# Nonlinear MPC for Collision-Free and Deadlock-Free Navigation of Multiple Nonholonomic Mobile Robots

Amir Salimi Lafmejani[a], Spring Berman[b,*]

[a]School of Electrical, Computer and Energy Engineering, Arizona State University, USA
[b]School for Engineering of Matter, Transport and Energy, Arizona State University, USA

ARTICLE INFO

*Keywords*:
Nonlinear model predictive control
Multi-robot systems
Wheeled mobile robots
Nonholonomic constraints
Collision avoidance
Deadlock avoidance

ABSTRACT

In this paper, we present an online nonlinear Model Predictive Control (MPC) method for collision-free, deadlock-free navigation by multiple autonomous nonholonomic Wheeled Mobile Robots (WMRs). Our proposed method solves a nonlinear constrained optimization problem at each time step over a specified horizon to compute a sequence of optimal control inputs that drive the robots to target poses along collision-free trajectories, where the robots' future states are predicted according to a unicycle kinematic model. To reduce the computational complexity of the optimization problem, we formulate it without stabilizing terminal constraints or terminal costs. We describe a computationally efficient approach to programming and solving the optimization problem, using open-source software tools for fast nonlinear optimization and applying the multiple-shooting method. We also provide rigorous proofs of the feasibility of the optimization problem and the stability of the proposed method. To validate the performance of our MPC method, we implement it in both 3D robot simulations and experiments with real nonholonomic WMRs for different multi-robot navigation scenarios with up to six robots. In all scenarios, the robots successfully navigate to their goal poses without colliding with one another or becoming trapped in a deadlock.

## 1. Introduction

Many applications of multi-robot systems require the robots to navigate through an environment while avoiding collisions with each other and with obstacles. Despite extensive research on this topic, there are still challenges to designing control strategies for multi-robot navigation that are computationally efficient and have theoretical guarantees on collision avoidance and absence of deadlocks. To address these challenges, we present a collision-free and deadlock-free multi-robot navigation strategy that is based on an online, computationally efficient Nonlinear Model Predictive Control (NMPC) method for pose stabilization of multiple nonholonomic Wheeled Mobile Robots (WMRs).

### 1.1. Related Works

According to the well-known Brockett's condition [1], the pose of a nonholonomic WMR cannot be stabilized using a linear controller or a smooth time-invariant controller. Thus, other types of control methods have been proposed in the literature for this objective, including smooth time-varying controllers [2], differential kinematic-based approaches [3], and guiding vector field (GVF) controllers [4]. However, these control methods do not incorporate constraints on the robot's control inputs, which are enforced by the limitations of the robot's actuators, and constraints on the robot's states, which are determined by the boundaries of the free space in which the robot can move.

Other navigation strategies for nonholonomic WMRs that do incorporate these constraints have been developed using Model Predictive Control (MPC) [5, 6], also called Receding Horizon Control (RHC). MPC is a powerful control method that can handle Multiple-Input Multiple-Output (MIMO) constrained control problems [7, 8]. Due to this capability, MPC-based methods have also been employed for collision-free multi-robot navigation, in which collision avoidance between pairs of robots is encoded as constraints on the robots' states.

The main limitation of *online* implementations of MPC methods, especially for multi-robot systems with large numbers of robots, is their high computational cost, arising from the fact that the controller runs a constrained optimization problem over a specified horizon at each time step [9]. If sufficient computational resources are in fact available, online MPC methods can be used to compute controllers for large-scale systems and systems with fast dynamics [10, 11].The stability of an MPC method can be ensured by introducing terminal constraints or terminal costs, e.g. [12, 13]. However, this increases the computational complexity of the associated optimization problem.

Additional MPC-based approaches for multi-robot navigation have been developed which can be implemented with lower computational resources than online methods. Some of these approaches calculate the reachable set of the robots' states or the optimal control solutions *offline*, prior to the robots' deployment. For example, the MPC method in [14] incorporates collision avoidance constraints in the optimization problem, and the feasibility and stability of the method are established by computing the reachable set of the robots' states. However, such offline computations scale poorly with the number of robots, and they cannot be used for real-time implementations in dynamic or uncertain environments. An-

This work was supported by the Arizona State University Global Security Initiative.

*Corresponding author
asalimil@asu.edu ( Amir Salimi Lafmejani); spring.berman@asu.edu ( Spring Berman)
ORCID(s): 0000-0001-8691-2139 ( Amir Salimi Lafmejani)

other approach to reducing the computational effort is to simplify the control problem by formulating it as a linear MPC method [15, 16]. This can be done by using linear models to describe the robots' motion, such as linear point-mass models or linearizations of the unicycle model [17, 18, 19]. For instance, in [17], a linear MPC is combined with the Optimal Reciprocal Collision Avoidance (ORCA) approach to synthesize navigation controllers for nonholonomic robots by computing holonomic reference trajectories for the robots, using the approximation that they are holonomic. However, many of these methods have been implemented only in simulation, e.g. [14, 16], not in real-world experiments. Moreover, applying these methods to nonholonomic robots, which are described by nonlinear kinematic models, will inevitably result in a tracking error between the reference trajectories and the robots' actual trajectories.

Various software tools and techniques have been developed for speeding up the solution of the optimization problem in an MPC method. In [20], the authors propose approaches to implementing fast MPC using online optimization with interior-point methods. The work [21] presents a solver for convex quadratic programs, implemented in the Operator Splitting Solver for Quadratic Programming (OSQP), which is typically ten times faster than the interior-point methods. However, these approaches are restricted to linear MPC problems [17, 22] or a linearized form of the original nonlinear problem [23, 24]. MPC methods for collision-free navigation by multiple nonholonomic WMRs involve the solution of a nonlinear optimal control problem. The software packages NLopt [25] and APOPT [26] can be used to solve this type of problem. Moreover, the open-source software tool CasADi [27] provides a symbolic programming framework for formulating the optimization problem, which further reduces the computational effort required to solve the problem and facilitates efficient implementation of NMPCs. Another way to accelerate the solution of the optimization problem is to define its decision variables as both the control inputs and the robot states, an approach called the *multiple-shooting method* [28], instead of just the control inputs, as is done in the *single-shooting method* [29]. The single-shooting method can exhibit slow convergence to the optimal solution and numerical instabilities [30]. The multiple-shooting method produces faster convergence to optimal solutions than the single-shooting method [31], since it lifts the optimization problem to a higher-dimensional space. The multiple-shooting method is employed for MPC in [32] to accelerate the convergence of the associated optimization problem.

MPC methods have been developed for different types of multi-robot control architectures. The appropriate control architecture in a given application is determined by the number of robots and their capabilities, the available computational resources, and the relative importance of preventing deadlocks and collisions. In *centralized* control approaches for multi-robot systems, a central computational unit designs and transmits control commands for all the robots. Centralized multi-robot navigation approaches, e.g. [33], can guarantee collision-free and deadlock-free navigation, but the required computational resources increase with the number of robots. One way to reduce the computational complexity of the control problem in an MPC method for multi-robot systems is to employ a *distributed* control approach [34, 35]. In these approaches, the optimal control problem is decomposed into subproblems, which are assigned to each robot to solve using its on-board computational resources, and neighboring robots communicate their solutions to each other to plan collision-free paths [36]. To eliminate the requirement for a central computational unit or inter-robot communication, *decentralized* MPC-based approaches have been developed for multi-robot navigation [18, 37], in which each robot computes its own control inputs using only local measurements, without communicating with other robots. These types of approaches can be scaled to large numbers of robots. However, existing approaches are not fully decentralized, since they rely on communication between neighboring robots to share sensor information and computation tasks in order to avoid collisions. Importantly, when using distributed and decentralized control architectures, it remains a challenge to design multi-robot navigation strategies that avoid all types of deadlocks [19, 38].

### 1.2. Paper Contributions and Organization

This paper presents an online NMPC method for collision-free, deadlock-free navigation by multiple nonholonomic WMRs. In Section 2, we define the kinematic model of a nonholonomic WMR and formulate an MPC design for navigation by a single nonholonomic WMR and the corresponding optimization problem. In Section 3, we present our MPC design for navigation by multiple noholonomic WMRs and the associated optimization problem. As we discuss in Section 3.2, our method extends a type of NMPC method without stabilizing terminal constraints or terminal costs, designed for pose stabilization of a single robot, to multi-robot systems with inter-robot collision avoidance. The following properties of our method differentiate it from existing MPC-based methods for multi-robot navigation:

- Our NMPC method uses the nonlinear unicycle model for nonholonomic WMRs, as opposed to a linearization of this model, which can be used to design linear MPCs but can produce inaccurate predictions of future robot states.
- Our method is executable online, which enables real-time implementation in dynamic or uncertain environments.
- We provide rigorous proofs of the feasibility and stability of our method, which guarantee collision-free and deadlock-free navigation by the robots. These proofs are given in Section 3.2.
- We significantly reduce the computational complexity of nonlinear MPC by designing the associated optimization problem without stabilizing terminal constraints or terminal costs, as described in Section 3.2.

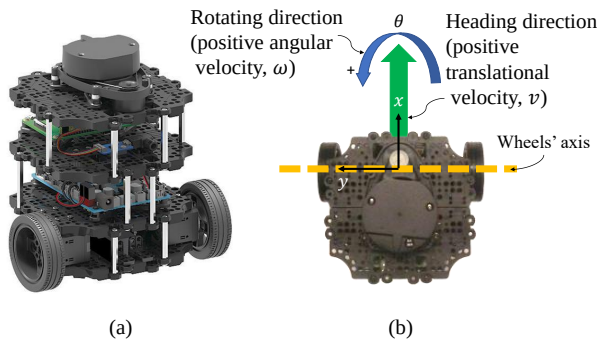


**Figure 1:** (a) 3-D view [42] of the Turtlebot3 Burger robot, and (b) overhead view with body-fixed coordinate frame.

- We use the multiple-shooting method to accelerate the convergence of the optimization problem in our NMPC.
- Unlike the implementations of other methods on real-world robots [19, 39, 40, 41], the experimental implementation of our method does not require global localization of the robots using an overhead camera or motion capture system.

We implement our method using CasADi in a standard software framework, the Robot Operating System (ROS), and validate its effectiveness at producing collision-free multi-robot navigation in six different scenarios, both in simulations and physical experiments, as described in Section 4. In Section 5, we provide recommendations for tuning the parameters of our method, describe how the method can prevent three types of deadlocks, and outline limitations of the method and how they can be overcome.

## 2. Preliminaries and Background

### 2.1. Kinematic Model of a Nonholonomic Robot

Figure 1 shows an example of a nonholonomic WMR, the Turtlebot3 Burger robot from Robotis®, which we use to validate our control strategy in simulations and physical experiments. The origin of the robot's local coordinate frame, defined in Fig. 1b, is located at position $(x(t), y(t))$ in the global coordinate frame at time $t$. The heading direction of the robot is aligned with the $x$-axis of the local coordinate frame. The robot's heading angle $\theta(t)$ at time $t$ increases when the robot rotates counter-clockwise about the $z$-axis of the local frame, and decreases when it rotates clockwise. The control inputs at time $t$ are the robot's linear velocity $v(t)$ and angular velocity $w(t)$. We define the state vector $\mathbf{x} \in \mathbb{R}^3$ and the control input vector $\mathbf{u} \in \mathbb{R}^2$ as:

$$\mathbf{x} := [x \quad y \quad \theta]^{\mathrm{T}}, \quad \mathbf{u} := [v \quad w]^{\mathrm{T}}. \tag{1}$$

The kinematic model of a nonholonomic WMR can be written as the following unicycle model in state-space form [43]:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \rightarrow \begin{cases} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= w \end{cases} \tag{2}$$

where $f(\cdot)$ denotes a nonlinear function of states and control inputs, representing the twist of the robot. From Eq. (2), we obtain the following discrete-time kinematic model of the robot [44]:

$$\begin{aligned} x(k+1) &= x(k) + v(k)\cos(\theta(k))T_s \\ y(k+1) &= y(k) + v(k)\sin(\theta(k))T_s \\ \theta(k+1) &= \theta(k) + w(k)T_s \end{aligned} \tag{3}$$

where $T_s$ is the sampling time and $k \in \mathbb{N} \cup \{0\}$ is an index for the time step, such that $t = kT_s$. We can write this model in state-space form as:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + f(\mathbf{x}(k), \mathbf{u}(k))T_s \tag{4}$$

The states and control inputs of the robot may be subject to particular constraints. For instance, if a robot is constrained to navigate in a bounded environment, then its state vector must evolve within the limits $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ defined by the coordinates of the boundaries. Moreover, in real-world implementations, the control inputs of the robot are constrained by a lower limit $\mathbf{u}_{\min}$ and an upper limit $\mathbf{u}_{\max}$, which are determined by the capabilities of the robots' actuators.

### 2.2. MPC Formulation for Single-Robot Navigation

In an MPC scheme, the future states of the robot are calculated over a prediction horizon and an objective function is minimized in order to find a sequence of optimal control solutions over this horizon. At each time step, only the first optimal control in the computed sequence is applied to the robot [9]. In the general MPC formulation, a control input $\mathbf{u}(k)$ that is close to a reference control input $\mathbf{u}_{\mathrm{ref}}(k)$ must be computed at each time step $k$ in order to drive the robot's state $\mathbf{x}(k)$ to follow a reference trajectory $\mathbf{x}_{\mathrm{ref}}(k)$. Using the notation $\|\mathbf{x}\|^2_{\mathbf{A}} := \mathbf{x}^T\mathbf{A}\mathbf{x}$ for a square matrix $\mathbf{A}$, we define the following loss function with weighting matrices $\mathbf{Q} \in \mathbb{R}^{3\times3}$ and $\mathbf{R} \in \mathbb{R}^{2\times2}$:

$$l(\mathbf{x}(k), \mathbf{u}(k)) = \left\|\mathbf{x}(k) - \mathbf{x}_{\mathrm{ref}}(k)\right\|^2_{\mathbf{Q}} + \left\|\mathbf{u}(k) - \mathbf{u}_{\mathrm{ref}}(k)\right\|^2_{\mathbf{R}} \tag{5}$$

where $\mathbf{Q}$ is positive semi-definite and $\mathbf{R}$ is positive definite. In this paper, we consider the case where the robot must be stabilized to a fixed goal pose, i.e., the *pose regulation problem*. The reference trajectory $\mathbf{x}_{\mathrm{ref}}(k)$ is defined as this single goal pose, and $\mathbf{u}_{\mathrm{ref}}(k)$ is set to zero so that no control input is driving the robot once it reaches the goal pose.

Given the loss function in Eq. (5), the discrete-time kinematic model Eq. (4), and the specified limits $\mathbf{x}_{\min}, \mathbf{x}_{\max}$ and $\mathbf{u}_{\min}, \mathbf{u}_{\max}$ on the state vector and control input vector, respectively, we can write the nonlinear optimization problem

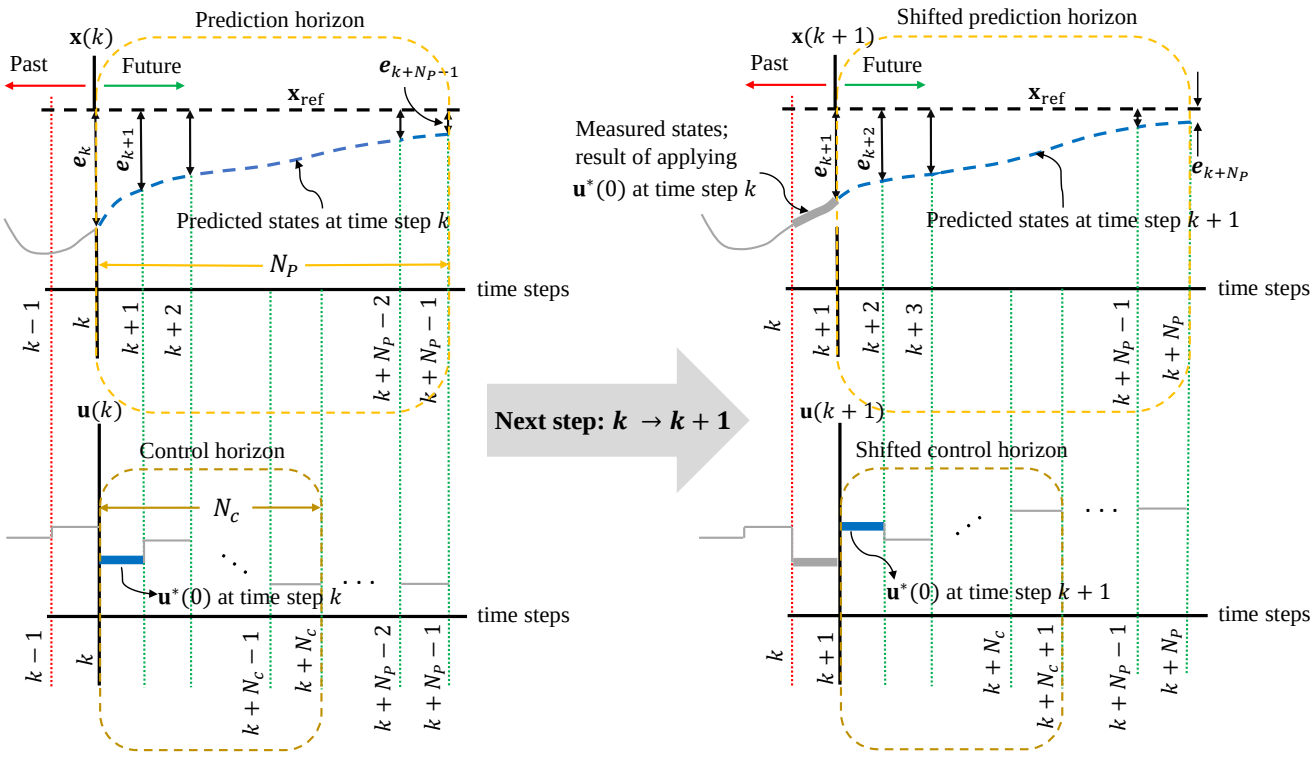


**Figure 2:** Illustration of the execution of an MPC method over two consecutive time steps, $k$ (*left*) and $k+1$ (*right*). The robot states $\mathbf{x}(k)$ (*upper plots*) and control inputs $\mathbf{u}(k)$ (*lower plots*) are computed over the prediction horizon $N_P$ and the control horizon $N_C$, respectively.

for the MPC method as follows:

$$\begin{aligned} J^*_{N_P}(\mathbf{x}(0), \mathbf{x}^*, \mathbf{u}^*) &= \min_{\mathbf{u}} \sum_{k=0}^{N_P-1} l(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{x}(k+1) &= \mathbf{x}(k) + f(\mathbf{x}(k), \mathbf{u}(k))T_s \\ \mathbf{x}_{\min} &\leq \mathbf{x}(k) \leq \mathbf{x}_{\max} \\ \mathbf{u}_{\min} &\leq \mathbf{u}(k) \leq \mathbf{u}_{\max} \\ \mathbf{x}(0) &= \mathbf{x}_c(k) \end{aligned} \tag{6}$$

where $N_P$ is the prediction horizon, $\mathbf{x}(0)$ is the initial state of the robot, and $\mathbf{x}_c(k)$ is the current measured state of the robot. The optimization problem in Eq. (6) is solved at each time step, and the initial state of the robot is updated at each step with the measured state of the robot at time step $k$. In this MPC design, the optimization problem is solved using the single-shooting method, in which only control inputs are decision variables.

Figure 2 illustrates the execution of the MPC method over two consecutive time steps, $k$ and $k+1$. The figure shows representative plots of the robot's state $\mathbf{x}$ and control input $\mathbf{u}$ over time at these two time steps. At time step $k$, the robot states at times $k$ through $k+N_P-1$, depicted by the blue dashed line in the upper left plot, are predicted according to the discrete-time model in Eq. (3). We define the error between the robot's state at time $k$ and its current goal pose as

$$\mathbf{e}(k) = \mathbf{x}(k) - \mathbf{x}_{\text{ref}}(k). \tag{7}$$

Then, we can compute a matrix $\mathbf{e}(k|k+N_P-1) \in \mathbb{R}^{3\times N_P}$ that consists of the errors between all future predicted robot states and the corresponding goal poses:

$$\mathbf{e}(k|k+N_P-1) = [\mathbf{e}(k) \quad \mathbf{e}(k+1) \quad \ldots \quad \mathbf{e}(k+N_P-1)] \tag{8}$$

The solution to the optimization problem in Eq. (6) is the sequence of optimal controls computed at time step $k$, $\mathbf{u}^*(k|k+N_P-1) \in \mathbb{R}^{2\times N_P}$. In accordance with the MPC method, the robot receives only the first optimal control input in this sequence, $\mathbf{u}^*(0) \in \mathbb{R}^{2\times 1}$, and applies this input until the next time step, as illustrated in the control input plots in Fig. 2.

The robot's measurements of its states will differ from the predicted states due to sensor noise, uncertainties in the kinematic model, and unmodeled dynamics of the environment. This disparity is depicted in the plots of the robot's state between time steps $k$ and $k+1$ in Fig. 2. To initialize the optimization problem at time step $k+1$, the prediction horizon is shifted forward by one time step and the robot's initial state is updated to its current measured state, $\mathbf{x}(0) = \mathbf{x}_c(k+1)$. The state errors over the prediction horizon, $\mathbf{e}(k+1|k+N_P)$, are computed for the new time step, and the optimization problem is solved to obtain the optimal control solutions. Again, the robot applies the first optimal control input until the next time step. This procedure is repeated until $||\mathbf{e}(k)|| = ||\mathbf{x}(k) - \mathbf{x}_{\text{ref}}(k)|| < \delta$, for some small positive constant $\delta$.

In order to reduce the computational complexity of the

optimization problem, we can compute the control inputs over a control horizon $N_C$ that is shorter than the prediction horizon $N_P$, while still predicting the future robot states over the prediction horizon. The solution to the optimization problem is then the sequence of optimal controls $\mathbf{u}^*(k|k+N_C-1) \in \mathbb{R}^{2\times N_C}$. The optimal controls for the remainder of the prediction horizon are defined as follows:

$$\mathbf{u}^*(\bar{k}) := \mathbf{u}^*(k+N_C-1), \quad \forall \bar{k} \in [k+N_C, k+N_P-1] \quad (9)$$

At the next time step, this control horizon shifts along with the prediction horizon. In this paper, we set the control horizon equal to the prediction horizon, $N_C = N_P$, in all simulations and experiments in order to predict future robot states as accurately as possible, as discussed in Section 5.1.

## 3. MPC Design for Multiple Nonholonomic Robots

In our proposed nonlinear MPC method for collision-free navigation by multiple nonholonomic WMRs, the optimal control solutions (i.e., the optimal linear and angular velocities) for all robots are obtained by solving a constrained optimization problem at each time step. The objective is to find optimal solutions such that each robot navigates to a preassigned goal pose while avoiding collisions with other robots.

### 3.1. MPC Formulation for Multi-Robot Navigation

Given $m$ robots, we define $\mathbf{x}_i$ and $\mathbf{u}_i$, $i \in \{1, 2, ..., m\}$, as the state and control input vectors, respectively, of the $i$-th robot. From Eq. (4), the discrete-time kinematic model of all robots is given by:

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + f(\mathbf{x}_i(k), \mathbf{u}_i(k))T_s, \quad i = 1, ..., m \quad (10)$$

We define vectors that contain all the robot states and control inputs at time step $k \in \{0, 1, ..., N_P - 1\}$:

$$\begin{aligned} \mathbf{X}(k) &= [\mathbf{x}_1^{\mathrm{T}}(k) \quad \mathbf{x}_2^{\mathrm{T}}(k) \quad ... \quad \mathbf{x}_m^{\mathrm{T}}(k)]^{\mathrm{T}} \in \mathbb{R}^{3m} \\ \mathbf{U}(k) &= [\mathbf{u}_1^{\mathrm{T}}(k) \quad \mathbf{u}_2^{\mathrm{T}}(k) \quad ... \quad \mathbf{u}_m^{\mathrm{T}}(k)]^{\mathrm{T}} \in \mathbb{R}^{2m} \end{aligned} \quad (11)$$

Similarly, $\mathbf{X}_c(k)$ will denote the vector containing all the robots' measured states at time step $k$. We also define $\mathbf{x}_{i,\min}$ and $\mathbf{x}_{i,\max}$ as lower and upper bounds on the $i$-th robot's state vector, and $\mathbf{u}_{i,\min}$ and $\mathbf{u}_{i,\min}$ as lower and upper bounds on its control input vector.

We consider the pose regulation problem, in which each robot must be stabilized to a fixed goal pose $\mathbf{x}_{g_i}$. We define the vector of reference states as:

$$\mathbf{X}_{\mathrm{ref}} = [\mathbf{x}_{g_1}^{\mathrm{T}} \quad \mathbf{x}_{g_2}^{\mathrm{T}} \quad ... \quad \mathbf{x}_{g_m}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{3m} \quad (12)$$

Note that $\mathbf{X}_{\mathrm{ref}}$ is a constant vector, since the goal poses of the robots are fixed. Given that the reference control inputs are set to zero in the pose regulation problem, the loss function in Eq. (5) in our multi-robot MPC formulation is defined as:

$$l(\mathbf{X}(k), \mathbf{U}(k)) = \|\mathbf{X}(k) - \mathbf{X}_{\mathrm{ref}}\|_{\mathbf{Q}}^2 + \|\mathbf{U}(k)\|_{\mathbf{R}}^2 . \quad (13)$$

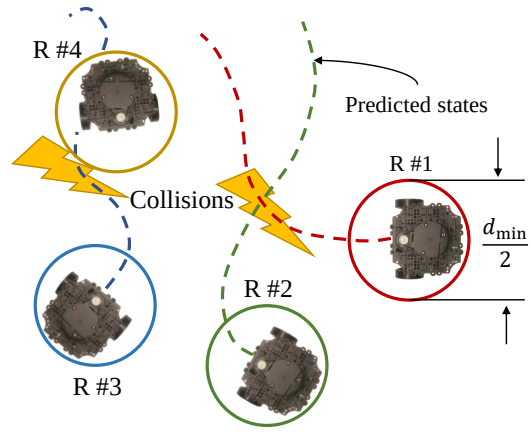


**Figure 3:** Examples of potential collisions, indicated by lightning bolts, between pairs of robots.

In order to achieve collision-free navigation of all $m$ robots, we define a set of constraints in the formulation of the MPC to prevent any collisions that may occur within the prediction horizon. These constraints restrict the optimization problem to compute optimal control commands that drive the robots along collision-free paths toward their goal poses. To this end, at every time step within the prediction horizon, we need to prevent collisions between each pair of robots. We define the minimum allowable distance between two robots, $d_{\min}$, as twice the diameter of the smallest circle that is centered at the origin of a robot's local coordinate frame and completely encloses the robot. Then, the following constraints prevent collisions between each pair of robots during the prediction horizon:

$$||\mathbf{x}_i(k|k+N_P-1) - \mathbf{x}_j(k|k+N_P-1)|| > d_{\min}, \quad \forall i \neq j, \quad (14)$$

where $i, j \in \{1, 2, ..., m\}$. We assume that the distance between the initial positions of each pair of robots, and the distance between their goal positions, is at least $d_{\min}$.

Figure 3 illustrates examples of collisions between robots that would occur if the robots followed their predicted trajectories. The $i$-th robot is labeled R#$i$. In this scenario, R#1, R#2, and R#3 are moving, and R#4 is stationary. The predicted trajectories of R#1 and R#2 intersect, and R#3 is predicted to collide with R#4. The optimal control solutions which would produce these collisions, and hence violate the collision avoidance constraints in Eq. (14), are considered to be infeasible solutions of the optimization problem at the time step $k$.

Given the loss function in Eq. (13), the discrete-time kinematic model in Eq. (10), the limits on the robots' state and control input vectors, and the collision avoidance constraints in Eq. (14), we can write the nonlinear optimization problem

for our proposed MPC method as follows, with $i = 1, ..., m$:

$$\begin{aligned} J^*_{N_P}(\mathbf{X}(0), \mathbf{X}^*, \mathbf{U}^*) = \min_{\mathbf{X},\mathbf{U}} \sum_{i=1}^{m} \sum_{k=0}^{N_P-1} l(\mathbf{X}(k), \mathbf{U}(k)) \\ ||\mathbf{x}_i(k+1) - \mathbf{x}_j(k+1)|| > d_{\min}, \quad \forall i \neq j, \\ \mathbf{x}_i(k+1) = \mathbf{x}_i(k) + f(\mathbf{x}_i(k), \mathbf{u}_i(k))T_s \\ \mathbf{x}_{i,\min} \leq \mathbf{x}_i(k) \leq \mathbf{x}_{i,\max} \\ \mathbf{u}_{i,\min} \leq \mathbf{u}_i(k) \leq \mathbf{u}_{i,\max} \\ \mathbf{X}(0) = \mathbf{X}_c(k) \end{aligned} \tag{15}$$

We solve this optimization problem using the multiple-shooting method.

### 3.2. Feasibility and Stability Analysis

In this section, we investigate the *feasibility* of the optimization problem in Eq. (15) and the *stability* of the proposed MPC method. These two properties of MPC schemes have been studied extensively in [45, 46]. The optimization problem is considered *feasible* if it has at least one solution and all its constraints are satisfied at each time step, and the MPC method is considered *stable* if all robots converge to their preassigned goal poses. A *feasible state* is a state $\mathbf{X}(k)$ that satisfies the bounds $\mathbf{x}_{i,\min} \leq \mathbf{x}_i(k) \leq \mathbf{x}_{i,\max}$ for all $i = 1, ..., m$, and a *feasible control input* is an input $\mathbf{U}(k)$ that satisfies the bounds $\mathbf{u}_{i,\min} \leq \mathbf{u}_i(k) \leq \mathbf{u}_{i,\max}$, $i = 1, ..., m$.

**Definition 3.1** (Feasible states and admissible control sets)**.** *Define $\mathbb{X}$ as the set of feasible states $\mathbf{X}$ of all robots, and $\mathbb{U}$ as the set of feasible robot control inputs $\mathbf{U}$. Then, the control inputs $\mathbf{U}(k|k+N_P-1)$ are admissible for any initial state $\mathbf{X}(0) \in \mathbb{X}$ if:*
***(1)*** *the control inputs are a subset of the feasible control set for all time steps in the prediction horizon, i.e. $\mathbf{U}(k|k+N_P-1) \subseteq \mathbb{U}$, and*
***(2)*** *the predicted states are a subset of the feasible state set for all time steps in the horizon, i.e. $\mathbf{X}(k|k+N_P-1) \subseteq \mathbb{X}$. To simplify the notation, we also define the set of all feasible states and feasible control inputs as $\mathbb{Z} := \mathbb{X} \times \mathbb{U}$.*

**Theorem 3.2** (**Feasibility**)**.** *Suppose that the initial state $\mathbf{X}(0)$ is known and is a feasible state, i.e. $\mathbf{X}(0) \in \mathbb{X}$. Then, the constrained nonlinear MPC optimization problem in Eq. (15) has feasible solutions if and only if for all time steps $k$ in the prediction horizon, the following statements hold:*
***(a)*** *The optimal control solutions and corresponding predicted states are a subset of the feasible set, i.e. $\mathbf{X}^*(k|k+N_P-1) \times \mathbf{U}^*(k|k+N_P-1) \subseteq \mathbb{Z}$.*
***(b)*** *The collision avoidance constraints are satisfied for all time steps in the horizon, i.e. $||\mathbf{x}_i(k|k+N_P+1) - \mathbf{x}_j(k|k+N_P+1)|| > d_{min}$, $\forall i \neq j$.*

*Proof.* Given the nonlinear system in Eq. (2), if the initial state and initial control input are in the feasible set, i.e. $\mathbf{X}(0) \times \mathbf{U}(0) \in \mathbb{Z}$, then the recursive feasibility, defined in [47], of the optimization problem in Eq. (15) trivially holds. In other words, the set of optimal control solutions $\mathbf{U}^*(k|k+N_P-1)$ is not the empty set. To prove that the collision avoidance constraints are satisfied for all time steps $k, ..., k+N_P-1$, we consider a contradictory case. Suppose that there exists an optimal control solution for which two robots have a point in common on their predicted paths that they will reach at the same time, leading to a collision (see robots R#1 and R#2 in Fig. 3). According to the *Nyquist-Shannon* sampling theorem, a discrete sequence of samples can be used to adequately reconstruct a continuous-time signal given a small enough sampling time, $T_s$. Thus, for $T_s$ small enough, the MPC method will identify the aforementioned point of collision between the robots and compute an optimal admissible control solution that avoids producing this collision. As a result, satisfying the collision avoidance constraints at each time step is sufficient to guarantee collision-free navigation by all the robots. ■

Proving the stability of our proposed NMPC method is challenging, since the method requires solving a finite-horizon optimal control problem at each time step. Most existing NMPC-based approaches for pose stabilization of nonholonomic robots ensure stability by incorporating stabilizing terminal constraints and/or stabilizing terminal costs into the optimization problem [44, 48, 49, 50, 51, 52, 2, 53]. However, constructing suitable stabilizing terminal constraints or costs is a challenging task, and moreover, this approach limits the operating region of the MPC and necessitates a relatively long prediction horizon [54]. These issues in turn make the NMPC method computationally prohibitive to implement. The stability of NMPC methods for nonholonomic robots has also been proved without the introduction of stabilizing terminal constraints and terminal costs [55, 56, 57]. We establish our proposed MPC method's stability using this approach, which is detailed in [58]. Our method extends these types of NMPC methods, which are designed for single robots, to multi-robot systems and introduces constraints into the NMPC that enforce inter-robot collision avoidance. The objective is to prove that there exists an optimal value function, defined as:

$$V_{N_P}(\mathbf{X}(0)) := \inf J_{N_P}(\mathbf{X}(0), \mathbf{X}, \mathbf{U}) \quad \forall \mathbf{X} \times \mathbf{U} \in \mathbb{Z}, \tag{16}$$

that guarantees the asymptotic stability of our NMPC design with prediction horizon $N_P$. We first state several definitions and assumptions that are needed in the stability proof.

**Definition 3.3** ($\mathcal{K}_\infty$-function)**.** *Continuous functions $\zeta(r) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that are strictly increasing, and for which $\zeta(0) = 0$, are categorized as the class of $\mathcal{K}$-functions. Those $\mathcal{K}$-functions which satisfy $\lim_{r\to\infty} \zeta(r) = \infty$ are called the class of $\mathcal{K}_\infty$-functions.*

**Assumption 3.4.** *Define $J_k$ as the following cost function, which is calculated over the first $k$ steps of the prediction horizon:*

$$J_k(\mathbf{X}(0), \mathbf{X}, \mathbf{U}) = \min_{\mathbf{X},\mathbf{U}} \sum_{i=1}^{m} \sum_{\bar{k}=0}^{k} l(\mathbf{x}_i(\bar{k}), \mathbf{u}_i(\bar{k})). \tag{17}$$

*Then, for any initial state $\mathbf{X}(0) \in \mathbb{X}$, the following holds:*

$$V_k(\mathbf{X}(0)) := \inf J_k(\mathbf{X}(0), \mathbf{X}, \mathbf{U}) \leq \eta_k l(\mathbf{X}^*, \mathbf{U}^*), \tag{18}$$

*where $\eta_k$, $k \in \mathbb{N}$, is a bounded sequence of monotonically increasing natural numbers.*

**Assumption 3.5.** *There exist two $\mathcal{K}$-functions $\zeta_1$ and $\zeta_2$ which satisfy the following:*

$$\zeta_1(||\mathbf{X}-\mathbf{X}_{\text{ref}}||) \leq l(\mathbf{X}^*, \mathbf{U}^*) \leq \zeta_2(||\mathbf{X}-\mathbf{X}_{\text{ref}}||). \quad (19)$$

**Definition 3.6** (Performance index)**.** *The performance index $\gamma_{N_P}$, which determines the minimum value of the prediction horizon $N_P$ that ensures the asymptotic stability of the MPC method [58], is defined as:*

$$\gamma_{N_P} := 1 - \frac{(\eta_{N_P}-1)\prod_{k=1}^{N_P-1}(\eta_k-1)}{\prod_{k=1}^{N_P-1}\eta_k - \prod_{k=1}^{N_P-1}(\eta_k-1)}. \quad (20)$$

**Theorem 3.7** (**Stability**)**.** *Suppose that the initial state $\mathbf{X}(0)$ is known and that the initial state and initial control inputs are feasible, i.e. $\mathbf{X}(0) \times \mathbf{U}(0) \in \mathbb{Z}$. Given Assumption 3.4, the closed-loop controller computed by the nonlinear constrained MPC in Eq. (15) is asymptotically stable with the prediction horizon $N_P$ if for a positive performance index $\gamma_{N_P} > 0$, the relaxed Lyapunov inequality holds:*

$$V_{N_P}(\mathbf{X}(k+1)) \leq V_{N_P}(\mathbf{X}(k)) - \gamma_{N_P} l(\mathbf{X}(k), \mathbf{U}(k)). \quad (21)$$

*Proof.* Suppose that Assumption 3.4 and the inequality in Eq. (21) hold, and the initial robot states and control inputs are feasible, i.e. $\mathbf{X}(0) \times \mathbf{U}(0) \in \mathbb{Z}$. Given a sequence $c_k \subseteq \mathbb{R}_{\geq 0}$, where $\sum_{k=0}^{\infty} c_k < \infty$, such that

$$l(\mathbf{X}(k), \mathbf{U}(k)) \leq c_k l(\mathbf{X}^*, \mathbf{U}^*), \quad \forall k \in \{1, 2, ..., N_P - 1\}, \quad (22)$$

the bounds in Eq. (18) are achieved by setting $\eta_k = \sum_{\bar{k}=0}^{k-1} c_{\bar{k}}$ [32]. Thus, an upper bound on the optimal value function at time step $k$, i.e. $V_k$, can be obtained as follows:

$$\begin{aligned} V_k(\mathbf{X}(0)) &\leq \sum_{\bar{k}=0}^{k-1} l(\mathbf{X}(\bar{k}), \mathbf{U}(\bar{k})) \\ &\leq \sum_{\bar{k}=0}^{k-1} c_{\bar{k}} l(\mathbf{X}^*(\bar{k}), \mathbf{U}^*(\bar{k})) = \eta_k l(\mathbf{X}^*(k), \mathbf{U}^*(k)), \end{aligned} \quad (23)$$

in which the sequence $\eta_k$ is monotonically increasing for $c_k \geq 0$. For further details, we refer the reader to [53, 54, 58, 59]. ■

**Corollary 3.8.** *Theorem 3.7 implies that the constrained nonlinear MPC design in Eq. (15) produces stable optimal control solutions if and only if the predicted sequence of states for each robot, i.e. $\mathbf{x}_i(k)$, $i = 1, ..., m$, converges to its desired goal pose $\mathbf{x}_{g_i}$ in finite time starting from any feasible initial state $\mathbf{X}(0) \in \mathbb{X}$.*

### 3.3. Algorithms for Implementation of the MPC

Figure 4 illustrates the framework for implementing our proposed MPC method in the Robot Operating System (ROS). We first specify the number of robots $m$; the robots' initial poses $\mathbf{X}(0)$ and goal poses $\mathbf{X}_{\text{ref}}$; the bounds on the robots' states and control inputs, i.e. $\mathbf{x}_{i,\min}$, $\mathbf{x}_{i,\max}$, $\mathbf{u}_{i,\min}$, and $\mathbf{u}_{i,\max}$; the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$; the sampling time $T_s$; and the distance $d_{\min}$. The bounds on the robots' linear and angular velocities are defined as $v_i \in [-0.22 \quad 0.22]$ (m/s) and $w_i \in [-2.84 \quad 2.84]$ (rad/s), according to the Burger robot's specifications [42]. We use the symbolic programming framework in CasADi [27] to formulate the optimization problem in Eq. (6) for the case of a single robot, or the one in Eq. (15) for multiple robots. The optimization problem is solved using the Interior Point OPTimizer (IPOPT), an open-source software library for large-scale nonlinear optimization problems that is available in CasADi. For details about IPOPT options and the tuning parameters in CasADi, see [27, 60].

Algorithms 1, 2, and 3 contain pseudocode that describes the implementation of our proposed MPC method in both simulation and experiment. A central supervisor runs these algorithms to solve the optimization problem and send optimal control commands to the robots. The robots estimate their current poses relative to their initial poses (which are known) using odometry, fusing their wheel encoder measurements and IMU sensor data to improve the accuracy of the pose estimates. Using ROS, the robots *publish* their odometry information on a *topic*, and the central supervisor *subscribes* to each robot's published odometry topic in order to obtain the poses of the robots. The supervisor publishes the optimal control solutions to the robots' velocity commands. These control commands are published to the robots' velocity topics, which are each defined as a ROS topic.

Algorithm 1 executes the main loop of the program. First, it initializes the optimization problem of the MPC, the ROS core node, the subscribers that read the robots' odometry measurements, and the velocity command publishers. It also sets the options for the Nonlinear Programming (NLP) solver, including the acceptable convergence tolerance (acceptable_tol), the stopping criterion based on the change in the objective function value (acceptable_obj_change_tol), and the maximum number of iterations (max_iter) for the IPOPT solver. Next, the following procedure repeats as long as the error between the current robot poses and their goal poses, $||\mathbf{X}-\mathbf{X}_{\text{ref}}||$, exceeds a small positive constant $\delta$. First, Algorithm 2 obtains the robots' measurements of their own poses by subscribing to each robot's odometry callback function. Then, the IPOPT solves the optimization problem in either Eq. (6) or Eq. (15). As described earlier, only the optimal control solutions computed at the first time step of the prediction horizon, i.e. $\mathbf{U}^*(0)$, are applied to the robots. Finally, Algorithm 3 shifts the prediction and control horizons forward by one time step and re-initializes the optimization problem using the *warm start* approach, in which the optimal solutions at the previous time step are defined as the initial solutions (also referred to as initial guesses) at the next time

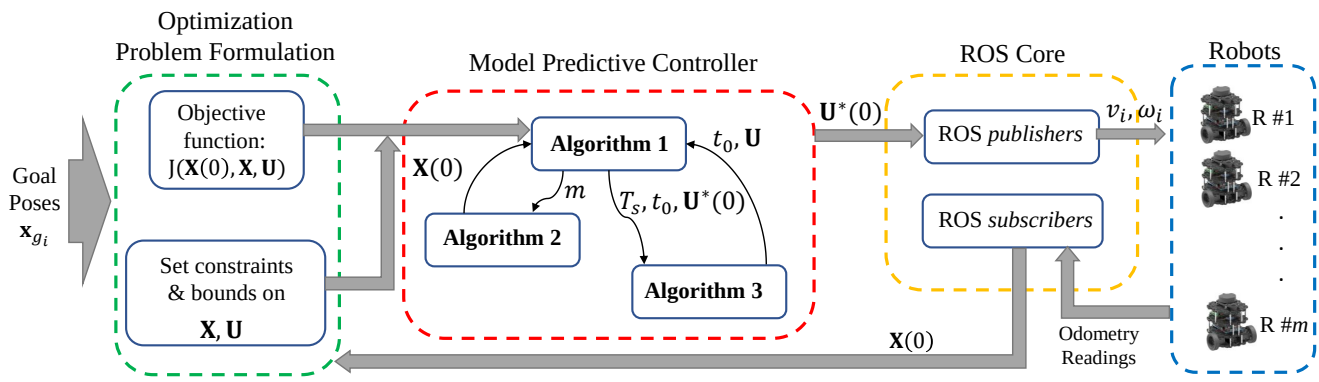


**Figure 4:** Framework for implementing the proposed MPC method for collision-free navigation by multiple WMRs.

**Algorithm 1** Main Function of MPC Method

**Input:** $\mathbf{X}(k|k+N_P-1)$, $\mathbf{X}(0)$, $\mathbf{X}_{\text{ref}}(k|k+N_P-1)$, $\mathbf{U}(k|k+N_P-1)$, $\mathbf{Q}$, $\mathbf{R}$, $N_P$, $m$, $\mathbf{x}_{i,\min}$, $\mathbf{x}_{i,\max}$, $\mathbf{u}_{i,\min}$, $\mathbf{u}_{i,\max}$, $f(\mathbf{x}_i(k), \mathbf{u}_i(k))$, $T_s$, $d_{\min}$

**Output:** $\mathbf{X}^*(k|k+N_P-1)$, $\mathbf{U}^*(k|k+N_P-1)$

Initialize ROS node, odometry subscribers, and velocity command publishers; import CasADi
Symbolically formulate the optimization problem in Eq. (6) or Eq. (15) using CasADi
Set NLP solver (IPOPT) options: acceptable_tol = $10^{-8}$, acceptable_obj_change_tol = $10^{-6}$, max_iter = 2000
**while** $||\mathbf{X}(k) - \mathbf{X}_{\text{ref}}(k)|| > \delta$ **do**
    Algorithm 2: Obtain the robots' pose measurements
    Solve optimization problem in Eq. (6) or Eq. (15)
    Publish $\mathbf{U}^*(0)$ to ROS topics of robots' velocity commands
    Algorithm 3: Shift the prediction and control horizons, re-initialize the optimization problem with warm start
**end while**

**Algorithm 2** Obtain Robots' Pose Measurements

**Input:** $m$, $k$

**Output:** $\mathbf{X}_c(k)$

$i \leftarrow 1$, $\mathbf{X}_c(k) \leftarrow [\,]$
**for** $i \leq m$ **do**
    Call odometry callback function of $i$-th robot
    Read pose measurement $\mathbf{x}_i \leftarrow [x_i \quad y_i \quad \theta_i]$
    Concatenate $\mathbf{X}_c(k)$ and $\mathbf{x}_i$
**end for**
Return $\mathbf{X}_c(k)$

step. The robots' states are re-initialized with their current poses.

## 4. Simulation and Experimental Results

In this section, we validate our NMPC method for collision-free navigation in both simulations and experiments with real-world robots. We performed the simulations in Gazebo, using

**Algorithm 3** Shift Horizons & Initialize Next Step

**Input:** $\mathbf{U}^*(0)$, $\mathbf{X}_c(k)$

**Output:** $\mathbf{U}(0)$, $\mathbf{X}(0)$

Update initial guess: $\mathbf{U}(0) \leftarrow \mathbf{U}^*(0)$
Update initial state: $\mathbf{X}(0) \leftarrow \mathbf{X}_c(k)$
Increment time step: $k \leftarrow k+1$
Return $\mathbf{U}(0)$, $\mathbf{X}(0)$

**Table 1**
Sampling time (s) and prediction horizon (time steps) for each scenario in the simulations and experiments.

| | Simulation | | | Experiment | | |
|---|---|---|---|---|---|---|
| **Scenario** | **1** | **2** | **3** | **1** | **2** | **3** |
| Sampling time | 0.1 | 0.1 | 0.35 | 0.1 | 0.1 | 0.3 |
| Prediction horizon | 50 | 35 | 35 | 45 | 40 | 35 |

the Turtlebot3 Burger (see Fig. 1) as the robot platform, and conducted the experiments with multiple Burger robots. The Burger robot is a two-wheeled differential-drive WMR that is equipped with Raspberry Pi and OpenCR boards, providing an integrated embedded system of sensors and processors which can be programmed in ROS to implement our MPC method. More details about this robot can be found in [42]. Videos of the simulations and experiments are available online in [61] and [62], respectively.

We tested our MPC method in six different scenarios, each implemented in both simulations and experiments. Three of the scenarios, in which there are $m \in \{1, 2, 3\}$ robots, are shown in the videos [61, 62] but are not discussed here for the sake of conciseness. The other three scenarios, in which there are $m \in \{4, 5, 6\}$ robots, are described below:

*Scenario 1 : Collision-free Position Swap of Four Robots.* Four robots are initially located at the vertices of a square and are all oriented toward the center of the square (the origin). Each robot must switch positions with the robot at the opposite vertex along the diagonal of the square. The robots' goal headings are the same as their initial headings.

*Scenario 2 : Collision-free Reconfiguration of Five Robots.* Five robots start in a V-formation and must reconfigure into a symmetric formation about the $y$-axis, with their goal headings the same as their initial headings. The goal positions of the robots are assigned such that they must avoid many potential collisions with one another as they navigate to their goal poses.

*Scenario 3 : Collision-free Position Swap of Six Robots.* Six robots start at the vertices of a regular hexagon, facing toward the center (the origin). Each robot must switch positions with the robot at the diametrically opposite vertex while avoiding collisions with one another. The robots' goal headings are the same as their initial headings. (*Note:* In the experiment for this scenario, the ranges of $v_i$ and $w_i$ were reduced to $v_i \in [-0.15 \;\; 0.15]$ (m/s), $w_i \in [-1.5 \;\; 1.5]$ (rad/s) to accommodate the relatively high density of robots in the testbed.)

Table 1 lists the sampling time $T_s$ and the prediction horizon $N_P$ that were used in each of these scenarios in the simulations and experiments. For these scenarios, and for the scenarios with $m = 2$ and $m = 3$ robots, the weighting matrices $\mathbf{Q}_m$ and $\mathbf{R}_m$ were defined as the Kronecker products of the matrices in Eq. (25) with the $m \times m$ identity matrix $\mathbf{I}_{m\times m}$:

$$\mathbf{Q}_m = \mathbf{Q} \otimes \mathbf{I}_{m\times m}, \quad \mathbf{R}_m = \mathbf{R} \otimes \mathbf{I}_{m\times m}, \tag{24}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are the weighting matrices for the scenario with $m = 1$ robot, defined as:

$$\mathbf{Q} = \mathrm{diag}(1, 5, 0.1), \quad \mathbf{R} = \mathrm{diag}(0.5, 0.05). \tag{25}$$

Figures 5–10 show snapshots of the robots at several time steps during simulations and experimental runs of the *Scenarios 1–3*. The robots' trajectories are plotted as colored dashed lines, and their goal positions and goal orientations are indicated by colored circles and arrows, respectively, with a different color assigned to each robot. In the figure captions, the goal position coordinates are given in meters, and the goal orientations are in radians. The snapshots show that in each scenario, all robots successfully navigate to their goal poses without colliding with one another or becoming trapped in a deadlock configuration. In addition, Figs. 11–22 plot the time evolution of the robots' optimal control inputs, i.e. the optimal linear and angular velocities computed by our MPC method, during the simulations and experimental runs of the scenarios.

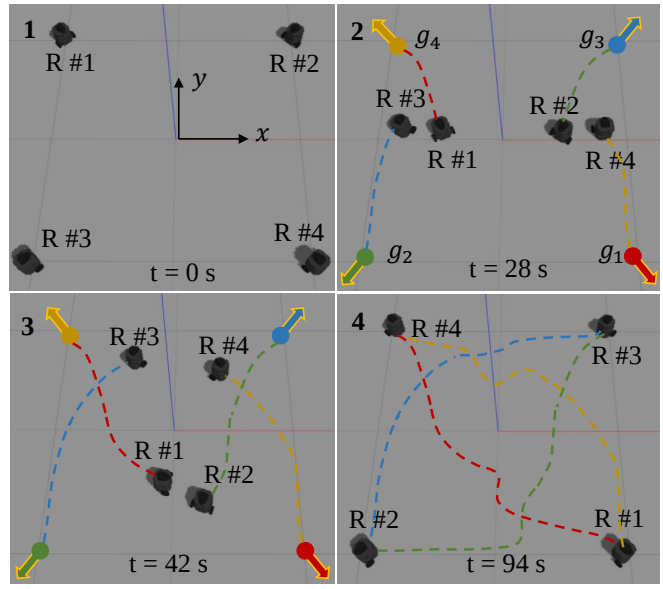


**Figure 5:** Snapshots of the Gazebo simulation of *Scenario 1*. The robots' goal poses are: $\mathbf{x}_{g_1} = [1 \;\; -1 \;\; -0.785]^\mathsf{T}$, $\mathbf{x}_{g_2} = [-1 \;\; -1 \;\; -2.356]^\mathsf{T}$, $\mathbf{x}_{g_3} = [1 \;\; 1 \;\; 0.785]^\mathsf{T}$, $\mathbf{x}_{g_4} = [-1 \;\; 1 \;\; 2.356]^\mathsf{T}$.

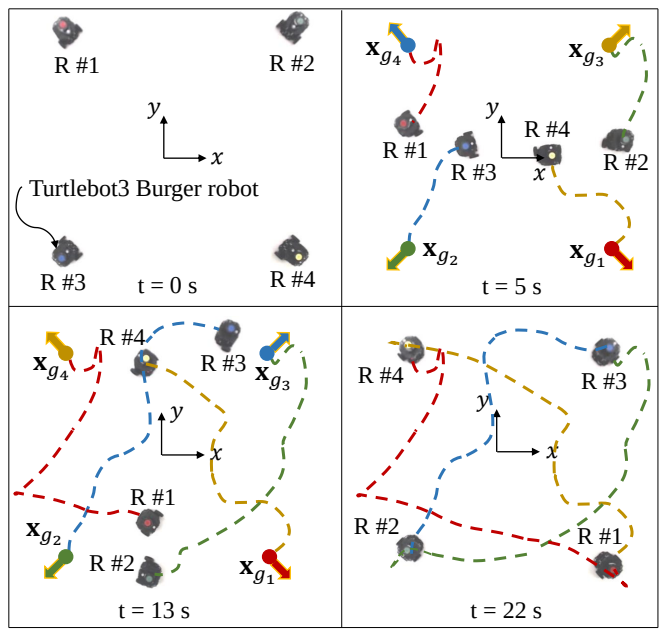


**Figure 6:** Snapshots of the experimental run of *Scenario 1*. The robots' goal poses are: $\mathbf{x}_{g_1} = [0.71 \;\; -0.71 \;\; -0.785]^\mathsf{T}$, $\mathbf{x}_{g_2} = [0.71 \;\; 0.71 \;\; -2.356]^\mathsf{T}$, $\mathbf{x}_{g_3} = [-0.71 \;\; -0.71 \;\; 0.785]^\mathsf{T}$, $\mathbf{x}_{g_4} = [0.71 \;\; -0.71 \;\; 2.356]^\mathsf{T}$.

## 5. Discussion and Analysis

In this section, we first give recommendations on tuning several parameters in our MPC method for the scenarios described in Section 4. Next, we describe types of deadlocks that can occur in each scenario and discuss how our method can resolve these deadlocks. Finally, we identify some limitations of our method.

### 5.1. Tuning MPC Parameters

Proper selection of the MPC parameters $T_s$, $N_P$, $N_C$, $\mathbf{Q}$, and $\mathbf{R}$ is important, since they can affect both the controller performance and the computational complexity of the method. Furthermore, these parameters may need to be retuned for different multi-robot navigation problems.

**Sampling time ($T_s$):** The sampling time determines the res-

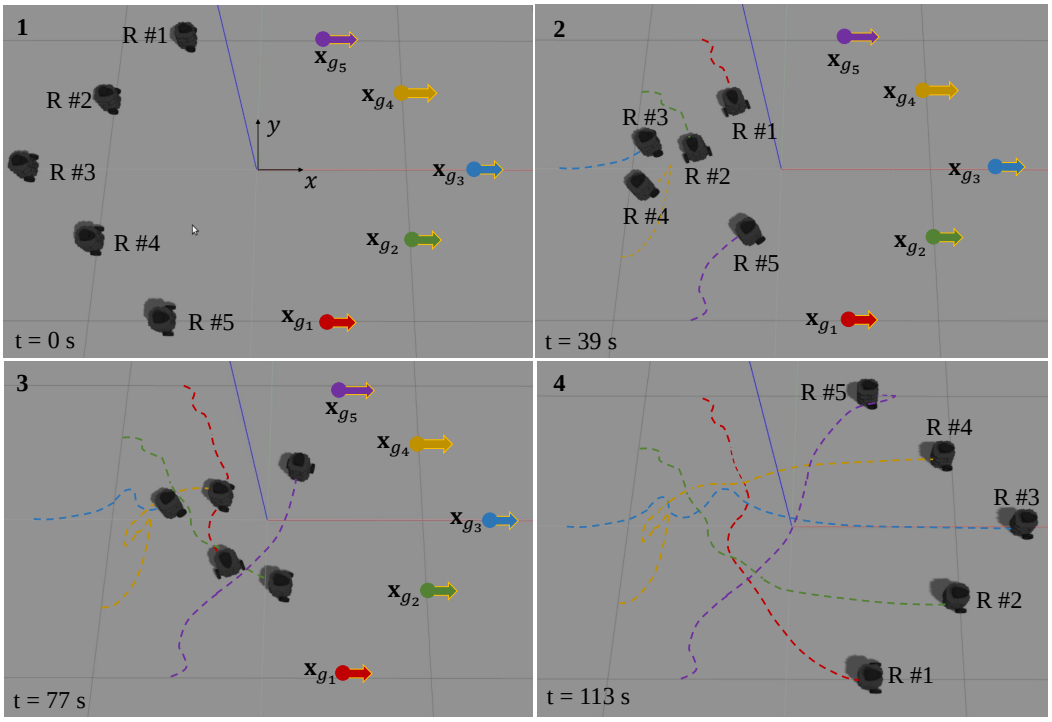


**Figure 7:** Snapshots of the Gazebo simulation of *Scenario 2*. The robots' goal poses are: $\mathbf{x}_{g_1} = [1 \;\; 0.5 \;\; 0]^\mathsf{T}$, $\mathbf{x}_{g_2} = [0 \;\; 0.75 \;\; -1.57]^\mathsf{T}$, $\mathbf{x}_{g_3} = [-0.5 \;\; -0.5 \;\; 3.14]^\mathsf{T}$, $\mathbf{x}_{g_4} = [-0.5 \;\; -0.75 \;\; 0.785]^\mathsf{T}$, $\mathbf{x}_{g_5} = [0.75 \;\; -0.75 \;\; -0.785]^\mathsf{T}$.

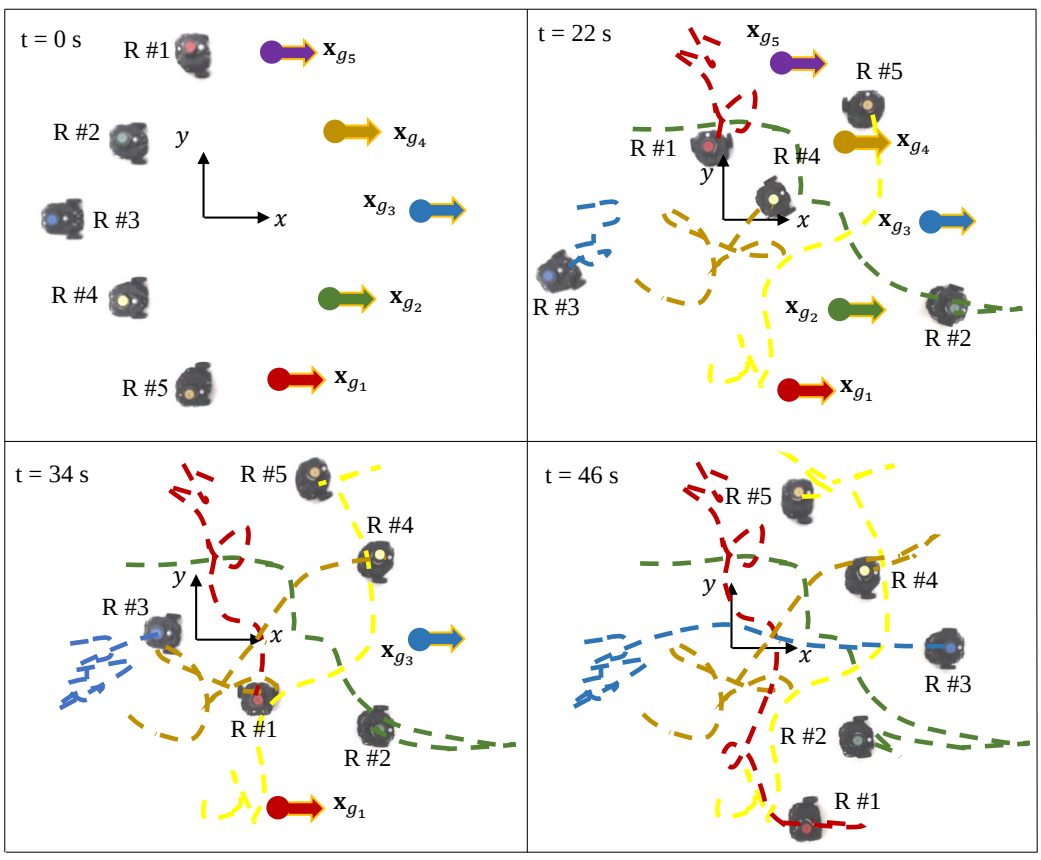


**Figure 8:** Snapshots of the experimental run of *Scenario 2*. The robots' goal poses are: $\mathbf{x}_{g_1} = [0.56 \;\; -0.71 \;\; 0]^\mathsf{T}$, $\mathbf{x}_{g_2} = [0.84 \;\; -0.36 \;\; 0]^\mathsf{T}$, $\mathbf{x}_{g_3} = [1.18 \;\; 0 \;\; 0]^\mathsf{T}$, $\mathbf{x}_{g_4} = [0.84 \;\; 0.36 \;\; 0]^\mathsf{T}$, $\mathbf{x}_{g_5} = [0.56 \;\; 0.71 \;\; 0]^\mathsf{T}$.

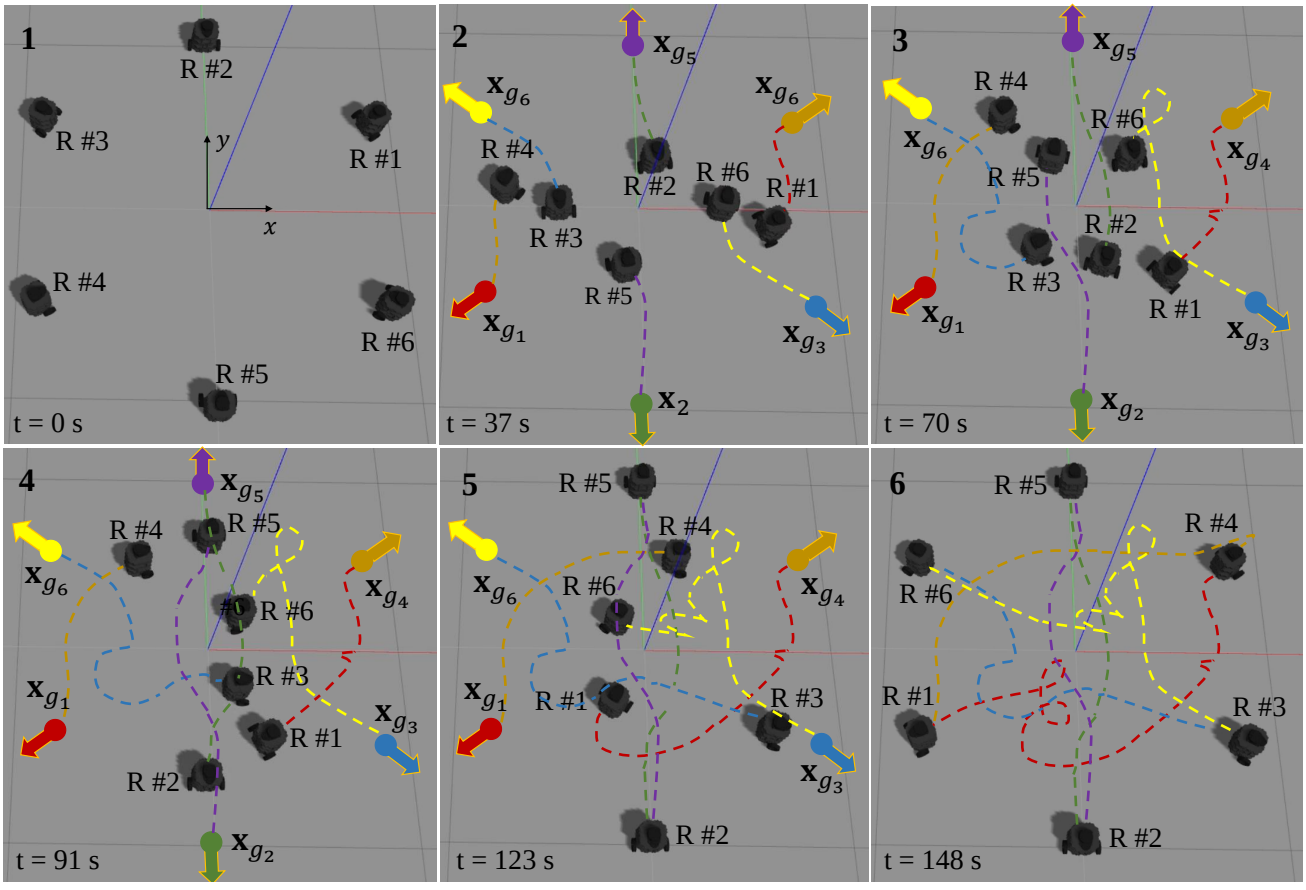


**Figure 9:** Snapshots of the Gazebo simulation of *Scenario 3*. The robots' goal poses are: $\mathbf{x}_{g_1} = [-0.866 \;\; -0.5 \;\; -2.618]^\mathsf{T}$, $\mathbf{x}_{g_2} = [0 \;\; -1 \;\; -1.57]^\mathsf{T}$, $\mathbf{x}_{g_3} = [0.866 \;\; -0.5 \;\; -0.523]^\mathsf{T}$, $\mathbf{x}_{g_4} = [0.866 \;\; 0.5 \;\; 0.523]^\mathsf{T}$, $\mathbf{x}_{g_5} = [0 \;\; 1 \;\; 1.57]^\mathsf{T}$, $\mathbf{x}_{g_6} = [-0.866 \;\; 0.5 \;\; 2.618]^\mathsf{T}$.

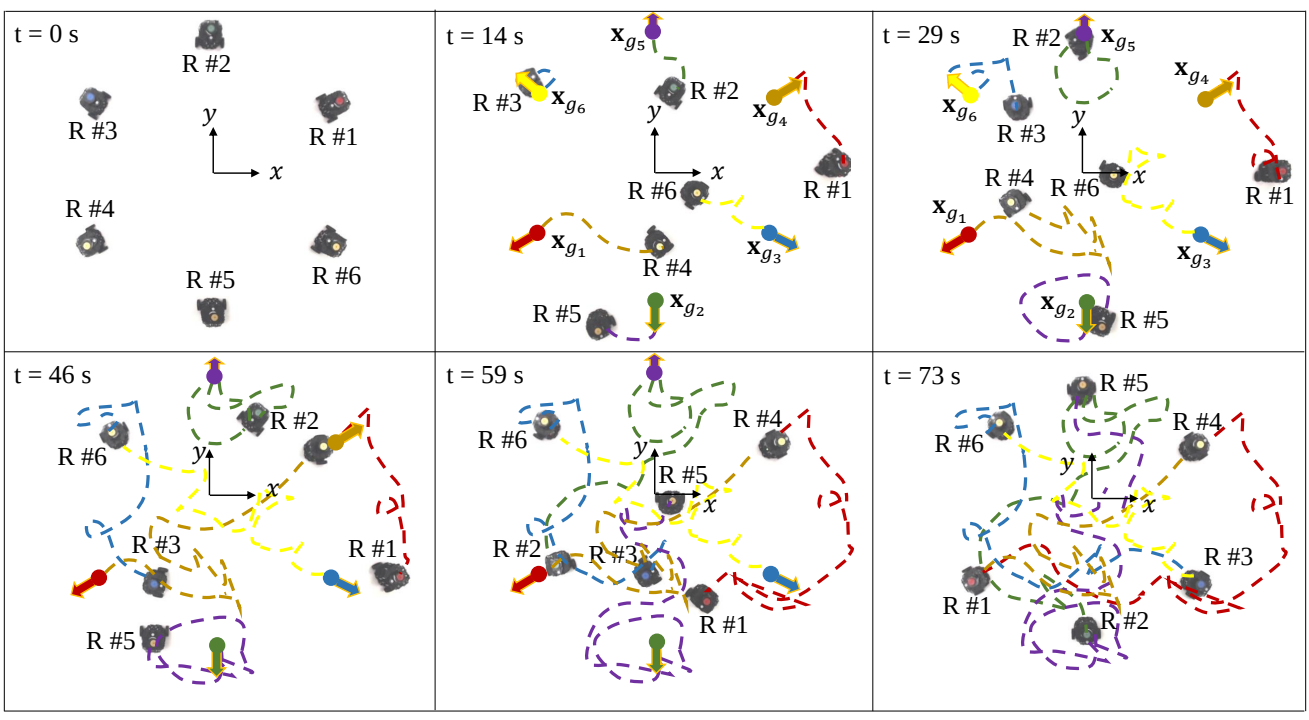


**Figure 10:** Snapshots of the experimental run of *Scenario 3*. The robots' goal poses are: $\mathbf{x}_{g_1} = [-0.7 \;\; -0.4 \;\; -2.618]^\mathsf{T}$, $\mathbf{x}_{g_2} = [0 \;\; -0.8 \;\; -1.57]^\mathsf{T}$, $\mathbf{x}_{g_3} = [0.7 \;\; -0.4 \;\; -0.523]^\mathsf{T}$, $\mathbf{x}_{g_4} = [0.7 \;\; 0.4 \;\; 0.523]^\mathsf{T}$, $\mathbf{x}_{g_5} = [0 \;\; 0.8 \;\; 1.57]^\mathsf{T}$, $\mathbf{x}_{g_6} = [-0.7 \;\; 0.4 \;\; 2.618]^\mathsf{T}$.

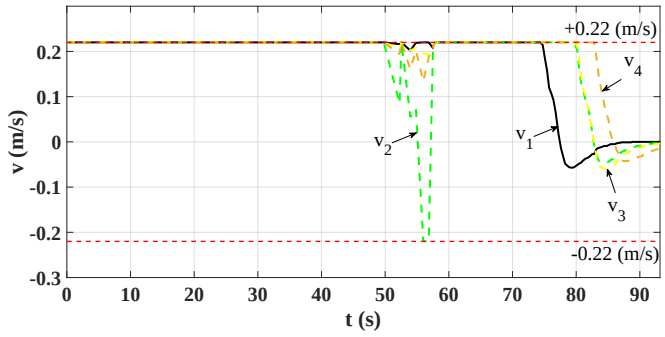


**Figure 11:** Optimal robot linear velocities over time in the simulation of *Scenario 1*

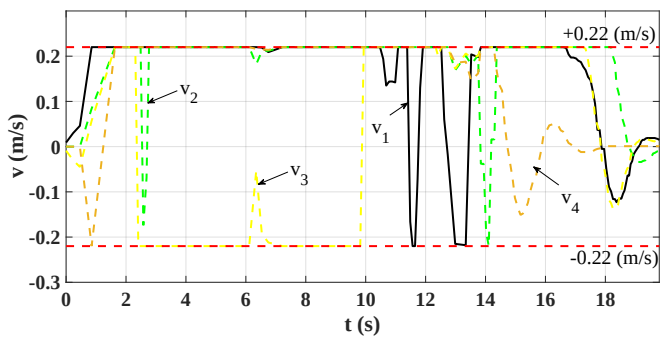


**Figure 12:** Optimal robot linear velocities over time during the experimental run of *Scenario 1*.

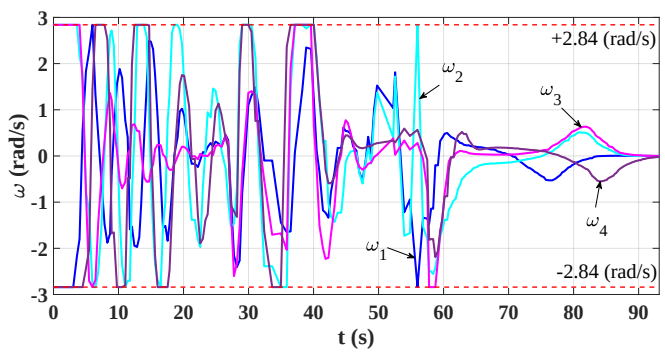


**Figure 13:** Optimal robot angular velocities over time in the simulation of *Scenario 1*.

olution of the discrete-time kinematic model in Eq. (3) and the rate at which the optimization problem in Eq. (15) is solved. If $T_s$ is too large, then the kinematic model will not accurately predict the robots' future states, and therefore potential collisions between robots may not be identified and the robots may navigate to points that are far from their goal positions. Moreover, collisions can occur between robots that follow optimal control velocities which are infrequently updated by the MPC method, and therefore are not adjusted for imminent collisions. On the other hand, a very small $T_s$ will result in more accurate predictions of the future states and identification of all possible robot collisions, as discussed in the proof of Theorem 3.2. However, reducing $T_s$ increases the computational cost to solve the optimization problem, as

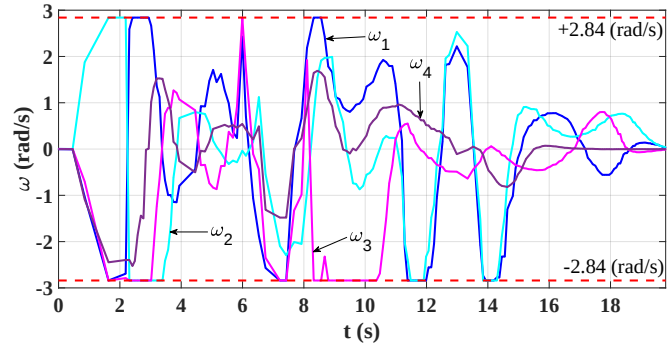


**Figure 14:** Optimal robot angular velocities over time during the experimental run of *Scenario 1*.

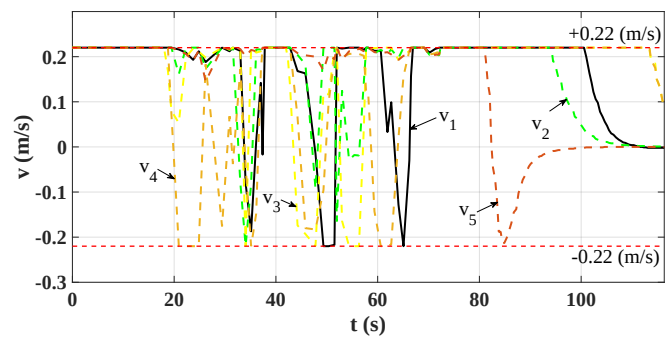


**Figure 15:** Optimal robot linear velocities over time in the simulation of *Scenario 2*.

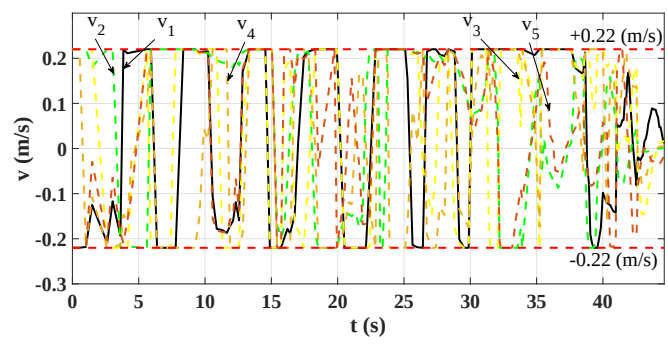


**Figure 16:** Optimal robot linear velocities over time during the experimental run of *Scenario 2*.

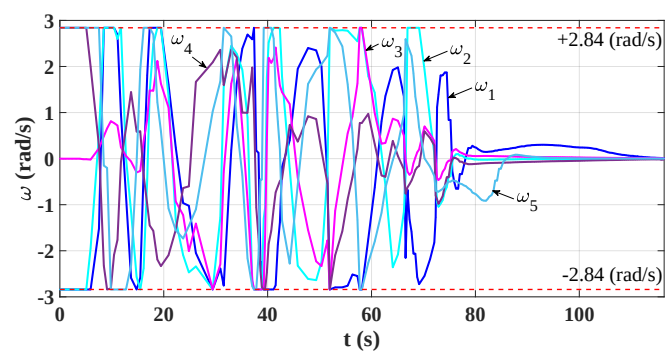


**Figure 17:** Optimal robot angular velocities over time in the simulation of *Scenario 2*.

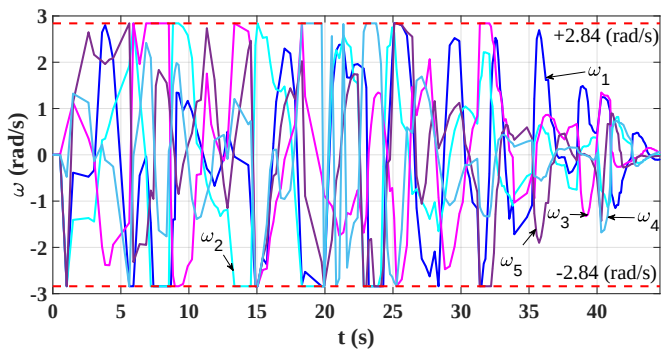


**Figure 18:** Optimal robot angular velocities over time during the experimental run of *Scenario 2*.

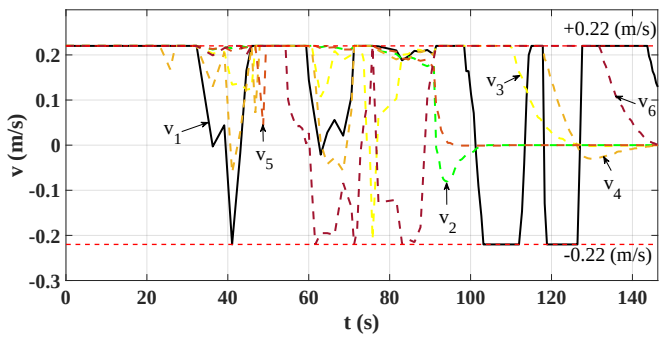


**Figure 19:** Optimal robot linear velocities over time in the simulation of *Scenario 3*.

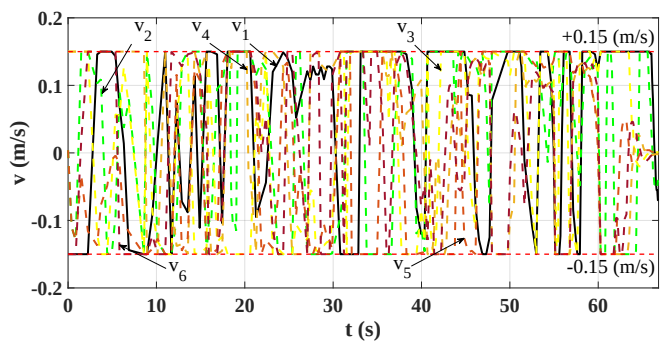


**Figure 20:** Optimal robot linear velocities over time during the experimental run of *Scenario 3*.

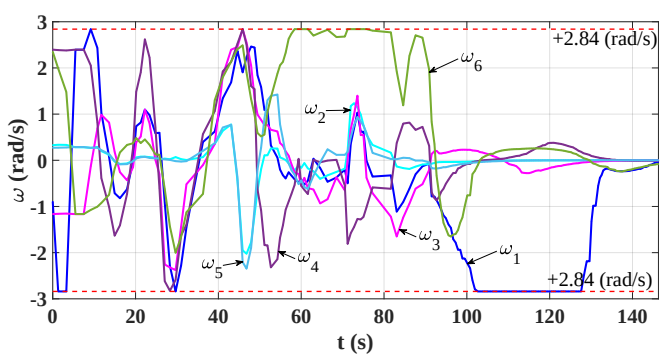


**Figure 21:** Optimal robot angular velocities over time in the simulation of *Scenario 3*.

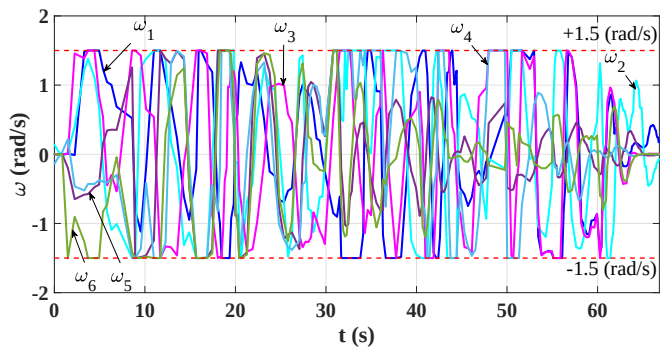


**Figure 22:** Optimal robot angular velocities over time during the experimental run of *Scenario 3*.

we observed in our simulations and experiments.

**Prediction horizon ($N_P$):** The prediction horizon determines the number of time steps over which the MPC method predicts the robots' future states. If $N_P$ is too small, then collisions between approaching robots may not be predicted, or even if robots avoid collisions, they can become stuck in immobile deadlock configurations (see Section 5.2). If $N_P$ is large, then the MPC method may produce overly conservative optimal control velocities that steer the robots away from each other prematurely, in an effort to prevent future collisions that are in fact too distant in time to accurately predict. Moreover, increasing $N_P$ raises the computational cost to solve the optimization problem, similar to the effect of decreasing $T_s$. Based on our empirical observations, our recommendation is to set $N_P$ between 25 and 110 time steps.

**Control horizon ($N_C$)**: The control horizon, $N_C$, can be set to any value such that $N_C \leq N_P$. In this paper, we choose $N_C = N_P$ in all simulations and experiments. In MPC design, the smaller the control horizon, the lower the computational cost to solve the optimization problem. However, increasing the control horizon improves the accuracy of the predicted future robot states, since they are computed using a longer sequence of optimal control inputs. Therefore, if sufficient computational resources are available, we recommend to set $N_C = N_P$ in order to achieve the best possible predictions of the future robot states. Alternatively, we may define $N_C$ as 10%–20% of $N_P$, as is standard in MPC design.

**Weighting matrices (Q and R)**: The weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ are used to define the objective function of the optimization problem in the MPC formulation. The MPC must simultaneously satisfy two competing objectives: (1) the robots' poses, $\mathbf{X}$, should converge as closely as possible to their goal poses, $\mathbf{X}_{\text{ref}}$; and (2) the optimal control solutions should be sufficiently smooth to avoid aggressive control maneuvers. The weighting matrices establish the relative importance of these two objectives. One way to satisfy the requirement that $\mathbf{Q}$ is positive semi-definite and $\mathbf{R}$ is positive definite is to define them as diagonal matrices with positive

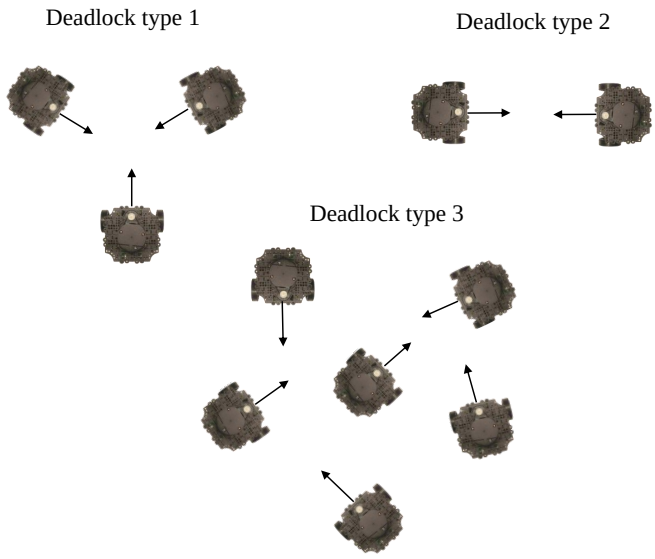


**Figure 23:** Three types of deadlocks that can occur in our multi-robot navigation scenarios. Illustration is based on Fig. 8 in [19].

entries $q_{ii}$, $r_{ii}$ along the diagonal:

$$\mathbf{Q} = \mathrm{diag}(q_{11}, q_{22}, q_{33}), \quad \mathbf{R} = \mathrm{diag}(r_{11}, r_{22}). \tag{26}$$

In the navigation problem that we consider, it is more important that the robots reach their goal positions than their goal orientations. Therefore, $q_{11}$ and $q_{22}$ can be defined as much higher weights (e.g., 10 to 50 times larger) than $q_{33}$. For the same reason, we can define $r_{11}$, the weight corresponding to a robot's linear velocity, as a much higher value (e.g., 10 times larger) than $r_{22}$, the weight on its angular velocity, which only affects its orientation. We note that the relative values of the entries of the weighting matrices affect the trajectories of the robots.

### 5.2. Resolving Deadlocks

In multi-robot navigation scenarios, a deadlock occurs when a robot stops moving before it reaches its goal position, as a consequence of becoming trapped in an equilibrium point of the system dynamics [63]. Deadlocks inevitably occur in multi-robot systems that use decentralized control approaches, since the robots can only respond to local information within their sensing or communication range [64]. Three types of deadlocks that can occur during multi-robot navigation are described and illustrated in [19]. These types of deadlocks are re-illustrated in Fig. 23. Given the scenarios defined in Section 4, deadlock type 1 can occur in *Scenarios 1* and *3*; deadlock type 2 can occur in *Scenario 3*; and deadlock type 3 can occur in *Scenarios 2* and *3*. Our centralized MPC method can predict all three types of deadlocks, since it models the future states of all robots at each time step over the prediction horizon $N_P$. Given an appropriate selection of $N_P$ and a sufficiently small sampling time $T_s$, as discussed earlier in this section, our method produces optimal control solutions for deadlock-free navigation.

### 5.3. Limitations of the Proposed MPC Method

Although our MPC method can guarantee deadlock-free, collision-free navigation by multiple nonholonomic WMRs, it has certain limitations that should be considered in the selection of its parameters and its implementation. First, the scalability of the method to large numbers of robots is limited by the available computational resources, since the method uses a centralized approach to computing robot control inputs. Second, as demonstrated in [58] with numerical simulations, the quadratic loss function in the optimization problem fails to satisfy the bounds specified in Assumption 3.4 when the prediction horizon $N_P$ is very large. This problem can be addressed by careful tuning of $N_P$ and the weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ for each navigation scenario. Third, errors in the robots' measured poses can arise from sensor noise in the robots' odometry readings, wheel skidding and sliding, and external mechanical disturbances such as unknown friction forces. In our experiments, as mentioned in Section 3.3, the robots fuse their wheel encoder readings and IMU sensor data to improve the accuracy of the odometry, although this sensor fusion did not completely eliminate errors in the odometry measurements. Techniques such as direct decentralized integration [65] can improve real-time estimation of the robots' positions and orientations. In addition, a robust controller such as an $H_\infty$ controller and an observer could be incorporated into our method to attenuate the effects of measurement noise and disturbances on the robots' trajectories.

## 6. Conclusion

In this paper, we propose a nonlinear MPC method for collision-free and deadlock-free navigation by multiple nonholonomic WMRs. This method incorporates an optimization problem for computing velocity control inputs that drive the robots to goal poses while avoiding collisions. We analyze the feasibility of the optimization problem and prove the stability of the resulting controller. To reduce the computational complexity of the MPC, we do not include any stabilizing terminal constraints or costs in the optimization problem. We implement the MPC using open-source software tools that provide a symbolic programming framework, which significantly accelerates the solution of the optimization problem. We demonstrate the effectiveness of our proposed method in both realistic 3D simulations and physical experiments for six different multi-robot navigation scenarios.

Future work includes the modification of this method to create a decentralized MPC scheme for multi-robot navigation, in which robots use their own distance measurements (e.g., from a LiDAR sensor) to detect nearby robots and unknown obstacles in the environment and autonomously adjust their velocities to avoid potential collisions and deadlocks. As another direction of future work, an LMI-based NMPC method could be designed with linear models of the robots' motion, which would reduce the computational complexity of our approach and thus enable controller synthesis for larger numbers of robots without a corresponding significant increase in computational effort.

## Acknowledgments

We are thankful to Dr. Mohamed W. Mehrez for his informative presentation on MPC and MHE implementation in MATLAB using CasADi [66], and to Mr. Mojtaba Zarei for his advice on the design of the MPC method.

## References

[1] R. W. Brockett, et al., Asymptotic stability and feedback stabilization, Differential geometric control theory 27 (1) (1983) 181–191.

[2] D. Gu, H. Hu, A stabilizing receding horizon regulator for nonholonomic mobile robots, IEEE Transactions on Robotics 21 (5) (2005) 1022–1028.

[3] W. E. Dixon, D. M. Dawson, F. Zhang, E. Zergeroglu, Global exponential tracking control of a mobile robot system via a PE condition, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 30 (1) (2000) 129–142.

[4] Y. A. Kapitanyuk, A. V. Proskurnikov, M. Cao, A guiding vector-field algorithm for path-following control of nonholonomic mobile robots, IEEE Transactions on Control Systems Technology 26 (4) (2017) 1372–1385.

[5] M. Saska, V. Spurnỳ, V. Vonásek, Predictive control and stabilization of nonholonomic formations with integrated spline-path planning, Robotics and Autonomous Systems 75 (2016) 379–397.

[6] M. Zarei, N. Kashi, A. Kalhor, M. T. Masouleh, Experimental study on shared-control of a mobile robot via a haptic device with an optimal velocity obstacle based receding horizon control approach, Journal of Intelligent & Robotic Systems 97 (2) (2020) 357–372.

[7] J. B. Rawlings, E. S. Meadows, K. R. Muske, Nonlinear model predictive control: A tutorial and survey, IFAC Proceedings Volumes 27 (2) (1994) 185–197.

[8] E. F. Camacho, C. B. Alba, Model predictive control, Springer Science & Business Media, 2013.

[9] F. Borrelli, A. Bemporad, M. Morari, Predictive control for linear and hybrid systems, Cambridge University Press, 2017.

[10] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, S. Kim, Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 1–9.

[11] U. Rosolia, A. D. Ames, Multi-rate control design leveraging control barrier functions and model predictive control policies, arXiv preprint arXiv:2004.01761 (2020).

[12] S. a. Keerthi, E. G. Gilbert, Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving-horizon approximations, Journal of optimization theory and applications 57 (2) (1988) 265–293.

[13] M. R. Azizi, J. Keighobadi, Point stabilization of nonholonomic spherical mobile robot using nonlinear model predictive control, Robotics and Autonomous Systems 98 (2017) 347–359.

[14] C. Huang, X. Chen, Y. Zhang, S. Qin, Y. Zeng, X. Li, Hierarchical model predictive control for multi-robot navigation, in: International Joint Conferences on Artificial Intelligence (IJCAI), AAAI Press., Conference Proceedings, 2016, p. 3140–3146.

[15] Y. Kuwata, A. Richards, T. Schouwenaars, J. P. How, Distributed robust receding horizon control for multivehicle guidance, IEEE Transactions on Control Systems Technology 15 (4) (2007) 627–641.

[16] A. Richards, J. P. How, Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility, in: Proceedings of the 2003 American Control Conference, 2003., Vol. 5, IEEE, 2003, pp. 4034–4040.

[17] R. Mao, H. Gao, L. Guo, A novel collision-free navigation approach for multiple nonholonomic robots based on ORCA and linear MPC, Mathematical Problems in Engineering 2020 (2020).

[18] R. Tallamraju, S. Rajappa, M. J. Black, K. Karlapalem, A. Ahmad, Decentralized MPC based obstacle avoidance for multi-robot target tracking scenarios, in: 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), IEEE, 2018, pp. 1–8.

[19] L. Wang, A. D. Ames, M. Egerstedt, Safety barrier certificates for collisions-free multirobot systems, IEEE Transactions on Robotics 33 (3) (2017) 661–674.

[20] Y. Wang, S. Boyd, Fast model predictive control using online optimization, IEEE Transactions on Control Systems Technology 18 (2) (2009) 267–278.

[21] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, S. Boyd, OSQP: An operator splitting solver for quadratic programs, Mathematical Programming Computation (2020).
URL https://doi.org/10.1007/s12532-020-00179-2

[22] C. Wang, X. Liu, X. Yang, F. Hu, A. Jiang, C. Yang, Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy, Applied Sciences 8 (2) (2018) 231.

[23] F. Kuhne, W. F. Lages, J. G. da Silva Jr, Model predictive control of a mobile robot using linearization, in: Proceedings of Mechatronics and Robotics, 2004, pp. 525–530.

[24] W. F. Lages, J. A. V. Alves, Real-time control of a mobile robot using linearized model predictive control, IFAC Proceedings Volumes 39 (16) (2006) 968–973.

[25] S. G. Johnson, The NLopt nonlinear-optimization package (2020).
URL http://github.com/stevengj/nlopt

[26] J. Hedengren, J. Mojica, W. Cole, T. Edgar, APOPT: MINLP solver for differential and algebraic systems with benchmark testing, in: Proceedings of the INFORMS National Meeting, Phoenix, AZ, USA, Vol. 1417, 2012, p. 47.

[27] J. Andersson, J. Åkesson, M. Diehl, CasADi: A symbolic package for automatic differentiation and optimal control, in: Recent Advances in Algorithmic Differentiation, Springer, 2012, pp. 297–307.

[28] H. G. Bock, K.-J. Plitt, A multiple shooting algorithm for direct solution of optimal control problems, IFAC Proceedings Volumes 17 (2) (1984) 1603–1608.

[29] L. Stella, A. Themelis, P. Sopasakis, P. Patrinos, A simple and efficient algorithm for nonlinear model predictive control, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE, 2017, pp. 1939–1944.

[30] A. S. Hussein, C. M. Elias, E. I. Morgan, A realistic model predictive control using single and multiple shooting in the formulation of nonlinear programming model, in: 2019 IEEE International Conference of Vehicular Electronics and Safety (ICVES), IEEE, 2019, pp. 1–6.

[31] J. Tamimi, P. Li, Nonlinear model predictive control using multiple shooting combined with collocation on finite elements, IFAC Proceedings Volumes 42 (11) (2009) 703–708.

[32] M. W. Mehrez, Optimization based solutions for control and state estimation in nonholonomic mobile robots: Stability distributed control and localization, Ph.D. thesis, Memorial University of Newfoundland (2017).

[33] S. Tang, J. Thomas, V. Kumar, Hold or take optimal plan (HOOP): A quadratic programming approach to multi-robot trajectory generation, The International Journal of Robotics Research 37 (9) (2018) 1062–1084.

[34] H. Ebel, E. S. Ardakani, P. Eberhard, Distributed model predictive formation control with discretization-free path planning for transporting a load, Robotics and Autonomous Systems 96 (2017) 211–223.

[35] M. Farina, A. Perizzato, R. Scattolini, Application of distributed predictive control to motion and coordination problems for unicycle autonomous robots, Robotics and Autonomous Systems 72 (2015) 248–260.

[36] R. Van Parys, G. Pipeleers, Distributed MPC for multi-vehicle systems moving in formation, Robotics and Autonomous Systems 97 (2017) 144–152.

[37] J. M. Mendes Filho, E. Lucet, D. Filliat, Real-time distributed receding horizon motion planning and control for mobile multi-robot dynamic systems, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 657–663.

[38] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, H. Kress-Gazit, Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles, Autonomous Robots 42 (4) (2018) 801–824.

[39] J. Snape, J. Van Den Berg, S. J. Guy, D. Manocha, Smooth and

collision-free navigation for multiple robots under differential-drive constraints, in: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2010, pp. 4584–4589.

[40] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, M. Egerstedt, The robotarium: A remotely accessible swarm robotics research testbed, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 1699–1706.

[41] J. Alonso-Mora, S. Baker, D. Rus, Multi-robot navigation in formation via sequential convex programming, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2015, pp. 4634–4641.

[42] Robots: Turtlebot 3, https://robots.ieee.org/robots/turtlebot3/ (2020).

[43] K. M. Lynch, F. C. Park, Modern Robotics, Cambridge University Press, 2017.

[44] J. Keighobadi, M. S. Sadeghi, K. A. Fazeli, Dynamic sliding mode controller for trajectory tracking of nonholonomic mobile robots, IFAC Proceedings Volumes 44 (1) (2011) 962–967.

[45] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. Scokaert, Constrained model predictive control: Stability and optimality, Automatica 36 (6) (2000) 789–814.

[46] R. Gondhalekar, J.-i. Imura, K. Kashima, Controlled invariant feasibility—a general approach to enforcing strong feasibility in MPC applied to move-blocking, Automatica 45 (12) (2009) 2869–2875.

[47] L. Grüne, J. Pannek, Nonlinear model predictive control, in: Nonlinear Model Predictive Control, Springer, 2017, pp. 45–69.

[48] T. Faulwasser, R. Findeisen, Nonlinear model predictive control for constrained output path following, IEEE Transactions on Automatic Control 61 (4) (2015) 1026–1039.

[49] T. Faulwasser, Optimization-based solutions to constrained trajectory-tracking and path-following problems, Ph.D. thesis, Otto von Guericke University Magdeburg (2012).

[50] A. Alessandretti, A. P. Aguiar, C. N. Jones, Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control, in: 2013 European Control Conference (ECC), IEEE, 2013, pp. 1371–1376.

[51] D. Lam, C. Manzie, M. C. Good, Multi-axis model predictive contouring control, International Journal of Control 86 (8) (2013) 1410–1424.

[52] S. Yu, X. Li, H. Chen, F. Allgöwer, Nonlinear model predictive control for path following problems, International Journal of Robust and Nonlinear Control 25 (8) (2015) 1168–1182.

[53] K. Worthmann, M. W. Mehrez, M. Zanon, G. K. Mann, R. G. Gosine, M. Diehl, Regulation of differential drive robots using continuous time MPC without stabilizing constraints or costs, IFAC-PapersOnLine 48 (23) (2015) 129–135.

[54] A. Boccia, L. Grüne, K. Worthmann, Stability and feasibility of state constrained MPC without stabilizing terminal constraints, Systems & Control Letters 72 (2014) 14–21.

[55] M. W. Mehrez, K. Worthmann, J. P. Cenerini, M. Osman, W. W. Melek, S. Jeon, Model predictive control without terminal constraints or costs for holonomic mobile robots, Robotics and Autonomous Systems 127 (2020) 103468.

[56] G. Grimm, M. J. Messina, S. E. Tuna, A. R. Teel, Model predictive control: for want of a local control Lyapunov function, all is not lost, IEEE Transactions on Automatic Control 50 (5) (2005) 546–558.

[57] J. B. Rawlings, D. Q. Mayne, Model predictive control: Theory and design, Nob Hill Pub., 2009.

[58] K. Worthmann, M. W. Mehrez, M. Zanon, G. K. Mann, R. G. Gosine, M. Diehl, Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs, IEEE Transactions on Control Systems Technology 24 (4) (2015) 1394–1406.

[59] K. Worthmann, Stability analysis of unconstrained receding horizon control schemes, Ph.D. thesis, University of Bayreuth (2011).

[60] A. Wächter, Short tutorial: getting started with IPOPT in 90 minutes, in: Dagstuhl Seminar Proceedings, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[61] Autonomous Collective Systems Laboratory YouTube channel, Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots (simulations), https://www.youtube.com/watch?v=8fVpX0D_OH4 (2020).

[62] Autonomous Collective Systems Laboratory YouTube channel, Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots (experiments), https://www.youtube.com/watch?v=eHKpTs5h3wY (2020).

[63] Y. Zhou, H. Hu, Y. Liu, Z. Ding, Collision and deadlock avoidance in multirobot systems: A distributed approach, IEEE Transactions on Systems, Man, and Cybernetics: Systems 47 (7) (2017) 1712–1726.

[64] J. Grover, C. Liu, K. Sycara, Deadlock analysis and resolution in multi-robot systems: The two robot case, arXiv preprint arXiv:1911.09146 (2019).

[65] H. Nourmohammadi, J. Keighobadi, Decentralized INS/GNSS system with MEMS-grade inertial sensors using QR-factorized CKF, IEEE Sensors Journal 17 (11) (2017) 3278–3287.

[66] Mohamed W. Mehrez, MPC and MHE implementation in Matlab using CasADi, https://www.youtube.com/watch?v=RrnkPrcpyEA (2019).