Journal of Field Robotics

WILEY

RESEARCH ARTICLE OPEN ACCESS

Real-time Nonlinear Model Predictive Control of a Robotic Arm Using Spatial Operator Algebra Theory

Tuğçe Yaren[1] | Selçuk Kizir[2]

[1]Department of Electrical and Electronic Engineering, Balikesir University, Balikesir, Turkey | [2]Department of Mechatronic Engineering, Kocaeli University, Kocaeli, Turkey

Correspondence: Tuğçe Yaren (tugce.yaren@balikesir.edu.tr)

Received: 8 November 2023 | **Revised:** 9 November 2024 | **Accepted:** 5 January 2025

Funding: This work was supported by the Scientific Research Projects Coordination Unit of Kocaeli University under grant numbers #2483 and #4253.

Keywords: nonlinear model predictive control | online optimization | real-time | robotics | spatial operator algebra

ABSTRACT

Nonlinear model predictive control (NMPC) has inherent challenges, such as high computational burden, nonconvex optimization, and the necessity of powerful and fast processors with large memory for real-time robotics. In this study, a new NMPC strategy is proposed using Spatial Operator Algebra (SOA) theory to address these challenges, and experimental results are presented for the five degrees of freedom robot manipulator. The proposed scheme is based on an NMPC controller using the SOA-based dynamic model to provide good tracking performance and ensure the satisfaction of constraints. Two novel control schemes, SOA–NMPC and SOA–NMPC proportional-derivative (PD), are introduced for a comprehensive analysis of the proposed innovative approach. The validity of the proposed scheme is experimentally tested through robustness analysis conducted across various tasks, including the addition of weight and exposure to internal/external disturbances. The effectiveness of the proposed approach is demonstrated through benchmarking against NE-NMPC using the Newton–Euler (NE) algorithm, classical MPC, MPC-PD, and PID techniques. The comparative results show that the SOA–NMPC controller provides effective performance and ensures constraints for the entire trajectory of the manipulator, even under varying conditions.

1 | Introduction

Model predictive control (MPC) is a very powerful control strategy used in the control of multivariable constrained systems. This method is very different from classical controllers and requires high design skills (Mayne 2014). Nonlinear model predictive control (NMPC) is an approach that uses an internal nonlinear model to predict the future behavior of the system, allowing deliberate control actions. There are many advantages of MPC over traditional control techniques, such as (i) it can be applied directly to linear and nonlinear systems, (ii) it is easy to implement for single-variable/multivariable systems, (iii) it can incorporate constraints and perform cost minimization, and (iv) it has intuitive tuning parameters (Grüne and Pannek 2017; Schwenzer et al. 2021; Lee 2011). However, one of the major challenges in the real-time implementation of NMPC controllers is the significant computational burden, as NMPC involves iterative optimization at each time step (Chemori, Kouki, and Bouani 2023). In addition, the reliance on accurate models for prediction can introduce errors due to model mismatches, uncertainties, or disturbances, thereby undermining the controller's performance. The nonconvex nature of the NMPC optimization problem arises from its incorporation of nonlinear constraints to describe model dynamics. Consequently, solving this problem often necessitates initialization and can pose challenges in identifying globally optimal solutions (Polak 1997).

There are many published techniques proposed to overcome these challenges, such as using simplified models (Krämer et al. 2020), linearizing (Bettega and Richiedei 2023; Narasingam, Son, and Kwon 2023), using fast solver methods (Chemori, Kouki, and

Bouani 2023), developing hybrid methods (Yang, Xu, and Yao 2023), offline computation (Bruder et al. 2021), and kinematics-decomposition (Reinhold, Baumann, and Meurer 2023). The proposed techniques (Polak 1997; Zhao, Mohan, and Vasudevan 2017) to solve nonconvex cost functions still take several seconds per iteration, which makes them too slow to be implemented in most real-time control applications (Narasingam, Son, and Kwon 2023). Integrating the decomposition of the differential kinematics of an anthropomorphic robot into the controller was proposed by Reinhold, Baumann, and Meurer (2023) to significantly reduce the computational costs of NMPC. This was validated through numerical simulations. Wu, Chen, and Xie (2019) suggested the implementation of NMPC with an economic objective function to decrease computational demands. Nevertheless, the efficacy of this approach was solely validated through numerical simulations. In Chaber (2020), the nonlinear optimization of NMPC was explored, leveraging a neural network for implementation within an embedded system. However, the optimization problem reformulation as a convex quadratic programming problem featuring elusive nonlinear terms was eventually addressed via a simplified dual-network solution. Carron et al. (2019) introduced an innovative scheme that combined inverse dynamics feedback linearization and a data-driven error model integrated within an MPC framework. The effectiveness of this algorithm was demonstrated through enhanced tracking performance on a robotic arm, comparing the performance with that of two other MPC schemes and a proportional, integral, and derivative (PID) controller.

Yang et al. proposed a multiloop composite control scheme comprising an inner loop centered around inverse dynamics control and an outer loop employing a set of distinct disturbance-observer-based tube model predictive composite controllers. The efficacy of this control scheme was validated through simulation experiments conducted on a PUMA 560 robotic manipulator (Yang, Xu, and Yao 2023). Krämer et al. (2020) analyzed an MPC scheme based on a hypergraph for online trajectory optimization of collaborative robots while the manipulator dynamics were neglected, and kinematic optimization problems were solved online. Bettega and Richiedei (2023) proposed a formulation of MPC, based on linearized models to address the challenge of tip trajectory tracking for time-varying references in a two-link, multibody system characterized as an underactuated and nonminimum phase. Narasingam, Son, and Kwon (2023) presented a Koopman-based MPC approach that integrated Koopman operator theory with Lyapunov-based MPC techniques by using Koopman eigenfunctions and nonlinear dynamics transform from state–space to function space. Nevertheless, it is worth noting that simplified or linearized models, while aiding in computational tractability, may inadvertently impose constraints on the robot's capabilities, thereby constraining the spectrum of attainable motions (Kleff et al. 2021). In addition, these models can have negative implications in terms of robustness when model–plant mismatches and uncertainties are present. Such scenarios, where the model does not precisely capture real-world conditions, can limit the ability of control strategies to exhibit the desired performance. The challenges and significance of considering the full dynamics in complex motion generation have previously been acknowledged (Kleff et al. 2021; Orin, Goswami, and Lee 2013; Wieber 2006).

In this study, a controller scheme that can be applied online using a nonlinear model is proposed, and its validity has been investigated. The aim was to develop an NMPC that can adapt to real-world conditions under the influence of model uncertainties with full dynamics by exploiting the advantages of Spatial Operator Algebra (SOA). SOA, one of the recursive methods based on spatial vector representation, was introduced to the literature in 1991 by Rodriguez, Jain, and Kreutz-Delgado (1991) and Rodriguez (1987) at NASA Jet Propulsion Laboratory. SOA is a high-performance algorithm used to model systems with a high number of degrees of freedom (DOFs), including solid and flexible components, underactuated systems, flexible joints, serial chains, and tree structures (Ozakyol, Karaman, and Bingul 2019; The Spatial Operator Algebra 2020). Since the SOA-based algorithm does not rely on gradient-based derivatives and is not affected by discontinuities or sudden changes in trajectory functions, it is well-suited for real-time programming and control of robotic systems (Ozakyol, Karaman, and Bingul 2017). One of its most significant innovations is the reduction of mathematical complexity, which is accomplished by minimizing the number of symbols that necessitate comprehension and analysis, even in highly intricate systems. In addition, SOA-based algorithms rely on a coordinate-free vector representation, allowing the designer the flexibility to select a set of free axes in the configuration, which greatly simplifies the design process. The SOA algorithm can achieve shorter cycle times, enabling a more efficient and powerful control of robot arms and robotic systems. Therefore, the use of SOA in robot control simultaneously enhances both robustness and computational speed (Yıldız 2014; Dong et al. 2018). In the literature, SOA has been explored for its capability to provide a unified framework for modeling and controlling complex systems, particularly those with multibody dynamics (Meldrum, Franklin, and Wiktor 1993; Wensing, Palmer, and Orin 2015; Hopler and Thummel 2004; Featherstone 1987; Nakanishi, Mistry, and Schaal 2007). In Yaren and Kizir (2024), the advantages and disadvantages of SOA are discussed, and its efficiency in control applications is evaluated. The study concluded that an SOA-based controller significantly reduced computational costs. Moreover, compared with a Newton–Euler (NE) based controller, it exhibited approximately 60%–70% faster performance. The main advantage of using SOA is computational efficiency in dynamic modeling. In addition, SOA is more straightforward and user-friendly compared with the NE method, which makes it highly suitable for real-time robotic control applications.

SOA algorithm has never been proposed in the NMPC, according to the best of the author's knowledge. This research presents a novel approach in which the significant computational burden, a major challenge in real-time applications of NMPC, has been reduced by capitalizing on the shorter cycle times enabled by SOA. Using the SOA nonlinear dynamic model, controller predictions are made over full dynamics without the need for model simplification/linearization. In this case, it is expected that the performance of the controller will not be limited by model mismatches, uncertainties, or disturbances.

Two control schemes, SOA–NMPC and SOA–NMPC proportional-derivative (PD), are introduced in the present study for a comprehensive analysis of the proposed approach. The validity of the proposed scheme was experimentally tested using a 5-DOF robot manipulator, with robustness analysis conducted across various tasks, including the addition of weight and exposure to internal and external disturbances. To gain a better understanding of the performance of the proposed method and to provide a metric for its

evaluation, a benchmark was established by comparing the proposed control applications with classical methods. The control methods compared include NE-NMPC designed using the NE algorithm, classical MPC, MPC-PD, and PID methods. Moreover, this comparison allowed the assessment of the potential advantages, areas for improvement, and contributions to industrial applications of the proposed method in greater detail.

This research presents three key contributions. First, it introduces a novel algorithm using the SOA dynamic model with NMPC, enabling precise tracking while addressing nonlinear dynamics and incorporating constraints for control inputs and states, thereby enhancing controller versatility. Second, it proposes an extended SOA–NMPC controller that integrates a PD term, allowing states predicted by SOA to improve the controller's sensitivity to dynamic changes. Third, comprehensive laboratory experiments and benchmark tests validate the effectiveness of the proposed method across various scenarios, even in the presence of model mismatches, uncertainties, and disturbances. Additionally, the proposed control scheme offers flexibility, allowing customization to achieve various desired behaviors by incorporating supplementary terms into the objective function. By highlighting the potential of SOA in complex robotic systems, the study encourages its adoption and suggests promising outcomes for multi-DOF systems and collaborative manipulators.

The remainder of this paper is organized as follows. The nonlinear dynamic modeling of the manipulator using SOA is presented in Section 2. Problem formulation and the SOA algorithm-based NMPC design approach are introduced in Section 3. The simulation and experimental results are compared and discussed in Section 4. Finally, the general conclusions and future work are presented in Section 5.

2 | Robotic Manipulator: Description and Modeling

To test the proposed and developed controller schemes, a practical and low-cost 5-DOF robotic manipulator was designed and produced in the laboratory. The computer-aided design view of the manipulator is shown in Figure 1. The first three joints are equipped with RMD-X8 Pro actuators, which are composed of a brushless direct current motor, high-precision planetary reducer, 18-bit magnetic encoder, and a field-oriented control driver for position, speed, and torque control. It has a peak torque of 40 Nm and supports controller area network (CAN) communication up to 1 Mbps. The last two joints are configured with a digital servo (QDS-15RO) motor. Joint connections were produced from three-dimensional (3D) printers using acrylonitrile butadiene styrene material.

2.1 | Robot Kinematics

The kinematic and dynamic models of the manipulator were obtained by SOA, a high-performance computational algorithm. Figure 2 provides representations of the calculations of link velocities and accelerations in each iteration, the computation of the Jacobian matrix, as well as forward and inverse velocity kinematics.

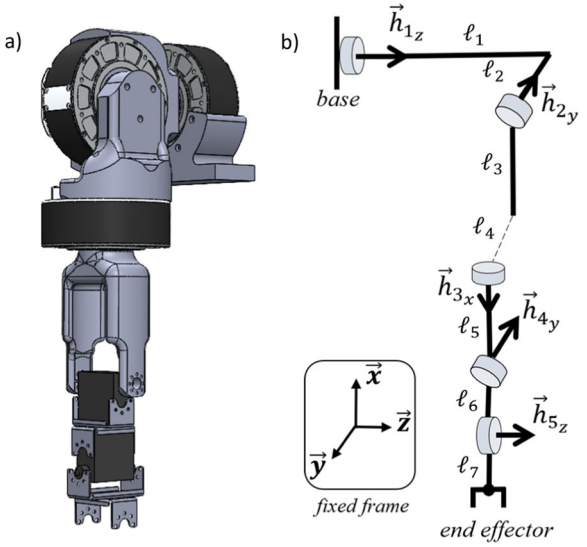

FIGURE 1 | (a) Computer-aided design view of the five degrees of freedom robotic manipulator and (b) kinematic parameters of the manipulator.

The actuated joint coordinates are defined by vector $q = [q_1, q_2, q_3, q_4, q_5]^T$. Joint positions q define the configuration of the manipulator. The relationship between this configuration vector and the end effector's pose can be determined through a kinematic analysis. In SOA, the kinematic relationship is defined using six-dimensional (6D) vectors whose three dimensions are angular velocities, and the other three dimensions are linear velocities.

The frame assignment for the 5-DOF manipulator is shown in Figure 1. The exact locations of the placed frames are the center of mass of the relevant link. The equations and spatial vectors are chosen in the initial configuration of the manipulator.

The link lengths and propagation operators for the robot above can be written as

$$\begin{aligned} \vec{\ell}_{0,1} &= [0 \;\; 0 \;\; 0]^T, \;\; \vec{\ell}_{1,2} = [0 \;\; -\ell_2 \;\; \ell_1]^T, \\ \vec{\ell}_{2,3} &= [-\ell_3 \;\; \ell_4 \;\; 0]^T, \;\; \vec{\ell}_{3,4} = [-\ell_5 \;\; 0 \;\; 0]^T, \\ \vec{\ell}_{4,5} &= [-\ell_6 \;\; 0 \;\; 0]^T, \;\; \vec{\ell}_{5,e} = [-\ell_7 \;\; 0 \;\; 0]^T, \end{aligned} \tag{1}$$

where the link lengths are $\ell_1 = 0.0625$, $\ell_2 = 0.0235$, $\ell_3 = 0.11365$, $\ell_4 = 0.0195$, $\ell_5 = 0.1105$, $\ell_6 = 0.08$, $\ell_7 = 0.0416$ m. The axes of rotation vectors are

$$\begin{aligned} \vec{h}_1 &= [0 \;\; 0 \;\; 1]^T, \;\; \vec{h}_2 = [0 \;\; -1 \;\; 0]^T, \;\; \vec{h}_3 = [-1 \;\; 0 \;\; 0]^T, \\ \vec{h}_4 &= [0 \;\; -1 \;\; 0]^T, \;\; \vec{h}_5 = [0 \;\; 0 \;\; 1]^T. \end{aligned} \tag{2}$$

The axis of rotation matrices are

$$\vec{\vec{H}}_k = [\vec{h}_k \;\; \vec{0}]^T, \;\; k = 1, ..., 5. \tag{3}$$

Using the given kinematic parameters of the manipulator, the kinematic model is obtained according to the flow diagram in Figure 2.

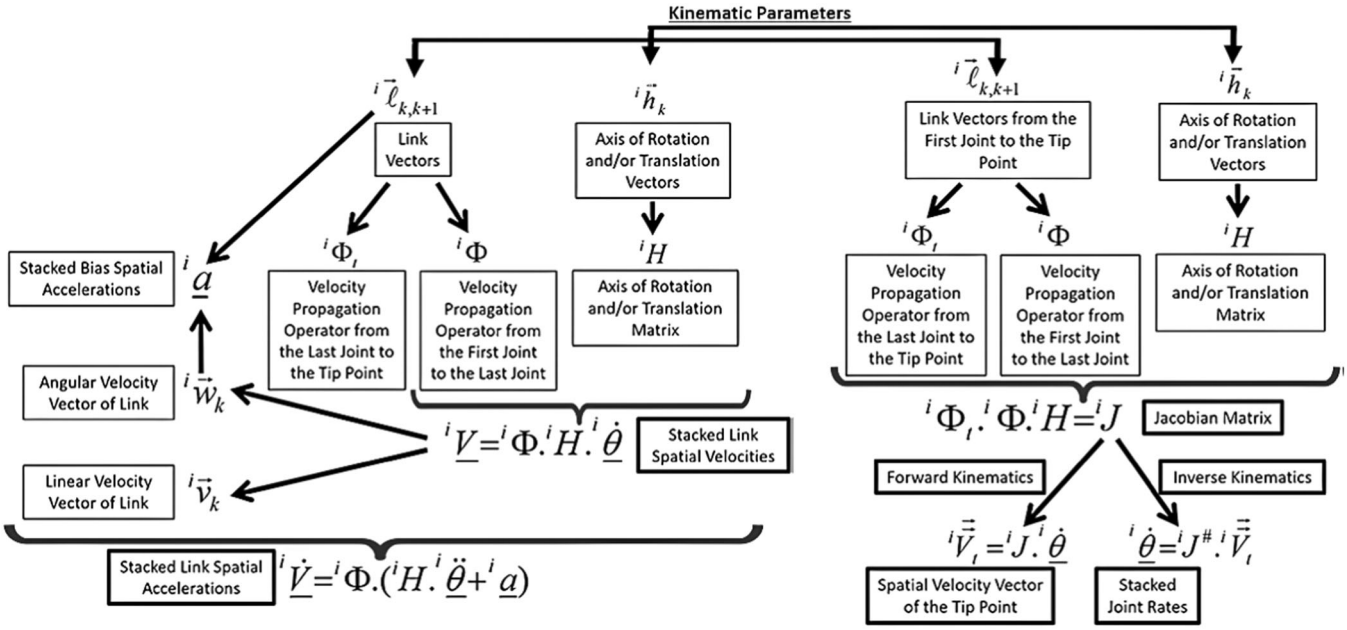

FIGURE 2 | Forward and inverse kinematic analysis with Spatial Operator Algebra (Ozakyol, Karaman, and Bingul 2019).

2.2 | Nonlinear Dynamic Model of the Manipulator

For dynamic analysis, it is necessary to examine the torque and force distributions of the links. The 6D spatial force transfer from the $k+1$ link to the k link is performed as follows:

$$\vec{\vec{F}}_k = \varnothing^T_{k+1,k}\vec{\vec{F}}_{k+1} + M_k\vec{\vec{V}}_k + \vec{\vec{b}}_k, \tag{4}$$

where the parameters of this equation are outlined in Table 1.

The torque applied to the joints is calculated using the following equation:

$$\underline{\tau} = \boldsymbol{M}\ddot{\underline{\theta}} + \underline{C} + J^T\vec{\vec{F}}_t. \tag{5}$$

Here,

$$\begin{aligned} \boldsymbol{M} &= H^T\varnothing^T M\varnothing H, \\ \underline{C} &= H^T\varnothing^T(M\varnothing\underline{a} + \underline{b}), \\ J^T &= H^T\varnothing^T\varnothing_t^T, \end{aligned} \tag{6}$$

where $\underline{\tau}$ are the torques that occur at the joints due to the acceleration values given to the joints and the forces applied to the endpoint, $\boldsymbol{M}$ is the generalized mass matrix, $\underline{C}$ represents the bias terms that include Coriolis and gravity, $\vec{\vec{F}}_t$ are the spatial forces at the tip point of the manipulator, $\underline{a}$ is the stacked bias spatial accelerations of the manipulator, and $\underline{b}$ is the stacked bias spatial forces of the manipulator.

The mass properties of the manipulator are given in Table 2. According to Tables 1 and 2, link mass matrices (M_k, $k = 1,2,3$) are obtained. By calculating the link mass matrices, the manipulator mass matrix can be calculated as

TABLE 1 | Parameter description of the spatial force equation.

Parameter	Definition	Equivalent
$\vec{\vec{F}}_k$	Six-dimensional vector of spatial forces at the origin of link k	$\begin{bmatrix}\vec{\tau}_k \\ \vec{f}_k\end{bmatrix}$
M_k	Mass matrix of link k	$\begin{bmatrix} I_k & m_k\hat{\ell}_{k,c} \\ -m_k\hat{\ell}_{k,c} & I_3 m_k\end{bmatrix}$
$\vec{\vec{b}}_k$	Bias spatial forces of link k	$\begin{bmatrix}\vec{\omega}_k \times I_k\vec{\omega}_k \\ m_k\vec{\omega}_k \times (\vec{\omega}_k \times \vec{\ell}_{k,c})\end{bmatrix}$

$$M = \begin{bmatrix} M_1 & 0_6 & 0_6 \\ 0_6 & M_2 & 0_6 \\ 0_6 & 0_6 & M_3 \end{bmatrix}. \tag{7}$$

The recursive process of the inverse dynamics algorithm is shown in Table 3. Rodrigues' formula, which plays a significant role in the SOA recursive dynamic process, is used to efficiently calculate the rotation of a vector in 3D space given an axis and angle of rotation.

$$e^{\hat{\omega}\theta} = I + \hat{\omega}\ \sin\theta + \hat{\omega}^2(1 - \cos\theta). \tag{8}$$

As seen from Equation (8), Rodrigues' formula provides an efficient method to compute $e^{\hat{\omega}\theta}$, which is the rotation matrix expressing a rotation by θ about axis ω. A rotation that can be represented with nine parameters in a 3×3 rotation matrix can be expressed using Rodrigues' formula with a 3D vector $\vec{\omega}$ and a rotation angle θ, totaling four parameters. Rodrigues' formula is an essential tool in SOA for the simple, efficient, and understandable representation and calculation of 3D rotations. The effect of the rotation is incorporated into the algorithm using

TABLE 2 | Mass properties of the manipulator (dynamic parameters of the manipulator).

		Inertia (I_k)					
	Mass (m_k)	I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{yz}	I_{zx}
Link 1	0.77412	837,720.17	1,252,091.31	809,756.36	43.0543.05	−14,981.81	4.80
Link 2	0.78018	1,087,199.53	1,087,199.53	1,129,158.39	−159,520.21	4.06	62.44
Link 3	0.12449	62,771.32	81,874.85	122,408.01	0.13	0.32	3.33
Link 4	0.15402	33,558.79	89,956.63	89,956.63	−3397.29	−178.78	3397.29
Link 5	0.09778	27,006.55	32,783.35	35,658.14	−42.71	10.12	−1249.17

Note: m_k (kg) and I_k (g mm^2).

TABLE 3 | Recursive dynamic process of the Spatial Operator Algebra.

The inverse dynamics process
Require: Rodrigues' formula (R)
Calculate: The effects of rotation on the axes of the rotation vector, link length, and link center of mass vectors
Require: Axis of rotation vectors
Calculate: Axis of the rotation matrix
Require: Link vectors
Calculate: The manipulator propagation matrix
Require: Link vectors, axis of rotation vectors
Calculate: Angular and linear velocities
Require: System parameters (mass, length, inertia matrix)
Calculate: Mass matrix of the manipulator
Require: System parameters (mass, length, inertia matrix, and center of mass) and angular velocities
Calculate: Stacked bias spatial accelerations and stacked bias spatial forces
Calculate: Generalized mass matrix from Equation (6)
Calculate: Bias terms from Equation (6)
Calculate: Jacobian matrix from Equation (6)
Calculate: The torque applied to the joints from Equation (5)

the Rodrigues formula, and the SOA algorithm operates iteratively, as shown in Table 3.

Using the dynamic parameters of the manipulator and all the system parameters obtained, the nonlinear dynamic model is obtained according to the SOA algorithm. The dynamic model presented in Equation (5) is well-suited for joint-space control design and serves as the foundation for the proposed controller scheme. It is noteworthy that although the proposed control scheme was designed for 5-DOF manipulators, it can be easily generalized to n-DOF manipulators via SOA and still remains valid. As SOA uses a recursive approach and coordinate-free vector notation, it can efficiently and rapidly model complex serial robots with high DOFs. These robots may incorporate various joint types, such as universal, spherical, prismatic, or rotational, each potentially having multiple DOFs, and can have either a mobile or a stationary base. Consequently, it becomes feasible to implement and experimentally validate our proposed SOA-based controller schemes with any other system by leveraging the identified recursive dynamic process.

3 | Nonlinear Model Predictive Control

MPC is a very powerful control strategy used to control multivariable constrained systems. It differs significantly from classical control methods and demands advanced design skills (Mayne 2014). MPC not only aims to track the system output to a reference input value but also strives to keep the process within predetermined system limits. By solving a constrained optimization problem at each sampling time, the future control sequence is obtained. MPC is basically a process where an optimal control problem is solved at each step.

The fundamental principle of MPC is to use the system model to minimize a given cost function over a future prediction horizon, aiming to compute the optimal input signal. The control signal is then applied by using the first element of the computed control signal throughout the control horizon. The goal is to minimize the cost function and achieve optimal control performance based on the system model and predicted future behavior.

In MPC, there are multiple variations depending on factors such as the type of model used (state–space, transfer function), linear or nonlinear model structure, linear or nonlinear constraints, and whether the controller itself is linear or nonlinear. These variations allow for flexibility in designing and implementing predictive control strategies that best suit the specific system requirements and characteristics. The choice of model type, linearity, and constraint formulation depends on the complexity and nature of the controlled system, as well as the desired control performance.

NMPC is an optimization-based technique used to control systems with nonlinear feedback. The application areas it prioritizes are stability and tracking problems. If there is a nonlinear system, nonlinear constraints, and nonlinear cost function in the control application, NMPC should be used for system control, as a linear MPC will not be capable of solving the control problem. This method is among the most powerful as it uses the most accurate representation of the system, that is, a nonlinear system model. Therefore, the predictions made by the controller are more accurate, leading to better control actions. However, it

is one of the most difficult problems to solve in real-time, because a nonconvex optimization problem is generated if there are nonlinear constraints and a nonlinear cost function. Therefore, the cost function may have more than one local optimum, and it may be difficult to find the global optimum.

3.1 | Control Scheme

An NMPC controller is proposed for the system in Equation (5) that exploits the SOA-based model to provide good tracking performance and ensure the satisfaction of state and control input constraints. After the basic SOA–NMPC structure has been proven to provide the expected advantages, different controller structures were constructed based on the identified system model.

The designed NMPC has 10 states, 5 outputs, and 5 inputs. For the 5-DOF robot manipulator system, we consider the torques $u\epsilon\mathcal{R}^5$ as system inputs, the joint positions $q\epsilon\mathcal{R}^5$ as system outputs, and the joint velocities and positions $q, \dot{q}\,\epsilon\mathcal{R}^5$ as system states.

For the control objective, the optimization problem is formulated as follows:

$$\min_{u} \mathcal{J} = \sum_{k=0}^{p} (r_k - q_k)^T Q (r_k - q_k) + u_{k-1}^T R u_{k-1} + q_k^T Q_q q_k + \dot{q}_k^T Q_v \dot{q}_k \tag{9a}$$

$$\begin{aligned} &\text{s.t.} \quad x_k = [0], \quad k = 0, \\ &\dot{x}_k = \mathcal{F}(x_k, u_k), \quad k\epsilon[0, p], \\ &x_L \le x_k \le x_H, \\ &u_L \le u_k \le u_H, \end{aligned} \tag{9b}$$

where $\mathcal{J}$ is the cost function that depends on the inputs (u), joint positions (q), joint velocities ($\dot{q}$), and reference trajectory (r). The matrices Q, Q_q, Q_v, and R are constant weight matrices that are positive-semidefinite. The objective function, represented by $\mathcal{J}$, incorporates penalties for various factors. It penalizes high joint velocities, deviations of joint positions from the home position, control signal magnitudes, and errors of trajectory following. The inclusion of a penalty for trajectory following errors is essential to achieve precise trajectory tracking. Penalizing high joint velocities helps prevent excessive input energy consumption and oscillations. In addition, penalizing high joint position values ensures that the joints remain within their constraints, resulting in a smoother and more natural motion.

The prediction horizon (p) defines the number of future time steps to be predicted. L and H denote the lower and upper limits of the states and control inputs in Equation (9b), respectively, which are defined as constraint functions. The optimization problem in Equation (9a) must be solved at each sampling time (k), yielding a sequence of optimal control law as $\{u^*(k|k), ..., u^*(k + c|k)\}$. The number of control moves to time step c is called the control horizon.

Constraints are very important in real-time applications because violating them can lead to undesired consequences. Considering that the amplitude of input torques is limited, one of the constraints is

$$\tau_{tL} \le \tau_t \le \tau_{tH}, \tag{10}$$

where τ_{tL} and τ_{tH} stand for the lower and upper bounds of torque of the servo actuator module. In addition, considering the physical limits of the robot joints, the following constraints should be included in the controller:

$$q_{tL} \le q_t \le q_{tH}, \tag{11}$$

where q_{tL} and q_{tH} are the lower and the upper bounds of the joint's positions, respectively, which are not identical for all joints. Furthermore, the following constraints of the joint velocities must be considered because of safety, stability, and actuator limitations:

$$\dot{q}_{tL} \le \dot{q}_t \le \dot{q}_{tH}, \tag{12}$$

where $\dot{q}_{tL}$ and $\dot{q}_{tH}$ are the lower and the upper bounds of the joint's velocity, respectively.

To begin determining constraints, it is essential to understand the mathematical model of the system thoroughly. This involves defining the state variables, control inputs, and system dynamics equations. State constraints should reflect the physical limits of the system, such as maximum and minimum positions and velocities, ensuring operational safety by preventing unsafe states, such as collision zones. Performance requirements should also be considered, ensuring accuracy and acceptable response times. Moreover, environmental factors, such as external disturbances, must be taken into account.

To understand the process of effectively setting and tuning the state constraints, let us assume that the state variables of a general control system are defined as velocity and position. The mechanical design and actuator specifications determine both position and velocity limits, thereby identifying the system's physical limits. For instance, joint limits may be established within a range of $[-\pi, \pi]$ radians. In this case, the position state constraints are set directly to these values, and since the physical constraints are hard constraints, no tuning is performed. Operating the motor beyond the specified maximum angle can result in mechanical damage or negatively impact its performance. On the other hand, the velocity state constraints are values that can be tuned. When tuning, the maximum velocity of the actuator should be considered first; however, it is advisable to reduce the maximum speeds by 10%–15% for safety. Additionally, in scenarios like precision assembly and manufacturing, where the desired control problem necessitates the robot's operation at low speeds, it is advisable to decrease the velocity constraints. The behavior of the closed-loop system is then observed until the expected performance is achieved, with the velocity constraints gradually decreased to ensure the system operates effectively within the specified limits.

For input constraints, it is important to consider the physical capabilities of actuators, including their maximum and minimum inputs, power, and energy limits, to prevent overheating or excessive power consumption. Control bandwidth should be respected to avoid instability or oscillatory behavior, and

saturation constraints should be implemented to maintain control performance. Safety and reliability are paramount, so constraints should include fail-safes to prevent inputs from driving the system into unsafe conditions and ensure redundancy and fault tolerance.

To set and tune input constraints effectively, the process typically begins by identifying the actuator's maximum and minimum operating thresholds, such as voltage, current, or force. Once these physical limits are defined, the next step is to establish safe operating margins, ensuring the system remains within limits under dynamic conditions. Tuning involves gradually adjusting these constraints based on the system's behavior in real-world or simulated environments. For instance, one can initially set inputs close to their maximum allowable values and then progressively scale them down to optimize performance, all while maintaining stability and safety. Continuous monitoring of system responses, such as actuator temperature, power consumption, and control accuracy, helps fine-tune the constraints to ensure optimal performance across varying conditions without compromising the integrity or longevity of the system. By adhering to these guidelines summarized in Table 4, system constraints can be determined for a robust and reliable control system.

By incorporating these constraints into the cost function, a balance is established between optimizing performance and respecting the physical limitations of the robot, leading to a more robust and reliable control solution. To solve the NMPC problem under the constraints, the *fmincon* solver with the SQP algorithm was used in the present study.

In NMPC, achieving a good and efficient structure is more challenging compared with the linear case. Solving Equation (9) quickly enough for real-time NMPC is often difficult. The optimal control inputs are computed offline, depending on the system and implementation. Herein, a real-time, applicable, and practical NMPC scheme has been proposed for advanced systems that can overcome these shortcomings.

3.2 | Proposed Schemes: SOA–NMPC and SOA–NMPC PD

The block diagram, given in Figure 3, illustrates two different schemes. The first is the SOA–NMPC controller, which was obtained by integrating the SOA dynamic model into the NMPC structure. This innovative control scheme facilitates the generation of anticipatory predictions by leveraging the dynamics inherent in the SOA dynamic model, thereby establishing a robust foundation for informed prediction within the control process. This approach enhances real-time adaptability by aligning SOA model predictions closely with actual system responses, optimizing trajectory tracking, and minimizing deviations. Integrating the SOA algorithm into the NMPC framework has the potential to improve control performance, enhance stability, and increase responsiveness, resulting in a more reliable and efficient control solution for diverse and complex operational scenarios. The recursive algorithm given in Table 3 operates online within the NMPC structure, generating predictions based on this algorithm.

The second proposed scheme, we have termed SOA–NMPC PD, emerged with the addition of the term PD to the SOA–NMPC structure. The reference of the PD is the position states predicted by optimization through the SOA model. This approach offers several advantages. First, it enhances the accuracy of the reference trajectory, enabling the controller to respond more precisely to dynamic changes. Second, by incorporating the insights provided by the SOA model, the controller achieves improved adaptability in complex and nonlinear environments. Finally, this method optimizes trajectory tracking, leading to enhanced control performance and the ability to effectively mitigate disturbances, resulting in more stable and efficient control outputs.

3.2.1 | Stability Analysis

NMPC stability depends on several factors, such as constraint handling, prediction horizon, online optimization, and model accuracy. Stability is ensured through rigorous analysis and tuning during the controller's design phase. In the proposed controller design, the cost function, constraints, and prediction-control horizon have been tuned to ensure the stability of the controller. In addition, the closed-loop stability of the SOA–NMPC was analyzed using Lyapunov's direct method. The theorem requires identifying a function, known as the Lyapunov function $\mathcal{V}(x)$, which must fulfill three conditions to prove stability. These are (1) $\mathcal{V}$ is positive definite: $\mathcal{V}(x) > 0$ for all $x \neq 0$ and $\mathcal{V}(0) = 0$; (2) $\mathcal{V}$ is radially unbounded: If $\|x\| \to \infty$,

TABLE 4 | The guidelines for setting state constraints and input constraints.

Setting state constraints	Setting input constraints
Understanding the system dynamics	Understanding the system dynamics
-Modeling the system	-Modeling the System
-Define state variables	-Define control inputs
Physical limits of the system	Actuator limits
-Identify position and velocity limits	-Maximum and minimum inputs
-Operational safety	-Power and energy limits
Performance requirements	Safety and reliability
-Accuracy, rise time, and so forth	-Control bandwidth, saturation, and so forth
Environmental factors	Rate limits

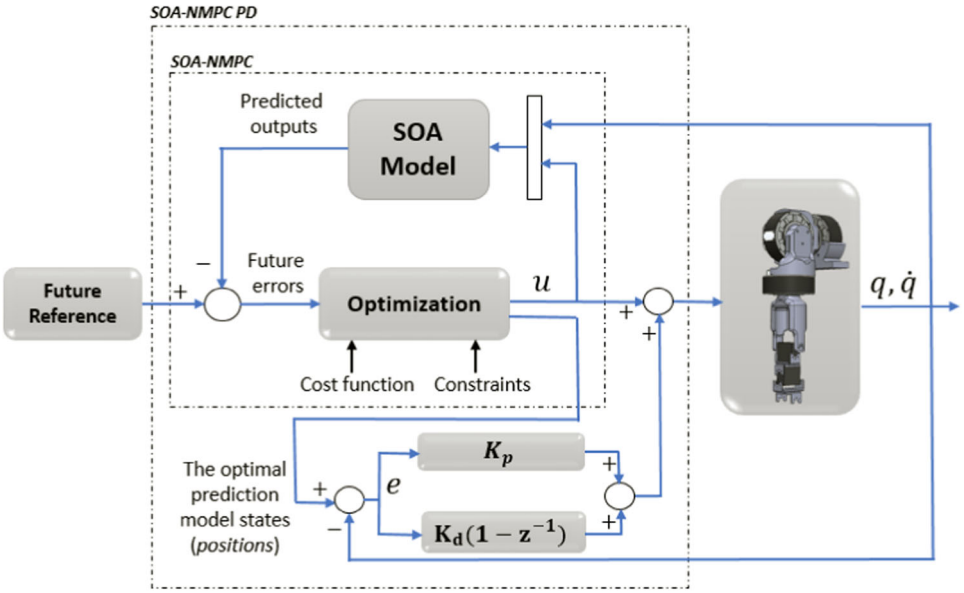

FIGURE 3 | The block diagram of the SOA–NMPC and SOA–NMPC PD. NMPC, nonlinear model predictive control; PD, proportional-derivative; SOA, Spatial Operator Algebra.

then $\mathcal{V}(x) \to \infty$; and (3) $\mathcal{V}$ decreases along trajectories: $\mathcal{V}(x(k+1)) - \mathcal{V}(x(k)) < 0$ and $\dot{\mathcal{V}}(x) < 0$.

On the basis of the cost function in Equation (9), the candidate Lyapunov function is defined as follows:

$$\mathcal{V}(x_k) = \min_{x,u} \sum_{j=k}^{N-1} F(x_j, u_j) = \min_{x,u} \sum_{j=k}^{N-1} (r_j - q_j)^T Q (r_j - q_j) + u_{j-1}^T R u_{j-1} + q_j^T Q_q q_j + \dot{q}_j^T Q_v \dot{q}_j. \tag{13}$$

The matrices Q, Q_q, Q_v, and R are the constant weight matrices that are positive-semidefinite, and thus the candidate function $\mathcal{V}$ satisfies the first and second conditions. For x_k, $\mathcal{V}$ takes the minimum value when the minimization problem over the N prediction horizon is solved, and the optimal variables are obtained as follows:

$$\mathcal{V}(x_k) \to \{u_k^*, ..., u_{k+N-1}^*, x_k^*, ..., x_{k+N}^*\} \tag{14}$$

In addition, all variables satisfy the constraints in $\mathcal{V}(x_k)$. For the next time step, a feasible point can be selected for the minimization problem in $\mathcal{V}(x_{k+1})$. The actual solution to the minimization problem will then be less than or equal to the value of $\mathcal{V}(x_{k+1})$ when using this feasible point, as shown in Equation (15), which proves the third condition.

$$\mathcal{V}(x_{k+1}) = \sum_{j=k+1}^{N+k-1} F\left(x_j^*, u_j^*\right) = \underbrace{\boxed{\sum_{j=k}^{N+k-1} F\left(x_j^*, u_j^*\right)}}_{\mathcal{V}(x_k)} - \underbrace{\boxed{F(x_k^*, u_k^*)}}_{>0}$$
$$\mathcal{V}(x_{k+1}) < \mathcal{V}(x_k) \tag{15}$$

Stability has been proven by demonstrating that the candidate Lyapunov function, employed as the cost function within the NMPC framework, satisfies all the conditions required of a Lyapunov function.

4 | Experimental Results

4.1 | Physical Setup

The validity of the SOA-based controller scheme was analyzed by examining whether successful tracking performance can be achieved while ensuring compliance with specified constraints. Besides, the SOA-based controller schemes and basic control methods were benchmarked on a 5-DOF manipulator.

The control architectures were implemented using the *Simulink Desktop Real-time* platform running on a laptop personal computer (PC) with an Intel Core i7-11800H 2.30 GHz processor without parallelization. The controller runs on a PC, and the motors and the PC communicate over a 32-bit Arm Cortex-M4-based microcontroller unit (MCU). Communication between the PC and the MCU is carried out with a baud rate of 1 Mbps and a sampling time of 0.005 s via universal asynchronous receiver/transmitter (UART), and between the MCU and the motors with a baud rate of 1 Mbps via CAN (see Figure 4).

The motor communication part was embedded in the MCU, while the host PC model was run on the Simulink Desktop Real-Time platform. As the manipulator completed its trajectory, online data recording was performed by the host PC. The algorithm ran on the PC, generating reasonable torque signals that were sent to the MCU via UART. The MCU then transmitted these torque signals to the motors via CAN, read the position information from the motor encoders, and sent it back to the host PC.

The first three joints of the manipulator were used in the torque mode, while the wrist joint was used in the position mode. The

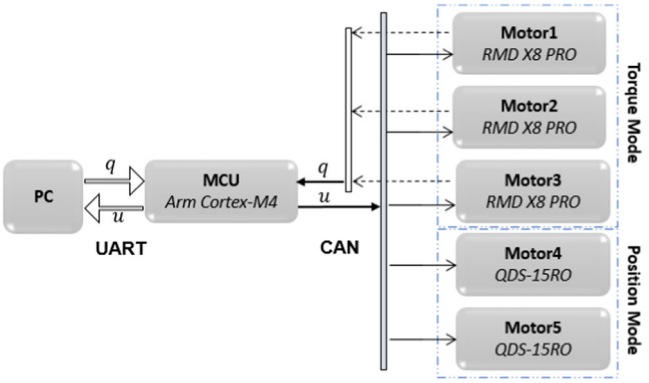

FIGURE 4 | Hardware detail of the physical setup. CAN, controller area network; MCU, microcontroller unit; PC, personal computer; UART, universal asynchronous receiver/transmitter.

controller algorithm was based on the SOA dynamic model with a 5-DOF manipulator. Therefore, the trajectory of the last two joints was obtained from the 5-DOF NMPC.

4.2 | SOA–NMPC Controller Scheme

The SOA–NMPC controller, based on the nonlinear model, solved the optimization problem as described in Equation (9). The controller operated in a closed loop with a 0.1 s sampling rate, a prediction horizon of 8, a control horizon of 1, and a prediction sampling rate of 0.05 s.

The control objective in this experiment was to steer the manipulator from the initial joint angular position $(0, 0, 0, 0, 0)$ to the goal angular position $(1, 1.3, 0.9, 0.785, 0.785)$ in the joint space while satisfying the constraints. The MPC parameters in the SOA–NMPC scheme are determined as follows: $Q = \text{diag}([1175\ 1125\ 325\ 30\ 30])$, $R = 1e-5$, $Q_q = \text{diag}([0.75\ \ 1\ \ 1\ \ 1\ \ 1])$, $Q_v = \text{diag}([1\ 1\ 1\ 1\ 1])$.

The constraints of the robot control system were defined following the guidelines presented in Table 4. The state constraints were determined based on the actuator's torque limits, ensuring that no control input exceeds the safe operational range of the motor. In our case, the torque values were limited to $[-20, 20]$ Nm for the first three joints and $[-10, 10]$ Nm for the last two joints, which correspond to the torque limits specified by the actuator manufacturers. On the basis of these limits, the minimum input constraint was set to $[-20, -20, -20, -10, -10]$, and the maximum input constraint was set to Yang, Xu, and Yao (2023) and Rodriguez, Jain, and Kreutz-Delgado (1991).

Depending on the system's physical limits, the minimum state constraint was set to $[-3, -0.2, -3, -3, -3, -2, -2, -2, -2, -2]$ and the maximum state constraint was set to Grüne and Pannek (2017) and Schwenzer et al. (2021). The -0.2 rad constraint for the second joint accounts for its limited range of movement. These hard constraints have been integrated into the controller, and no additional constraints were deemed necessary. However, constraints may vary depending on the control application. For example, a safety distance constraint should be added for obstacle avoidance applications.

The proposed SOA–NMPC controller scheme was first designed and developed with simulation studies and then implemented in real-time. The validity of the real-time experimental results was examined by comparing them with the simulation results. The comparison of the simulation and experimental test results of the SOA–NMPC controller can be seen in Figure 5. In the figure, the reference signals represent the target angular positions, which correspond to the final points of the trajectories. The NMPC algorithm automatically planned and adjusted the trajectory to reach these target angles. Also, Table 5 shows the control performances in terms of the cost of the tracking errors, and transient and steady-state response specifications. The cost functions are defined by the root mean squared error (RMSE) for this comparison.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(e(t))^2}, \qquad \text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|e(i)|, \tag{16}$$

where N denotes the number of recorded samples and e defines the tracking error of the related joint. On the basis of both tabular data and graphs, it was evident that the controller performance was satisfactory for both simulation and real-time experiments. Moreover, real-time results in terms of overshoot criteria were much more promising.

In addition, Figure 6a illustrates the real-time control signals (torque signals) with input constraints for the first three joints. From the figure, it can be easily verified from the actual control signal that the SOA–NMPC controller satisfied the input constraints. Besides, if we look at the actual velocity signals of the first three joints, it is also seen in Figure 6b that the velocity state constraint was also satisfied. Velocity signals were obtained using first-order derivative filters. According to the results shown in Figures 5 and 6, the controller performance was within the admissible limits. Satisfying both input and state constraints demonstrates the feasibility of the SOA–NMPC scheme in a real-time testbed.

4.3 | SOA–NMPC PD Controller Scheme

One of the other proposed controller schemes was the SOA–NMPC PD structure, illustrated by the block diagram in Figure 3. All controller parameters were identical to those of the SOA–NMPC. The validity of the proposed scheme was tested. PD terms are $P = [2.15\ \ 1.75\ \ 1.25]$, $D = [0.08\ \ 0.04\ \ 0.05]$.

The experimental results obtained for the controllers are shown in Figure 7a, and the control performance evaluation is reported in Table 6. Since the trajectories followed by the last two joints were equal, they were not included in the RMSE calculation. Figure 7a illustrates that the proposed SOA–NMPC PD controller outperforms the SOA–NMPC control strategy in terms of tracking performance. To quantitatively assess the improvement brought about by the proposed SOA–NMPC PD approach, the RMSE-based criterion introduced earlier was evaluated for each controller, and the results are summarized in Table 6. It is evident from the table that the SOA–NMPC PD control strategy featuring PD feedback gains effectively reduced tracking errors across all joints, providing better results. There was an improvement of 10.24% in the tracking error of joint 1, 16.02% tracking error of joint 2, and 18.71% in the tracking error of joint 3. The transient responses of the

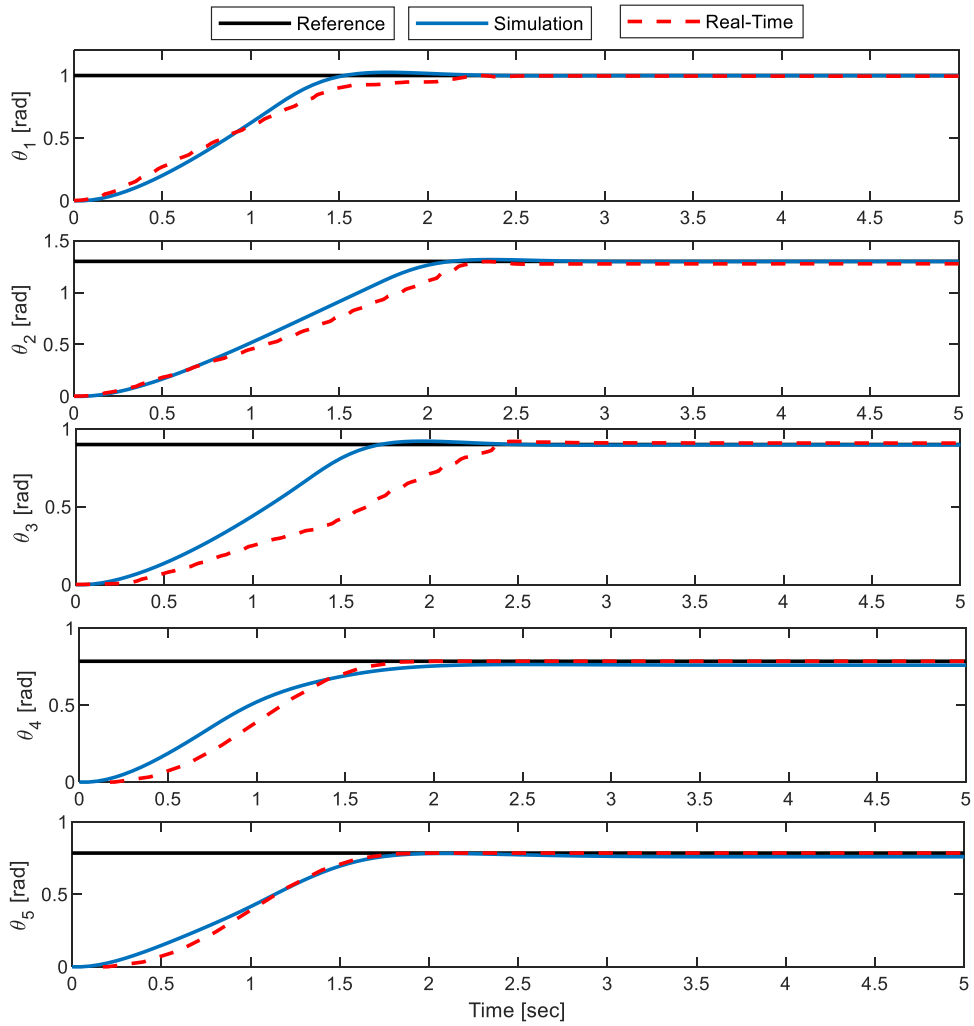

FIGURE 5 | Experimental and simulation results with SOA–NMPC controller. NMPC, nonlinear model predictive control; SOA, Spatial Operator Algebra.

TABLE 5 | Control performance evaluation with simulation and real-time.

	Settling time (s)	Rising time (s)	Overshoot (%)	Steady-state error (rad)	RMSE (rad)
Joint1					
Simulation	2.25	1.3	2	0.001	0.0181
Real-time	2.525	1.5	0	0.005	0.0140
Joint2					
Simulation	2.7	1.45	1.7	0.001	0.0272
Real-time	2.525	1.675	0	0.025	0.0225
Joint3					
Simulation	2.375	1.5	2	0.001	0.0176
Real-time	2.825	2.175	0	0.01	0.0169
Joint4					
Simulation	2.15	1.6	0	0.025	0.0140
Real-time	1.8	1.55	0	0	0.0303
Joint5					
Simulation	2.15	1.6	0	0.025	0.0140
Real-time	1.8	1.55	0	0	0.0303

Abbreviation: RMSE, root mean squared error.

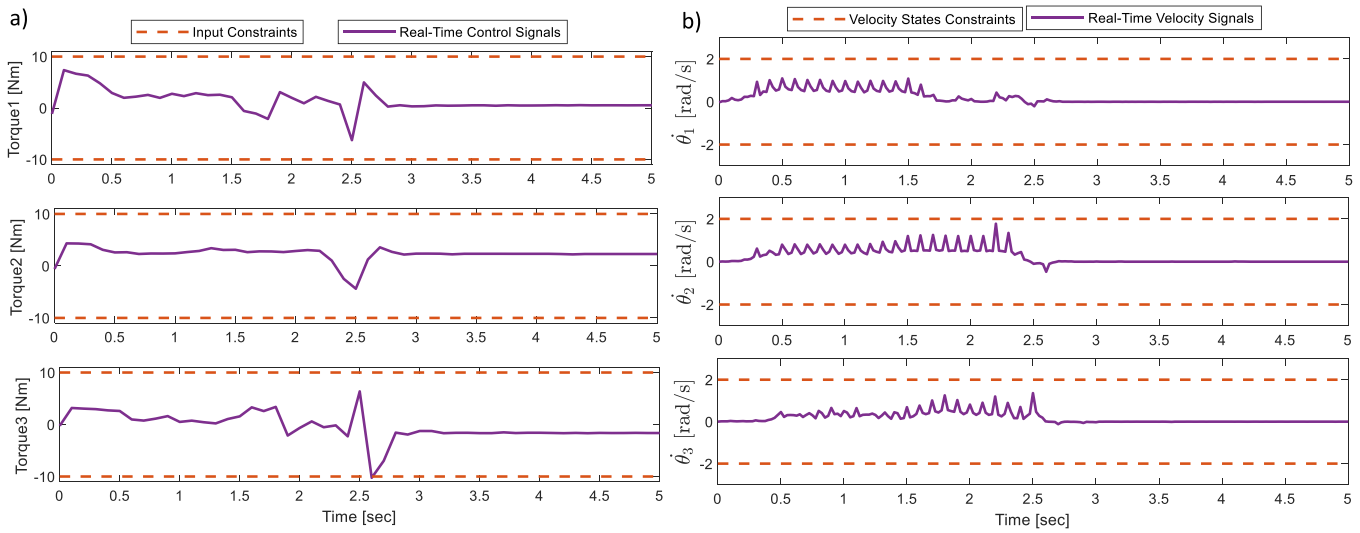

FIGURE 6 | (a) Control signals with input constraints for the first three joints and (b) velocity signals with state constraints for the first three joints.

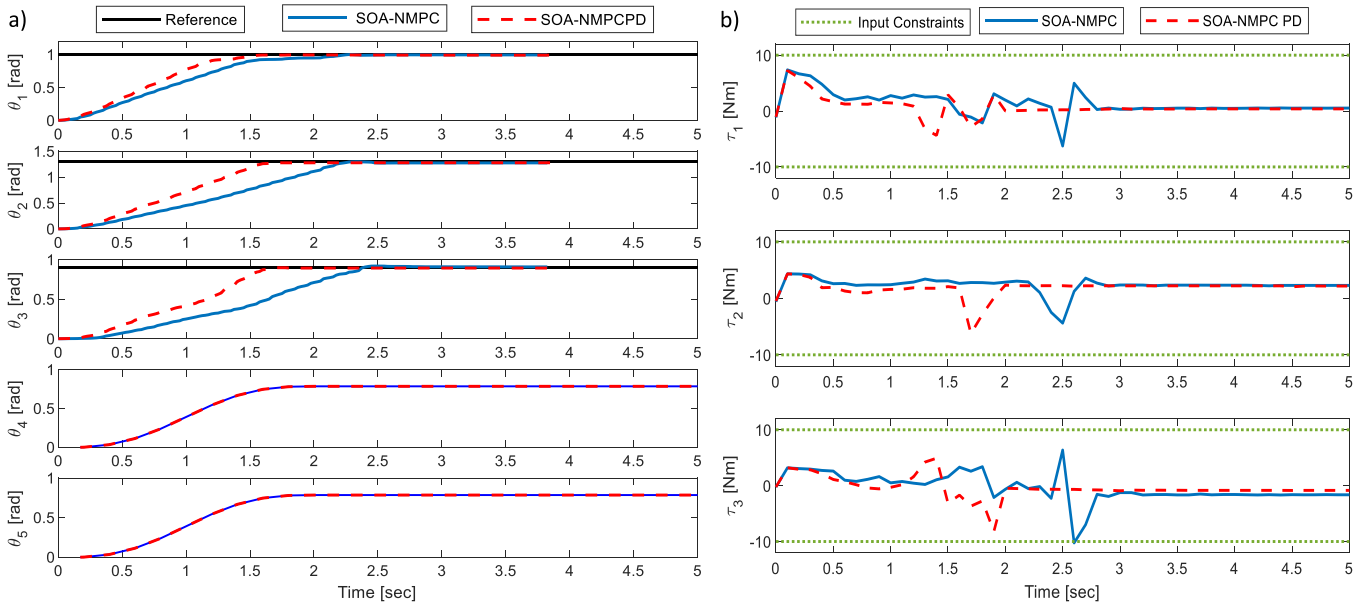

FIGURE 7 | (a) Experimental results with the SOA–NMPC and SOA–NMPC PD controllers and (b) evolution of the control input torques versus time (SOA–NMPC and SOA–NMPC PD). NMPC, nonlinear model predictive control; PD, proportional-derivative; SOA, Spatial Operator Algebra.

SOA–NMPC PD controller were better than those of the SOA–NMPC controller. Clearly, there was an improvement of between 20% and 36.63% in the settling and rising time performance criteria.

The control input torques of the direct drive RMD X8 Pro motors for both controllers are depicted in Figure 7b. The generated control signals for both controllers remained within permissible ranges and lacked high-frequency components. Furthermore, it is evident that both controllers ensure the specified constraints and can be applied in real-time.

4.4 | Benchmark

To demonstrate the relevance and effectiveness of the proposed control solutions, the performance of the proposed schemes (SOA–NMPC and SOA–NMPC PD) was compared with the NE-NMPC, conventional PID, and linear MPC (based on state–space model), and MPC-PD for the nominal case ($[1, 1.3, 0.9, 0.785, 0.785]$ rad).

It has been reported by Carron et al. (2019) that NMPC is computationally too demanding for six links in real-time based on their hardware. For this reason, the experiments were performed only on the first three links in Carron et al. (2019). All SOA–NMPC experiments, in contrast, were able to run on five links thanks to the novel approach proposed in our study. In addition, a 3-DOF dynamic model was used for the NE-NMPC algorithm since the lightweight fourth and fifth links were neglected. Therefore, NE-NMPC has the advantage of lower computational burden compared with SOA–NMPC.

TABLE 6 | Control performance evaluation with the SOA–NMPC and SOA–NMPC PD.

	Settling time (s)	Rising time (s)	Overshoot (%)	Steady-state error (rad)	RMSE (rad)
Joint1					
SOA–NMPC	2.525	1.5	—	0.005	0.0322
SOA–NMPC PD	1.6	1.2	—	0.008	0.0289
Improvements (%)	36.63	20	—	—	10.24
Joint2					
SOA–NMPC	2.525	1.675	—	0.025	0.0518
SOA–NMPC PD	1.7	1.185	—	0.02	0.0435
Improvements (%)	32.67	29.25	—	—	16.02
Joint3					
SOA–NMPC	2.825	2.175	—	0.01	0.0390
SOA–NMPC PD	1.8	1.49	—	0.005	0.0317
Improvements (%)	36.28	31.49	—	—	18.71

Abbreviations: NMPC, nonlinear model predictive control; PD, proportional-derivative; RMSE, root mean squared error; SOA, Spatial Operator Algebra.

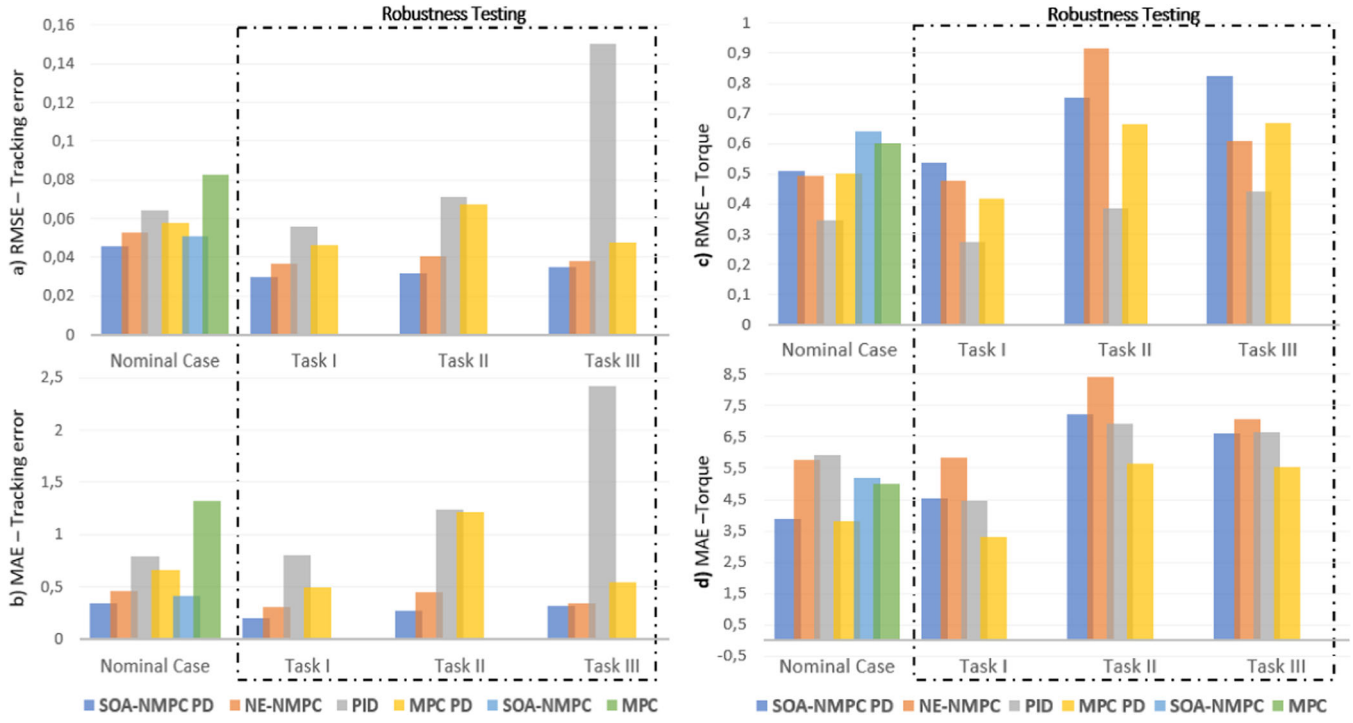

FIGURE 8 | RMSE and MAE over the entire trajectory for all controllers. MAE, mean absolute error; NMPC, nonlinear model predictive control; RMSE, root mean squared error; SOA, Spatial Operator Algebra.

After conducting the nominal test, four additional tasks were performed to undertake a detailed analysis of the performance of the SOA–NMPC PD. These tasks were robustness testing, which included the normal case (Task I), the addition of weight (Task II), internal disturbance (Task III), and output disturbance (Task IV). In all tasks, the target joint angular positions were set to $[1.1, 0.8, 0.9, 0.785, -1.18]$ radians. The objective of these tasks was to compare the performance of SOA–NMPC PD with that of PID, MPC-PD, and NE-NMPC controllers (one controller selected from each group was analyzed).

After all tasks were performed on the manipulator for each controller, charts were created in terms of RMSE and mean absolute error (MAE) performance criteria to compare the results more clearly. However, since the external disturbance in each controller in Task IV would not be identical, it was not included in the general comparison and the results are examined under a separate heading.

In Figure 8a,b, the total tracking errors of the joints are seen as RMSE and MAE criteria. Furthermore, Figure 8c,d shows the RMSE and MAE values of the total torque data. According to the total errors given in Figure 8a,b, SOA–NMPC PD gave the best result in all tests. The RMSE-based results of SOA–NMPC PD showed an improvement of 28.8%, 44.26%, 21.0%, and 13.45% compared with PID, MPC, MPC-PD, and NE-NMPC for the

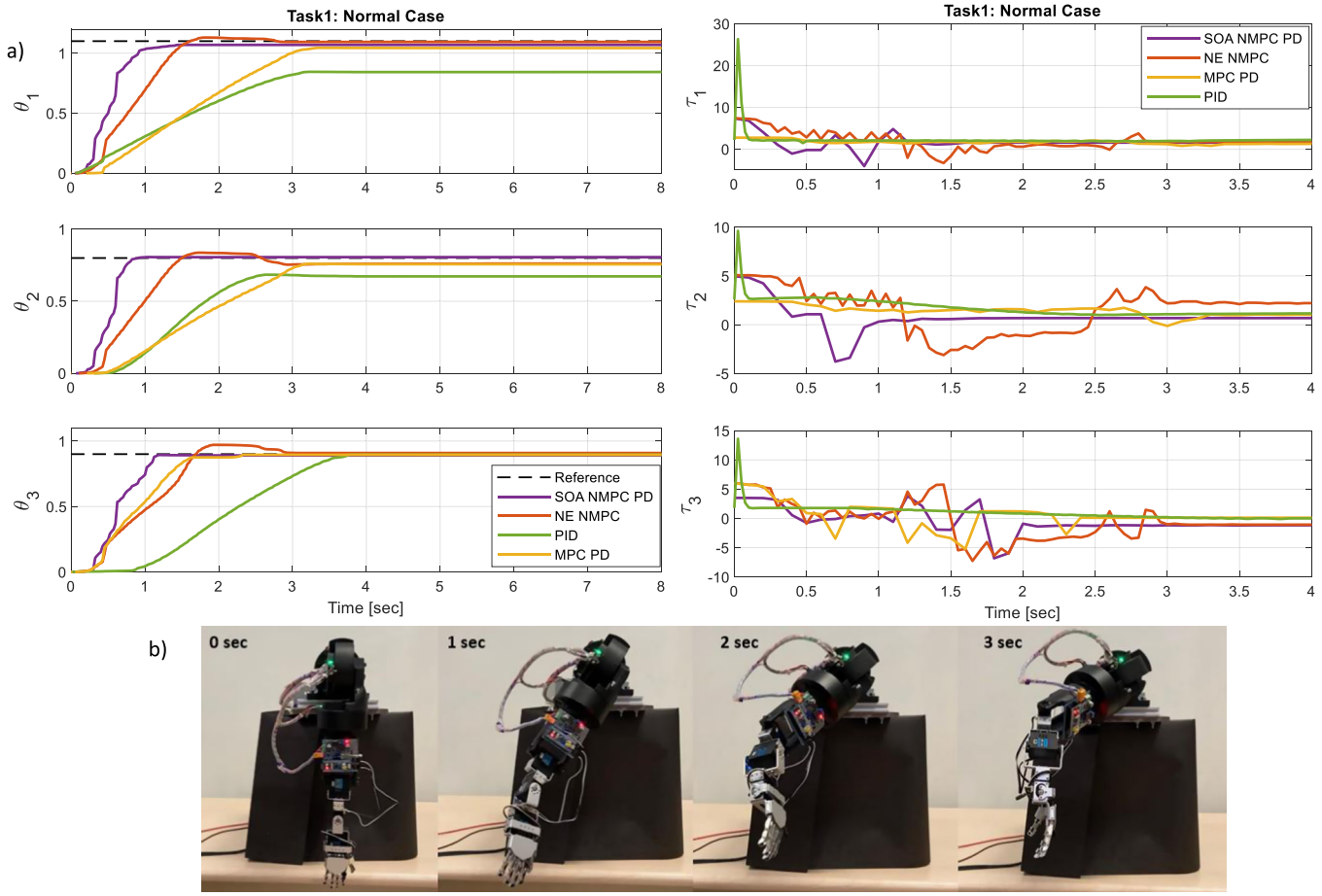

FIGURE 9 | (a) Evolution of joint positions and control input torques versus time (Task I) and (b) real-time action of manipulator (Task I/SOA–NMPC PD). NE, Newton–Euler; NMPC, nonlinear model predictive control; PD, proportional-derivative; PID, proportional, integral, and derivative; SOA, Spatial Operator Algebra.

nominal case, respectively. In Task I, an improvement of 17.8% over NE-NMPC, 46.6% over PID, and 35.06% over MPC-PD was observed. In Task II, SOA–NMPC PD showed enhancements of 22.49% compared with NE-NMPC, 55.41% compared with PID, and 53.03% compared with MPC-PD. For Task III, SOA–NMPC PD exhibited improvements of 8.8% over NE-NMPC, 76.84% over PID, and 26.37% over MPC-PD. Moreover, the MAE-based results of SOA–NMPC PD show an improvement of 56.12%, 73.86%, 47.88%, and 24.5% compared with PID, MPC, MPC-PD, and NE-NMPC for the nominal case, respectively. In Task I, an improvement of 32.1% over NE-NMPC, 74.22% over PID, and 58.29% over MPC-PD was observed. In Task II, SOA–NMPC PD showed enhancements of 38.41% compared with NE-NMPC, 77.96% compared with PID, and 77.55% compared with MPC-PD. For Task III, SOA–NMPC PD exhibited improvements of 6.78% over NE-NMPC, 86.95% over PID, and 41.38% over MPC-PD.

The detailed analysis of each task and the torque data of that task are discussed under the following headings.

4.4.1 | Task I: Normal Case

The control objective in this experiment was to steer the manipulator from the initial joint angular position $(0, 0, 0, 0, 0)$ to the goal angular position $(1.1, 0.8, 0.9, 0.785, -1.18)$ in the joint space while satisfying the constraints. The controller parameters were equivalent in all tasks, and no tuning was performed.

The joint positions and control inputs of all controllers are given in Figure 9a, and Table 7 lists the RMSE values of the joints tracking for Task 1. The real-time actions of the graphs in Figure 9a are depicted in Figure 9b. To show computational indications of the control performances, the criteria MAE defined in Equation (16) was also added into Table 7.

SOA–NMPC PD showed the best performance, while MPC-PD and PID exhibited slow behavior. In addition, PID did not provide the desired control input constraint (± 10 Nm) since there was no constraint provision feature in its structure. In the NE torque data, the presence of high-frequency components is clear. This can potentially lead to several disadvantages in control applications. First, the presence of high-frequency components can result in increased energy consumption, as the control system may unnecessarily expend energy to reduce these oscillations. Second, these high-frequency components can lead to mechanical wear and tear in the robot's joints and components, reducing their overall lifespan. Moreover, such oscillations can introduce instability into the control system, making it less robust, especially in scenarios with model uncertainties or disturbances. To mitigate these disadvantages, it is

TABLE 7 | Control performance evaluation (Task I).

	Joint 1 tracking error		Joint 2 tracking error		Joint 3 tracking error	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
SOA–NMPC PD	0.0113	0.0885	0.0084	0.0476	0.0103	0.0693
MPC-PD	0.0197	0.2310	0.0149	0.1763	0.0116	0.0856
PID	0.0220	0.3725	0.0154	0.2252	0.0188	0.1997
NE-NMPC	0.0139	0.1012	0.0105	0.1011	0.0121	0.1005

Note: The cells highlighted in the table represent the minimum values in the respective data comparisons.
Abbreviations: MAE, mean absolute error; NE, Newton–Euler; NMPC, nonlinear model predictive control; PD, proportional-derivative; PID, proportional, integral, and derivative; RMSE, root mean squared error; SOA, Spatial Operator Algebra.

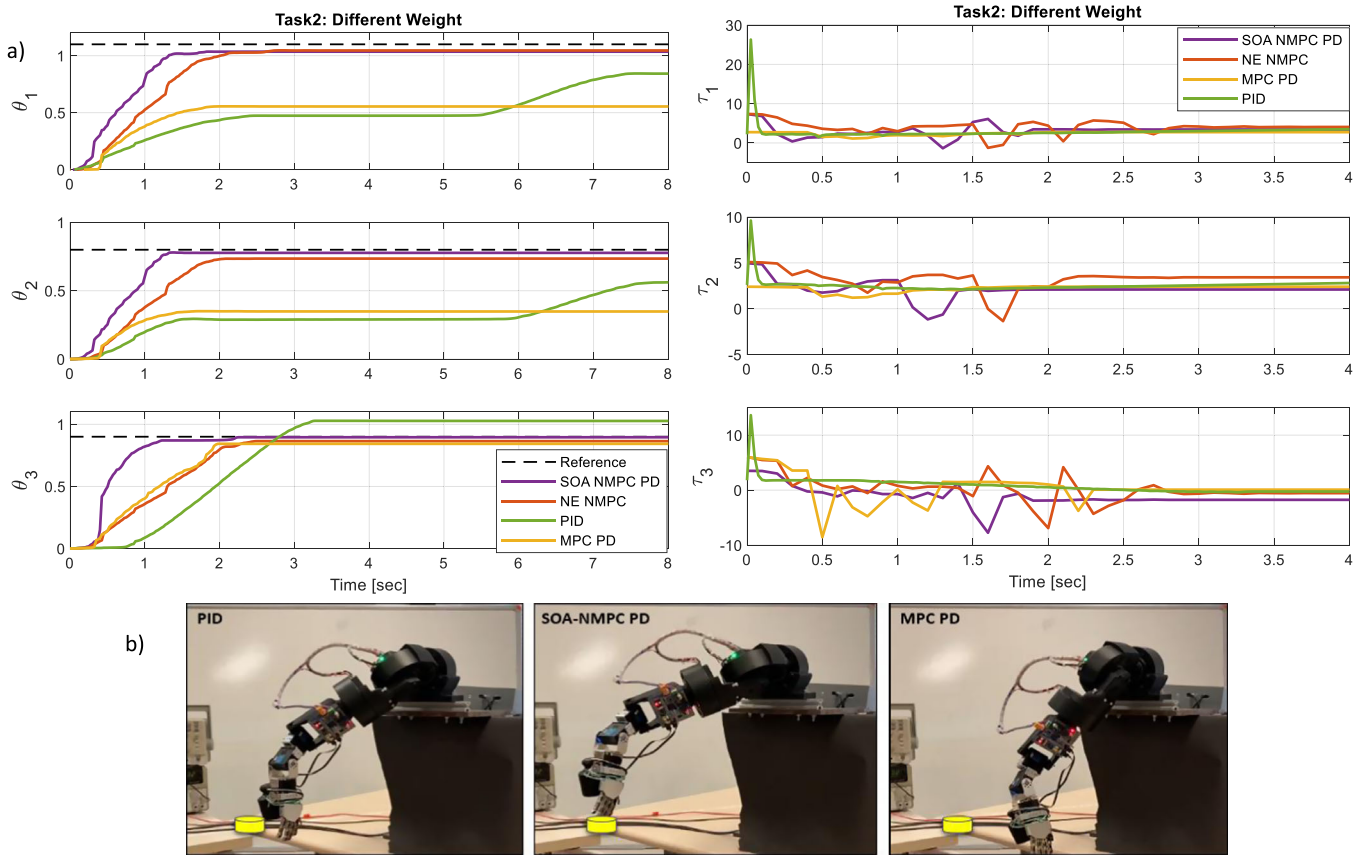

FIGURE 10 | (a) Evolution of the joint positions and the control input torques versus time (Task II) and (b) success of the end effector in reaching the target point in Task II. NE, Newton–Euler; NMPC, nonlinear model predictive control; PID, proportional, integral, and derivative; SOA, Spatial Operator Algebra.

essential to employ control strategies that address and suppress these high-frequency components, ensuring smoother and more efficient control operations.

4.4.2 | Task II: Different Weights (the Addition of Weight)

In this task, a weight of 500 g was attached to the end effector of the manipulator. This task is significant because it simulates real-world scenarios in which the robot interacts with external objects or payloads, thereby assessing the controller's adaptability and responsiveness to varying task conditions. With weight-based tasks, the controller's ability to accommodate changes in load, maintain stability, and ensure accurate positioning can be comprehensively assessed. Incorporating such tests into the evaluation framework enhances the reliability and robustness of the controller, substantiating its suitability for practical applications demanding versatility and precise performance even in the presence of external disturbances or modifications.

As can be seen from the graphs in Figure 10a, PID and MPC-PD did not perform well when weight was added. The end effector was quite far from the target (see Figure 10b). The NE-NMPC and SOA–NMPC PD showed approximate performance at the arrival of the end effector at the target point. However, the SOA–NMPC PD gave better results in the transient response controller criteria. In addition, SOA stands out in the

TABLE 8 | Control performance evaluation (Task II).

	Joint 1 tracking error		Joint 2 tracking error		Joint 3 tracking error	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
SOA–NMPC PD	0.0129	0.1373	0.0093	0.0776	0.0096	0.1067
MPC-PD	0.0303	0.5924	0.0241	0.4764	0.0112	0.0583
PID	0.0294	0.5420	0.0235	0.4449	0.0181	0.2531
NE-NMPC	0.0157	0.1643	0.0118	0.1403	0.0135	0.1390

Note: The cells highlighted in the table represent the minimum values in the respective data comparisons.
Abbreviations: MAE, mean absolute error; NE, Newton–Euler; NMPC, nonlinear model predictive control; PD, proportional-derivative; PID, proportional, integral, and derivative; RMSE, root mean squared error; SOA, Spatial Operator Algebra.

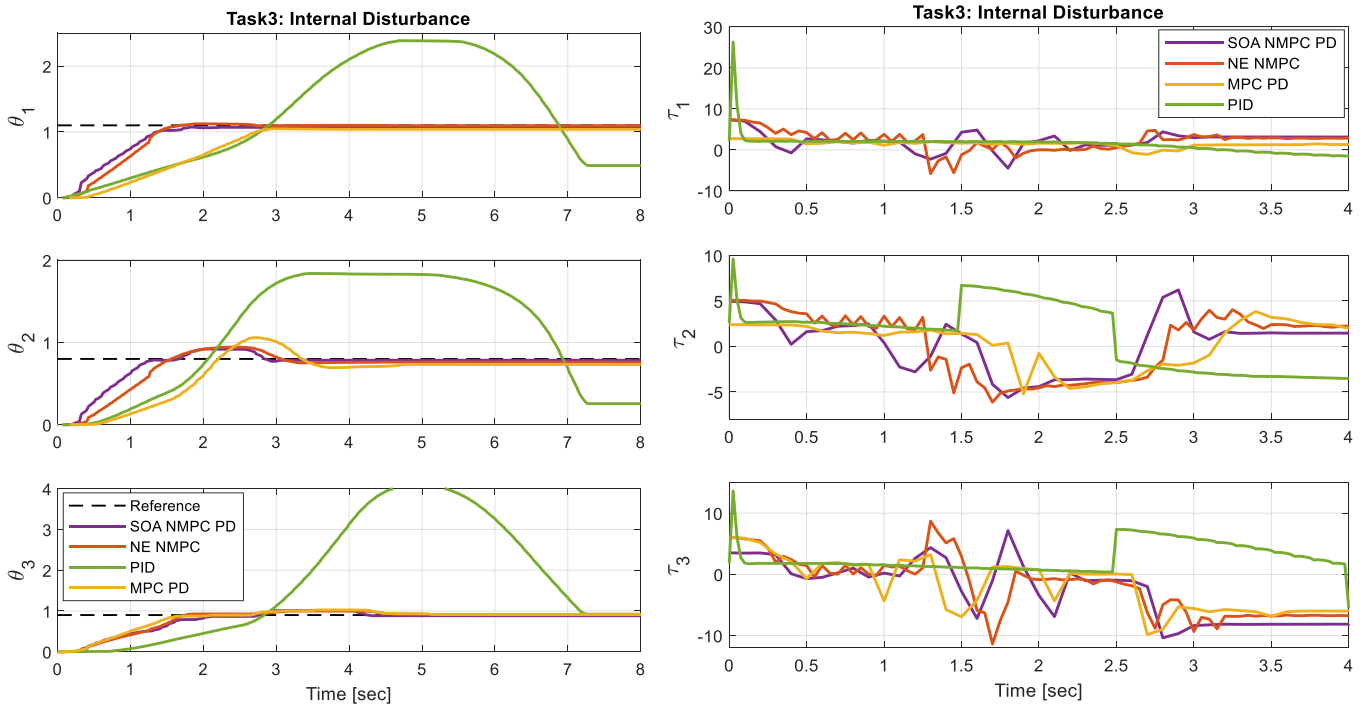

FIGURE 11 | Evolution of the joint positions and control input torques versus time (Task III). NE, Newton–Euler; NMPC, nonlinear model predictive control; PID, proportional, integral, and derivative; SOA, Spatial Operator Algebra.

performance index comparisons given in Table 8. Moreover, similar to Task 1, PID could not meet the control input constraint.

4.4.3 | Task III: Internal Disturbance

For the internal disturbance task, a pulse signal with 10-s periods, 10% pulse width, 1.5-s phase delay, and 5 Nm amplitude was applied to the second joint's control signal, while another pulse signal with 10-s periods, 15% pulse width, 2.5-s phase delay, and 7 Nm amplitude was applied to the third joint's control signal. Figure 11 shows the response of the controllers for the internal disturbance task. As a result of this task, the controller was able to eliminate the disturbing effects and maintain system stabilization. Controllers other than PID showed valid performances in this task. It can be seen from the figure that there were high-frequency components of the NE torque data, as well as the PID and NE-NMPC, that did not satisfy the control input constraint. Performance index comparisons are shown in Table 9.

4.4.4 | Task IV: External Disturbance

The significance of Task IV lies in its assessment of the robot's performance under unexpected perturbations during the initial motion from the home position. By introducing a 1.5-s obstruction where the robot was impeded by the presence of a hand, the evaluation provided insights into the system's response to external disturbances and its ability to recover from such interruptions.

This test scenario reflects practical situations in which robots operate in dynamic environments with the potential for unforeseen obstacles. Analyzing the robot's reaction to this controlled interference aided in understanding its real-world adaptability, collision avoidance mechanisms, and capability to resume the intended trajectories following disruptions.

TABLE 9 | Control performance evaluation (Task III).

	Joint 1 tracking error		Joint 2 tracking error		Joint 3 tracking error	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
SOA–NMPC PD	0.0129	0.1130	0.0093	0.0792	0.0121	0.1237
MPC-PD	0.0200	0.2326	0.0152	0.1943	0.0126	0.1120
PID	0.0399	0.7245	0.0362	0.6725	0.0745	1.0242
NE-NMPC	0.0143	0.1034	0.0112	0.1174	0.0127	0.1180

Note: The cells highlighted in the table represent the minimum values in the respective data comparisons.
Abbreviations: MAE, mean absolute error; NE, Newton–Euler; NMPC, nonlinear model predictive control; PD, proportional-derivative; PID, proportional, integral, and derivative; RMSE, root mean squared error; SOA, Spatial Operator Algebra.

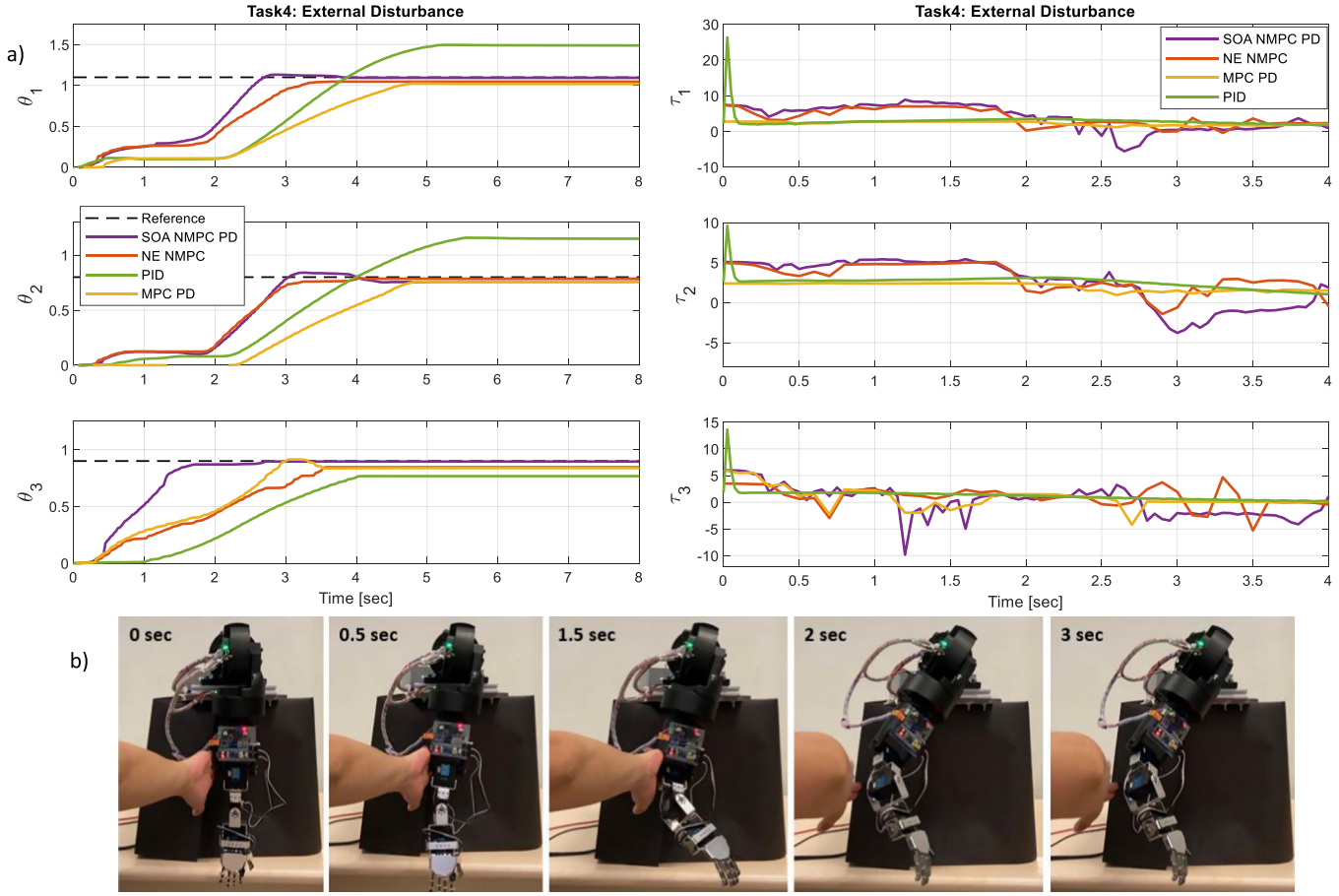

FIGURE 12 | (a) Evolution of the joint positions and the control input torques versus time (Task IV) and (b) real-time action of the manipulator (Task IV/SOA–NMPC PD). NE, Newton–Euler; NMPC, nonlinear model predictive control; PD, proportional-derivative; PID, proportional, integral, and derivative; SOA, Spatial Operator Algebra.

Incorporating such obstacle-induced tests not only enhanced the robot's reliability but also validated its suitability for safe and robust operation, a critical consideration in real-world applications requiring consistent and obstacle-tolerant performance. The data given in Figure 12a and Table 10 show that the SOA–NMPC PD scheme was also successful in this task.

5 | Conclusion and Future Works

This study presents a novel approach to address the challenges associated with NMPC in real-time robotic implementation. By using the principles of SOA theory, the proposed SOA–NMPC and SOA–NMPC PD control schemes offer significant advantages, including enhanced tracking performance and the satisfaction of state and control input constraints. Experimental validation using a 5-DOF robot manipulator, along with robustness analysis under various conditions, demonstrated the effectiveness and versatility of these schemes. Furthermore, benchmarking against alternative control methods served to highlight the superiority of the SOA–NMPC PD controller in providing precise trajectory tracking and ensuring constraints were met throughout the manipulator's trajectory, making it a valuable contribution to the field of advanced robotics and control systems. On the basis of MAE criteria, the proposed SOA–NMPC PD showed average enhancements of 25.44%

TABLE 10 | Control performance evaluation (Task IV).

	Joint 1 tracking error		Joint 2 tracking error		Joint 3 tracking error	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
SOA–NMPC PD	0.0201	0.2000	0.0171	0.2036	0.0117	0.0878
MPC-PD	0.0278	0.3890	0.0224	0.3064	0.0159	0.1971
PID	0.0297	0.5083	0.0231	0.4070	0.0218	0.3265
NE-NMPC	0.0213	0.2585	0.0168	0.1868	0.0167	0.2121

Note: The cells highlighted in the table represent the minimum values in the respective data comparisons.
Abbreviations: MAE, mean absolute error; NE, Newton–Euler; NMPC, nonlinear model predictive control; PD, proportional-derivative; PID, proportional, integral, and derivative; RMSE, root mean squared error; SOA, Spatial Operator Algebra.

compared with NE-NMPC, and 62.77% compared with MPC in tracking errors. Furthermore, in terms of control input (torque) cost, SOA–NMPC had a lower cost than NE-NMPC while it was very close to the MPC cost.

In particular, the superiority demonstrated over the commonly used NE-NMPC is quite important. SOA–NMPC PD emerged as a preferable alternative for real-time robotic applications because of the absence of high-frequency components found in NE torque data and superior tracking performance. Moreover, it is predicted that SOA will provide greater advantages as the DOF and complexity of the systems increase. In studies involving dual-arm robots and collaborative robot implementations, the application of SOA dynamic modeling to NMPC will be expected to enhance feasibility. SOA-based NMPC can easily be extended to handle more complex and larger robotic systems. In conclusion, this study highlighted the significant benefits of integrating SOA with NMPC as a novel step toward effective control of complex systems, offering a substantial contribution to future research and robotic applications. Additional future work will aim to implement the verified advantages of SOA–NMPC in obstacle avoidance applications and dual-arm robot systems.

Acknowledgments

This work was supported by the Scientific Research Projects Coordination Unit of Kocaeli University under grant numbers #2483 and #4253.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that supports the findings of this study are available in the supplementary material of this article.

References

Spatial Operator Algebra. 2020. *The Spatial Operator Algebra*. NASA Jet Propulsion Laboratory (JPL). https://dartslab.jpl.nasa.gov/SOA/index.php.

Bettega, J., and D. Richiedei. 2023. "Trajectory Tracking in an Underactuated, Non-Minimum Phase Two-Link Multibody System Through Model Predictive Control With Embedded Reference Dynamics." *Mechanism and Machine Theory* 180: 105165. https://doi.org/10.1016/j.mechmachtheory.2022.105165.

Bruder, D., X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan. 2021. "Data-Driven Control of Soft Robots Using Koopman Operator Theory." *IEEE Transactions on Robotics* 37, no. 3: 948–961. https://doi.org/10.1109/TRO.2020.3038693.

Carron, A., E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger. 2019. "Data-Driven Model Predictive Control for Trajectory Tracking With a Robotic Arm." *IEEE Robotics and Automation Letters* 4, no. 4: 3758–3765. https://doi.org/10.1109/LRA.2019.2929987.

Chaber, P. 2020. "Fast Nonlinear Model Predictive Control Algorithm With Neural Approximation for Embedded Systems: Preliminary Results." In *Advances in Intelligent Systems and Computing*, vol. 1196, edited by A. Bartoszewicz, J. Kabziński, and J. Kacprzyk, 1067–1078. Cham: Springer.

Chemori, A., R. Kouki, and F. Bouani. 2023. "A New Fast Nonlinear Model Predictive Control of Parallel Manipulators: Design and Experiments." *Control Engineering Practice* 130: 105367. https://doi.org/10.1016/j.conengprac.2022.105367.

Dong, C., F. Tian, X. Dong, X. Zhao, and F. Li. 2018. "The Structure and Control Analysis of AMR Automatic Harvesting Robot." In *Advances in Intelligent Systems and Computing*, vol. 690, edited by F. Qiao, S. Patnaik, and J. Wang, 457–463. Cham: Springer.

Featherstone, R. 1987. *Robot Dynamics Algorithms*. Boston, MA: Springer US.

Grüne, L., and J. Pannek. 2017. *Nonlinear Model Predictive Control*, 45–69. Springer International Publishing.

Hopler, R., and M. Thummel. 2004. "Symbolic Computation of the Inverse Dynamics of Elastic Joint Robots." In *Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, 2004, ICRA'04, Vol. 5*, 4314–4319. IEEE. https://doi.org/10.1109/ROBOT.2004.1302396.

Kleff, S., A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti. 2021. "High-Frequency Nonlinear Model Predictive Control of a Manipulator." In *2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China*, 7330–7336. IEEE. https://doi.org/10.1109/ICRA48506.2021.9560990.

Krämer, M., C. Rösmann, F. Hoffmann, and T. Bertram. 2020. "Model Predictive Control of a Collaborative Manipulator Considering Dynamic Obstacles." *Optimal Control Applications and Methods* 41, no. 4: 1211–1232. https://doi.org/10.1002/oca.2599.

Lee, J. H. 2011. "Model Predictive Control: Review of the Three Decades of Development." *International Journal of Control, Automation and Systems* 9, no. 3: 415–424. https://doi.org/10.1007/s12555-011-0300-6.

Mayne, D. Q. 2014. "Model Predictive Control: Recent Developments and Future Promise." *Automatica* 50, no. 12: 2967–2986. https://doi.org/10.1016/j.automatica.2014.10.128.

Meldrum, D. R., G. F. Franklin, and P. J. Wiktor. 1993. "Control of Manipulators With Some Unactuated Joints." *IFAC Proceedings Volumes* 26, no. 2: 345–348. https://doi.org/10.1016/S1474-6670(17)48746-5.

Nakanishi, J., M. Mistry, and S. Schaal. 2007. "Inverse Dynamics Control With Floating Base and Constraints." In *Proceedings of the 2007*

IEEE International Conference on Robotics and Automation, Rome, Italy, 1942–1947. IEEE. https://doi.org/10.1109/ROBOT.2007.363606.

Narasingam, A., S. H. Son, and J. S.-I. Kwon. 2023. "Data-Driven Feedback Stabilisation of Nonlinear Systems: Koopman-Based Model Predictive Control." *International Journal of Control* 96, no. 3: 770–781. https://doi.org/10.1080/00207179.2021.2013541.

Orin, D. E., A. Goswami, and S.-H. Lee. 2013. "Centroidal Dynamics of a Humanoid Robot." *Autonomous Robots* 35, no. 2–3: 161–176. https://doi.org/10.1007/s10514-013-9341-4.

Ozakyol, H., C. Karaman, and Z. Bingul. 2017. "Robotics Toolbox for Kinematic Analysis And Design of Hybrid Multibody Systems." In *2017 18th International Conference on Advanced Robotics, ICAR 2017*, 401–406. IEEE. https://doi.org/10.1109/ICAR.2017.8023639.

Ozakyol, H., C. Karaman, and Z. Bingul. 2019. "Advanced Robotics Analysis Toolbox for Kinematic and Dynamic Design and Analysis of High-DOF Redundant Serial Manipulators." *Computer Applications in Engineering Education* 27, no. 6: 1429–1452. https://doi.org/10.1002/cae.22160.

Polak, E. 1997. *Optimization* (Vol. 124). New York, NY: Springer.

Reinhold, J., H. Baumann, and T. Meurer. 2023. "Constrained-Differential-Kinematics-Decomposition-Based NMPC for Online Manipulator Control With Low Computational Costs." *Robotics* 12, no. 1: 7. https://doi.org/10.3390/robotics12010007.

Rodriguez, G. 1987. "Kalman Filtering, Smoothing, and Recursive Robot Arm Forward and Inverse Dynamics." *IEEE Journal on Robotics and Automation* 3, no. 6: 624–639. https://doi.org/10.1109/JRA.1987.1087147.

Rodriguez, G., A. Jain, and K. Kreutz-Delgado. 1991. "A Spatial Operator Algebra for Manipulator Modeling and Control." *International Journal of Robotics Research* 10, no. 4: 371–381. https://doi.org/10.1177/027836499101000406.

Schwenzer, M., M. Ay, T. Bergs, and D. Abel. 2021. "Review on Model Predictive Control: An Engineering Perspective." *International Journal of Advanced Manufacturing Technology* 117, no. 5–6: 1327–1349. https://doi.org/10.1007/s00170-021-07682-3.

Wensing, P. M., L. R. Palmer, and D. E. Orin. 2015. "Efficient Recursive Dynamics Algorithms for Operational-Space Control With Application to Legged Locomotion." *Autonomous Robots* 38, no. 4: 363–381. https://doi.org/10.1007/s10514-015-9420-9.

Wieber, P. B. 2006. "Holonomy and Nonholonomy in the Dynamics of Articulated Motion." In *Fast Motions in Biomechanics and Robotics. Lecture Notes in Control and Information Sciences*, vol. 340, 411–425. Berlin, Heidelberg: Springer.

Wu, X., J. Chen, and L. Xie. 2019. "Fast Economic Nonlinear Model Predictive Control Strategy of Organic Rankine Cycle for Waste Heat Recovery: Simulation-Based Studies." *Energy* 180: 520–534. https://doi.org/10.1016/j.energy.2019.05.023.

Yang, Y., H. Xu, and X. Yao. 2023. "Inverse-Dynamics- and Disturbance-Observer-Based Tube Model Predictive Tracking Control of Uncertain Robotic Manipulator." *Journal of the Franklin Institute* 360, no. 10: 6906–6927. https://doi.org/10.1016/j.jfranklin.2023.04.005.

Yaren, T., and S. Kizir. 2024. "Efficiency Assessment of SOA-Based Computed Torque Control: A Comparative Analysis With NE-Based Approach." *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 238, no. 8: 1410–1424. https://doi.org/10.1177/09596518241238414.

Yıldız, G. 2014. *Nonlinear Control Methods of Industrial Serial Robots*. Istanbul, Turkey: Istanbul Technical University.

Zhao, P., S. Mohan, and R. Vasudevan. 2017. "Control Synthesis for Nonlinear Optimal Control via Convex Relaxations." In *2017 American Control Conference (ACC), Seattle, WA, USA*, 2654–2661. IEEE. https://doi.org/10.23919/ACC.2017.7963353.

Supporting Information

Additional supporting information can be found online in the Supporting Information section.