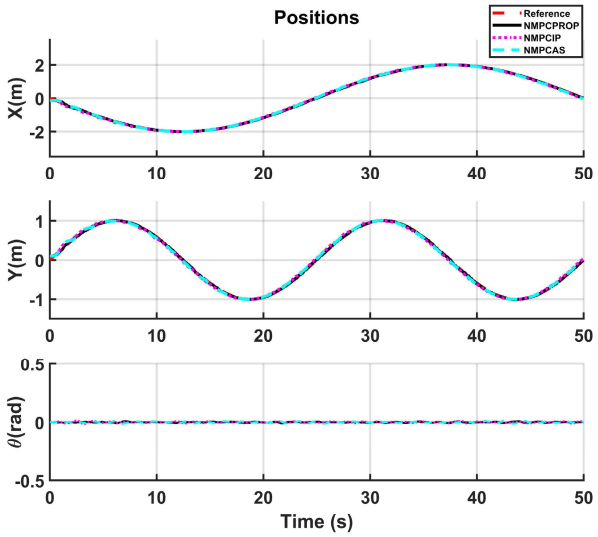



**FIGURE 9.** Real-time position tracking.

**TABLE 3.** Root mean square for different strategies: Real-time experiment.

| | NMPCPROP | NMPCIP | NMPCAS |
|---|---|---|---|
| $M_x$ (m) | 0.0169 | 0.0372 | 0.0307 |
| $M_y$ (m) | 0.0140 | 0.0347 | 0.0275 |
| $M_\theta$ (rad) | 0.0034 | 0.0049 | 0.0052 |

fair comparison in real-time scenarios. In the first experiment, we aim to drive the robot to follow the eight-shaped trajectory given in (21) where $P = 50$. The initial position is chosen to be $q_0 = [-0.15,\ 0.1,\ 0,\ 0, 0, 0]$. The penalization matrices for NMPCPROP are set as follows:

$$R_k = diag(4, 4, 4),$$
$$Q_k = Q_0 = diag(100, 100, 100, 5, 5, 8),$$

and for NMPCIP and NMPCAS, the penalization matrices are selected as:

$$R_{nk} = diag(1, 1, 1),$$
$$Q_{nk} = Q_{n0} = diag(750, 750, 750, 10, 10, 10).$$

Figure 8 shows the real-time tracking performance of the faster convergence compared to the other methods which take a longer to converge due to the higher computational time required in the transitory phase. Position tracking and velocity tracking are given in Figures 9 and 10 respectively. While the position tracking performance of three methods was similar, the superiority of the NMPCPROP is clear in speed tracking thanks to its computational advantage over the other two methods. The position tracking errors are displayed in Figure 11 where the faster convergence of NMPCPROP can be seen. Table 3 provides RMS errors from real-time experiments. The results highlight that NMPCPROP reduced

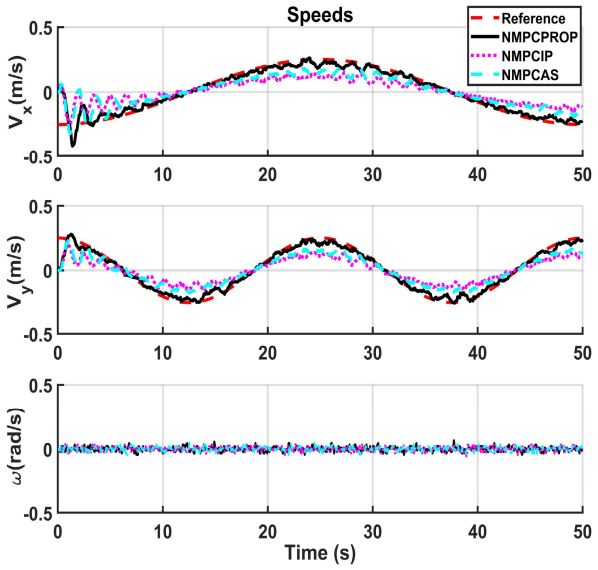



**FIGURE 10.** Real-time velocities tracking.

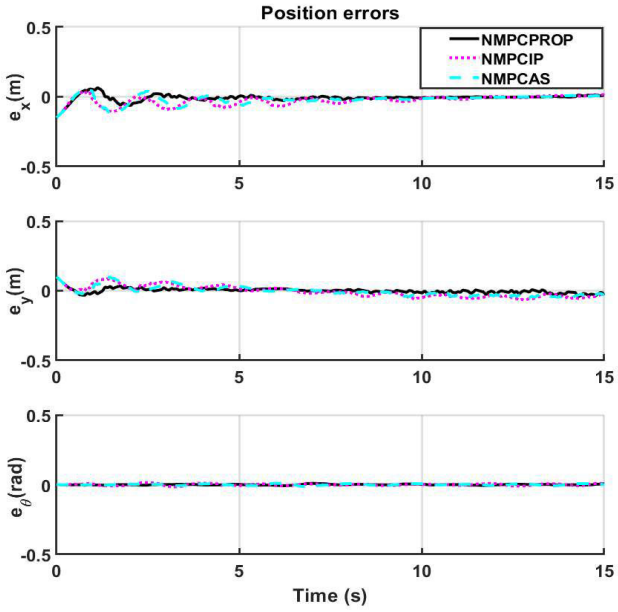



**FIGURE 11.** Real-time position tracking errors.

tracking errors by approximately 50% compared to NMPCIP and NMPCAS. Additionally, NMPCPROP achieved faster convergence and better velocity tracking, validating its suitability for real-time control applications.

In the second experiment, the aim is to track an eight-shaped trajectory with time-variant orientation given as follows:

$$x_d = 0.5\sin((4\pi/P)t),$$
$$y_d = 1.5\sin((2\pi/P)t),$$
$$\theta_d = \text{atan}\,(\dot{y}_d/\dot{x}_d) + k\pi,\ k = 0, 1,$$

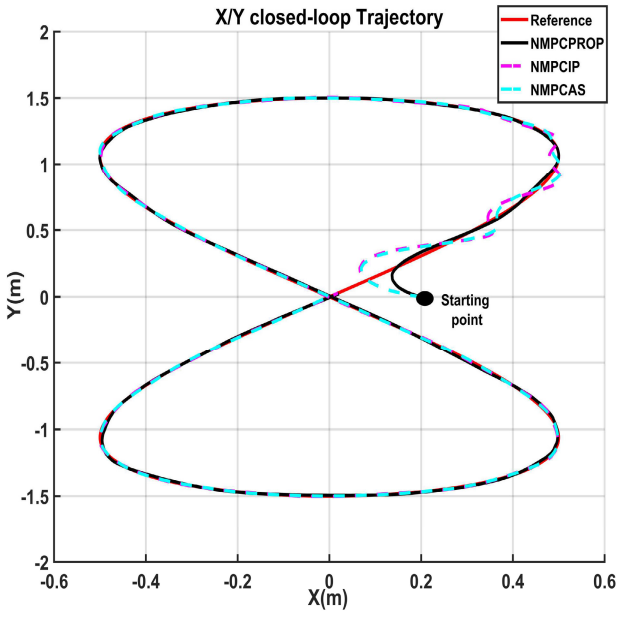



**FIGURE 12.** Tracking of the eight-shaped trajectory (time-variant orientation).

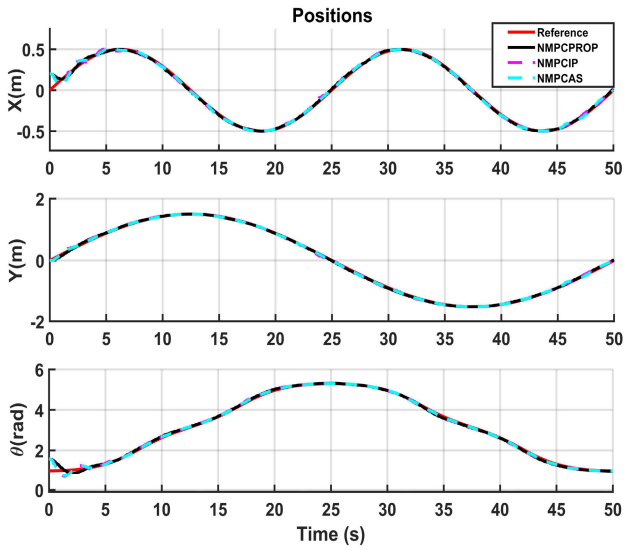



**FIGURE 13.** Position tracking (time-variant orientation).

$$v_{xd} = \sqrt{\dot{x}_d^2 + \dot{y}_d^2}, \; v_{yd} = 0, \tag{23}$$

where $P = 50$. The initial states are set to:

$$q_0 = [0.2, \; 0, \; \pi/2, \; 0, 0, 0]. \tag{24}$$

Figure 12 illustrates the performance of the three methods when tracking the eight-shaped trajectory. NMPCPROP demonstrated faster convergence compared to NMPCIP and NMPCAS. Furthermore, Figures 13 and 14 depict the positions and speeds tracking, respectively. The superior performance of NMPCPROP is evident, particularly in the

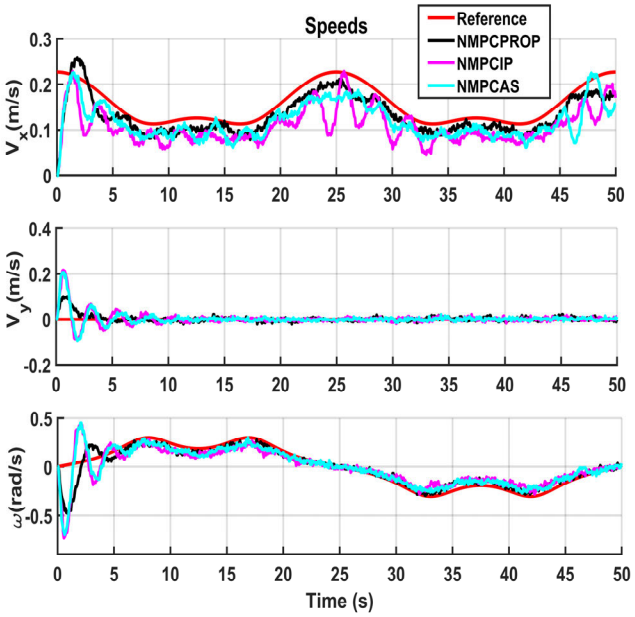



**FIGURE 14.** Velocities tracking (time-variant orientation).

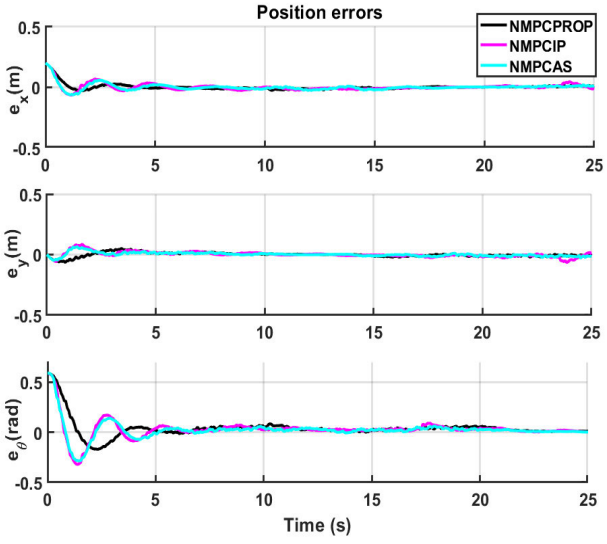



**FIGURE 15.** Position tracking errors (time-variant orientation).

velocities, where it exhibits better tracking compared to the other two methods.

The position tracking errors are presented in Figure 15, and the RMS errors are given in Table 4. It is evident that NMPCPROP not only attained faster convergence but also maintained comparable position errors with the other two methods.

In our real-time experiments we encountered several practical challenges: transient deviations resulting form floor irregularities and wheel slip, occasional control lag due to solver convergence limits, and degraded state estimation caused by sensor noise, bias, and latency. To address these

**TABLE 4. Root mean square for different strategies: Real-time experiment (time-variant orientation).**

| | NMPCPROP | NMPCIP | NMPCAS |
|---|---|---|---|
| $M_x$ (m) | 0.0207 | 0.0230 | 0.0212 |
| $M_y$ (m) | 0.0173 | 0.0197 | 0.0156 |
| $M_\theta$ (rad) | 0.0802 | 0.0766 | 0.0716 |

issues, this work can be extended to incorporate robust or adaptive control schemes that compensate for parameter shifts, automate filtering and outlier rejection to improve sensor data quality, and allow real-time adjustment of optimization settings. These enhancements would further narrow the gap between idealized simulation and real-world deployment, ensuring reliable trajectory tracking under realistic operating conditions.

## V. CONCLUSION AND FUTURE WORK

In this study, a NMPC controller for the trajectory tracking problem of mobile robots was presented. The proposed controller used RCPROP algorithm to solve the optimization problem leading to rapid convergence, precise tracking, and low computational burden. The capability of this algorithm to solve constraint problems during online optimization was successfully demonstrated.

To validate the performance of the proposed controller, comparison studies were conducted against benchmark methods, namely Interior Point and Active Set. Simulation and experimental results proved that NMPCPROP outperformed both NMPCIP and NMPCAS in terms of computational efficiency, convergence speed and in real-time tracking performance. The results of this study highlighted the superiority of NMPCPROP in practical applications.

However, some limitations remain. While the controller achieves high tracking accuracy and efficiency under nominal conditions, its robustness to model uncertainties and external disturbances has not been extensively evaluated. Moreover, the current parameter tuning was done manually, which may limit adaptability and scalability across different operating scenarios. These limitations highlight opportunities for further improvement.

In future work, we aim to evaluate the robustness of the proposed method and explore the use of automated tuning algorithms to optimize the tuning process.

## REFERENCES

[1] K. Zhang, J. Wang, X. Xin, X. Li, C. Sun, J. Huang, and W. Kong, "A survey on learning-based model predictive control: Toward path tracking control of mobile platforms," *Appl. Sci.*, vol. 12, no. 4, p. 1995, Feb. 2022.

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

[3] R. Raj and A. Kos, "A comprehensive study of mobile robot: History, developments, applications, and future research perspectives," *Appl. Sci.*, vol. 12, no. 14, p. 6951, Jul. 2022.

[4] B. Fernandez, P. J. Herrera, and J. A. Cerrada, "A simplified optimal path following controller for an agricultural skid-steering robot," *IEEE Access*, vol. 7, pp. 95932–95940, 2019.

[5] D. Topolsky, I. Topolskaya, I. Plaksina, P. Shaburov, N. Yumagulov, D. Fedorov, and E. Zvereva, "Development of a mobile robot for mine exploration," *Processes*, vol. 10, no. 5, p. 865, Apr. 2022.

[6] J. Szrek, J. Jakubiak, and R. Zimroz, "A mobile robot-based system for automatic inspection of belt conveyors in mining industry," *Energies*, vol. 15, no. 1, p. 327, Jan. 2022.

[7] I. Rangapur, B. K. S. Prasad, and R. Suresh, "Design and development of spherical spy robot for surveillance operation," *Proc. Comput. Sci.*, vol. 171, no. 4, pp. 1212–1220, Jan. 2020.

[8] A. J. Sathyamoorthy, U. Patel, M. Paul, Y. Savle, and D. Manocha, "COVID surveillance robot: Monitoring social distancing constraints in indoor scenarios," *PLoS ONE*, vol. 16, no. 12, Dec. 2021, Art. no. e0259713.

[9] J. Zhong, C. Ling, A. Cangelosi, A. Lotfi, and X. Liu, "On the gap between domestic robotic applications and computational intelligence," *Electronics*, vol. 10, no. 7, p. 793, Mar. 2021.

[10] E. E. Turhanlar, B. Y. Ekren, and T. Lerher, "Autonomous mobile robot travel under deadlock and collision prevention algorithms by agent-based modelling in warehouses," *Int. J. Logistics Res. Appl.*, vol. 27, no. 8, pp. 1–20, Aug. 2024.

[11] H. Kim and Y. Choi, "Autonomous driving robot that drives and returns along a planned route in underground mines by recognizing road signs," *Appl. Sci.*, vol. 11, no. 21, p. 10235, Nov. 2021.

[12] H. Hewawasam, M. Y. Ibrahim, and G. K. Appuhamillage, "Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments," *IEEE Open J. Ind. Electron. Soc.*, vol. 3, no. 1, pp. 353–365, Aug. 2022.

[13] F. Tian, R. Zhou, Z. Li, L. Li, Y. Gao, D. Cao, and L. Chen, "Trajectory planning for autonomous mining trucks considering terrain constraints," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 4, pp. 772–786, Dec. 2021.

[14] Z. Chen, "Path planning method for underground survey robot in dangerous scenarios," *Highlights Sci., Eng. Technol.*, vol. 43, pp. 207–214, Apr. 2023.

[15] N. Hassan and A. Saleem, "Analysis of trajectory tracking control algorithms for wheeled mobile robots," in *Proc. IEEE Ind. Electron. Appl. Conf. (IEACon)*, Nov. 2021, pp. 236–241.

[16] Z. Xu, S. X. Yang, and S. A. Gadsden, "Enhanced bioinspired backstepping control for a mobile robot with unscented Kalman filter," *IEEE Access*, vol. 8, pp. 125899–125908, 2020.

[17] Z. Li and J. Zhai, "Super-twisting sliding mode trajectory tracking adaptive control of wheeled mobile robots with disturbance observer," *Int. J. Robust Nonlinear Control*, vol. 32, no. 18, pp. 9869–9881, Dec. 2022.

[18] J. A. Rodríguez-Arellano, R. Miranda-Colorado, L. T. Aguilar, and M. A. Negrete-Villanueva, "Trajectory tracking nonlinear $H_\infty$ controller for wheeled mobile robots with disturbances observer," *ISA Trans.*, vol. 142, pp. 372–385, Jul. 2023.

[19] R. Miranda-Colorado, "Observer-based proportional integral derivative control for trajectory tracking of wheeled mobile robots with kinematic disturbances," *Appl. Math. Comput.*, vol. 432, Nov. 2022, Art. no. 127372.

[20] A. Khadhraoui, A. Zouaoui, and M. Saad, "Barrier Lyapunov function and adaptive backstepping-based control of a quadrotor UAV," *Robotica*, vol. 41, no. 10, pp. 1–23, Oct. 2023.

[21] K. Shao, J. Zheng, R. Tang, X. Li, Z. Man, and B. Liang, "Barrier function based adaptive sliding mode control for uncertain systems with input saturation," *IEEE/ASME Trans. Mechatronics*, vol. 27, no. 6, pp. 4258–4268, Dec. 2022.

[22] J. Moreno-Valenzuela, "Constrained trajectory tracking control of a mobile robot by limited integrator anti-windup," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1422–1426, Mar. 2022.

[23] J. Moreno-Valenzuela, L. Montoya-Villegas, R. Pérez-Alcocer, and J. Sandoval, "A family of saturated controllers for UWMRs," *ISA Trans.*, vol. 100, pp. 495–509, May 2020.

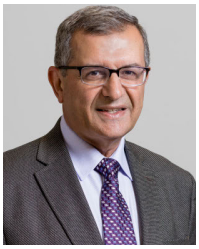
[24] O. Tutsoy, D. Asadi, K. Ahmadi, S. Y. Nabavi-Chashmi, and J. Iqbal, "Minimum distance and minimum time optimal path planning with bioinspired machine learning algorithms for faulty unmanned air vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 8, pp. 9069–9077, Aug. 2024.

[25] R. H. Abiyev, N. Akkaya, and I. Gunsel, "Control of omnidirectional robot using Z-number-based fuzzy system," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 1, pp. 238–252, Jan. 2019.

[26] P. Harasim and M. Trojnacki, "State of the art in predictive control of wheeled mobile robots," *J. Autom., Mobile Robot. Intell. Syst.*, vol. 10, no. 1, pp. 34–42, Feb. 2016.
[27] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control*. Cham, Switzerland: Springer, 2017, pp. 45–69.
[28] K. Wang, 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," 2023, *arXiv:2302.08039*.
[29] J. Wei and B. Zhu, "Model predictive control for trajectory-tracking and formation of wheeled mobile robots," *Neural Comput. Appl.*, vol. 34, no. 19, pp. 16351–16365, Oct. 2022.
[30] M. El-Sayyah, M. R. Saad, and M. Saad, "Enhanced MPC for omnidirectional robot motion tracking using Laguerre functions and non-iterative linearization," *IEEE Access*, vol. 10, pp. 118290–118301, 2022.
[31] C. Müller, D. E. Quevedo, and G. C. Goodwin, "How good is quantized model predictive control with horizon one?" *IEEE Trans. Autom. Control*, vol. 56, no. 11, pp. 2623–2638, Nov. 2011.
[32] M. H. Korayem, H. R. Adriani, and N. Y. Lademakhi, "Intelligent time-delay reduction of nonlinear model predictive control (NMPC) for wheeled mobile robots in the presence of obstacles," *ISA Trans.*, vol. 141, pp. 414–427, Oct. 2023.
[33] Y. Hamada, T. Tsukamoto, and S. Ishimoto, "Receding horizon guidance of a small unmanned aerial vehicle for planar reference path following," *Aerosp. Sci. Technol.*, vol. 77, pp. 129–137, Jun. 2018.
[34] T. Ohtsuka, "A tutorial on C/GMRES and automatic code generation for nonlinear model predictive control," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2015, pp. 73–86.
[35] Y. Hu, H. Su, J. Fu, H. R. Karimi, G. Ferrigno, E. D. Momi, and A. Knoll, "Nonlinear model predictive control for mobile medical robot using neural optimization," *IEEE Trans. Ind. Electron.*, vol. 68, no. 12, pp. 12636–12645, Dec. 2021.
[36] Z. Li, C. Yang, C.-Y. Su, J. Deng, and W. Zhang, "Vision-based model predictive control for steering of a nonholonomic mobile robot," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 2, pp. 553–564, Mar. 2016.
[37] H. Merabti, K. Belarbi, and B. Bouchemal, "Nonlinear predictive control of a mobile robot: A solution using metaheuristcs," *J. Chin. Inst. Eng.*, vol. 39, no. 3, pp. 282–290, Apr. 2016.
[38] N. Ito, H. Okuda, and T. Suzuki, "Model predictive driving for tractor-trailer mobile robot with an omni-directional tractor," in *Proc. 59th Annu. Conf. Soc. Instrum. Control Eng. Jpn. (SICE)*, Sep. 2020, pp. 1530–1533.
[39] R. G. Patel and J. J. Trivedi, "Nonlinear model predictive control of steam-assisted-gravity-drainage well operations for real-time production optimization," *SPE Prod. Oper.*, vol. 35, no. 3, pp. 564–578, Aug. 2020.
[40] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.
[41] R. Quirynen and S. Di Cairano, "PRESAS: Block-structured preconditioning of iterative solvers within a primal active-set method for fast model predictive control," *Optim. Control Appl. Methods*, vol. 41, no. 6, pp. 2282–2307, Nov. 2020.
[42] J. C. L. Barreto S., A. G. S. Conceição, C. E. T. Dórea, L. Martinez, and E. R. de Pieri, "Design and implementation of model-predictive control with friction compensation on an omnidirectional mobile robot," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 2, pp. 467–476, Apr. 2014.
[43] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, Mar. 1993, pp. 586–591.
[44] T. M. Bailey, "Convergence of Rprop and variants," *Neurocomputing*, vol. 159, pp. 90–95, Jul. 2015.
[45] A. D. Anastasiadis, G. D. Magoulas, and M. N. Vrahatis, "New globally convergent training scheme based on the resilient propagation algorithm," *Neurocomputing*, vol. 64, pp. 253–270, Mar. 2005.
[46] T. Nascimento and M. Saska, "Embedded fast nonlinear model predictive control for micro aerial vehicles," *J. Intell. Robot. Syst.*, vol. 103, no. 4, pp. 1–11, Dec. 2021.
[47] N. Guo, B. Lenzo, X. Zhang, Y. Zou, R. Zhai, and T. Zhang, "A real-time nonlinear model predictive controller for yaw motion optimization of distributed drive electric vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 4935–4946, May 2020.
[48] D. Pršić, V. Stojanović, and V. Đordević, "A constructive approach to teaching with robotino," in *Proc. 7th Int. Sci. Conf. Technics Inform. Educ.*, Serbia, Balkans, 2018, pp. 273–278.

**MAHMOUD EL-SAYYAH** received the B.S. and M.S. degrees in instrumentation, control and cyber-systems technologies from Lebanese University, Tripoli, Lebanon, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the Université de Québec en Abitibi-Témiscamingue (UQAT), QC, Canada. His research interests include nonlinear, adaptative and robust control theories, and their application to robotics and autonomous systems.

**MOHAMAD R. SAAD** (Senior Member, IEEE) received the Ph.D. degree in electrical engineering and control systems from the École Polytechnique de Montréal, in 2001. Then, he joined the Performance Systems Department, CAE Inc., as a Flight Control Specialist. In 2006, he joined the School of Engineering, Université du Québec en Abitibi-Témiscamingue (UQAT). His research interest includes nonlinear control applied to robotic systems.



**MAAROUF SAAD** (Senior Member, IEEE) received the bachelor's and master's degrees in electrical engineering from École Polytechnique de Montréal, in 1982 and 1984, respectively, and the Ph.D. degree in electrical engineering from McGill University, in 1988. He joined École de Technologie Supérieure, in 1987, where he is teaching control theory and robotics courses. His research is mainly in nonlinear control and optimization applied to robotics and autonomous systems.