sensors

Article

Modeling of Cooperative Robotic Systems and Predictive Control Applied to Biped Robots and UAV-UGV Docking with Task Prioritization

Baris Taner † **and Kamesh Subbarao** *,†

Department of Mechanical and Aerospace Engineering, The University of Texas at Arlington, 500 W. First St., Arlington, TX 76019, USA
* Correspondence: subbarao@uta.edu; Tel.: +1-817-272-7467
† These authors contributed equally to this work.

Abstract: This paper studies a cooperative modeling framework to reduce the complexity in deriving the governing dynamical equations of complex systems composed of multiple bodies such as biped robots and unmanned aerial and ground vehicles. The approach also allows for an optimization-based trajectory generation for the complex system. This work also studies a fast–slow model predictive control strategy with task prioritization to perform docking maneuvers on cooperative systems. The method allows agents and a single agent to perform a docking maneuver. In addition, agents give different priorities to a specific subset of shared states. In this way, overall degrees of freedom to achieve the docking task are distributed among various subsets of the task space. The fast–slow model predictive control strategy uses non-linear and linear model predictive control formulations such that docking is handled as a non-linear problem until agents are close enough, where direct transcription is calculated using the Euler discretization method. During this phase, the trajectory generated is tracked with a linear model predictive controller and addresses the close proximity motion to complete docking. The trajectory generation and modeling is demonstrated on a biped robot, and the proposed MPC framework is illustrated in a case study, where a quadcopter docks on a non-holonomic rover using a leader–follower topology.

Keywords: cooperation; model predictive control; rover; quadcopter; docking

Citation: Taner, B.; Subbarao, K. Modeling of Cooperative Robotic Systems and Predictive Control Applied to Biped Robots and UAV-UGV Docking with Task Prioritization. *Sensors* **2024**, *24*, 3189. https://doi.org/10.3390/s24103189

Academic Editor: Andrey V. Savkin

Received: 15 March 2024
Revised: 13 May 2024
Accepted: 15 May 2024
Published: 17 May 2024

1. Introduction

Planning a trajectory for complex robots such as a biped is a complex task as the free-floating base of the robot is moved by the discontinuous contact forces acting on their feet. This propulsion method requires attention in planning the motion of the contact points and the contact forces while considering the dynamic effects on the robot [1]. In addition, a straightforward optimal control formulation results in intractable non-linear programs, as stated in the literature [2,3].

There are multiple causes of complexity in planning the trajectory of a floating base robot. First of all, the pose of the floating base is described as unactuated base coordinates with six degrees of freedom (DoF). Then, actuated joint coordinates of legs are added on top of that, which results in relatively large joint coordinates [1]. Therefore, the trajectory optimization of such a system should find optimal values for all these coordinates. Secondly, contact between the feet and the terrain must comply certain contact conditions such as unilateral contact forces [4]. In the literature, this problem is handled as a trajectory optimization problem using numerical optimization techniques such as direct optimal control methods [5], indirect optimal control methods [6], dynamic programming [7], and sequential methods [8]. In this way, trajectory of the robot, along with calculated joint coordinates, torques, and contact forces, are calculated by considering the upper-level constraints on the task.

The complexity of the problem is reduced in multiple ways. One way to carry this out is to use low-fidelity dynamic models for the robot such as single rigid body dynamics model, where lumped inertia is attached to the body frame and actuated links are assumed to be moving slowly and having low inertia [2]. Another way to reduce complexity is to split the the optimization into smaller sub-problems and to pre-define some portions of the trajectory such as footholds [9].

However, simplifications in robot model and optimization problems are compromised with the complexity of the motion that can be calculated with trajectory optimization. For this reason, having means to simplify full-body dynamics without compromising the fidelity of the model is vital. Recently, the trajectory optimization problem was distributed into smaller alternating sub-problems, where one first satisfies dynamic constraints of the problem by finding robot momentum and contact forces, and then one finds the leg kinematics that satisfy robot dynamics. This is denoted as centroidal and whole-body model splitting and was first introduced in the work [8], where a sequential optimal control formulation was represented. In another work based on the same splitting of whole-body motion and centroidal dynamics, the locomotion problem is cast into a mathematical framework based on Alternating Direction Method of Multipliers (ADMM) [10]. This paper exploits the natural splitting between centroidal and manipulator dynamics and ensures consensus between these models. Another recent example uses the same splitting and introduces an accelerated ADMM algorithm to solve the locomotion problem. [3].

Besides the centroidal and whole body splitting of the dynamic model of a legged robot, there is obviously another split between the portions of the body. This splitting is naturally defined by the design of legged robots and referred to as body and limbs. Depending on the design, they can contain a single body or a set of multi-bodies and interconnections between these sets maintained over a connection node, which is a joint. A formal way to describe the cooperation between these sets already exists in the literature within a cooperative system framework [11]. This framework defines a relationship between independent agents using communication graphs, which dictate a mapping to the information flow among agents. Cooperative system frameworks are used in many areas such as increasing performance of a sensory network in localization of data [12] or calculating the communication topology between dynamic agents within the cooperative system [13]. It is also used in designing controllers for agents within a network [14]. All the applications have one thing in common, which is the distributed modeling of the cooperative system and the distributed calculation of the variables to reach the desired objective.

Autonomous aerial systems have become essential in numerous practical fields, such as in public safety, as a surveillance tool and first response [15–17]. It has also been studied for the delivery of goods by the industry [18,19]. A critical task during the operation of an aerial system is docking maneuver, where it might be approaching a stationary or a moving platform [20,21] to drop/load cargo. In addition, aerial vehicles require refueling or recharging to extend their workspace [22]. Docking is an intricate maneuver that necessitates the awareness of the docking path's constraints regarding flight safety and the docked platform's attitude [23]. In addition, nonlinear effects such as the wake of the leading agent due to the proximity flight must be considered [24] as well as the ground effect [25]. Thus it is clear that the characteristic of this maneuver requires the agent to handle certain constraints and uncertainties.

A popular control method to handle previously described constraints and calculate control actions is model-predictive control (MPC). An MPC reaches the desired control action by minimizing a given objective using linear and non-linear optimization theory, as applied in the case of a tracking controller [26]. MPC also allows the integration of secondary tasks into decision-making, as in [27], which makes it a unified strategy to handle docking maneuvers without requiring ad-hoc integration of multiple frameworks that increase complexity. Implementing MPC-based docking strategies for space applications exists in case studies, where line of sight constraints are satisfied, energy-saving strategies are pursued, and docking on tumbling objects are executed [28–30].

Autonomous docking for aerial vehicles requires state information of the docked platform, which receives the docking agent. Therefore it certainly requires an on-board or external mechanism to sense and estimate states. On-board sensors such as cameras or LiDAR are widely implemented solutions (see [31]), yet they are bounded with range limitations. External mechanisms to obtain state information about the docked platform are sensors placed on the agents, except for the docking agent or external observers. As a result, this information is shared over a communication grid as illustrated in [32]. Besides state information, some of the autonomous docking literature can be grouped in terms of cooperation at the controller level. In one of the works, a central controller calculates lateral and longitudinal velocity commands for both of the docking maneuver devices: an uncrewed ground vehicle (UGV) and an uncrewed aerial vehicle (UAV). Then, these control signals are fed into these agents [33]. In the extension of this work, an MPC strategy is utilized on the same system, where both agents are cooperatively trying to execute docking maneuvers [34]. In another work, a multirotor docks onto a fixed-wing platform, where only the multirotor executes the docking maneuver [21]. The aerial refueling problem is addressed as a docking problem in [22]. However, only the docking platform and the boom are manipulated. Recent work studies an uncrewed sea surface vehicle (USV) landing and designs an MPC controller for a multirotor, where cooperative docking is executed [35]. However, the docking trajectory calculation is calculated on the multirotor and shared back to the USV.

Motivation

In this context, this paper aims to keep the model fidelity of the biped robot higher than a single body dynamics and adopt a cooperative system framework in defining dynamics of the robot. As a result, the robot will be defined by multiple cooperating agents, where cooperation is introduced to the trajectory optimization by an adjacency matrix as a constraint. In this way, lumped trajectory optimization is converted to a separated objectives and constraints that are linearly related.

When multiple agents are communicating with each other and obeying specific rules, i.e., collision avoidance, velocity matching, and staying within the vicinity of the neighbors, the aggregate of these agents are called cooperative agents [36] and the application of these systems are vast due to the advantages. Popular applications include uncrewed vehicles [37] and space applications [38]. The nature of autonomous docking makes the system performing this maneuver a cooperative system. However, it is not formally addressed as such in the vast majority of the literature. This is due to the fact that in most scenarios, one of the vehicles is assumed to be tightly controlled and hence passive. The control laws are then derived for the remaining active vehicle. On the other hand, controllers cooperatively addressing this problem are mainly centralized on a ground controller or in one of the agents, which could suffer in performance or even fail due to uncertainties in the communication. Apart from the centralized implementation of cooperative docking, there are a few applications where both agents take part in the docking maneuver. Since the cooperative control framework allows various communication topologies, methods based on MPC with local neighbor state information can be applied. Besides the cooperative aspect, a prioritization of the states to track during docking is typically not studied in the literature. As summarized before, agents that are too far apart from each other first could first minimize the positional difference, which is carried out to be in a feasible solution set when the MPC for docking is initiated. Instead of handling this problem as two separate sub-problems, an automated prioritization of tracking certain states can be defined so that linear states can be given higher priority over the angular states (see [39]).

Contributions

The key contributions of this paper are as follows. The methods introduced in this paper are used to divide the lumped multi-body model of a large robot-system into cooperative multi-bodies and address the trajectory optimization problem using this distributed model. In the case of the biped robot, although it has light weight legs compared to the shoulder and the floating base, the biped robot will be modeled as three cooperative agents, which are the floating base, right shoulder, and left shoulder. All agents are defined as multi-bodies, as defined in the following sections. The contributions of this paper are summarized as follows:

- This method divides the EoM of the biped robot into smaller cooperative agents, which has simpler EoMs. Agents with simpler dynamics result in simpler equality constraints for the trajectory optimization.
- The non-linear programming formulation given in this paper cast the trajectory optimization problem with single objective and single augmented Hamiltonian into split objectives and constraints.
- This paper also proposes a cooperative control strategy based on MPC for docking. The designed strategy implements a non-linear and a linear MPC for the coarse approach (long distance) and the delicate docking maneuver (short distance) based on the same objective function with tailored optimization strategies. A leader–follower type of topology is adopted, where the quadcopter docks on the UGV. As a showcase, this controller performs short- and long-distance docking of a quadcopter on a UGV.
- Formulation of the MPC includes task prioritization, which is based on a null-space projection of the tasks being ranked. The formulation is adopted from [40] by defining the docking task in terms of the docking agents' Degrees of Freedom (DoF).

The organization of the paper is such that Section 2 expresses governing equations of agents and the underlying graph structure. Following that, Section 8 provides a task-prioritization and MPC strategy for docking. The simulation results are shared in Section 9. Finally, conclusions are provided in Section 11.

2. Notation and Preliminaries

The notation of this work is as follows. $s \in \mathbb{R}$, $\boldsymbol{v} \in \mathbb{R}^{n_v}$ and $\boldsymbol{M} \in \mathbb{R}^{n_r \times n_c}$ represent the arbitrary scalar, vector, and matrix. $\mathcal{F}_I$ and $\mathcal{F}_B$ represent the inertial and body frames, respectively, where the expression of a vector for these frames is written as ${}_i\boldsymbol{v}$, $i = I, B$. Unit vectors in orthonormal frames are denoted as $\boldsymbol{u}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$, $\boldsymbol{u}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$ and $\boldsymbol{u}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. The composite rotation matrix ${}_I\boldsymbol{C}_B \in SO(3)$ is written from $\mathcal{F}_B$ to $\mathcal{F}_I$, the rotation sequence is depicted as $z - y - x$, and associated Euler angles are denoted as ψ, θ and ϕ. $g = 9.81\ \mathrm{m/s^2}$ is the gravitational acceleration. $col\{\cdot\}_{m=1}^{M}$, $row\{\cdot\}_{l=1}^{M}$, and $diag\{\cdot\}_{m=1}^{M}$ represent column, row and diagonal concatenation, respectively, of the entity within the parenthesis.

3. Underlying Graph Structure

The communication among agents in the cooperative systems are described by the graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, which consists of node set $\mathcal{N}$ and edge set $\mathcal{E}$ [41]. Edge set $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ is given between nodes $i \in \mathcal{N}$ and $j \in \mathcal{N}$ such that $(j, i) \in \mathcal{E}$ denotes node i receives information from j. Let n_w be an arbitrary signal dimension, and then the adjacency matrix; shared signal sizes among agents are assumed to be identical and equal to n_w. $\boldsymbol{A} = [a_{ij}] \otimes \boldsymbol{I}_{n_w} \in \mathbb{R}^{N \cdot n_w \times N \cdot n_w}$ of $\mathcal{G}$ is composed of weighting scalars a_{ij}, where a_{ij} quantifies the strength of the connection from node j to node i. N is the number of agents in the cooperative system (CS). Formally, A_{ij} is described by the following equation.

$$\boldsymbol{A}_{ij} = \begin{cases} a_{ij} > 0, & j \neq i, \ \ (j, i) \in \mathcal{E} \\ a_{ij} = 0, & otherwise \end{cases} \tag{1}$$

4. ASLB—A Bipedal Robot for Dynamics Locomotion

4.1. ASLB System Composition

ASLB is a floating bipedal platform in which each leg is composed of hybrid structure with three degrees of freedom (DoFs). Specifically, starting from the body, the kinematic structure of legs are a revolutionjoint (at shoulder) followed by a parallel 5R mechanism. This kinematics result in three actuated and two passive joint coordinates in each leg. A rendered 3D model and manufactured prototype of ASLB is provided in Figure 1.

Table 1 provides a summary of the key variables and terms used for the development of the ASLB dynamic model.

Table 1. Summary of key variables used for the description of the ASLB kinematics and dynamics.

$\mathbf{q}_B$	set of unactuated joints
$\mathbf{q}_{t,B}$	translational joints
$\mathbf{q}_{r,B}$	rotational joints
$\mathbf{r}_B()$	translation matrix from one frame to another
$\mathbf{C}_{IB}()$	rotation matrix from frame I to frame B
$\boldsymbol{\theta}_i$	vector of joints for each leg
$\boldsymbol{\theta}_{a,i}$	vector of active joints for leg, i
$\boldsymbol{\theta}_{p,i}$	vector of passive joints for leg, i
$\mathbf{r}_{v,i}$	position of point v
$\mathbf{R}_*$	elementary rotation matrix for $*$ axis
$\mathbf{v}_{r,i}$	velocity relationship
$\mathbf{J}_{r,i}$	Jacobian relating velocity to active and passive joints
$a_{.,.}$	lengths between joints
$\mathbf{M}()$	generalized mass matrix
$\mathbf{C}(.,.)$	Coriolis and centrifugal terms
$\mathbf{G}()$	gravitational terms
$\mathbf{S}$	control selection matrix for actuated joints of respective legs
$\boldsymbol{\tau}$	actuated joint torques
$\boldsymbol{F}_{C,i}$	external force on tip of i^{th} leg
$\boldsymbol{J}_{C,i}$	geometric Jacobian of tip point for i^{th} leg
ω_0	angular speed for the linear inverted pendulum model (LIPM)
${}^{p}\bar{\mathbf{x}}_{c,K}$	contact state $\bar{x}_c$ at instant K in phase p

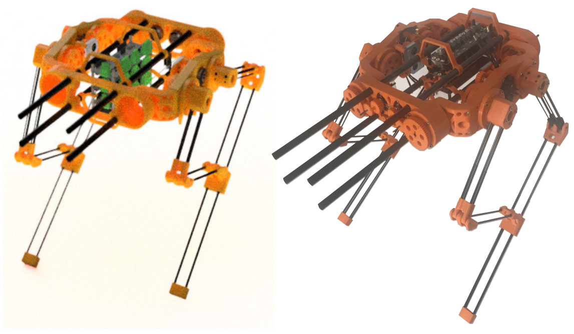


Figure 1. Three-dimensional model (**left**) and manufactured prototype of ASLB (**right**).

4.2. ASLB Kinematics

Kinematic model of the floating platform starts with a set of unactuated joints that gives six DoFs to the mobile platform. These joints are collected in a column matrix $\boldsymbol{q}_B \in \mathbb{R}^3 \times SO(3)$. Starting from the inertial frame, joints are located in a sequence such that first translational joints $\boldsymbol{q}_{t,B} = [q_1, q_2, q_3]^T \in \mathbb{R}^3$ are located in respective x, y and z directions. Assuming the euler sequence of YZX, rotational joints are $\boldsymbol{q}_{r,B} = [q_4, q_5, q_6]^T \in \mathbb{R}^3$. Body frame $\mathcal{F}_B$ is kinematically represented with respect to $\mathcal{F}_I$ by $\boldsymbol{q}_B = [\boldsymbol{q}_{t,B}^T, \boldsymbol{q}_{r,B}^T]^T$ such that $\boldsymbol{r}_B(\boldsymbol{q}_{t,B})$ and $\boldsymbol{C}_{IB}(\boldsymbol{q}_{r,B})$ are translation and rotation matrices from $\mathcal{F}_I$ to $\mathcal{F}_B$, respectively. There are two legs attached to the body, and respective joints are represented by $\boldsymbol{\theta}_i \in \mathbb{R}^{n_i}$, where $n_i = 5$ and $i = R, L$. In total, the combined DoF for legs are given as $n_a = 10$.

The composition of $\boldsymbol{\theta}_i$ for each leg is given as $\boldsymbol{\theta}_i = [\theta_{0,i}\ \theta_{1,i}\ \theta_{2,i}\ \theta_{3,i}\ \theta_{4,i}]^T$, where there are three active and two passive joints, respectively $\boldsymbol{\theta}_{a,i} = [\theta_{0,i}\ \theta_{1,i}\ \theta_{3,i}]^T \in \mathbb{R}^3$ and $\boldsymbol{\theta}_{p,i}(\boldsymbol{\theta}_{a,i}) = [\theta_{2,i}(\boldsymbol{\theta}_{a,i})\ \theta_{4,i}(\boldsymbol{\theta}_{a,i})]^T$. Passive joints can be written as a function of $\boldsymbol{\theta}_{a,i}$ based on a velocity constraint as described in Section 4.2.1. As a result, total joint space of ASLB is given by $\boldsymbol{q} = [\boldsymbol{q}_B,\ \boldsymbol{\theta}_{a,R},\ \boldsymbol{\theta}_{a,L}]^T \in \mathbb{R}^{12}$. The kinematic structure of the right leg is illustrated in Figure 2.

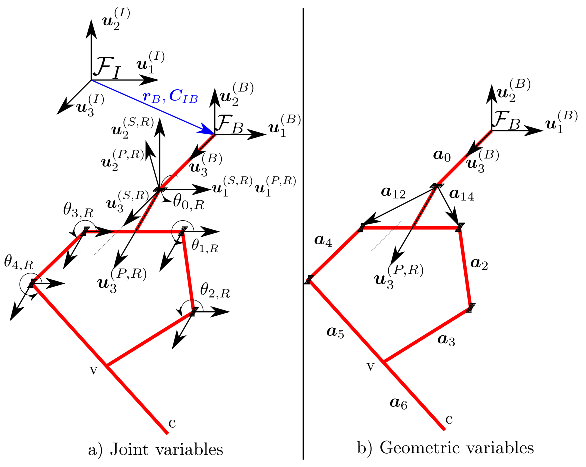


Figure 2. Kinematic structure of ASLB.

4.2.1. Passive—Active Joint Relation

As described in [42] starting from the origin of the body frame $\mathcal{F}_B$, there are two chains to reach point $\boldsymbol{v}$ on both legs. Let $\boldsymbol{R}_*$ represent elementary rotation matrix, where $* = x, y, z$ are active axes. Then these two chains can be written as in Equation (2).

$$\begin{aligned} \boldsymbol{r}_{v,1} &= \boldsymbol{a}_{12} + \boldsymbol{R}_z(\theta_{1,R})\boldsymbol{a}_2 + \boldsymbol{R}_z(\theta_{1,R} + \theta_{2,R})\boldsymbol{a}_2 \\ \boldsymbol{r}_{v,2} &= \boldsymbol{a}_{14} + \boldsymbol{R}_z(\theta_{3,R})\boldsymbol{a}_2 + \boldsymbol{R}_z(\theta_{3,R} + \theta_{4,R})\boldsymbol{a}_2 \end{aligned} \tag{2}$$

Differentiating Equation (2) results in velocity equations of $\boldsymbol{v}_{r,1}$ and $\boldsymbol{v}_{r,2}$, which are equal. Writing this relationship as below relates the passive joints to active joints as described in Equation (3). Let $\overline{\boldsymbol{\theta}}_{a,i} = [\theta_{1,i}\ \theta_{3,i}]^T$ denote the set of active joints related to the closed loop, then $\boldsymbol{J}_{a,i} \in \mathbb{R}^2$ becomes a square matrix.

$$\begin{aligned} \boldsymbol{v}_{r,1} &= \boldsymbol{J}_{r,1}\begin{bmatrix} \overline{\boldsymbol{\theta}}_{a,i}^T & \boldsymbol{\theta}_{p,i}^T \end{bmatrix}^T \\ \boldsymbol{v}_{r,2} &= \boldsymbol{J}_{r,2}\begin{bmatrix} \overline{\boldsymbol{\theta}}_{a,i}^T & \boldsymbol{\theta}_{p,i}^T \end{bmatrix}^T \\ \boldsymbol{0} &= (\boldsymbol{J}_{r,1} - \boldsymbol{J}_{r,2})\begin{bmatrix} \overline{\boldsymbol{\theta}}_{a,i}^T & \boldsymbol{\theta}_{p,i}^T \end{bmatrix}^T \\ &= \begin{bmatrix} \boldsymbol{J}_a \mid \boldsymbol{J}_p \end{bmatrix}\begin{bmatrix} \overline{\boldsymbol{\theta}}_{a,i}^T & \boldsymbol{\theta}_{p,i}^T \end{bmatrix}^T \end{aligned} \tag{3}$$

Finally, passive joints are related to active joints as given in Equation (4).

$$\begin{aligned} \boldsymbol{\theta}_{p,i} &= \boldsymbol{J}_{pa}\overline{\boldsymbol{\theta}}_{a,i} \\ \boldsymbol{J}_{pa} &= -\boldsymbol{J}_{p,i}^{-1}\boldsymbol{J}_{a,i} \end{aligned} \tag{4}$$

4.2.2. Forward Kinematics

Forward kinematics for each leg are used to calculate the position of the contact point c with respect to origin of $\mathcal{F}_B$, as illustrated in Figure 2. The aforementioned position vector is denoted as $\boldsymbol{r}_c$. As mentioned in Section 4.2.1 the active and passive joint angles are related to each other and unless the passive joints are measured by sensors, they have to be calculated from this relationship. To do so, position vectors $\boldsymbol{r}_{v,1}$ and and $\boldsymbol{r}_{v,2}$ are arranged as shown below.

$$\begin{aligned}
&\begin{bmatrix} \cos(\theta_{3,R}+\theta_{4,R}) \\ \sin(\theta_{3,R}+\theta_{4,R}) \\ 0 \end{bmatrix} a_{5,x} = \\
&-\begin{bmatrix} \cos(\theta_{1,R}+\theta_{2,R}) \\ \sin(\theta_{1,R}+\theta_{2,R}) \\ 0 \end{bmatrix} a_{3,x} + \begin{bmatrix} a_{12,x}-a_{14,x} \\ a_{12,y}-a_{14,y} \\ a_{12,z}-a_{14,z} \end{bmatrix} \\
&+\begin{bmatrix} \cos(\theta_{1,R}) \\ \sin(\theta_{1,R}) \\ 0 \end{bmatrix} a_{2,x} - \begin{bmatrix} \cos(\theta_{3,R}) \\ \sin(\theta_{3,R}) \\ 0 \end{bmatrix} a_{4,x}
\end{aligned} \tag{5}$$

The first two rows of Equation (5) can be written in a minimal form as provided in Equation (6) such that terms including $\theta_{3,R}$, $\theta_{4,R}$ are left alone.

$$\begin{aligned}
\begin{bmatrix} \cos(\theta_{3,R}+\theta_{4,R}) \\ \sin(\theta_{3,R}+\theta_{4,R}) \end{bmatrix} &= \begin{bmatrix} T_x \\ T_y \end{bmatrix} + \frac{a_{3,x}}{a_{5,x}} \begin{bmatrix} \cos(\theta_{1,R}+\theta_{2,R}) \\ \sin(\theta_{1,R}+\theta_{2,R}) \end{bmatrix} \\
T_x &= \frac{1}{a_{5,x}}(a_{12,x}-a_{14,x}+\cos(\theta_{1,R})+\cos(\theta_{3,R})) \\
T_y &= \frac{1}{a_{5,x}}(a_{12,y}-a_{14,y}+\sin(\theta_{1,R})+\sin(\theta_{3,R}))
\end{aligned} \tag{6}$$

Elements of Equation (6) are squared and summed to obtain Equation (7). Then trigonometric expressions are written in terms of tangents of the half angles, which leads to the a solution to $\theta_{12,R} = \theta_{1,R} + \theta_{2,R}$ as provided in Equation (8).

$$T_x^2 + T_y^2 + \frac{a_{3,x}}{a_{5,x}}^2 + 2T_x \frac{a_{3,x}}{a_{5,x}} \cos(\theta_{12,R}) + 2B \sin(\theta_{12,R}) = 1 \tag{7}$$

$$\begin{aligned}
\theta_{12,R} &= 2\tan^{-1}(T_{12}) \\
T_{12} &= \frac{-2C_3 \pm \sqrt{(2C_3)^2 - 4(C_1-C_2)(C_1+C_2)}}{2(C_1-C_2)} \\
C_1 &= T_x^2 + T_y^2 + \frac{a_{3,x}}{a_{5,x}}^2 - 1 \\
C_2 &= 2T_x \frac{a_{3,x}}{a_{5,x}} \\
C_3 &= 2T_y \frac{a_{3,x}}{a_{5,x}}
\end{aligned} \tag{8}$$

Using $\theta_{12,R}$, one can write Equation (9) and solve for $\theta_{34,R} = \theta_{3,R} + \theta_{4,R}$. Finally, tip point location $\boldsymbol{r}_c$ is calculated as provided in Equation (10).

$$\begin{aligned}
\theta_{34,R} &= atan2(s_{34,R}, c_{34,R}) \\
\begin{bmatrix} c_{34,R} \\ s_{34,R} \end{bmatrix} &= \begin{bmatrix} T_x + \frac{a_{3,x}}{a_{5,x}} \cos(\theta_{12,R}) \\ T_y + \frac{a_{3,x}}{a_{5,x}} \sin(\theta_{12,R}) \end{bmatrix}
\end{aligned} \tag{9}$$

$$\begin{aligned}
\boldsymbol{r}_c = \; & \boldsymbol{a}_0 + \boldsymbol{R}_x(\theta_{0,R})(\boldsymbol{a}_{14} + \boldsymbol{R}_z(\theta_{3,R})\boldsymbol{a}_4 \\
& + \boldsymbol{R}_z(\theta_{34,R})(\boldsymbol{a}_5 + \boldsymbol{a}_6))
\end{aligned} \tag{10}$$

4.2.3. Inverse Kinematics

Inverse kinematics is illustrated on right leg, and the calculations provided here can be duplicated for left leg. Inverse kinematics solutions in this work rely on the geometric calculation of $\theta_{0,R}$ as provided in Equation (11). The geometric entities are illustrated in Figure 3.

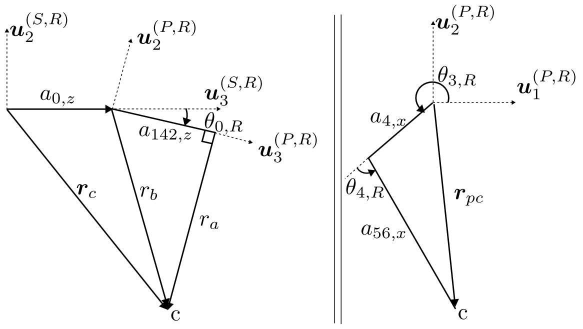


Figure 3. Geometric entities related to calculation of $\theta_{0,R}$.

$$\begin{aligned}
\theta_{0,R} &= atan2(\sin(\theta_{0,R}), \cos(\theta_{0,R})) \\
\begin{bmatrix} \cos(\theta_{0,R}) \\ \sin(\theta_{0,R}) \end{bmatrix} &= \begin{bmatrix} a_{142,z} & -r_a \\ -r_a & -a_{142,z} \end{bmatrix}^{-1} \begin{bmatrix} r_z - a_{0,z} \\ r_y - a_{0,y} \end{bmatrix} \\
\boldsymbol{r}_1 &= \boldsymbol{r} - \boldsymbol{a}_0 \\
r_a &= \sqrt{r_{1,y}^2 + r_{1,z}^2 - a_{14,z}^2} \\
a_{142,z} &= a_{14,z} + a_{2,z}
\end{aligned} \tag{11}$$

To calculate active joints $\theta_{1,R}$ and $\theta_{3,R}$ x and y components of the position vector $\boldsymbol{r}_{pc}$ as given in Equation (12) are expanded in Equation (13).

$$\begin{aligned}
r_{pc,x} &= a_{4,x}\cos(\theta_{3,R}) + a_{56,x}\cos(\theta_{3,R} + \theta_{4,R}) \\
r_{pc,y} &= a_{4,x}\sin(\theta_{3,R}) + a_{56,x}\sin(\theta_{3,R} + \theta_{4,R})
\end{aligned} \tag{12}$$

$$\begin{aligned}
r_{pc,x} &= a_{4,x}\cos(\theta_{3,R}) + a_{56,x}(\cos(\theta_{3,R})\cos(\theta_{4,R}) - \sin(\theta_{3,R})\sin(\theta_{4,R})) \\
r_{pc,y} &= a_{4,x}\sin(\theta_{3,R}) + a_{56,x}(\cos(\theta_{3,R})\sin(\theta_{4,R}) + \sin(\theta_{3,R})\cos(\theta_{4,R}))
\end{aligned} \tag{13}$$

Equation (13) is rewritten as in Equation (14), and each scalar equation can be squared and summed as in Equation (15).

$$\begin{aligned}
\cos(\theta_{3,R}) &= (r_{pc,x}/a_{4,x}) + (a_{56,x}/a_{4,x})\cos(\theta_{3,R} + \theta_{4,R}) \\
&= C_4 + C_5\cos(\theta_{3,R} + \theta_{4,R}) \\
\sin(\theta_{3,R}) &= (r_{pc,y}/a_{4,x}) + (a_{56,x}/a_{4,x})(\sin(\theta_{3,R} + \theta_{4,R}) \\
&= C_6 + C_5\sin(\theta_{3,R} + \theta_{4,R})
\end{aligned} \tag{14}$$

$$\begin{aligned}
1 &= C_4^2 + C_6^2 + C_5^2 + 2C_4C_5\cos(\theta_{3,R} + \theta_{4,R}) + 2C_6C_5\sin(\theta_{3,R} + \theta_{4,R}) \\
0 &= (C_4^2 + C_6^2 + C_5^2 - 1) + (2C_4C_5)\cos(\theta_{3,R} + \theta_{4,R}) + (2C_6C_5)\sin(\theta_{3,R} + \theta_{4,R}) \\
0 &= C_7 + C_8\cos(\theta_{3,R} + \theta_{4,R}) + C_9\sin(\theta_{3,R} + \theta_{4,R})
\end{aligned} \tag{15}$$

Then, using tangents of the half angle, Equation (15) can be further manipulated in Equation (16). The solution to $\theta_{34,R}$ can be calculated by solving the quadratic problem for T_{34} as illustrated in Equation (8).

$$0 = (C_7 - C_8)T_{34}^2 + 2C_9T_{34} + (C_7 + C_8) \tag{16}$$

After finding the solution to $\theta_{34,R}$, these values are inserted into Equation (12) to calculate $\theta_{3,R}$. This completes the solution to one chain of the leg mechanism.

Geometric definitions such as $\boldsymbol{r}_{pcl}$, $\boldsymbol{r}_{cl}$ are provided in Figure 4 to calculate the joint variables on the other chain that contains $\boldsymbol{a}_2$ and $\boldsymbol{a}_3$.

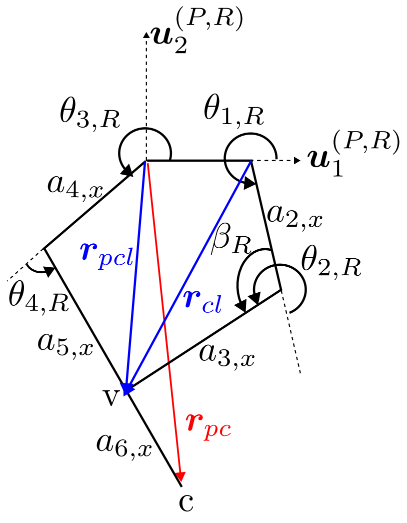


Figure 4. Geometric entities related to calculation of $\theta_{1,R}$.

Using the known joint variables, point v is represented from origin of the joint **3** with a vector denoted as $\boldsymbol{r}_{pcl}$. Alternatively, point v can also be represented from the origin of the joint **1** with a vector that is $\boldsymbol{r}_{cl}$. These vectors are defined on (P, R) plane, where the 5R mechanism lies , and provided in Equations (17) and (18), respectively.

$$\begin{aligned} r_{pcl,x} &= r_{pc,x} - a_{6,x}\cos(\theta_{3,R} + \theta_{4,R}) \\ r_{pcl,y} &= r_{pc,y} - a_{6,x}\sin(\theta_{3,R} + \theta_{4,R}) \end{aligned} \tag{17}$$

$$\begin{aligned} r_{cl,x} &= a_{2,x}\cos(\theta_{1,R}) + a_{3,x}\cos(\theta_{1,R} + \theta_{2,R}) \\ r_{cl,y} &= a_{2,x}\sin(\theta_{1,R}) + a_{3,x}\sin(\theta_{1,R} + \theta_{2,R}) \end{aligned} \tag{18}$$

Scalar equations $r_{cl,x}$ and $r_{cl,y}$ of Equation (18) are squared, summed, and reorganized as provided in Equation (19) to calculate β_R, which leads to $\theta_{2,R}$.

$$\begin{aligned} \cos(\beta_R) &= -\left(\frac{r_{cl,x}^2 + r_{cl,y}^2 - a_{2,x}^2 - a_{3,x}^2}{2a_{2,x}a_{3,x}}\right) \\ \beta_R &= acos\left(-\frac{r_{cl,x}^2 + r_{cl,y}^2 - a_{2,x}^2 - a_{3,x}^2}{2a_{2,x}a_{3,x}}\right) \\ \theta_{2,R} &= \pi + \beta_R \end{aligned} \tag{19}$$

To calculate $\theta_{1,R}$, $r_{cl,x}$ and $r_{cl,y}$ are represented as a function of r_{pcl}. This is given in Equation (20) in matrix form, where terms with $\theta_{1,R}$ are the unknown variables. The solution to Equation (20) for $\theta_{1,R}$ finishes the inverse kinematic solution of the right leg. The procedure for the left leg is identical.

$$\begin{aligned} \begin{bmatrix} r_{pcl,x} + (a_{14,x} - a_{12,x}) \\ r_{pcl,y} \end{bmatrix} &= \\ \begin{bmatrix} a_{2,x} + a_{3,x}\cos(\theta_{2,R}) & -a_{3,x}\sin(\theta_{2,R}) \\ a_{3,x}\sin(\theta_{2,R}) & a_{2,x} + a_{3,x}\cos(\theta_{2,R}) \end{bmatrix} \begin{bmatrix} \cos(\theta_1) \\ \sin(\theta_1) \end{bmatrix} \\ \theta_{1,R} &= atan2(\sin(\theta_1), \cos(\theta_1)) \end{aligned} \tag{20}$$

4.3. ASLB Dynamics

Based on the generalized coordinates, multi-body dynamics of ASLB are formulated as in (Equation (21))

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}_{C,i}^T \boldsymbol{F}_{C,i} \tag{21}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{12\times12}$, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^{12}$, and $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^{12}$ are the generalized mass, Coriolis and centrifugal, and gravitational terms, respectively. $\boldsymbol{S} \in \mathbb{R}^{12\times6}$, $\boldsymbol{\tau} \in \mathbb{R}^{6}$, $\boldsymbol{F}_{C,i} \in \mathbb{R}^{3}$ and $\boldsymbol{J}_{C,i} \in \mathbb{R}^{3\times12}$ are the selection matrix for the actuated joints of respective legs, actuated joint torques, geometric Jacobian of the tip point of the i^{th} leg, and external force on the tip of i^{th} leg.

Let $\boldsymbol{x} = [\boldsymbol{q},\ \dot{\boldsymbol{q}}]^T \in \mathbb{R}^{24}$, $\boldsymbol{u}_\tau = \boldsymbol{\tau}$, and $\boldsymbol{u}_c = \boldsymbol{F}_{C,i}\ \forall i$ be the states, inputs to motors and external forces acting on tip point of the legs, respectively; then, the non-linear dynamics of the robot can be written as in Equation (22).

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}_\tau,\boldsymbol{u}) \\ \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}_\tau,\boldsymbol{u}) &= \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}(\boldsymbol{q})^{-1}\boldsymbol{\Theta} \end{bmatrix} \\ \boldsymbol{\theta} &= -\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{S}\boldsymbol{\tau} + \sum_i \boldsymbol{J}_{C,i}^T \boldsymbol{F}_{C,i} \end{aligned} \tag{22}$$

The application of the graph-theoretic modeling for the ASLB by separating the legs, which are defined as **Agent 2** and **Agent 3**, from the floating base introduces simplifications to the overall complexity of the model and optimization. One of the simplifications appears in modeling as leg dynamics does not necessarily need to be modeled with respect to $\mathcal{F}_I$ using $\boldsymbol{q}_B$ but rather is better defined with respect to the $\mathcal{F}_B$. This local representation of leg dynamics leads to following outcomes for the dynamics of the agents:

- The leg is only used to find adjacency and contact forces on the floating base and ground,
- The state space representation of the leg dynamics can be kept at velocity level.

Note that the contact forces and geometric properties of contact point must be converted to $\mathcal{F}_B$.

For ASLB, the floating base of the robot is denoted as **Agent 1**, the right leg is denoted as **Agent 2**, and the left leg is denoted as **Agent 3**. **Agent 1** is defined with two nodes, **node 1** and **node 2**; **Agent 2** is defined with **node 3**; and **Agent 3** is defined with **node 4**. The locations of these nodes are illustrated in Figure 5, and it should be noted that **node 1** and **node 2** coincide. Without any interconnection constraint, agents are independent of each other; however, there are rigid joints connecting them. In this case there are two bi-directional edges and these are $(\textbf{node 3}, \textbf{node 1}) \in \mathcal{E}$ and $(\textbf{node 2}, \textbf{node 4}) \in \mathcal{E}$. Connections between nodes are assumed to be rigid, and then the adjacency matrix is composed of edge weights $a_{mn} = 1$. This yields an adjacency matrix as in (23).

$$\mathbb{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{23}$$

Finally, the relationships between cooperative agents are given in (24) by the Laplacian matrix. $\boldsymbol{I}_p$ represents identity matrix with size p.

$$\mathbb{L} = \boldsymbol{I}_4 - \mathbb{A} \tag{24}$$

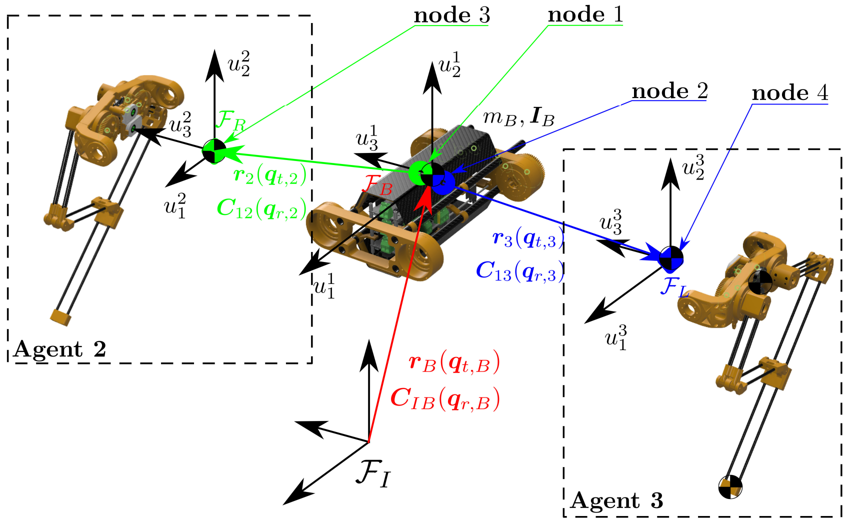
Let $\boldsymbol{W} \in \mathbb{R}^p$ be the signal that is being shared between agents; then, $\mathbb{L}$ can be expanded as in (25) to comply with the signal dimension. $\otimes$ is the Kronecker product operator.

$$\boldsymbol{L} = \mathbb{L} \otimes \boldsymbol{I}_p \tag{25}$$

Recalling that **node 1** and **node 2** are coincident, one can define the following adjacency constraints using the extended Laplacian definition given in (25)

$$
\begin{aligned}
\mathcal{L}_x &= \boldsymbol{L}\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \boldsymbol{x}_3 \end{bmatrix} = \boldsymbol{0} \\
\mathcal{L}_w &= \boldsymbol{L}\begin{bmatrix} \boldsymbol{W}_{A,1} \\ \boldsymbol{W}_{A,2} \\ \boldsymbol{W}_{A,3} \\ \boldsymbol{W}_{A,4} \end{bmatrix} = \boldsymbol{0}
\end{aligned}
\tag{26}
$$

Another aspect of splitting a lumped multi-body model into distributed cooperating multi-body models is generalized coordinates. This operation necessarily duplicates the generalized coordinates of the floating base in a lumped model to the distributed models. In addition to that, **Agent 2** and **Agent 3** have actuated joints of $\boldsymbol{q}_{a,i}$, $i = 2, 3$. Therefore, generalized coordinates of the agents are defined as $\boldsymbol{q}_1 = \begin{bmatrix} \boldsymbol{q}_{t,1}^T \ \boldsymbol{q}_{r,1}^T \end{bmatrix}^T \in \mathbb{R}^6$, $\boldsymbol{q}_2 = \begin{bmatrix} \boldsymbol{q}_{t,2}^T \ \boldsymbol{q}_{r,2}^T \ \boldsymbol{q}_{a,2}^T \end{bmatrix}^T \in \mathbb{R}^9$ and $\boldsymbol{q}_3 = \begin{bmatrix} \boldsymbol{q}_{t,3}^T \ \boldsymbol{q}_{r,3}^T \ \boldsymbol{q}_{a,3}^T \end{bmatrix}^T \in \mathbb{R}^9$, respectively, for **Agent 1**, **Agent 2**, and **Agent 3**. This is illustrated in Figure 5.



node 3
node 1
node 2
node 4
u_2^2
u_3^2
$\mathcal{F}_B$
u_1^2
$\boldsymbol{r}_2(\boldsymbol{q}_{t,2})$
$\boldsymbol{C}_{12}(\boldsymbol{q}_{r,2})$
u_2^1
u_3^1
$m_B, \boldsymbol{I}_B$
$\mathcal{F}_B$
u_1^1
$\boldsymbol{r}_3(\boldsymbol{q}_{t,3})$
$\boldsymbol{C}_{13}(\boldsymbol{q}_{r,3})$
u_2^3
u_3^3
$\mathcal{F}_L$
u_1^3
$\boldsymbol{r}_B(\boldsymbol{q}_{t,B})$
$\boldsymbol{C}_{IB}(\boldsymbol{q}_{r,B})$
$\mathcal{F}_I$
Agent 2
Agent 3

Figure 5. Cooperative interconnection between agents and generalized coordinates at every agent.

4.4. Agent Kinematics

The YZX Euler sequence is used in defining the composite rotations $\boldsymbol{C}_{Ii}(\boldsymbol{q}_{r,i})$ from $\mathcal{F}_I$ to $\mathcal{F}_i$, where subscript i is the agent index and $\mathcal{F}_i$ is the local origin of the agent. The contact point position and velocity of **Agent 2** and **Agent 3** are calculated with respect to $\mathcal{F}_B$, which is denoted as in (27). $\mathcal{K}^+(\boldsymbol{q}_i)$ represents forward kinematics of leg i in Equation (27), which is explained in Section 4.2.

$$
\begin{aligned}
\boldsymbol{r}_{c,i}(\boldsymbol{q}_i) &= \mathcal{K}^+(\boldsymbol{q}_i) \\
\dot{\boldsymbol{r}}_{c,i}(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i) &= \boldsymbol{J}_{c,i}\dot{\boldsymbol{q}}_i
\end{aligned}
\tag{27}
$$

4.5. Agent Dynamics

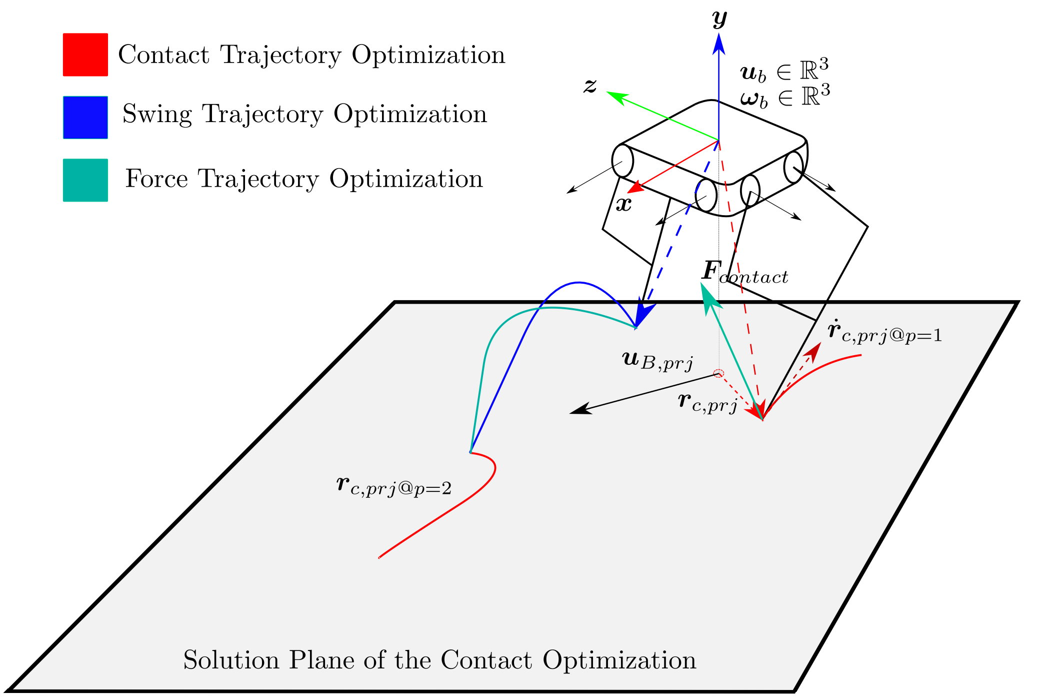
The dynamics of agents yield a similar equation as given in (22), and, for brevity, a representative Equation of Motion (EoM) is given in this section.

Generalized velocities are assigned to states of each agent as $\boldsymbol{x}_i = \dot{\boldsymbol{q}}_i$. External forces in distributed notation are divided into two, where the first one is denoted as $\boldsymbol{F}_{C,i}$ and acts on the agents as a result of ground contact. The second external force is denoted as $\boldsymbol{F}_{A,m}$ and exerted on the agents from the adjacent nodes. Adjacent nodes also transmit moment, $\boldsymbol{M}_{A,m}$; therefore it is convenient to collect forces and moments at adjacent nodes such as a wrench, denoted as $\boldsymbol{W}_{A,m} = \begin{bmatrix} \boldsymbol{F}_{A,m}^T \ \ \boldsymbol{M}_{A,m}^T \end{bmatrix}^T$. As a result, the non-linear dynamics of each agent are written as given in (28). Let $\boldsymbol{x}_i = \dot{\boldsymbol{q}} \in \mathbb{R}^{n_{xi}}$, $\boldsymbol{u}_{\tau,i} = \boldsymbol{\tau} \in \mathbb{R}^{n_{ti}}$, $\boldsymbol{u}_{c,i} = \boldsymbol{F}_{C,i} \in \mathbb{R}^{n_{fi}}$, and $\boldsymbol{u}_{w,i} = \boldsymbol{W}_{A,i} \in \mathbb{R}^{n_{wi}}$ be the states, torques, contact forces, and adjacency wrench, respectively; then, the non-linear dynamics of the robot can be written as in Equation (22). The signal sizes for **Agent 1** are $n_{x1} = 6, n_{t1} = 0, n_{f1} = 0, n_{w1} = 6$, while **Agent 2** and **Agent 3** have signal sizes of $n_{xi} = 9$, $n_{ti} = 3$, $n_{fi} = 3$, and $n_{wi} = 6$ for $i = 2, 3$.

$$\begin{aligned}
\dot{\boldsymbol{x}}_i &= \boldsymbol{f}_i(\boldsymbol{q}_i, \boldsymbol{u}_{\tau,i}, \boldsymbol{u}_{c,i}) \\
\boldsymbol{f}_i &= \boldsymbol{M}_i(\boldsymbol{q}_i)^{-1}\boldsymbol{\Theta}_i \\
\boldsymbol{\Theta}_i &= -\boldsymbol{C}_i(\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i) - \boldsymbol{G}_i(\boldsymbol{q}_i) \\
&\quad + \boldsymbol{S}_i\boldsymbol{\tau}_i + \boldsymbol{J}_{C,i}^T\boldsymbol{F}_{C,i} + \sum_m \boldsymbol{J}_{A,m}^T\boldsymbol{W}_{A,m}
\end{aligned} \tag{28}$$

5. Cooperative Graph-Theoretic Online Trajectory Generation for ASLB

Graph-theoretic online trajectory generation relies on the cooperative modeling of the robot and is composed of a series of optimizations. These are the contact phase, swing phase, and force optimizations, where contact optimization finds the optimal finite horizon for the current contact phase and a sequence of contact trajectories that will keep the robot states bounded for defined phase horizon. For this reason, the resultant contact phase trajectories, except for the one associated with the current contact leg, are not passed to the next optimization. Another cardinal data set that is passed to the next optimization from contact phase is the initial point of the subsequent contact phase. This information is required to generate a rough swing trajectory for the leg that is in the air. It should be noted that both contact and swing trajectories are calculated to provide an initial trajectory for the force optimization, where trajectories are refined using a cooperative system framework. Trajectories to previously defined optimizations are illustrated in Figure 6.



🟥 Contact Trajectory Optimization
🟦 Swing Trajectory Optimization
🟩 Force Trajectory Optimization
$\boldsymbol{y}$
$\boldsymbol{z}$
$\boldsymbol{x}$
$\boldsymbol{u}_b \in \mathbb{R}^3$
$\boldsymbol{\omega}_b \in \mathbb{R}^3$
$\boldsymbol{F}_{contact}$
$\dot{\boldsymbol{r}}_{c,prj@p=1}$
$\boldsymbol{u}_{B,prj}$
$\boldsymbol{r}_{c,prj}$
$\boldsymbol{r}_{c,prj@p=2}$
Solution Plane of the Contact Optimization

Figure 6. Cooperative interconnection between agents and generalized coordinates at every agent.

5.1. Contact Phase Optimization

The contact-phase optimization calculates a set of trajectories using the linear inverted pendulum model (LIPM) and contact constraints. The LIPM dynamics we used in this work are widely used in the vast majority of the literature. As shown in Section 5.3, the dynamics are written with respect to body frame $\mathcal{F}_B$. Besides that, the dynamics are also kept at the velocity level. Under these circumstances ,the contact conditions for the leg in contact need to be defined accordingly.

5.1.1. Contact Condition

Under contact conditions with the no-slip assumption, $\boldsymbol{r}_{c,i}$ has no relative motion with respect to the ground if this condition is observed from the inertial frame $\mathcal{F}_I$. This condition is observed from $\mathcal{F}_B$ as if $\dot{\boldsymbol{r}}_{c,i} = -\boldsymbol{u}_B$, which is illustrated in Equation (29) for agents $i = 2,3$. Recall that states of agents $i = 2,3$ are denoted as $\boldsymbol{x}_i = \left[\boldsymbol{q}_{t,i}^T\, \boldsymbol{q}_{r,i}^T\, \boldsymbol{q}_{a,i}^T\right]^T,\ i = 2,3$, and $\dot{\boldsymbol{q}}_{t,i}$,where $i = 2,3$ is duplicate of $\boldsymbol{u}_B$, assuming that the connections between nodes are as defined in Equation (26).

$$\boldsymbol{0} = \begin{bmatrix} -\boldsymbol{I} & -\boldsymbol{J}_{r,i} & \boldsymbol{J}_{a,i} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{t,i} \\ \dot{\boldsymbol{q}}_{r,i} \\ \dot{\boldsymbol{q}}_{a,i} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{I} & -\boldsymbol{J}_{r,i} & \boldsymbol{J}_{a,i} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_B \\ \dot{\boldsymbol{q}}_{r,i} \\ \dot{\boldsymbol{q}}_{a,i} \end{bmatrix} \tag{29}$$

Assuming the body is slowly rotating, $\dot{\boldsymbol{q}}_{r,i} \approx \boldsymbol{0},\ i = 2,3$

5.1.2. LIPM Model

The implementation of the LIPM model in this work has some nuances compared to the the work where it is proposed [43,44] and illustrated in Figure 7. Let $\boldsymbol{F}_{xz} \in \mathbb{R}^2$ be the virtual force created on the $x - z$ plane due to the displacement between origin and contact locations ${}_{xz}\boldsymbol{r}_{c,2}$ and ${}_{xz}\boldsymbol{r}_{c,3}$. Note that the origin represents the center of mass (CoM) and is not the origin of the $\mathcal{F}_B$. Although CoM moves with respect to the origin of $\mathcal{F}_B$, in practice, it is assumed to be fixed with an offset from $\mathcal{F}_B$.

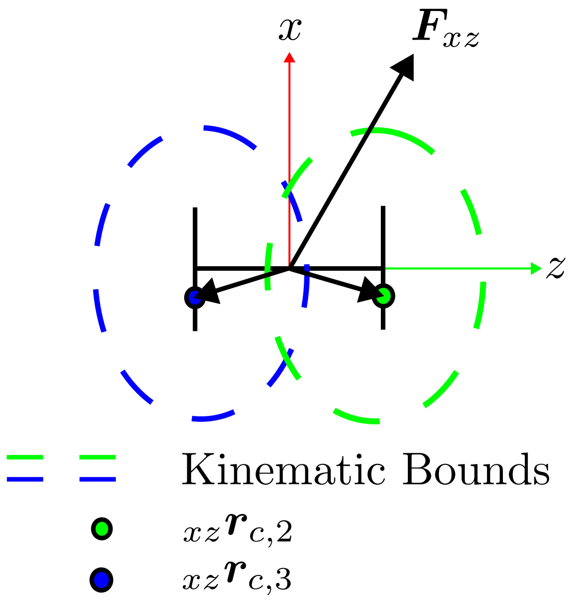


Figure 7. LIPM dynamics projected on the $x - z$ plane.

Under these assumptions, the LIPM dynamics are provided as in Equation (30) in state space form. States for this system are denoted as $\boldsymbol{x}_c$ and defined as the contact point, and this point is denoted as $\boldsymbol{\Delta r}_{xz}$, as any of the two contact locations can be assigned to it, which are ${}_{xz}\boldsymbol{r}_{c,2}$ and ${}_{xz}\boldsymbol{r}_{c,3}$, respectively. Specifically, states are defined as $\overline{\boldsymbol{x}}_{c,1} = \boldsymbol{\Delta r}_{xz} \in \mathbb{R}^2$, $\overline{\boldsymbol{x}}_{c,2} = \frac{d}{dt}\boldsymbol{\Delta r}_{xz} \in \mathbb{R}^2$, which are combined as $\overline{\boldsymbol{x}}_c = \left[\overline{\boldsymbol{x}}_{c,1}^T\ \overline{\boldsymbol{x}}_{c,2}^T\right]^T$.

$$\dot{\overline{\boldsymbol{x}}}_c = \begin{bmatrix} \dot{\overline{\boldsymbol{x}}}_{c,1} \\ \dot{\overline{\boldsymbol{x}}}_{c,2} \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{0} & \boldsymbol{I} \\ \hline \omega_0^2 \boldsymbol{I} & \boldsymbol{0} \end{array}\right] \begin{bmatrix} \overline{\boldsymbol{x}}_{c,1} \\ \overline{\boldsymbol{x}}_{c,2} \end{bmatrix} = \boldsymbol{A}_{lipm}\overline{\boldsymbol{x}}_c \tag{30}$$

This system is an inherently unstable system; therefore, what is being pursued with this system is to find a set of initial conditions, denoted as ${}^{0}\overline{\boldsymbol{x}}_{c}$, that will propel the CoM of the robot toward the desired velocity vector. While carrying that out, a set of state bounds are also satisfied. Equation (30) is discretized using Euler propagation as provided below, where Δt is the sampling time.

$$\overline{\boldsymbol{x}}_{c,k+1} = (\boldsymbol{I} + \Delta t \boldsymbol{A}_{lipm})\overline{\boldsymbol{x}}_{c,k} \tag{31}$$

5.1.3. Contact Phase Optimization

This method relies on finding a set of trajectories that will keep the proceeding steps within bounds; therefore, a phase horizon is defined as $N_p \in \mathbb{N}$, which represents the number of phases to be calculated during the optimization, including the current phase. A finite horizon for each phase is defined as $N_n \in \mathbb{N}$, which will be minimized for $N_p = 1$ and kept at its nominal for $N_p > 1$. The states of the LIPM dynamics in each phase are denoted as ${}^{p}\overline{\boldsymbol{x}}_{c,k}$, where $p = [1, \cdots, N_p]$ and $k = [1, \cdots, N_n]$. The combined states for each phase are denoted as ${}^{p}\boldsymbol{x}_{c,K}$ as provided below.

$${}^{p}\overline{\boldsymbol{x}}_{c,K} = \begin{bmatrix} {}^{p}\overline{\boldsymbol{x}}_{c,1} \\ \vdots \\ {}^{p}\overline{\boldsymbol{x}}_{c,N_n} \end{bmatrix}, \ \forall\, p \tag{32}$$

Equation (33) represents the quadratic problem that runs in contact optimization phase, where $\mathcal{L}_c({}^{p}\overline{\boldsymbol{x}}_{c,1}, \boldsymbol{u}_{B,ref})$ is the objective function, and $\mathcal{C}_{cont}({}^{p}\overline{\boldsymbol{x}}_{c,1})$ and $\mathcal{C}_{bounds}({}^{p}\overline{\boldsymbol{x}}_{c,1})$ are equality and inequality constraints to ensure continuity of the states between phases and to keep the states within predefined bounds.

$$\begin{aligned} \underset{{}^{p}\overline{\boldsymbol{x}}_{c,1}}{\text{minimize}} \quad & \sum_{k=1}^{M} \mathcal{L}_c({}^{p}\overline{\boldsymbol{x}}_{c,1}, \boldsymbol{u}_{B,ref}) \\ \text{s. t.} \quad & \mathcal{C}_{cont}({}^{p}\overline{\boldsymbol{x}}_{c,1}) = \boldsymbol{0} \\ & \mathcal{C}_{bounds}({}^{p}\overline{\boldsymbol{x}}_{c,1}) \leq \boldsymbol{0} \end{aligned} \tag{33}$$

5.1.4. Contact Phase: Continuity Constraint

As explained earlier, contact optimization seeks to find several contact phase trajectories, and these trajectories should ideally be continuous. This is achieved by an equality constraint defined as in Equation (34) for $p = 1, \cdots, N_p - 1$.

$$\begin{aligned} {}^{p}\overline{\boldsymbol{x}}_{c,N_n} &= {}^{p+1}\overline{\boldsymbol{x}}_{c,1} \\ \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} {}^{p}\overline{\boldsymbol{x}}_{c,N_n} &= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} {}^{p+1}\overline{\boldsymbol{x}}_{c,1} \end{aligned} \tag{34}$$

5.1.5. Contact Phase: Constraint for State Bounds

State bounds are defined based on the leg in contact; therefore, state bounds are switching between the bounds of **Agent 2** and **Agent 3**. Let $c_2 = [0,1]$ and $c_3 = [0,1]$ be the contact indicators of **Agent 2** and **Agent 3**, respectively. Under contact $c_i = 1$; otherwise, $c_i = 0$ for $i = 2,3$ at phase p. The bounds for **Agent 2** and **Agent 3** are defined as $\boldsymbol{S}_i = \left[\boldsymbol{S}_{i,UB}^{T}, \boldsymbol{S}_{i,LB}^{T}\right]^{T}$ and assigned to $\boldsymbol{S}_p$ such that $\boldsymbol{S}_p = \boldsymbol{S}_i$ if $c_i = 1$. Note that we are assuming a single point of contact. Using the bounds, the state boundary constraints are defined as in Equation (35).

$$\begin{bmatrix} \boldsymbol{I} \\ -\boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} {}^{p}\overline{\boldsymbol{x}}_{c,k} \leq \begin{bmatrix} \boldsymbol{S}_{p,UB} \\ \boldsymbol{S}_{p,LB} \end{bmatrix} \tag{35}$$

5.1.6. Contact Phase: Cost Function

Contact phase optimization aims to reach the $x - z$ projection of reference velocity that is provided by the user $\boldsymbol{u}_{B,ref}$, which is $\boldsymbol{u}_{B,xz}$. Recall that $\dot{\boldsymbol{r}}_{c,i} = -\boldsymbol{u}_B$; therefore, the cost function is written as in Equation (36).

$$\begin{aligned} J_{contact} = & \left[\boldsymbol{u}_{B,xz} + \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \end{bmatrix} {}^p\overline{\boldsymbol{x}}_{c,K}\right]^T \boldsymbol{Q}_s \left[\boldsymbol{u}_{B,xz} + \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \end{bmatrix} {}^p\overline{\boldsymbol{x}}_{c,K}\right]^T + \\ & \left[\begin{bmatrix} \mathbf{0} & \boldsymbol{I} \end{bmatrix} {}^p\overline{\boldsymbol{x}}_{c,K}\right]^T \boldsymbol{Q}_p \left[\begin{bmatrix} \mathbf{0} & \boldsymbol{I} \end{bmatrix} {}^p\overline{\boldsymbol{x}}_{c,K}\right]^T \end{aligned} \quad (36)$$

5.2. Swing Phase Optimization

Swing phase optimization calculates a rough trajectory for the swinging leg by connecting the current position of the tip of the swinging leg to the initial point of the proceeding contact phase trajectory with a bezier curve. Instances of the swing trajectory are denoted as ${}^p\boldsymbol{x}_{s,k}$, and the current and final positions of the swing trajectory are denoted as ${}^p\boldsymbol{x}_{s,1}$ and ${}^p\boldsymbol{x}_{s,N_n}$, respectively. ${}^p\overline{\boldsymbol{x}}_{s,1}$ and ${}^p\overline{\boldsymbol{x}}_{s,N_n}$ are the projections of the vectors on the $x - z$ plane, and the initial point of the proceeding contact phase trajectory is ${}^{p+1}\overline{\boldsymbol{x}}_{c,1}$. Note that the contact phase optimization generates a planar trajectory. Therefore, the implementation of swing-phase trajectory optimization requires a modification of these vectors by adding the height of the CoM to the y axis of any projected vector if it needs to be passed to the swing trajectory. Figure 8 illustrates previously mentioned vectors. Dashed lines represent the projection of the swing trajectory, while solid lines show the contact trajectory. $\boldsymbol{n}_0$, $\boldsymbol{n}_f$, and $\boldsymbol{L}$ represent the unit vector to CoM, the unit vector from CoM, and the straight line on the $x - z$ plane between initial and final positions of the swing trajectory. Formal definitions for $\boldsymbol{n}_0$, $\boldsymbol{n}_f$ and $\boldsymbol{L}$ are provided in Equation (37).

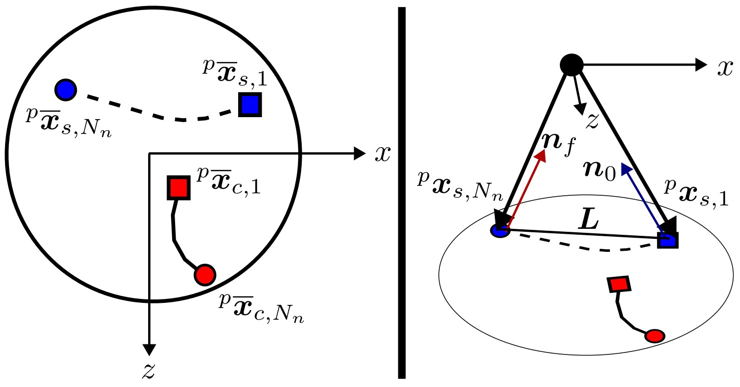


Figure 8. Swing phase trajectory generation.

$$\begin{aligned} \boldsymbol{n}_0 &= -\frac{{}^{p=1}\boldsymbol{x}_{s,1}}{|{}^{p=1}\boldsymbol{x}_{s,1}|} \\ \boldsymbol{n}_f &= \frac{{}^{p=1}\boldsymbol{x}_{s,N_n}}{|{}^{p=1}\boldsymbol{x}_{s,N_n}|} \\ \boldsymbol{L} &= {}^{p=1}\boldsymbol{x}_{s,N_n} - {}^{p=1}\boldsymbol{x}_{s,1} \end{aligned} \quad (37)$$

5.2.1. Swing Phase Optimization

Swing phase optimization is run for the current phase; therefore, unlike for the contact phase optimization, $p = 1$. The finite horizon for this phase is the N_n at $p = 1$. Note that at $p = 1$, N_n is optimized at the contact phase optimization.

Let $b_{x,j}$, $b_{y,j}$, and $b_{z,j}$ be the coefficients of the bezier curve, where $j = 1, \cdots, nb$. $\boldsymbol{c}_b$ and $\boldsymbol{v}_b$, which are sorted collections of $b_{x,j}$, $b_{y,j}$, and $b_{z,j}$, are defined. This classification collects coefficients related to initial and final positions of the curve under $\boldsymbol{c}_b$ and coefficients that are being optimized under $\boldsymbol{v}_b$. Specifically $\boldsymbol{c}_b$ is defined as $\boldsymbol{c}_b = \begin{bmatrix} {}^{p=1}\boldsymbol{x}_{s,1}^T & {}^{p=1}\boldsymbol{x}_{s,N_n} \end{bmatrix}^T$.

Based on thepreviously described notation, bezier curves for swing trajectory are defined as in Equation (39), where $\boldsymbol{J}_{b,c}$ and $\boldsymbol{J}_{b,v}$ are matrix-valued functions of k, which can be populated for $k = 1 < \cdots, N_n$ and maps $\boldsymbol{c}_b$ and $\boldsymbol{v}_b$ to ${}^{p=1}\boldsymbol{x}_{s,k} \in \mathbb{R}^3$. Similarly, $\boldsymbol{c}_b$ and $\boldsymbol{v}_b$ are mapped to ${}^{p=1}\dot{\boldsymbol{x}}_{s,k} \in \mathbb{R}^3$ using ${}_d\boldsymbol{J}_{b,c}$ and ${}_d\boldsymbol{J}_{b,v}$.

Equation (38) represents the quadratic problem that runs in the swing optimization phase, where $\mathcal{L}_s(\boldsymbol{v}_b)$ is the objective function and $\mathcal{S}_{bounds}(\boldsymbol{v}_b)$ is the set of inequality constraints to keep states within predefined bounds.

$$\begin{aligned} \underset{\boldsymbol{v}_b}{\text{minimize}} \quad & \sum_{k=1}^{M} \mathcal{L}_s(\boldsymbol{v}_b) \\ \text{s. t.} \quad & \mathcal{C}_{bounds}(\boldsymbol{v}_b) \leq \boldsymbol{0} \end{aligned} \tag{38}$$

$$\begin{aligned} {}^{p=1}\boldsymbol{x}_{s,k} &= \begin{bmatrix} \boldsymbol{J}_{b,c}(k) & \boldsymbol{J}_{b,v}(k) \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{v}_b \end{bmatrix} \\ {}^{p=1}\dot{\boldsymbol{x}}_{s,k} &= \begin{bmatrix} {}_d\boldsymbol{J}_{b,c}(k) & {}_d\boldsymbol{J}_{b,v}(k) \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{v}_b \end{bmatrix} \end{aligned} \tag{39}$$

The implementation of swing phase optimization requires the calculation of ${}^0_d\boldsymbol{J}_{b,c}$, ${}^f_d\boldsymbol{J}_{b,c}$, ${}^0_d\boldsymbol{J}_{b,v}$, ${}^f_d\boldsymbol{J}_{b,v}$ using Equation (39) at $k = 1$ and $k = N_n$.

5.2.2. Swing Phase: Constraint for State Bounds

State bounds are defined based on the leg in the swing; therefore, state bounds switch between the bounds of **Agent 2** and **Agent 3**. Bounds for **Agent 2** and **Agent 3** are defined as $\boldsymbol{S}_i = \left[\boldsymbol{S}_{i,UB}^T, \boldsymbol{S}_{i,LB}^T\right]^T$ and assigned to $\boldsymbol{S}_p$ such that $\boldsymbol{S}_p = \boldsymbol{S}_i$ if $c_i = 0$. Using these bounds, state boundary constraints are defined as in Equation (40).

$$\begin{aligned} \begin{bmatrix} \boldsymbol{J}_{b,c}(k)\boldsymbol{c}_b + \boldsymbol{J}_{b,v}(k)\boldsymbol{v}_b \\ -\boldsymbol{J}_{b,c}(k)\boldsymbol{c}_b - \boldsymbol{J}_{b,v}(k)\boldsymbol{v}_b \end{bmatrix} &\leq \boldsymbol{S}_p \\ \begin{bmatrix} \boldsymbol{J}_{b,v}(k) \\ -\boldsymbol{J}_{b,v}(k) \end{bmatrix} \boldsymbol{v}_b &\leq \boldsymbol{S}_p - \begin{bmatrix} \boldsymbol{J}_{b,c}(k) \\ -\boldsymbol{J}_{b,c}(k) \end{bmatrix} \boldsymbol{c}_b \end{aligned} \tag{40}$$

5.2.3. Swing Phase: Cost Function

Swing phase optimization aims to pull the swinging leg to the CoM in the beginning of the swing motion and then pushes it toward the final position. Along with these, it also tries to approach the straight line $\boldsymbol{L}$. Therefore, the cost function of the swing phase optimization is written as in Equation (41).

$$\begin{aligned} J_{swing} = {} & \left(\boldsymbol{n}_0 - \begin{bmatrix} {}^0_d\boldsymbol{J}_{b,c} & {}^0_d\boldsymbol{J}_{b,v} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{v}_b \end{bmatrix}\right)^T \boldsymbol{Q}_0 \left(\boldsymbol{n}_0 - \begin{bmatrix} {}^0_d\boldsymbol{J}_{b,c} & {}^0_d\boldsymbol{J}_{b,v} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{v}_b \end{bmatrix}\right) + \\ & \left(\boldsymbol{n}_f - \begin{bmatrix} {}^f_d\boldsymbol{J}_{b,c} & {}^f_d\boldsymbol{J}_{b,v} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{v}_b \end{bmatrix}\right)^T \boldsymbol{Q}_f \left(\boldsymbol{n}_f - \begin{bmatrix} {}^f_d\boldsymbol{J}_{b,c} & {}^f_d\boldsymbol{J}_{b,v} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{v}_b \end{bmatrix}\right) + \\ & \left(\boldsymbol{L} - \begin{bmatrix} \boldsymbol{J}_{b,c} & \boldsymbol{J}_{b,v} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{v}_b \end{bmatrix}\right)^T \boldsymbol{Q}_t \left(\boldsymbol{L} - \begin{bmatrix} \boldsymbol{J}_{b,c} & \boldsymbol{J}_{b,v} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{v}_b \end{bmatrix}\right) \end{aligned} \tag{41}$$

5.3. Cooperative Force Optimization

Approximate trajectories are obtained in contact and swing phase optimizations, and these trajectories are used in the cooperative force optimization as the initial trajectory. In order to follow the method easily, agent dynamics are rewritten in Equation (43), where $\boldsymbol{M}_h$ and $\boldsymbol{C}_h$ are partitioned as given in Equation (42). In addition, $\boldsymbol{M}_i$, $\boldsymbol{C}_i$, and $\boldsymbol{G}_i$ for $i = 2, 3$ are assigned to $\boldsymbol{M}_h$, $\boldsymbol{C}_h$ and $\boldsymbol{G}_h$, where $h = S$ represents swing, $h = C$ represents contact, and $h = B$ represents floating base matrices. Minimal representation of the dynamics are provided in Equation (44). Floating base dynamics do not switch; however, **Agent 2** and **Agent 3** dynamics are assigned to $h = C$ or $h = S$ depending on c_i for $i = 2, 3$. Similarly, wrenches $\boldsymbol{W}_{A,m}$ are assigned to $\boldsymbol{W}_{A,h}$ based on c_i such that if $c_2 = 1$, then $\boldsymbol{W}_{A,C} = \boldsymbol{W}_{A,2}$ and if $c_3 = 1$, then $\boldsymbol{W}_{A,C} = \boldsymbol{W}_{A,3}$ for $i = 2, 3$. Finally, $\tilde{\boldsymbol{r}}_C$ is the tip point position $\boldsymbol{r}_{c,i}$ of the leg with $c_i = 1$. $\boldsymbol{F}_C$ is the interaction between the contact leg and the ground.

$$\begin{aligned} \boldsymbol{M}_h &= \left[\begin{array}{c|c} {}_{bb}\boldsymbol{M}_h & {}_{bq}\boldsymbol{M}_h \\ \hline {}_{qb}\boldsymbol{M}_h & {}_{qq}\boldsymbol{M}_h \end{array}\right] \\ \boldsymbol{C}_h &= \left[\begin{array}{c|c} {}_{bb}\boldsymbol{C}_h & {}_{bq}\boldsymbol{C}_h \\ \hline {}_{qb}\boldsymbol{C}_h & {}_{qq}\boldsymbol{C}_h \end{array}\right] \end{aligned} \tag{42}$$

$$\begin{aligned} {}_{bb}\boldsymbol{M}_B\dot{\boldsymbol{x}}_B + {}_{bb}\boldsymbol{C}_B\boldsymbol{x}_B + {}_{bb}\boldsymbol{G}_B &= \overline{\boldsymbol{W}}_{A,C} + \overline{\boldsymbol{W}}_{A,S} \\ \left[\begin{array}{c|c} {}_{bb}\boldsymbol{M}_S & {}_{bq}\boldsymbol{M}_S \end{array}\right]\dot{\boldsymbol{x}}_S + \left[\begin{array}{c|c} {}_{bb}\boldsymbol{C}_S & {}_{bq}\boldsymbol{C}_S \end{array}\right]\boldsymbol{x}_S + {}_{bb}\boldsymbol{G}_S &= \boldsymbol{W}_{A,S} \\ \left[\begin{array}{c|c} {}_{bb}\boldsymbol{M}_C & {}_{bq}\boldsymbol{M}_C \end{array}\right]\dot{\boldsymbol{x}}_C + \left[\begin{array}{c|c} {}_{bb}\boldsymbol{C}_C & {}_{bq}\boldsymbol{C}_C \end{array}\right]\boldsymbol{x}_S + {}_{bb}\boldsymbol{G}_C &= \boldsymbol{W}_{A,C} + \begin{bmatrix} \boldsymbol{I} \\ \tilde{\boldsymbol{r}}_C \end{bmatrix}\boldsymbol{F}_C \end{aligned} \tag{43}$$

$$\begin{aligned} \hat{\boldsymbol{M}}_B\dot{\boldsymbol{x}}_B + \hat{\boldsymbol{C}}_B\boldsymbol{x}_B + \hat{\boldsymbol{G}}_B &= \overline{\boldsymbol{W}}_{A,C} + \overline{\boldsymbol{W}}_{A,S} \\ \hat{\boldsymbol{M}}_S\dot{\boldsymbol{x}}_S + \hat{\boldsymbol{C}}_S\boldsymbol{x}_S + \hat{\boldsymbol{G}}_S &= \boldsymbol{W}_{A,S} \\ \hat{\boldsymbol{M}}_C\dot{\boldsymbol{x}}_C + \hat{\boldsymbol{C}}_C\boldsymbol{x}_S + \hat{\boldsymbol{G}}_C &= \boldsymbol{W}_{A,C} + \begin{bmatrix} \boldsymbol{I} \\ \tilde{\boldsymbol{r}}_C \end{bmatrix}\boldsymbol{F}_C \end{aligned} \tag{44}$$

Then, continuous models in Equation (43) are converted into discrete models using Euler discretization, and Equation (45) provides the discrete system model that is used in cooperative force optimization. The current states are denoted as $\boldsymbol{x}_{h,k}$, where h and k represent the model identifier and the prediction step, respectively.

$$\begin{aligned} \begin{bmatrix} \Delta t\hat{\boldsymbol{C}}_{B,k} - \hat{\boldsymbol{M}}_{B,k} & \hat{\boldsymbol{M}}_{B,k} \end{bmatrix}\begin{bmatrix} \boldsymbol{x}_{B,k} \\ \boldsymbol{x}_{B,k+1} \end{bmatrix} - \Delta t\overline{\boldsymbol{W}}_{S,k} - \Delta t\overline{\boldsymbol{W}}_{C,k} &= \Delta t\hat{\boldsymbol{G}}_B \\ \begin{bmatrix} \Delta t\hat{\boldsymbol{C}}_{S,k} - \hat{\boldsymbol{M}}_{S,k} & \hat{\boldsymbol{M}}_{S,k} \end{bmatrix}\begin{bmatrix} \boldsymbol{x}_{S,k} \\ \boldsymbol{x}_{S,k+1} \end{bmatrix} - \Delta t\boldsymbol{W}_{S,k} &= \Delta t\hat{\boldsymbol{G}}_{S,k} \\ \begin{bmatrix} \Delta t\hat{\boldsymbol{C}}_{C,k} - \hat{\boldsymbol{M}}_{C,k} & \hat{\boldsymbol{M}}_{C,k} \end{bmatrix}\begin{bmatrix} \boldsymbol{x}_{C,k} \\ \boldsymbol{x}_{C,k+1} \end{bmatrix} - \Delta t\boldsymbol{W}_{C,k} - \Delta t\hat{\boldsymbol{J}}_{C,k}\boldsymbol{F}_{C,k} &= \Delta t\hat{\boldsymbol{G}}_{C,k} \end{aligned} \tag{45}$$

For brevity, Equations (45) are represented with a minimal representation as follows.

$$\begin{aligned} \tilde{\boldsymbol{M}}_{h,k} &= \begin{bmatrix} \tilde{\boldsymbol{M}}_{h_1,k} & \tilde{\boldsymbol{M}}_{h_2,k} \end{bmatrix} = \begin{bmatrix} \Delta t\hat{\boldsymbol{C}}_{B,k} - \hat{\boldsymbol{M}}_{B,k} & \hat{\boldsymbol{M}}_{B,k} \end{bmatrix} \\ \tilde{\boldsymbol{J}}_{C,k} &= -\Delta t\hat{\boldsymbol{J}}_{C,k} \\ \tilde{\boldsymbol{G}}_{h,k} &= \Delta t\hat{\boldsymbol{G}}_{h,k} \\ \tilde{\boldsymbol{P}}_{h,k} &= -\Delta t\hat{\boldsymbol{I}} \end{aligned} \tag{46}$$

5.3.1. Cooperative Force Optimization Problem

This method relies on initially provided trajectories that is provided by contact and swing phase. In this phase, decision variables are defined as corrections to the nominal trajectories, and a complete trajectory is defined as such. Nominal trajectories for states are denoted with ${}_0\boldsymbol{x}_{h,k}$, and corrections to the nominal trajectories at every instant are denoted as $\Delta\boldsymbol{x}_{h,k}$. Similarly, force trajectories are defined in the same fashion such that ${}_0\boldsymbol{F}_{C,k}$ and ${}_0\boldsymbol{W}_{A,m}$ are the nominal force trajectories, while $\Delta\boldsymbol{F}_{C,k}$ and $\Delta\boldsymbol{W}_{A,m}$ are the corrections to the relevant trajectories.

$$\begin{aligned} \boldsymbol{x}^T_{h,k} &= {}_0\boldsymbol{x}_{h,k} + \Delta\boldsymbol{x}_{h,k} \\ \boldsymbol{F}^T_{C,k} &= {}_0\boldsymbol{F}_{C,k} + \Delta\boldsymbol{F}_{C,k} \\ \boldsymbol{W}^T_{h,k} &= {}_0\boldsymbol{W}_{h,k} + \Delta\boldsymbol{W}_{h,k} \end{aligned} \tag{47}$$

States, contact forces, and wrenches for the entire trajectory are combined as follows.

$$\begin{aligned}\boldsymbol{x}_{h,K}^T &= \begin{bmatrix} \boldsymbol{x}_{h,1}^T & \cdots & \boldsymbol{x}_{h,N_n}^T \end{bmatrix}^T \\ \boldsymbol{F}_{C,K}^T &= \begin{bmatrix} \boldsymbol{F}_{C,1}^T & \cdots & \boldsymbol{F}_{C,N_n}^T \end{bmatrix}^T \\ \boldsymbol{W}_{h,K}^T &= \begin{bmatrix} \boldsymbol{W}_{h,1}^T & \cdots & \boldsymbol{W}_{h,N_n}^T \end{bmatrix}^T\end{aligned} \tag{48}$$

The relationship between $\boldsymbol{x}_{h,K}$, $\boldsymbol{F}_{C,K}$, and $\boldsymbol{W}_{h,K}$ can be written for the entire trajectory using combined matrices as provided in Equation (49).

$$\begin{aligned}\tilde{\boldsymbol{M}}_{h,K} &= \begin{bmatrix} \tilde{\boldsymbol{M}}_{h_1,1} & \tilde{\boldsymbol{M}}_{h_2,2} & \boldsymbol{0} & \cdots \\ \boldsymbol{0} & \tilde{\boldsymbol{M}}_{h_1,2} & \tilde{\boldsymbol{M}}_{h_2,3} & \\ \vdots & & \ddots & \ddots \end{bmatrix} \\ \tilde{\boldsymbol{J}}_{C,K} &= \begin{bmatrix} -\Delta t \hat{\boldsymbol{J}}_{C,1} & \boldsymbol{0} & \cdots \\ \boldsymbol{0} & -\Delta t \hat{\boldsymbol{J}}_{C,2} & \\ \vdots & & \ddots \end{bmatrix} \\ \tilde{\boldsymbol{G}}_{h,K} &= \begin{bmatrix} \tilde{\boldsymbol{G}}_{h,1} \\ \tilde{\boldsymbol{G}}_{h,2} \\ \vdots \end{bmatrix} \\ \tilde{\boldsymbol{P}}_{h,K} &= \begin{bmatrix} \tilde{\boldsymbol{P}}_{h,1} & \boldsymbol{0} & \cdots \\ \boldsymbol{0} & \tilde{\boldsymbol{P}}_{h,2} & \\ \vdots & & \ddots \end{bmatrix}\end{aligned} \tag{49}$$

The dynamics for the floating base, contacting, and swinging bodies are written as provided in Equation (50).

$$\begin{aligned}\begin{bmatrix} \tilde{\boldsymbol{M}}_{B,K} & \tilde{\boldsymbol{P}}_{S,K} & \tilde{\boldsymbol{P}}_{C,K} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{B,K} \\ \overline{\boldsymbol{W}}_{S,K} \\ \overline{\boldsymbol{W}}_{C,K} \end{bmatrix} &= \boldsymbol{G}_{B,K} \\ \begin{bmatrix} \tilde{\boldsymbol{M}}_{S,K} & \tilde{\boldsymbol{P}}_{S,K} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{S,K} \\ \boldsymbol{W}_{S,K} \end{bmatrix} &= \boldsymbol{G}_{S,K} \\ \begin{bmatrix} \tilde{\boldsymbol{M}}_{C,K} & \tilde{\boldsymbol{P}}_{C,K} & \tilde{\boldsymbol{J}}_{C,K} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{C,K} \\ \boldsymbol{W}_{C,K} \\ \boldsymbol{F}_{C,K} \end{bmatrix} &= \boldsymbol{G}_{C,K}\end{aligned} \tag{50}$$

The optimization for the three agents is represented in a single objective Equation (51) and a set of constraints in this work; however, the problem is readily available for distributed optimization.

$$\begin{aligned}\underset{\Delta \boldsymbol{x}_{h,k}, \Delta \boldsymbol{F}_{C,k}, \Delta \boldsymbol{W}_{h,k}}{\text{minimize}} \quad & \sum_{k=1}^{M} \mathcal{L}_c(\boldsymbol{x}_{h,k}, \boldsymbol{F}_{C,k}, \boldsymbol{W}_{h,k}) \\ \text{s. t.} \quad & \mathcal{C}_{dyn}(\Delta \boldsymbol{x}_{h,k}, \Delta \boldsymbol{F}_{C,k}, \Delta \boldsymbol{W}_{h,k}) = \boldsymbol{0} \\ & \mathcal{C}_{coop}(\Delta \boldsymbol{x}_{h,k}, \Delta \boldsymbol{F}_{C,k}, \Delta \boldsymbol{W}_{h,k}) = \boldsymbol{0} \\ & \mathcal{C}_{cntct}(\Delta \boldsymbol{x}_{h,k}) = \boldsymbol{0} \\ & \mathcal{C}_{fc}(\Delta \boldsymbol{F}_{C,k}) \leq \boldsymbol{0}\end{aligned} \tag{51}$$

Based on the dynamics given in Equation (49) and the representation of the trajectories provided in Equation (47), the matrices for the equality constraints are denoted as $\boldsymbol{A}_{h,dyn}$ and $\boldsymbol{B}_{h,dyn}$. The equality constraint is provided for only the floating base for brevity.

$$\begin{bmatrix} \tilde{M}_{B,K} & \tilde{P}_{S,K} & \tilde{P}_{C,K} \end{bmatrix} \begin{bmatrix} \Delta x_{B,K} \\ \Delta \overline{W}_{S,K} \\ \Delta \overline{W}_{C,K} \end{bmatrix} = G_{B,K} - \begin{bmatrix} \tilde{M}_{B,K} & \tilde{P}_{S,K} & \tilde{P}_{C,K} \end{bmatrix} \begin{bmatrix} {}_0x_{B,K} \\ {}_0\overline{W}_{S,K} \\ {}_0\overline{W}_{C,K} \end{bmatrix} \tag{52}$$

5.3.2. Cooperative Force Optimization: Contact Constraint

The contact constraint was provided previously in Equation (29). This condition is modified for the definition of the trajectory provided in Equation (47). Put simply, the contact point velocity has to be equal to the body velocity in the opposite direction in the no-slip condition, and Equation (53) projects the relationship on the decision variables for the quadratic optimization.

$$\begin{aligned} \mathbf{0} &= \begin{bmatrix} -I & J_{a,i} \end{bmatrix} x_{B,K} \\ \mathbf{0} &= \begin{bmatrix} -I & J_{a,i} \end{bmatrix} \Delta x_{B,K} + \begin{bmatrix} -I & J_{a,i} \end{bmatrix} {}_0x_{B,K} \\ -\begin{bmatrix} -I & J_{a,i} \end{bmatrix} \Delta x_{B,K} &= \begin{bmatrix} -I & J_{a,i} \end{bmatrix} {}_0x_{B,K} \end{aligned} \tag{53}$$

5.3.3. Cooperative Force Optimization: Force Cone Constraint

The coooperative force optimization phase is designed as a quadratic problem; therefore, constraints have to be set accordingly. Contact constraints are dedicated to keep tangential forces small so that no slipping occurs. To do so, a friction pyramid is created inside a friction cone. The friction cone is a geometric interpretation of the magnitude of the allowable tangential force that can be applied on the ground. The allowable limit is calculated by simply multiplying the normal component of the applied force by the contact point with the friction coefficient. The nrmal component of the force is denoted as ${}_Bf_{c,n}$ and the tangential components of the applied force are denoted as ${}_Bf_{c,t}$, ${}_Bf_{c,s}$, respectively. It should be noted that the applied force is defined with respect to $\mathcal{F}_B$. The friction coefficient is denoted as μ. The radius of the friction cone is defined as $\boldsymbol{r}_{fc} = \mu_B f_{c,n}$. The friction pyramid is defined such that it is always upper-bounded by the $\boldsymbol{r}_{fc}$, and this is achieved by setting linear bounds that are denoted as $r_{fc,s}$ and $r_{fc,t}$. These bounds are calculated such that $|r_{fc,s}| \approx 0.707\boldsymbol{r}_{fc}$ and $|r_{fc,s}| \approx 0.707\boldsymbol{r}_{fc}$. It should be noted that the bounding $\boldsymbol{r}_{fc}$ creates a non-linear relationship and makes optimization a non-linear problem; however, bounds that are defined for tangential components can be implemented in a linear fashion. The geometric interpretation of the friction cone (red solid line) and pyramid (blue solid line) is provided in Figure 9.

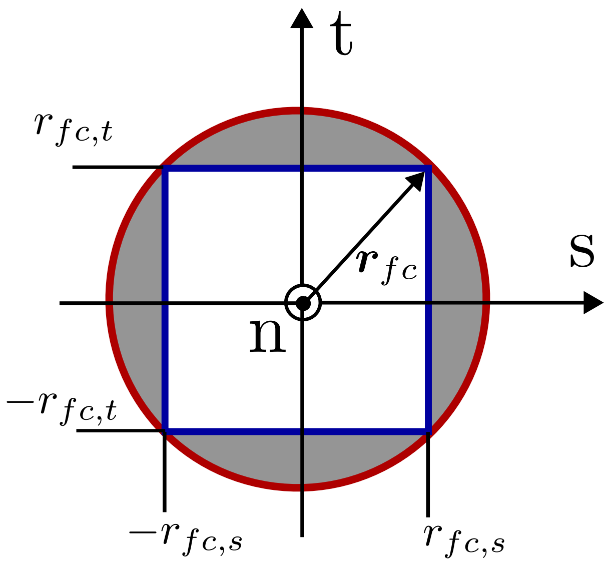


Figure 9. Geometric interpretation of the friction pyramid.

Based on the linear and conservative bounds, the following linear constraints are defined for the tangential forces.

$$\begin{aligned} -0.707\mu_B f_{c,n} \ \leq \ {}_B f_{c,t} \ \leq \ 0.707\mu_B f_{c,n} \\ -0.707\mu_B f_{c,n} \ \leq \ {}_B f_{c,s} \ \leq \ 0.707\mu_B f_{c,n} \\ -2\times 0.707\mu_B f_{c,n} \ \leq \ {}_B f_{c,t} + {}_B f_{c,s} \ \leq \ 2\times 0.707\mu_B f_{c,n} \end{aligned} \tag{54}$$

The relationship given in Equation (54) is written compactly as provided in Equation (55), where ${}_B\boldsymbol{F}_{C,k}$ is the vector containing the decision variables. ${}_B\boldsymbol{n}$, ${}_B\boldsymbol{t}$ and ${}_B\boldsymbol{s}$ are the unit vectors attached on the contact points and defined in $\mathcal{F}_B$. As a practical note, calculating these unit vectors with respect to $\mathcal{F}_B$ is a straightforward calculation when there are passive joints at the ankle of the contact legs. Depending on the kinematic structure of the leg, certain unit vectors can be assumed to be in the same direction with the axes of the body frame.

$$\begin{aligned} \left(-2\times 0.707\mu_B \boldsymbol{n}^T - {}_B\boldsymbol{t}^T - {}_B\boldsymbol{s}^T\right){}_B\boldsymbol{F}_{C,k} \ \leq \ \boldsymbol{0} \\ \left(2\times 0.707\mu_B \boldsymbol{n}^T + {}_B\boldsymbol{t}^T + {}_B\boldsymbol{s}^T\right){}_B\boldsymbol{F}_{C,k} \ \leq \ \boldsymbol{0} \end{aligned} \tag{55}$$

Within the current work, ${}_B\boldsymbol{n}$, ${}_B\boldsymbol{t}$ and ${}_B\boldsymbol{s}$ are assumed to be constant and defined as ${}_B\boldsymbol{n}^T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$, ${}_B\boldsymbol{t} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ and ${}_B\boldsymbol{s} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

5.3.4. Cooperative Force Optimization: Cost Function

Force phase optimization tries to keep the commanded motion intact while minimizing the disturbance injected on the system due to the joint accelerations. The joint accelerations affect each agent due to the $\mathcal{C}_{coop}$, which is provided in Equation (26). The constraints that are introduced in this chapter previously maintained contact, and force cone constraints are satisfied. The cost function for this optimization is defined in Equation (56).

$$\begin{aligned} &\mathcal{L}_c(\boldsymbol{x}_{h,k}, \boldsymbol{F}_{C,k}, \boldsymbol{W}_{h,k}) \ = \\ &\left(-\boldsymbol{u}_{B,xz} + \left[\begin{array}{ccc|c|c} 1 & 0 & 0 & \boldsymbol{0} & \boldsymbol{0} \\ 0 & 0 & 1 & \boldsymbol{0} & \boldsymbol{0} \end{array}\right]\boldsymbol{x}_{B,k}\right)^T \boldsymbol{Q}_T \left(-\boldsymbol{u}_{B,xz} + \left[\begin{array}{ccc|c|c} 1 & 0 & 0 & \boldsymbol{0} & \boldsymbol{0} \\ 0 & 0 & 1 & \boldsymbol{0} & \boldsymbol{0} \end{array}\right]\boldsymbol{x}_{B,k}\right) + \\ &\boldsymbol{x}_{B,k}^T \boldsymbol{II}_Q^T \boldsymbol{Q}_W \boldsymbol{II}_Q \boldsymbol{x}_{B,k} \end{aligned} \tag{56}$$

where $\boldsymbol{II}_Q$ is a matrix that selects the states that are the joint velocities of contact and swing legs and approximate the acceleration of these selected states. The joint velocities within states $\boldsymbol{x}_{h,k}$ are denoted as $\boldsymbol{q}_{h,k}$ for $h = C, S$, and the selection of these states is defined in Equation (57).

$$\begin{aligned} \dot{\boldsymbol{q}}_{h,k} \ &= \left[\begin{array}{c|c|ccc} \boldsymbol{0} & \boldsymbol{0} & 1 & 0 & 0 \\ \boldsymbol{0} & \boldsymbol{0} & 0 & 1 & 0 \\ \boldsymbol{0} & \boldsymbol{0} & 0 & 0 & 1 \end{array}\right]\boldsymbol{x}_{h,k} \\ \dot{\boldsymbol{q}}_{h,k} \ &= \boldsymbol{II}_q \boldsymbol{x}_{h,k} \end{aligned} \tag{57}$$

The approximation for acceleration is given below, where states are assumed to propagate with the first-order Euler method.

$$\ddot{\boldsymbol{q}}_{h,k} \ = \ \boldsymbol{II}_q \frac{(\boldsymbol{x}_{h,k+1} - \boldsymbol{x}_{h,k})}{\Delta t} \tag{58}$$

Finally, accelerations $\ddot{\boldsymbol{q}}_{h,k}$ for $h = C, S$ for $k = 1, \cdots, N_n$ is given as provided in Equation (59)

$$\ddot{q}_{h,k} = \begin{bmatrix} -\Delta t \boldsymbol{II}_q & \Delta t \boldsymbol{II}_q & & \\ & -\Delta t \boldsymbol{II}_q & \Delta t \boldsymbol{II}_q & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} x_{h,1} \\ x_{h,2} \\ \vdots \\ x_{h,N_n} \end{bmatrix} \tag{59}$$

$$\ddot{q}_{h,k} = \boldsymbol{II}_Q x_{h,K}$$

6. Preliminary Results for the ASLB Platform

This section presents the simulation of the algorithm for the proposed method. The simulation is not executed in a physics environment, and trajectories illustrated in this section are estimated trajectories only. Solutions are obtained on `Intel(R) Core i7-4720HQ CPU @2.60 GHz 16GB RAM PC` with MATLAB 2019b software.

The sampling time Δt for the discrete model is selected to be $\Delta t = 0.05$s. Nominal values for N_n and N_p are selected as $N_n = 20$ and $N_p = 4$, respectively. $\boldsymbol{Q}_s$ and $\boldsymbol{Q}_p$ for contact phase optimization is selected to be $\boldsymbol{Q}_s = 5$ and $\boldsymbol{Q}_p = 2$. $\boldsymbol{Q}_0$, $\boldsymbol{Q}_f$, and $\boldsymbol{Q}_t$ for swing phase optimization is selected to be $\boldsymbol{Q}_0 = 12.5$, $\boldsymbol{Q}_f = 12.5$, and $\boldsymbol{Q}_t = 30$. Finally, $\boldsymbol{Q}_u$ and $\boldsymbol{Q}_w$ for cooperative force optimization are selected to be $\boldsymbol{Q}_u = 100$ and $\boldsymbol{Q}_w = 5$.

Based on these settings, and from the initial conditions of $\boldsymbol{u}_B(t=0) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, $\boldsymbol{r}_2(t=0) = \begin{bmatrix} 0.01 & -0.344 & 0.063 \end{bmatrix}^T$, and $\boldsymbol{r}_3(t=0) = \begin{bmatrix} 0.045 & -0.344 & -0.239 \end{bmatrix}^T$, ASLB is asked to move forward by $\boldsymbol{u}_{B,ref} = \begin{bmatrix} 0.1 & 0 & 0 \end{bmatrix}^T$. The results are provided in the following figures.

Figure 10 illustrates the calculated swing and contact trajectories. N_n for the first optimization is minimized to $N_n = 6$ from a nominal 20 as the contact initial position for ${}^{p=1}\boldsymbol{r}_{c,1}$ is already provided by the sensor information. For a feasible finite horizon, $N_n = 6$, and contact leg, which is the right leg or **Agent 2**, is used to calculate an approximate trajectory for $\boldsymbol{r}_c$ starting from $\boldsymbol{r}_2(t=0)$ and diverge from the CoM toward the (+) x and (+) z directions. Note that this trajectory is calculated with respect to $\mathcal{F}_B$. The swing phase optimization connects $\boldsymbol{r}_3(t=0)$ to the ${}^{p=2}\boldsymbol{r}_{c,1}$ without violating the kinematic bounds.

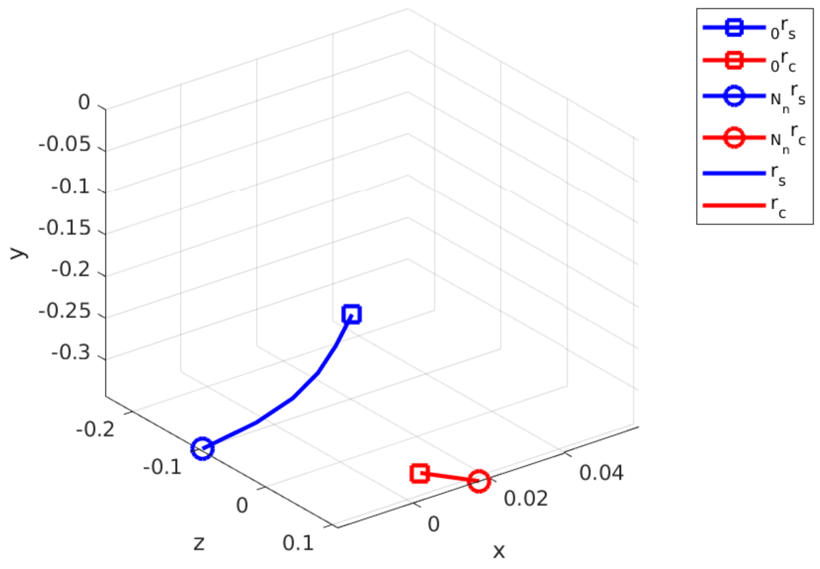


Figure 10. Calculated swing and contact leg trajectories for first step.

Figure 11 illustrates the nominal force trajectory for the contact leg as it interacts with the ground. ${}_0\boldsymbol{F}_C$ is the initial trajectory that is calculated by substituting $\boldsymbol{q}_{a,i}$ and $\boldsymbol{x}_i$ into the agent dynamics. ${}_{opt}\boldsymbol{F}_C$ is the resultant contact force trajectory. It is visible from the figure that there is not a significant change in the y-axis. However, ${}_{opt}\boldsymbol{F}_C$ is shifted in the (+) x direction by approximately 0.4 N.

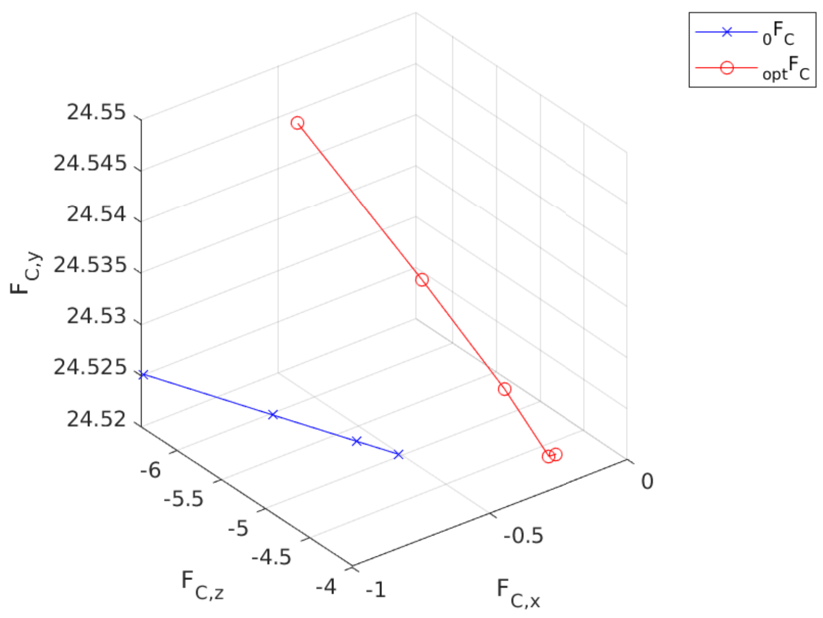


Figure 11. Calculated force trajectory for the first step.

Figure 12 shows the contact and swing trajectories for the second step, where N_n is minimized to $N_n = 16$. Similar to the first step, ${}^{p=1}\boldsymbol{r}_{c,1}$ for contact phase optimization is the initial position of the current contact leg, which is **Agent 3**. $\boldsymbol{r}_c$ in Figure 12 converges to the CoM in the beginning of this phase and then pushes away toward the (-) x and (-) z directions.

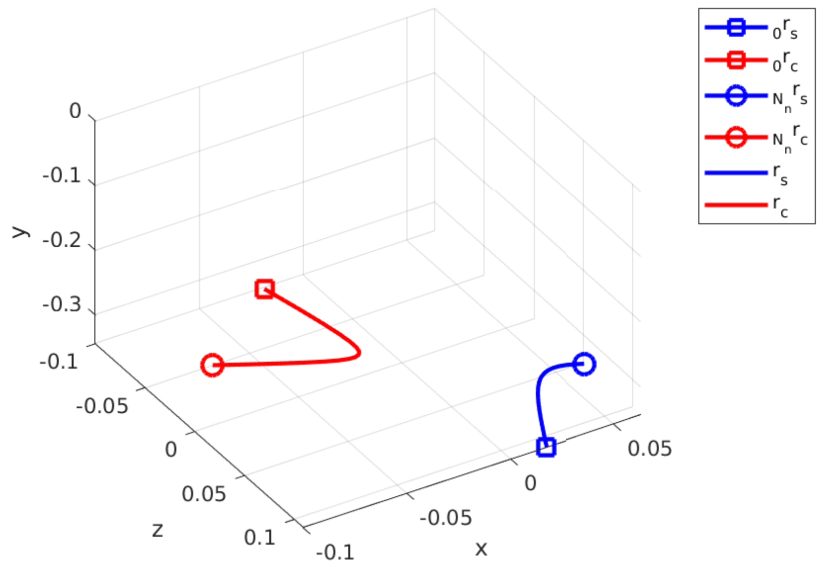


Figure 12. Calculated swing and contact leg trajectories for the second step.

A smaller correction occurs in the contact force as ${}_0\boldsymbol{F}_C$ and ${}_{opt}\boldsymbol{F}_C$ have slight differences in all three directions as seen in Figure 13.

Figure 14 provides the $\boldsymbol{u}_B$ trajectory throughout the walking simulation along with the $\boldsymbol{r}_{c,i}$ for $i = 2, 3$. This figure illustrates the characteristic difference in the method, which calculates the contact positions and forces them to track a reference velocity $\boldsymbol{u}_{B,ref}$. The black circle in Figure 14 indicates the CoM and decision variables are defined with respect to that. The trajectory of $\boldsymbol{u}_B$, which is given with the blue line, settles in a cyclic pattern. The mean of the magnitude in x direction is approximately 0.28 m/s, while it is almost zero for z direction.

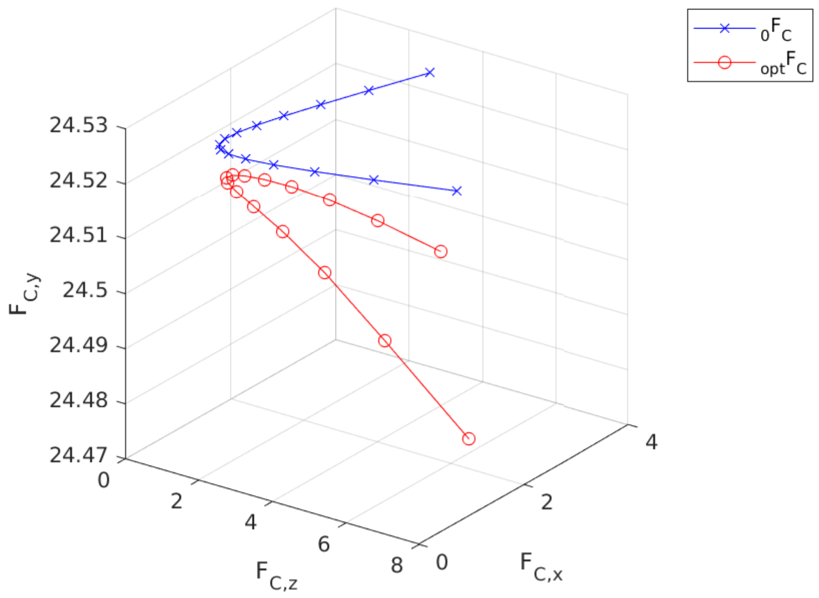


Figure 13. Calculated force trajectory for the second step.

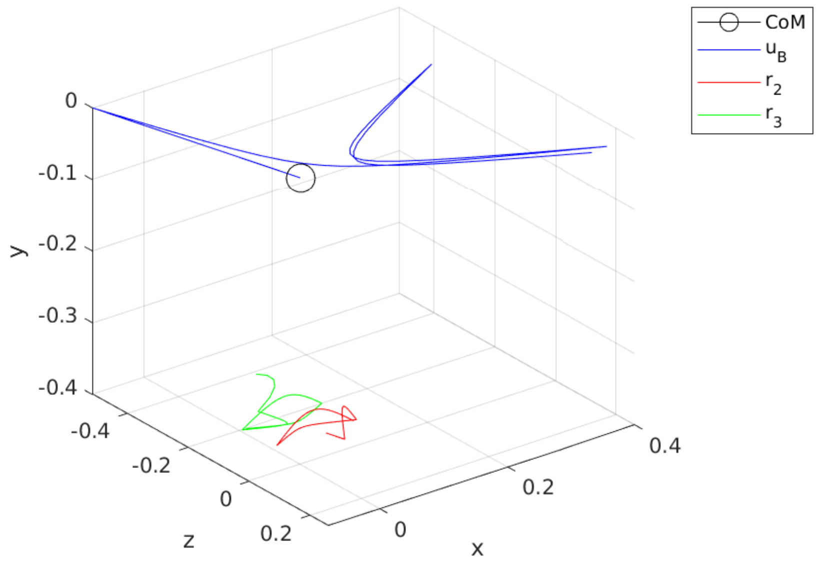


Figure 14. u_B trajectory plot along with leg motion with respect to CoM.

Figure 15 provides a more intuitive illustration of the motion of the robot as ${}_I\boldsymbol{r}_B$ is calculated from $\boldsymbol{u}_B$. ${}_I\boldsymbol{r}_B$ gives the position of the origin of the $\mathcal{F}_B$ with respect to the $\mathcal{F}_I$. The trajectory of ${}_I\boldsymbol{r}_B$ reveals that body drifts away from the line $z = 0$ while achieving forward motion, as desired. The drift is due to the first contact leg, which is the **Agent 2**, and cannot be corrected.

With the proposed method, a velocity command could be tracked with some offset. During the simulation of the algorithm, it was observed that relaxation of the state bounds decreases the offsets between velocity commands and actual velocity. In addition, since this method is designed for velocity level dynamics, a drift in the walking does occur, which can be mitigated by an appropriate control scheme. That said, extracting the dynamics from the kinematics and developing a controller for this simpler set equations allows faster calculation of the future steps.

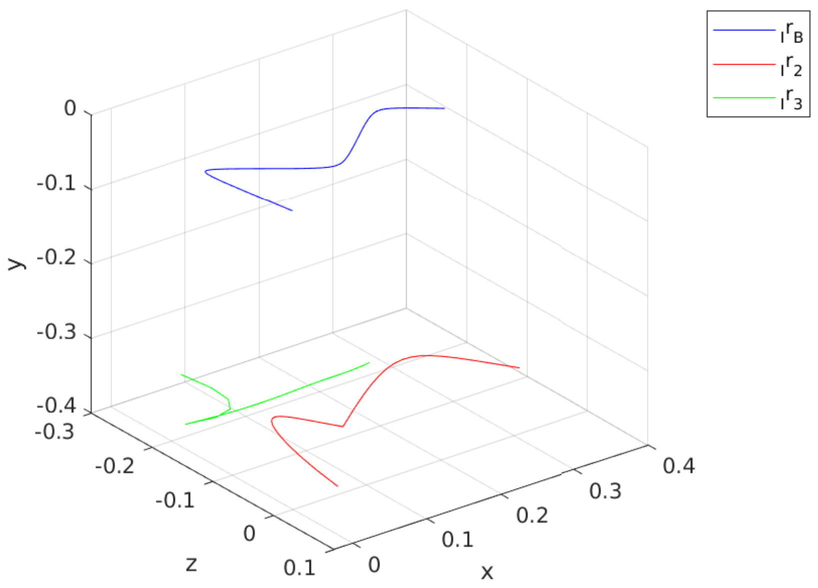


Figure 15. r_B plot along with leg motion with respect to $\mathcal{F}_I$.

7. Cooperative UAV-UGV Docking with Task Prioritization

Motivated by the results from the preceding section, we extend the framework to study systems that are not on the same platform for example a cooperative UAV-UGV system similar to that in [39], with the objective to have the UAV (quadcopter) dock on the moving UGV (rover). Note that this framework further illustrates that it can accommodate very different dynamical systems, unlike the ASLB system, where the left and right leg dynamics were identical.

Table 2 provides a summary of the key variables and terms used for the development of the ASLB dynamical model.

Table 2. Summary of key variables used for description of the quadcopter and rover kinematics and dynamics.

$\mathcal{F}_B$	body fixed reference frame (quadcopter)
m_q	mass of the quadcopter
$\mathbf{J}_q$	moment of inertia of the quadcopter
$\boldsymbol{x}_q$	state vector of quadcopter
${}_I\boldsymbol{C}_q(\boldsymbol{\theta}_q) = {}_I\boldsymbol{C}_q$	rotation matrix for quadcopter
f_q	total thrust generated by motors in the body frame
$\boldsymbol{t}_q$	moments generated on the body
$\boldsymbol{u}_q$	input vector for the quadcopter model
g	acceleration due to gravity
$[\boldsymbol{u}_1\ \boldsymbol{u}_2\ \boldsymbol{u}_3]^T$	unit vectors representing the body frame
m_r	mass of the rover
$\boldsymbol{J}_r$	moment of inertia of the rover
$\boldsymbol{x}_r$	state vector for rover dynamics
${}_I\boldsymbol{C}_r$	rotation matrix of the rover
$\boldsymbol{u}_r$	inputs to the rover

7.1. Quadcopter Dynamics

The six-degree-of-freedom (DoF) rigid body dynamics with mass (m_q) and moment of inertia (J_q) of the quadcopter are represented in (60). The model's states are denoted as $\boldsymbol{x}_q = \begin{bmatrix} \boldsymbol{p}_q^T & \boldsymbol{v}_q^T & \boldsymbol{\theta}_q^T & \boldsymbol{w}_q^T \end{bmatrix}^T \in \mathbb{R}^{12}$, where $\boldsymbol{p}_q = {}_I\boldsymbol{p}_q \in \mathbb{R}^3$, $\boldsymbol{v}_q = {}_I\boldsymbol{v}_q \in \mathbb{R}^3$, $\boldsymbol{\theta}_q \in \mathbb{R}^3$ and $\boldsymbol{w}_q = {}_B\boldsymbol{w}_q \in \mathbb{R}^3$ are inertial position, inertial velocity, Euler angles, and angular velocity, respectively. Given the Euler angles, the rotation matrix for the quadcopter is denoted as

${}_I\boldsymbol{C}_q(\boldsymbol{\theta}_q) = {}_I\boldsymbol{C}_q$. The inputs of the system are denoted as $\boldsymbol{u}_q = \begin{bmatrix} f_q & \boldsymbol{t}_q^T \end{bmatrix}^T \in \mathbb{R}^4$, where f_q is the total thrust generated by the motors on $\mathcal{F}_B$ and $\boldsymbol{t}_q \in \mathbb{R}^3$ is the column matrix of moments generated on the body defined in $\mathcal{F}_B$. $\boldsymbol{E}_q(\boldsymbol{\theta}_q)$ is the mapping between $\boldsymbol{w}_q$ and $\dot{\boldsymbol{\theta}}_q$ such that $\boldsymbol{w}_q = \boldsymbol{E}_q(\boldsymbol{\theta}_q)\dot{\boldsymbol{\theta}}_q$ for pre-defined rotation sequence. The quadcopter properties are taken from the work [45].

$$\begin{aligned}\dot{\boldsymbol{x}}_q &= \begin{bmatrix} \dot{\boldsymbol{p}}_q \\ \dot{\boldsymbol{v}}_q \\ \dot{\boldsymbol{e}}_q \\ \dot{\boldsymbol{w}}_q \end{bmatrix} = \boldsymbol{f}(\boldsymbol{x}_q, \boldsymbol{u}_q) \\ &= \begin{bmatrix} \boldsymbol{v}_q \\ g\boldsymbol{u}_3 - (1/m_q)\,{}_I\boldsymbol{C}_q f_q \boldsymbol{u}_3 \\ \boldsymbol{E}^{-1}(\boldsymbol{\theta}_q)\boldsymbol{w}_q \\ \boldsymbol{J}_q^{-1}(-\boldsymbol{w}_q \times \boldsymbol{J}_q\boldsymbol{w}_q + \boldsymbol{t}_q) \end{bmatrix}\end{aligned} \tag{60}$$

7.2. Rover Dynamics

The dynamics of the rover are calculated with mass m_r and moment of inertia $\boldsymbol{J}_r$, assuming that it runs on a flat surface and provided in (61). Based on this assumption, model states are denoted as $\boldsymbol{x}_r = \begin{bmatrix} p_{r,x}\ p_{r,y}\ \Psi_r\ v_{r,x}\ w_{r,z} \end{bmatrix}^T$, where $p_{r,x}$ and $p_{r,y}$ represent inertial position on the $x - y$ plane, Ψ_r is the Euler angle about the z axis, ${}_Bv_{r,x} = v_{r,x}$ is the horizontal velocity of the rover, and ${}_Bw_{r,z} = w_{r,z}$ is the angular velocity of the rover. The composite rotation matrix for the rover is denoted as ${}_I\boldsymbol{C}_r = {}_I\boldsymbol{C}_r(\Psi_r)$. Inputs to the system are denoted as $\boldsymbol{u}_r = [f_{r,1}\ f_{r,2}]^T$, which represent the traction forces applied on the surface by a set of wheels on the right- and left-hand sides of the rover, respectively. m_r is taken as 1 kg, and $\boldsymbol{J}_r = diag\{0.1\}$ kg·m^2.

$$\begin{aligned}\dot{\boldsymbol{x}}_r &= \begin{bmatrix} \dot{p}_{r,x} \\ \dot{p}_{r,y} \\ \dot{\Psi}_r \\ \dot{v}_{r,x} \\ \dot{w}_{r,z} \end{bmatrix} = \boldsymbol{f}(\boldsymbol{x}_r, \boldsymbol{u}_r) \\ &= \begin{bmatrix} \boldsymbol{u}_1^T\,{}_I\boldsymbol{C}_r v_{r,x}\boldsymbol{u}_1 \\ \boldsymbol{u}_2^T\,{}_I\boldsymbol{C}_r v_{r,x}\boldsymbol{u}_1 \\ w_{r,z} \\ (1/m_r)(f_{r,1} + f_{r,2}) \\ \boldsymbol{J}^{-1}(\boldsymbol{r}_1 \times f_{r,1} + \boldsymbol{r}_2 \times f_{r,2}) \end{bmatrix}\end{aligned} \tag{61}$$

8. Cooperative Model-Predictive Control Methodology

This section introduces a unified MPC strategy to maintain a docking approach for the long range and a finer docking maneuver in the short range by accommodating a non-linear and a linear MPC designed with edge weight information and task prioritization. This section is divided into four subsections, where non-linear MPC (NMPC), linear MPC (LMPC), cooperative task prioritization, and implementation of the control strategy are described in Sections 8.1, 8.2, 8.3, and 8.4, respectively.

The introduced method can be applied on all of the agents as the formulation only considers local neighbor information; therefore, formulations will be provided for the agent denoted as i, which is defined by the states $\boldsymbol{x}_i(t) \in \mathbb{R}^{n_x}$, inputs $\boldsymbol{u}_i(t) \in \mathbb{R}^{n_u}$, and outputs $\boldsymbol{y}_i(t) \in \mathbb{R}^{n_y}$. Given the states and inputs, the non-linear dynamics of agent i are given in the following form:

$$\begin{aligned}\dot{\boldsymbol{x}}_i(t) &= \boldsymbol{f}_i(\boldsymbol{x}_i(t),\ \boldsymbol{u}_i(t)) \\ \boldsymbol{y}_i &= \boldsymbol{h}_i(\boldsymbol{x}_i(t),\ \boldsymbol{u}_i(t))\end{aligned} \tag{62}$$

Equation (63) provides the discrete linear representation of the agents' dynamics. This is obtained by linearizing the current state and input, which is denoted as $(\boldsymbol{x}_{i,c},\, \boldsymbol{u}_{i,c})$ and then by converting the continuous time system in a discrete system using Euler discretization (see [46]).

$$\begin{aligned} \boldsymbol{\Delta x}_{i,k+1} &= \boldsymbol{A}_i \boldsymbol{\Delta x}_{i,k} + \boldsymbol{B}_i \boldsymbol{\Delta u}_{i,k} \\ \boldsymbol{\Delta y}_{i,k} &= \boldsymbol{C}_i \boldsymbol{\Delta x}_{i,k} + \boldsymbol{D}_i \boldsymbol{\Delta u}_{i,k} \end{aligned} \tag{63}$$

where k is the current sample such that $\boldsymbol{\Delta x}_{i,k} = \boldsymbol{x}_{i,k} - \boldsymbol{x}_{i,c}$, and $\boldsymbol{\Delta u}_{i,k} = \boldsymbol{u}_{i,k} - \boldsymbol{u}_{i,c}$. In this work, we assumed that full state information is shared among the agents. Therefore, matrices $\boldsymbol{C}_i$ and $\boldsymbol{D}_i$ are assumed to be $\boldsymbol{I}$ and $\boldsymbol{0}$, respectively, with compatible sizes.

8.1. Non-linear MPC Formulation

Equation (64) represents the non-linear problem that runs in NMPC where $\mathcal{L}_i(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k},\, \boldsymbol{x}_{i,ref})$ is the objective function, and $\mathcal{C}_{eq}(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k})$ and $\mathcal{C}_{ineq}(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k})$ are equality and inequality constraints, respectively. M in (64) denotes the prediction horizon for the NMPC.

$$\begin{aligned} \underset{\boldsymbol{x}_{i,k},\boldsymbol{u}_{i,k}}{\text{minimize}} \quad & \sum_{k=1}^{M} \mathcal{L}_i(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k},\, \boldsymbol{x}_{i,ref}) \\ \text{s. t.} \quad & \dot{\boldsymbol{x}}_i(t) - \boldsymbol{f}_i(\boldsymbol{x}_i(t),\, \boldsymbol{u}_i(t)) = \boldsymbol{0} \\ & \mathcal{C}_{eq}(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k}) = \boldsymbol{0} \\ & \mathcal{C}_{ineq}(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k}) \leq \boldsymbol{0} \end{aligned} \tag{64}$$

The objective function implemented in this work has the following quadratic form.

$$\begin{aligned} & \mathcal{L}_i(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k},\, \boldsymbol{x}_{i,ref}) = \\ & \sum_{k=1}^{M} \mathcal{L}_{i,tp}(\boldsymbol{x}_{i,ref},\, \boldsymbol{x}_{i,k}) + \boldsymbol{u}_{i,k}^T \boldsymbol{R} \boldsymbol{u}_{i,k} + \boldsymbol{x}_{i,k}^T \boldsymbol{Q} \boldsymbol{x}_{i,k} \end{aligned} \tag{65}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}$ are square matrices. This objective function drives the system to achieve the cooperative task defined by $\mathcal{L}_{i,tp}(\boldsymbol{x}_{i,ref},\, \boldsymbol{x}_{i,k})$, which brings a nuanced cooperativeness with task prioritization to the tracking and is explained in Section 8.3.

The equality and inequality constraints serve the following purposes. $\mathcal{C}_{eq}(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k})$ is introduced to ensure the initial conditions, which are $\boldsymbol{x}_{i,c} = \boldsymbol{x}_i(t=0)$ and $\boldsymbol{u}_{i,c} = \boldsymbol{u}_i(t=0)$, and $\mathcal{C}_{ineq}(\boldsymbol{x}_{i,k},\, \boldsymbol{u}_{i,k})$ is introduced to enforce states and inputs to stay in the predefined ranges. These constraints are provided in [45].

8.2. Linear MPC Formulation

The LMPC method implemented in this work is the implicit LMPC applied in [26,45], and details can be found therein. Therefore, the method is summarized here for completeness. The quadratic problem for the LMPC is provided in (66).

$$\begin{aligned} \underset{\boldsymbol{u}_{i,K}}{\text{minimize}} \quad & \overline{\mathcal{L}}_i(\boldsymbol{x}_{i,K},\, \boldsymbol{\Delta u}_{i,K},\, \boldsymbol{x}_{i,ref}) \\ & \overline{\mathcal{C}}_{eq}(\boldsymbol{x}_{i,K},\, \boldsymbol{u}_{i,K}) = \boldsymbol{0} \\ & \overline{\mathcal{C}}_{ineq}(\boldsymbol{x}_{i,K},\, \boldsymbol{u}_{i,K}) \leq \boldsymbol{0} \end{aligned} \tag{66}$$

The quadratic objective function is denoted as $\overline{\mathcal{L}}_i(\boldsymbol{x}_{i,K},\, \boldsymbol{\Delta u}_{i,K},\, \boldsymbol{x}_{i,ref})$ and given in (67), where the decision variable is $\boldsymbol{\Delta u}_{i,K}$ and $\boldsymbol{x}_{i,K}$ is a function of $\boldsymbol{\Delta u}_{i,K}$.

$$\begin{aligned} & \overline{\mathcal{L}}_i(\boldsymbol{x}_{i,K},\, \boldsymbol{\Delta u}_{i,K},\, \boldsymbol{x}_{i,ref}) = \\ & \overline{\mathcal{L}}_{i,tp}(\boldsymbol{x}_{i,ref},\, \boldsymbol{x}_{i,K}) + \boldsymbol{u}_{i,K}^T \overline{\boldsymbol{R}} \boldsymbol{u}_{i,K} + \boldsymbol{x}_{i,K}^T \overline{\boldsymbol{Q}} \boldsymbol{x}_{i,K} \end{aligned} \tag{67}$$

where $\overline{\boldsymbol{R}} = diag\{\boldsymbol{R}\}_1^M$, $\overline{\boldsymbol{Q}} = diag\{\boldsymbol{Q}\}_1^M$ and $\overline{\mathcal{L}}_{i,tp}(\boldsymbol{x}_{i,ref},\, \boldsymbol{x}_{i,K})$ is calculated in Section 8.3.

Due to the implicit formulation, state predictions are described with respect to the decision variables and combined in a lumped representation. Similarly, decision variables, which are the inputs, are also collected under a lumped term. These lumped states and inputs are denoted as $\boldsymbol{\Delta x}_{i,K}$ and $\boldsymbol{\Delta u}_{i,K}$ and provided in (68).

$$\begin{aligned} \boldsymbol{\Delta x}_{i,K} &= \begin{bmatrix} \boldsymbol{\Delta x}_{i,k} \\ \vdots \\ \boldsymbol{\Delta x}_{i,k+M-1} \end{bmatrix} \\ \boldsymbol{\Delta u}_{i,K} &= \begin{bmatrix} \boldsymbol{\Delta u}_{i,k} \\ \vdots \\ \boldsymbol{\Delta u}_{i,k+M-1} \end{bmatrix} \end{aligned} \tag{68}$$

The relationship between $\boldsymbol{\Delta x}_{i,K}$ and $\boldsymbol{\Delta u}_{i,K}$ is provided below with matrices $\boldsymbol{F}_i \in \mathbb{R}^{Mn_x \times n_x}$ and $\boldsymbol{H}_i \in \mathbb{R}^{Mn_x \times Mn_u}$ for the prediction horizon of M.

$$\begin{aligned} \boldsymbol{\Delta x}_{i,K} &= \boldsymbol{F}_i \boldsymbol{x}_{i,c} + \boldsymbol{H}_i \boldsymbol{\Delta u}_{i,K} \\ \boldsymbol{F}_i &= col\{\boldsymbol{F}_{i,m}\}_{m=1}^{M} \\ \boldsymbol{F}_{i,m} &= \boldsymbol{A}_i^{(m-1)} && , m = [1, \cdots, M] \\ \boldsymbol{H}_i &= col\{\boldsymbol{H}_{i,m}\}_{m=1}^{M} \\ \boldsymbol{H}_{i,m} &= \boldsymbol{0} && m = 1 \\ \boldsymbol{H}_{i,m} &= row\{\boldsymbol{h}_{i,l}\} && m = [2, \cdots, M] \\ \boldsymbol{h}_{i,l} &= \boldsymbol{A}_i^{(m-l-1)} \boldsymbol{B}_i && , l = [1, \cdots, m-1] \end{aligned} \tag{69}$$

Terminal states are given as a function of $\boldsymbol{x}_{i,k}$ and $\boldsymbol{\Delta u}_{i,K}$ below.

$$\begin{aligned} \boldsymbol{\Delta x}_{i,k+M} &= \boldsymbol{A}_i^M \boldsymbol{x}_{i,k} + \overline{\boldsymbol{B}}_i \boldsymbol{\Delta u}_{i,K} \\ \overline{\boldsymbol{B}}_i &= \begin{bmatrix} \boldsymbol{A}_i^{M-1} \boldsymbol{B}_i & \cdots & \boldsymbol{A}_i^1 \boldsymbol{B}_i & \boldsymbol{B}_i \end{bmatrix} \end{aligned} \tag{70}$$

$\overline{\mathcal{C}}_{eq}(\boldsymbol{x}_{i,K}, \boldsymbol{u}_{i,K})$ and $\overline{\mathcal{C}}_{ineq}(\boldsymbol{x}_{i,K}, \boldsymbol{u}_{i,K})$ serve the same purpose with $\mathcal{C}_{eq}(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k})$ and $\mathcal{C}_{ineq}(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k})$; however, they are modified to comply with LMPC notation. The calculations of these constraints are provided in [26].

8.3. Cooperative Task Prioritization

Let $\boldsymbol{\epsilon}_{i,t} \in \mathbb{R}^{n_{\epsilon t}}$, as provided in (71), represent a cooperative task for agent i based on local neighbor state information that is received from neighboring agents, denoted as j. Assume that $\boldsymbol{\epsilon}_{i,t} \in \mathbb{R}^{n_{\epsilon t}}$ is defined for a subset of states denoted as $\hat{\boldsymbol{x}}_i \subset \boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_j \subset \boldsymbol{x}_j$, where $\hat{\boldsymbol{x}}_i \in \mathbb{R}^{n_s}$ and $\hat{\boldsymbol{x}}_j \in \mathbb{R}^{n_s}$. It should be noted that $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$ are assumed to be measured with respect to the same coordinate frame. Otherwise, the task $\boldsymbol{\epsilon}_{i,t}$ becomes non-linear.

$$\boldsymbol{\epsilon}_{i,t} = a_{ij}(\hat{\boldsymbol{x}}_j - \hat{\boldsymbol{x}}_i), \ \ t = 1, \cdots, T \tag{71}$$

where T is the maximum number of tasks. The first derivative of $\boldsymbol{\epsilon}_{i,t}$ is calculated as follows:

$$\begin{aligned} \dot{\boldsymbol{\epsilon}}_{i,t} &= a_{ij}(\dot{\hat{\boldsymbol{x}}}_j - \dot{\hat{\boldsymbol{x}}}_i) \\ \dot{\boldsymbol{\epsilon}}_{i,t} &= \begin{bmatrix} a_{ij}\boldsymbol{I}_{n_s} & -a_{ij}\boldsymbol{I}_{n_s} \end{bmatrix} \begin{bmatrix} \dot{\hat{\boldsymbol{x}}}_j \\ \dot{\hat{\boldsymbol{x}}}_i \end{bmatrix} \end{aligned} \tag{72}$$

Collecting the terms on the right-hand side $\boldsymbol{M}_i(a_{ij}) = \begin{bmatrix} a_{ij}\boldsymbol{I}_{n_s} & -a_{ij}\boldsymbol{I}_{n_s} \end{bmatrix}$ and $\dot{\boldsymbol{S}}_{i,t} = \begin{bmatrix} \dot{\hat{\boldsymbol{x}}}_j^T & \dot{\hat{\boldsymbol{x}}}_i^T \end{bmatrix}^T \in \mathbb{R}^{n_{st}}$, the mapping in (72) between task space (TS) velocities and local state space (LSS) velocities takes the form

$$\dot{\boldsymbol{\epsilon}}_{i,t} = \boldsymbol{M}_i(a_{ij})\dot{\boldsymbol{S}}_{i,t} \tag{73}$$

Following that, the inverse mapping from TS to LSS is calculated as

$$\dot{\boldsymbol{S}}_{i,t} = \boldsymbol{M}_i^{+}\dot{\boldsymbol{\epsilon}}_{i,t} \tag{74}$$

where $\boldsymbol{M}_i^{+}$ is the pseudo inverse of $\boldsymbol{M}_{i,t}$ and a_{ij} is dropped from the expression for brevity. Note that $\boldsymbol{x}_j$ is the state of a neighboring agent.

As illustrated in [47] for robots, excessive (redundant) LSS can be utilized to manage secondary tasks. Let $\boldsymbol{\epsilon}_{i,1}$ and $\boldsymbol{S}_{i,1}$ denote the first TS and LSS and $\boldsymbol{\epsilon}_{i,t}$, and let $\boldsymbol{S}_{i,t}$, $t = 2, \cdots, T$ denotes the remaining ones; then, following task management methodology can be used [40]:

$$\begin{aligned} \dot{\boldsymbol{S}}_{i,1} &= \boldsymbol{M}_{i,1}^{+}\dot{\boldsymbol{\epsilon}}_1 \\ \dot{\boldsymbol{S}}_{i,t} &= \dot{\boldsymbol{S}}_{i,t-1} + (\boldsymbol{M}_{i,t}\boldsymbol{\Phi}_{i,t-1})^{+}(\dot{\boldsymbol{\epsilon}}_{i,t} - \boldsymbol{M}_{i,t}\dot{\boldsymbol{S}}_{i,t-1}) \end{aligned} \tag{75}$$

where $\boldsymbol{\Phi}_{i,t}$ is the null space projection matrix and calculated is as follows:

$$\begin{aligned} \boldsymbol{\Phi}_{i,1} &= \boldsymbol{I}_{n_{st}} - \boldsymbol{M}_{i,1}^{+}\boldsymbol{M}_{i,1} \\ \boldsymbol{\Phi}_{i,t} &= \boldsymbol{\Phi}_{i,t-1} - (\boldsymbol{M}_{i,t}\boldsymbol{\Phi}_{i,t-1})^{+}(\boldsymbol{M}_{i,t}\boldsymbol{\Phi}_{i,t-1}) \end{aligned} \tag{76}$$

The formulation defined in (75) is collected in a minimal representation as in (77).

$$\dot{\boldsymbol{S}} = \boldsymbol{\Psi}\dot{\boldsymbol{\epsilon}} \tag{77}$$

where $\dot{\boldsymbol{S}}_i = \begin{bmatrix} \dot{\boldsymbol{S}}_{i,1}^T & \dot{\boldsymbol{S}}_{i,t}^T & \cdots & \dot{\boldsymbol{S}}_{i,T}^T \end{bmatrix}^T$ and $\dot{\boldsymbol{\epsilon}}_i = \begin{bmatrix} \dot{\boldsymbol{\epsilon}}_{i,1}^T & \dot{\boldsymbol{\epsilon}}_{i,t}^T & \cdots & \dot{\boldsymbol{\epsilon}}_{i,T}^T \end{bmatrix}^T$. Thus, a quadratic function can be written as in (78) based on (77).

$$L(\dot{\boldsymbol{\epsilon}}_i) := \dot{\boldsymbol{S}}^T\dot{\boldsymbol{S}} = \dot{\boldsymbol{\epsilon}}_i^T\boldsymbol{\Psi}^T\boldsymbol{\Psi}\dot{\boldsymbol{\epsilon}}_i \tag{78}$$

Finally, based on (78), $\mathcal{L}_{i,tp}(\boldsymbol{x}_{i,ref}, \boldsymbol{x}_{i,k})$ and $\overline{\mathcal{L}}_{i,tp}(\boldsymbol{x}_{i,ref}, \boldsymbol{x}_{i,K})$ are calculated as given in (79) and (83), respectively, where $\boldsymbol{Q}_s^T = \boldsymbol{Q}_s \geq \boldsymbol{0} \in \mathbb{R}^{n_{et}\times n_{et}}$.

$$\begin{aligned} \mathcal{L}_{i,tp}(\boldsymbol{x}_{i,ref}, \boldsymbol{x}_{i,k}) &= \dot{\boldsymbol{\epsilon}}_i^T\boldsymbol{\Psi}^T\left[\begin{array}{c|c} \boldsymbol{Q}_s & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{Q}_s \end{array}\right]\boldsymbol{\Psi}\dot{\boldsymbol{\epsilon}}_i \\ &= \dot{\boldsymbol{\epsilon}}_i^T\boldsymbol{Q}_\epsilon\dot{\boldsymbol{\epsilon}}_i \end{aligned} \tag{79}$$

Remark 1. *If states of the agents i and j are linearly related, then $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$ are linearly related; therefore $\boldsymbol{M}_{i,1}$ is constant. As a result,*

- *$L(\dot{\boldsymbol{\epsilon}}_i)$ outputs a quadratic function of $\dot{\boldsymbol{\epsilon}}_{i,t}$ with scalars γ_t such that $L(\dot{\boldsymbol{\epsilon}}_i) = \sum_{t=1}^{T}\gamma_t\dot{\boldsymbol{\epsilon}}_{i,t}^2$, where $\gamma_1 \geq \cdots \geq \gamma_t \geq \cdots \geq \gamma_T$.*
- *The relationship given for derivatives of state space and task space in (78) is also valid for the state space and task space as follows:*

$$L(\boldsymbol{\epsilon}_i) := \boldsymbol{S}^T\boldsymbol{S} = \boldsymbol{\epsilon}_i^T\boldsymbol{\Psi}^T\boldsymbol{\Psi}\boldsymbol{\epsilon}_i \tag{80}$$

which results in

$$\begin{aligned} \mathcal{L}_{i,tp}(\boldsymbol{x}_{i,ref}, \boldsymbol{x}_{i,k}) &= \boldsymbol{\epsilon}_i^T\boldsymbol{\Psi}^T\left[\begin{array}{c|c} \boldsymbol{Q}_s & \boldsymbol{0} \\ \hline \boldsymbol{0} & \boldsymbol{Q}_s \end{array}\right]\boldsymbol{\Psi}\boldsymbol{\epsilon}_i \\ &= \boldsymbol{\epsilon}_i^T\boldsymbol{Q}_\epsilon\boldsymbol{\epsilon}_i \end{aligned} \tag{81}$$

Let $\boldsymbol{Q}_\epsilon$ be partitioned as in (82), and let $\boldsymbol{X}_{i,ref} := \overline{\boldsymbol{1}} \otimes \boldsymbol{x}_{i,ref}$, where $\overline{\boldsymbol{1}} = [1 \ \cdots \ 1]^T \in \mathbb{R}^M$. Then, the overall task matrix $\overline{\boldsymbol{\epsilon}}_i$ can be written as an affine function of $\boldsymbol{X}_{i,ref}$, and $\boldsymbol{x}_{i,K})$ and $\overline{\mathcal{L}}_{i,tp}(\boldsymbol{x}_{i,ref}, \boldsymbol{x}_{i,K})$ is calculated as provided in (83).

$$\boldsymbol{Q}_{\epsilon} = \left[\begin{array}{c|c} \boldsymbol{Q}_{\epsilon,11} & \boldsymbol{Q}_{\epsilon,12} \\ \hline \boldsymbol{Q}_{\epsilon,21} & \boldsymbol{Q}_{\epsilon,22} \end{array}\right] \tag{82}$$

$$\overline{\mathcal{L}}_{i,tp}(\boldsymbol{x}_{i,ref}, \boldsymbol{x}_{i,K}) = \overline{\boldsymbol{\epsilon}}_i^T \overline{\boldsymbol{Q}}_{\epsilon} \overline{\boldsymbol{\epsilon}}_i \tag{83}$$

where $\overline{\boldsymbol{Q}}_{\epsilon,\star*} = diag\{\boldsymbol{Q}_{\epsilon,\star*}\}_1^M$ and $\star, * = \{1,2\}$.

$$\overline{\boldsymbol{Q}}_{\epsilon} = \left[\begin{array}{c|c} \overline{\boldsymbol{Q}}_{\epsilon,11} & \overline{\boldsymbol{Q}}_{\epsilon,12} \\ \hline \overline{\boldsymbol{Q}}_{\epsilon,21} & \overline{\boldsymbol{Q}}_{\epsilon,22} \end{array}\right] \tag{84}$$

8.4. Implementation of the MPCs

The docking controller is designed to have several layers to decide how to perform the docking maneuver based on the distance between the agent i and the neighboring agents. The layers, as mentioned earlier, are the selector NMPC and LMPC. Essentially, the selector works as a governor and decides the reference trajectory that the LMPC tracks. If the absolute distance between agent i and neighboring agents is greater than a threshold, the NMPC generates a trajectory toward the neighboring agents using neighboring state information. Note that NMPC does not aim to achieve the terminal state constraint, but it repeatedly minimizes the state error within the given prediction horizon. Then, the generated trajectory, which creeps toward the neighbors, is passed to LMPC. If the distance is smaller than the threshold, LMPC receives the neighboring state information to generate control inputs without needing NMPC. The distance and threshold are denoted as $\boldsymbol{e} = \|[\boldsymbol{p}_j] - [\boldsymbol{p}_i]\|_2^2 \in \mathbb{R}$ and $e_d \in \mathbb{R}$. The currently available state and input information are denoted as $\boldsymbol{x}_i(t-1)$, $\boldsymbol{x}_j(t-1)$ and $\boldsymbol{u}_i(t-1)$, where t represents current sample and $(t-1)$ represents previous sample. The pipeline for the control strategy described above is given in Figure 16.

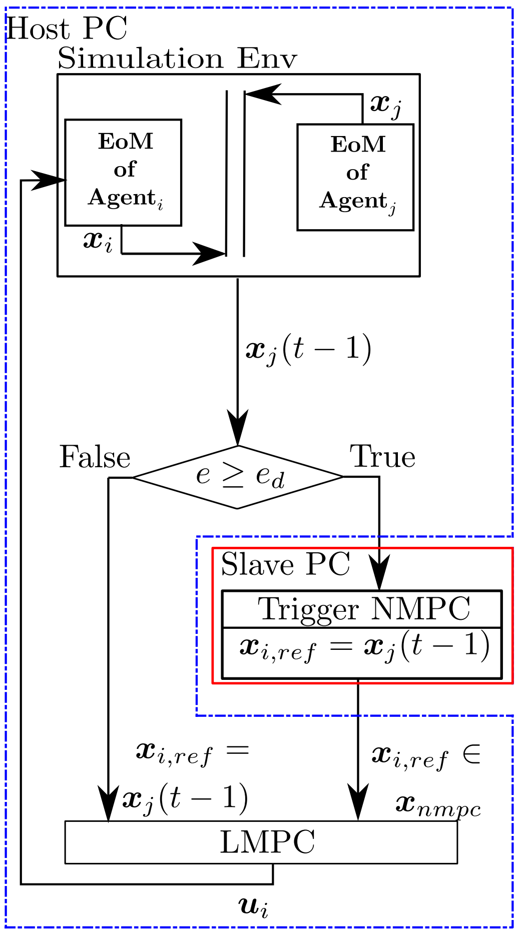


Figure 16. Unified MPC framework for docking.

When triggered, NMPC generates a rough trajectory toward agent j for a given prediction horizon M based on the formulation given in (64). Then, a portion of this trajectory that is determined by the control horizon M_u is extracted, and this is denoted as $\boldsymbol{x}_{nmpc}$. The optimization parameters for NMPC such as M, t_f, and M_u are selected empirically to achieve a rough trajectory and allow a time interval to queue a new trajectory to guide the agent i to the vicinity of other agents. The selection of the optimization parameters with the right-hand-side sparsity template described in [46] provides a longer time to queue a new trajectory. Before the LMPC receives the output of the NMPC or the neighboring state information, i.e., $\boldsymbol{x}_{nmpc}$ ir $\boldsymbol{x}_j(t-1)$, respectively, a line or a set of line segments are populated based on the prediction horizon of the LMPC. Finally, the tracking based on the optimization provided in (66) is performed by the LMPC.

9. Cooperative MPC Simulation and Results

This section presents the simulation results for the proposed strategy based on the simulation setup of a quadcopter and a rover, as illustrated in Figure 17.

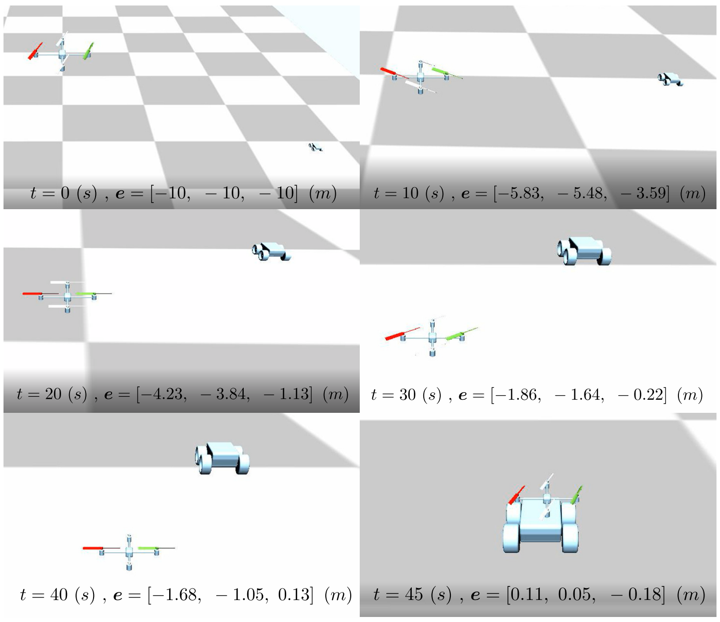

$t = 0\ (s)\ ,\ \boldsymbol{e} = [-10,\ \ -10,\ \ -10]\ \ (m)$
$t = 10\ (s)\ ,\ \boldsymbol{e} = [-5.83,\ \ -5.48,\ \ -3.59]\ \ (m)$
$t = 20\ (s)\ ,\ \boldsymbol{e} = [-4.23,\ \ -3.84,\ \ -1.13]\ \ (m)$
$t = 30\ (s)\ ,\ \boldsymbol{e} = [-1.86,\ \ -1.64,\ \ -0.22]\ \ (m)$
$t = 40\ (s)\ ,\ \boldsymbol{e} = [-1.68,\ \ -1.05,\ \ 0.13]\ \ (m)$
$t = 45\ (s)\ ,\ \boldsymbol{e} = [0.11,\ \ 0.05,\ \ -0.18]\ \ (m)$

Figure 17. Snaps of the realized trajectory during long range docking maneuver (see [39]).

The NMPC and LMPC parameters are provided in Figure 18, and the quadcopter properties are given in [45] in Table 1. Two scenarios are illustrated—(1) proximity docking on a moving agent and (2) large range docking on a moving agent—where LMPC and NMPC-LMPC strategies are tested, respectively. Let $\mathcal{P}_s = \{1, 2, 3, 4\}$ represent the priority of states, where 1 is the highest priority and 4 is the lowest priority, and $s = p, v, e, w$ represents the subset of states that $\mathcal{P}$ is representing. On this basis, LMPC uses the priority map of $\{\mathcal{P}_p, \mathcal{P}_v, \mathcal{P}_e, \mathcal{P}_w\} = \{2, 3, 1, 4\}$. Priority mapping is used to arrange tasks defined as $\boldsymbol{\epsilon}_{i,t}$, where $i \in \{1, 2, 3, 4\}$. This mapping is valid for both LMPC and NMPC.

Property	Value
M	14
M_u	1
Q	$3\boldsymbol{I}$
R	$3\boldsymbol{I}$
Q_s	Solution to Discrete Algebraic Ricatti Equation*

(a)

Property	Value
M	10
M_u	3
t_f	1 (s)
Q	$5\boldsymbol{I}$
R	$1\boldsymbol{I}$
Q_s	$40\boldsymbol{I}$
E_d	2 (m)

(b)

Figure 18. (**a**) LMPC parameters. (**b**) NMPC parameters. t_f is the relevant time for prediction horizon. * represents the discrete algebraic Ricatti solution; see [39,48].

9.1. Case Study 1: Proximity Docking on the Rover

The initial conditions for the quadcopter and the rover are $\boldsymbol{x}_{q,0} = [0\ 0.5\ -10\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ and $\boldsymbol{x}_{q,0} = [0\ 0\ 0\ 0.5\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$, respectively.

The initial inputs for the quadcopter are $\boldsymbol{u}_{i,c} = \left[9.81m_q\ 0\ 0\ 0\right]^T$ to temper the behavior of the $\boldsymbol{f}_q$. Given the initial values, the quadcopter is guided to the rover with -0.1 m offset in the z direction, as illustrated in Figure 19. As shown in Figure 19, the quadcopter makes a free fall in $3/4^{th}$ of a second, and then it recovers. During this phase, it loses altitude and gains speed in the w component up to 6 m/s. The duration of the free fall is correlated to $\boldsymbol{R}$ in the cost function and inequality constraints provided to the LMPC, which is $\overline{C}_{ineq}(\boldsymbol{x}_{i,K},\ \boldsymbol{u}_{i,K})$. After the free fall, after approximately $t = 1$ (s), the controller makes a correction in $\boldsymbol{e}$ and starts approaching the rover in the x- and y-directions.

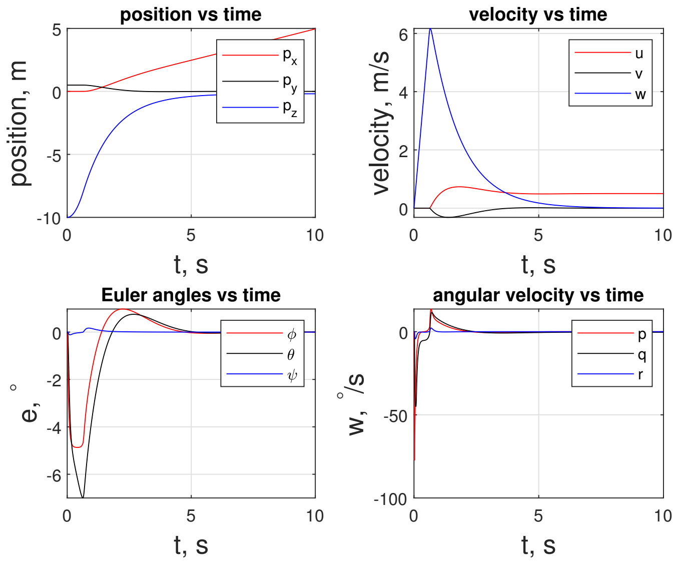


Figure 19. State trajectory of the agent of the quadcopter.

As mentioned above, free fall can also be seen in Figure 20 in the f_q subplot, where f_q stays at 0 N for that period. During the maneuver, f_q is upper-bounded by 12 N. The maximums for $\boldsymbol{t}_q$ are reached in the x-direction, where -0.52 and 0.05 Nm of toque are observed at the extremes.

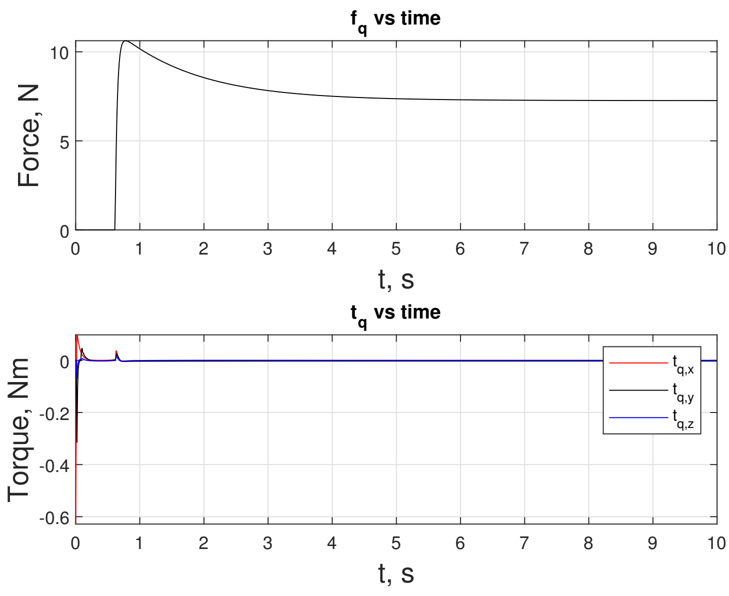


Figure 20. Inputs f_q and $\boldsymbol{t}_q$ over the duration of the docking maneuver.

Figure 21 provides an overview of the performed trajectory along with the heading direction of the quadcopter, which is aligned with the $\boldsymbol{u}_1$ axis of $\mathcal{F}_B$.

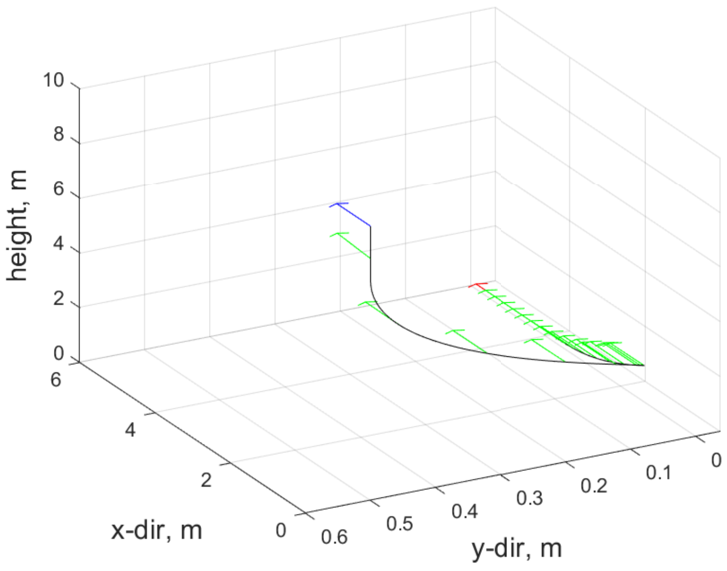


Figure 21. Trajectory and heading of the quadcopter.

9.2. Case Study 2: Long-Range Docking on the Rover

The initial conditions for the quadcopter and the rover are $\boldsymbol{x}_{q,0} = [0\ 0\ -10\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ and $\boldsymbol{x}_{r,0} = [10\ 10\ 0\ 0.5\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$, respectively.

The initial inputs for the quadcopter are $\boldsymbol{u}_{i,c} = [9.81m_q\ 0\ 0\ 0]^T$. This case illustrates the long-range capability of the proposed control strategy. The overview of the scenario is provided as a set snapshot in Figure 17, and the video of the performed scenario can be found in https://www.youtube.com/watch?v=QJ-NuwZQ8zw (accessed on 12 May 2024).

In this scenario, LMPC tracks the $\boldsymbol{x}_{nmpc}$ that the NMPC generates until $\boldsymbol{e}$ is less than $\boldsymbol{e}_d$. At about 41 s, a transition happens, and LMPC starts tracking the $\boldsymbol{x}_j(t-1)$. Figure 22 shows the states of the quadcopter. An important feature of this motion is that the signal oscillates at a velocity level in both the linear and angular motion. The same oscillatory behavior is visible for inputs f_q and $\boldsymbol{t}_q$, as provided in Figure 23, which is because the quadcopter's current states are changing as the $\boldsymbol{x}_{nmpc}$ is generated on the NMPC side, sent to the LMPC, and populated as line segments. Figures 22 and 24 reveal the difference in proximity and long-range docking maneuvers, to which LMPC is more sensitive and can be seen from the magnitude of the tracking error $\boldsymbol{e}$.

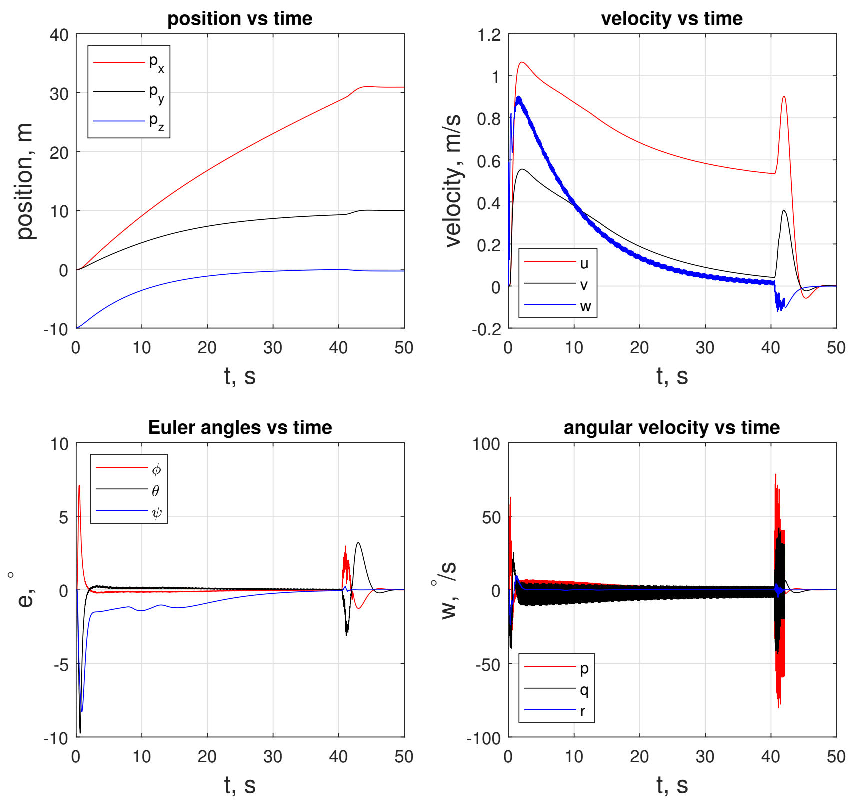


Figure 22. State trajectory of the agent of the quadcopter.

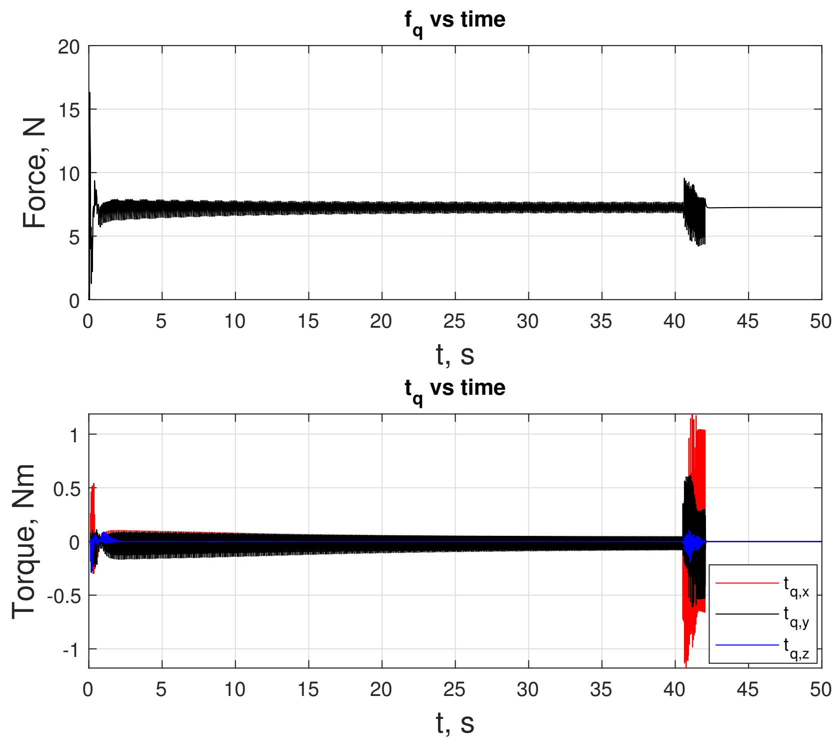


Figure 23. Inputs f_q and $\boldsymbol{t}_q$ over the duration of the docking maneuver.

The positional errors between the rover and the quadcopter in Figure 24 illustrate that the positional errors in the $x-$ and $y-$directions approach zero, while the error in the $z-$direction approach -0.1 m.

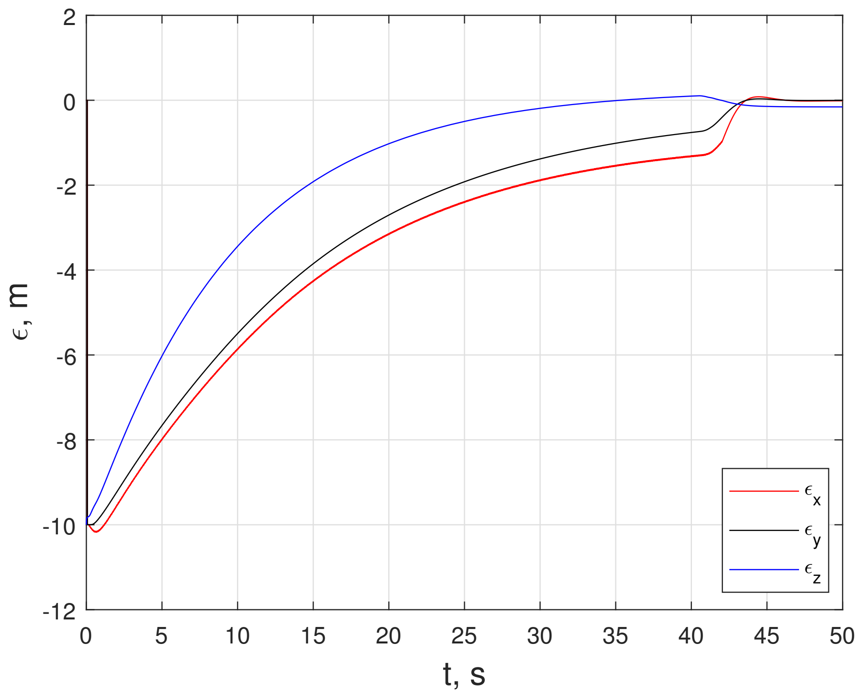


Figure 24. Position error between the rover and the quadcopter.

Figure 25 illustrates the calculation time of each NMPC trigger. Initially, the generation costs approximately 0.1 s, and as the quadcopter settles on the trajectory trajectory generation, the times decrease to 0.05 s. For a given elapsed time profile, the overall calculation time is calculated to be 13.91 s. The upper bound line illustrates the control horizon M_u for the NMPC; see [39]).

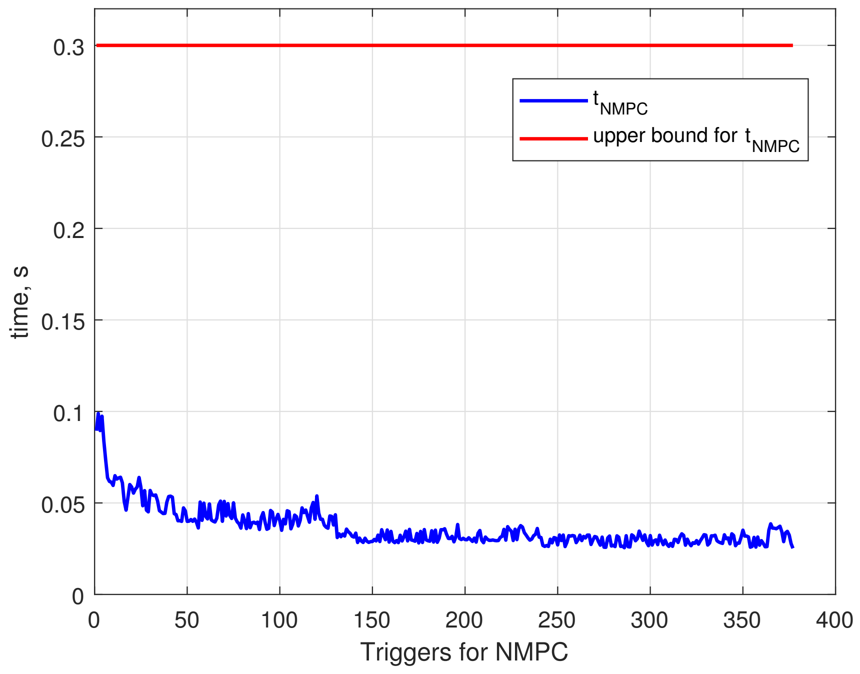


Figure 25. Calculation time for the NMPC problem at every instant.

10. Discussion

The *central premise* of this paper is to show that a complex dynamical system and/or a control system can be decomposed into parts. These parts can be constrained to develop the governing equations to study the dynamics or to synthesize a control strategy for the composite system using the the interconnection between the sub-systems as constraints. Note thatwhile these are two separate applications, they can be studied using the same framework. In the biped dynamics, the legs (left and right) and the torso are considered as sub-systems, which are coupled via constraints and thus modeled using the cooperative system framework. The trajectory for the biped robot is computed using a sequence of optimizations. Given a desired amount of forward movement of the center of mass, one would begin from the contact phase optimization, the results of which then feed into the swing phase optimization, hwich provides the necessary smooth trajectories for moving to the new location. This is finally passed on to the cooperative force optimization, which computes the necessary generalized forces to accomplish the movement.

While there are promising early results from this approach, further study is warranted to investigate the sensitivity of the optimization approaches to parameters such as the number of phases, the number of sample instants, the sampling period, the contact force models (friction cone), and the asymmetry in the legs, sensor noise, and other unmodeled terms in the dynamics.

Similarly, in the case of a cooperative docking scenario, the quadcopter and ground rover are two separate sub-systems of the overall system and are constrained to work together in the cooperative setup. In this context, the framework also allows one to choose different control schemes and combine them together for the overall cooperative mission. For example, a nonlinear MPC was chosen to bring the quadcopter closer to the ground rover faster because we wish for a faster approach to the goal; however, when one is close to the goal, a linear MPC controller is activated since when it is closer to the desired state, the linear terms typically dominate higher-order polynomial terms and/or harmonic functions.

In the cooperative task decomposition for the UAV-UGV problem, one would have to further study the impact of communication delays and environmental disturbances such as unstructured terrain models to fully understand the efficacy of the proposed cooperative control formulation.

11. Summary and Conclusions

This paper presents a cooperative modeling framework that is shown to reduce the complexity of deriving the governing dynamical equations of complex systems composed of multiple bodies. The approach also allows for an optimization-based trajectory generation for the complex system.

The results show that such cooperative modeling enables online trajectory generation through a series of optimizations that generate approximate trajectories and pass them to a final optimization that refines the motion with the cooperative system framework.

The paper also proposes a unified MPC control strategy capable of handling long-range to proximity docking maneuvers. A rough but fast NMPC method is run to propagate the agent to a closer vicinity, and then the LMPC handles more sensitive proximity motion. The sensitivity of the LMPC is apparent in Figures 22 and 24, where a smooth approach suddenly becomes relatively violent. Based on the simulations, the selection of the matrices $\boldsymbol{Q}$, $\boldsymbol{R}$ and $\boldsymbol{Q}_s$ makes the transition from NMPC-LMPC to LMPC smoother. Note that the docking of the quadcopter is almost tangential to the $x - y$ plane, and during the final section of the maneuver, it penetrates the docking platform. The latter problem can be resolved with an approach constraint to the MPC controllers. Most importantly, the proposed method is readily applicable among dynamic systems, including non-linear systems, since the controller's structure stays the same. Using local neighbor information instead of external sensors or observers naturally makes this control strategy decentralized.

Author Contributions: Conceptualization, B.T. and K.S.; methodology, B.T. and K.S.; software, B.T.; formal analysis, B.T. and K.S.; writing—original draft presentation, B.T.; investigation, B.T.; writing—review and editing, K.S., supervision, K.S.; project administration, K.S.; funding acquisition, K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Office of Naval Research via award number N00014-18-1-2215.

Data Availability Statement: All data was synthetically generated using the numerical simulation models presented in this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mistry, M.; Nakanishi, J.; Cheng, G.; Schaal, S. Inverse kinematics with floating base and constraints for full body humanoid robot control. In Proceedings of the Humanoids 2008—8th IEEE-RAS International Conference on Humanoid Robots, Daejeon, Republic of Korea, 1–3 December 2008; pp. 22–27. [CrossRef]
2. Winkler, A.W.; Bellicoso, C.D.; Hutter, M.; Buchli, J. Gait and Trajectory Optimization for Legged Systems through Phase-Based End-Effector Parameterization. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1560–1567. [CrossRef]
3. Zhou, Z.; Zhao, Y. Accelerated ADMM based Trajectory Optimization for Legged Locomotion with Coupled Rigid Body Dynamics. In Proceedings of the 2020 American Control Conference (ACC), Denver, CO, USA, 1–3 July 2020; pp. 5082–5089.
4. Yamamoto, K.; Kamioka, T.; Sugihara, T. Survey on model-based biped motion control for humanoid robots. *Adv. Robot.* **2020**, *34*, 1353–1369. [CrossRef]
5. Pardo, D.; Möller, L.; Neunert, M.; Winkler, A.W.; Buchli, J. Evaluating Direct Transcription and Nonlinear Optimization Methods for Robot Motion Planning. *IEEE Robot. Autom. Lett.* **2016**, *1*, 946–953. [CrossRef]
6. Mastalli, C.; Budhiraja, R.; Merkt, W.; Saurel, G.; Hammoud, B.; Naveau, M.; Carpentier, J.; Vijayakumar, S.; Mansard, N. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 2536–2542.
7. Budhiraja, R.; Carpentier, J.; Mastalli, C.; Mansard, N. Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics. In Proceedings of the 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), Beijing, China, 6–9 November 2018; pp. 1–9. [CrossRef]
8. Herzog, A.; Schaal, S.; Righetti, L. Structured contact force optimization for kino-dynamic motion generation. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016; pp. 2703–2710.
9. Winkler, A.W.; Mastalli, C.; Havoutis, I.; Focchi, M.; Caldwell, D.G.; Semini, C. Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 5148–5154. [CrossRef]
10. Budhiraja, R.; Carpentier, J.; Mansard, N. Dynamics Consensus between Centroidal and Whole-Body Models for Locomotion of Legged Robots. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6727–6733. [CrossRef]
11. Lewis, F.L.; Zhang, H.; Hengster-Movric, K.; Das, A. Algebraic Graph Theory and Cooperative Control Consensus. In *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*; Springer, London, UK, 2013; Chapter 2, pp. 23–70. [CrossRef]
12. Stanoev, A.; Filiposka, S.; In, V.; Kocarev, L. Cooperative method for wireless sensor network localization. *Ad Hoc Netw.* **2016**, *40*, 61–72. [CrossRef]
13. Yan, Z.; Gu, G.; Zhao, K.; Wang, Q.; Li, G.; Nie, X.; Yang, H.; Du, S. Integer Linear Programming Based Topology Design for GNSSs With Inter-Satellite Links. *IEEE Wirel. Commun. Lett.* **2021**, *10*, 286–290. [CrossRef]
14. van Horssen, E.P.; Weiland, S. Synthesis of Distributed Robust H-Infinity Controllers for Interconnected Discrete Time Systems. *IEEE Trans. Control Netw. Syst.* **2016**, *3*, 286–295. [CrossRef]
15. Efaz, E.T.; Mowlee, M.M.; Jabin, J.; Khan, I.; Islam, M.R. Modeling of a high-speed and cost-effective FPV quadcopter for surveillance. In Proceedings of the 2020 23rd International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 19–21 December 2020; IEEE: Piscataway, NJ, USA, 2020. [CrossRef]
16. Andersson, O.; Doherty, P.; Lager, M.; Lindh, J.O.; Persson, L.; Topp, E.A.; Tordenlid, J.; Wahlberg, B. WARA-PS: A research arena for public safety demonstrations and autonomous collaborative rescue robotics experimentation. *Auton. Intell. Syst.* **2021**, *1*, 9. [CrossRef]
17. Saha, H.; Basu, S.; Auddy, S.; Dey, R.; Nandy, A.; Pal, D.; Roy, N.; Jasu, S.; Saha, A.; Chattopadhyay, S.; et al. A low cost fully autonomous GPS (Global Positioning System) based quad copter for disaster management. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]

18. Guo, D.; Leang, K.K. Image-Based Estimation, Planning, and Control of a Cable-Suspended Payload for Package Delivery. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2698–2705. [CrossRef]
19. Estrada, C.; Sun, L. Trajectory Tracking Control of a Drone-Guided Hose System for Fluid Delivery. In Proceedings of the AIAA Scitech 2021 Forum, Virtual, 11–21 January 2021; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2021. [CrossRef]
20. Nguyen, T.M.; Qiu, Z.; Cao, M.; Nguyen, T.H.; Xie, L. An Integrated Localization-Navigation Scheme for Distance-Based Docking of UAVs. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
21. Caruso, B.; Fatakdawala, M.; Patil, A.; Chen, G.; Wilde, M. Demonstration of In-Flight Docking Between Quadcopters and Fixed-Wing UAV. In Proceedings of the 2021 IEEE Aerospace Conference (50100), Big Sky, MT, USA, 6–13 March 2021; IEEE: Piscataway, NJ, USA, 2021. [CrossRef]
22. Löbl, D.; Holzapfel, F.; Weiss, M.; Shima, T. Cooperative Docking Guidance and Control with Application to Civil Autonomous Aerial Refueling. *J. Guid. Control Dyn.* **2021**, *44*, 1638–1648. [CrossRef]
23. Thanh, H.L.N.N.; Phi, N.N.; Hong, S.K. Simple nonlinear control of quadcopter for collision avoidance based on geometric approach in static environment. *Int. J. Adv. Robot. Syst.* **2018**, *15*, 172988141876757. [CrossRef]
24. DeVries, L.D.; Paley, D.A. Wake Estimation and Optimal Control for Autonomous Aircraft in Formation Flight. In Proceedings of the AIAA Guidance, Navigation, and Control (GNC) Conference, Boston, MA, USA, 19–22 August 2013; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2013. [CrossRef]
25. Matus-Vargas, A.; Rodriguez-Gomez, G.; Martinez-Carranza, J. Ground effect on rotorcraft unmanned aerial vehicles: A review. *Intell. Serv. Robot.* **2021**, *14*, 99–118. [CrossRef]
26. Ru, P.; Subbarao, K. Nonlinear Model Predictive Control for Unmanned Aerial Vehicles. *Aerospace* **2017**, *4*, 31. [CrossRef]
27. Falanga, D.; Foehn, P.; Lu, P.; Scaramuzza, D. PAMPC: Perception-Aware Model Predictive Control for Quadrotors. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
28. Weiss, A.; Baldwin, M.; Erwin, R.S.; Kolmanovsky, I. Model Predictive Control for Spacecraft Rendezvous and Docking: Strategies for Handling Constraints and Case Studies. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 1638–1647. [CrossRef]
29. Stesina, F. Tracking Model Predictive Control for Docking Maneuvers of a CubeSat with a Big Spacecraft. *Aerospace* **2021**, *8*, 197. [CrossRef]
30. Dong, K.; Luo, J.; Dang, Z.; Wei, L. Tube-based robust output feedback model predictive control for autonomous rendezvous and docking with a tumbling target. *Adv. Space Res.* **2020**, *65*, 1158–1181. [CrossRef]
31. Araar, O.; Aouf, N.; Vitanov, I. Vision Based Autonomous Landing of Multirotor UAV on Moving Platform. *J. Intell. Robot. Syst.* **2016**, *85*, 369–384. [CrossRef]
32. Wenzel, K.E.; Masselli, A.; Zell, A. Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle. *J. Intell. Robot. Syst.* **2010**, *61*, 221–238. [CrossRef]
33. Muskardin, T.; Balmer, G.; Persson, L.; Wlach, S.; Laiacker, M.; Ollero, A.; Kondak, K. A novel landing system to increase payload capacity and operational availability of high altitude long endurance UAV. In Proceedings of the 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 7–10 June 2016; IEEE: Piscataway, NJ, USA, 2016. [CrossRef]
34. Persson, L.; Muskardin, T.; Wahlberg, B. Cooperative rendezvous of ground vehicle and aerial vehicle using model predictive control. In Proceedings of the 2017 IEEE 56th Annual Conference on Decision and Control (CDC), Melbourne, VIC, Australia, 12–15 December 2017; IEEE: Piscataway, NJ, USA, 2017. [CrossRef]
35. Persson, L.; Wahlberg, B. Model Predictive Control for Autonomous Ship Landing in a Search and Rescue Scenario. In Proceedings of the AIAA Scitech 2019 Forum, San Diego, CA, USA, 7–11 January 2019; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2019. [CrossRef]
36. Reynolds, C.W. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Comput. Graph.* **1987**, *21*, 25–34. [CrossRef]
37. Zhang, S.; Zhang, H.; Di, B.; Song, L. Cellular Cooperative Unmanned Aerial Vehicle Networks with Sense-and-Send Protocol. *IEEE Internet Things J.* **2019**, *6*, 1754–1767. [CrossRef]
38. Yao, F.; Li, J.; Chen, Y.; Chu, X.; Zhao, B. Task allocation strategies for cooperative task planning of multi-autonomous satellite constellation. *Adv. Space Res.* **2019**, *63*, 1073–1084. [CrossRef]
39. Taner, B.; Subbarao, K., Model Predictive Control for Cooperative Systems with Task Prioritization applied to Vehicle Rendezvous and Docking. In Proceedings of the AIAA SCITECH 2023 Forum, National Harbor, MD, USA, & Online, 23–27 January 2023. [CrossRef]
40. Baerlocher, P.; Boulic, R. Task-priority formulations for the kinematic control of highly redundant articulated structures. In Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190), Victoria, BC, Canada, 17 October 1998; IEEE: Piscataway, NJ, USA, 1998. [CrossRef]
41. Anderson, T.; Chang, C.Y.; Martínez, S. Distributed approximate Newton algorithms and weight design for constrained optimization. *Automatica* **2019**, *109*, 108538. [CrossRef]
42. Lynch, K.M.; Park, F.C. *Modern Robotics: Mechanics, Planning, and Control*, 1st ed.; Cambridge University Press: New York, NY, USA, 2017.

43. Khadiv, M.; Herzog, A.; Moosavian, S.A.A.; Righetti, L. Walking Control Based on Step Timing Adaptation. *IEEE Trans. Robot.* **2020**, *36*, 629–643. [CrossRef]
44. Daneshmand, E.; Khadiv, M.; Grimminger, F.; Righetti, L. Variable Horizon MPC with Swing Foot Dynamics for Bipedal Walking Control. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2349–2356. [CrossRef]
45. Bhattacharjee, D.; Subbarao, K. Robust Control Strategy for Quadcopters using Sliding Mode Control and Model Predictive Control. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020; American Institute of Aeronautics and Astronautics: Reston, VA, USA, 2020. [CrossRef]
46. Betts, J.T. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2010. [CrossRef]
47. Uzunoğlu, E.; Tatlicioğlu, E.; Dede, M.İ.C. A Multi-Priority Controller for Industrial Macro-Micro Manipulation. *Robotica* **2020**, *39*, 217–232. [CrossRef]
48. Ferrante, A.; Ntogramatzidis, L. The generalised discrete algebraic Riccati equation in linear-quadratic optimal control. *Automatica* **2013**, *49*, 471–478. [CrossRef]