Distributed Receding Horizon Control of Multiagent Systems

Thesis by

William B. Dunbar

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2004
(Defended April 6, 2004)

Acknowledgements

I would like to thank my advisor and committee chair, Prof. Richard Murray, for his guidance, funding and enthusiasm for research. Thanks to Prof. Jerry Marsden for inviting me to come to Caltech, for being on my committee and for being the patriarch of the CDS department. Prof. Jeff Shamma was a great help in completing the work in this dissertation, and I thank him for his friendship and being on my committee. Also, thanks to Prof. Jason Hickey for reading group discussions and being on my committee.

Thanks very much to the other graduate students, postdocs and visiting professors in the CDS department who have helped me to mature as a researcher. In particular: Dr. Mark Milam, for demanding quality and providing CDS with NTG; Prof. Nicolas Petit, for also demanding quality and inviting me to Ecole des Mines; Dr. Reza Olfati Saber, for teaching me aspects of his research and for our collaborations. I especially thank the crew of my past office mates, fellow system administrators, and the department staff for making the administrative life of every CDS student easy.

I would not be here if it wasn't for my family, and I would not be able to stay if it weren't for my beautiful wife, Rebekah. This work is dedicated to my wife and the glorious memories I have of Mike Finch.

Abstract

Multiagent systems arise in several domains of engineering. Examples include arrays of mobile sensor networks for aggregate imagery, autonomous highways, and formations of unmanned aerial vehicles. In these contexts, agents are governed by vehicle dynamics and often constraints, and the control objective is achieved by cooperation. Cooperation refers to the agreement of the agents to 1) have a common objective with neighboring agents, with the objective typically decided offline, and 2) share information online to realize the objective. To be viable, the control approach for multiagent systems should be distributed, for autonomy of the individual agents and for scalability and improved tractability over centralized approaches.

Optimization-based techniques are suited to multiagent problems, in that such techniques can admit very general objectives. Receding horizon control is an optimization-based approach that is applicable when dynamics and constraints on the system are present. Several researchers have recently explored the use of receding horizon control to achieve multi-vehicle objectives. In most cases, the common objective is formulated, and the resulting control law implemented, in a centralized way.

This dissertation provides a distributed implementation of receding horizon control with guaranteed convergence and performance comparable to a centralized implementation. To begin with, agents are presumed to be individually governed by heterogeneous dynamics, modelled by a nonlinear ordinary differential equation. Coupling between agents occurs in a generic quadratic cost function of a single optimal control problem. The distributed implementation is generated by decomposition of the single optimal control problem into local problems, and the inclusion of local compatibility constraints in each local problem. The coordination requirements are globally synchronous timing and local information exchanges between neighboring agents. For sufficiently fast update times, the distributed implementation is proven to be asymptotically stabilizing. Extensions for handling inter-agent coupling constraints and partially synchronous timing are also explored. The venue of multi-vehicle formation stabilization demonstrates the efficacy of the implementation in numerical experiments. Given the generality of the receding horizon control mechanism, there is great potential for the implementation presented here in dynamic and constrained distributed systems.

Contents

List of Figures

Chapter 1

Introduction

Multiagent system is a phrase used here to describe a general class of systems comprised of autonomous agents that cooperate to meet a system level objective. Cooperation refers to the agreement of the agents to 1) have a common objective with other agents, with the objective typically decided offline, and 2) share information online to realize the objective. Agents are autonomous in that they are individually capable of sensing their environment and possibly other agents, communicating with other agents, and computing and implementing control actions to meet their portion of the objective. An inherent property of multiagent systems is that they are distributed, by which we mean that each agent must act autonomously based on local information exchanges with neighboring agents.

Examples of multiagent systems arise in several domains of engineering. Examples of immediate relevance to this dissertation include arrays of mobile sensor networks for aggregate imagery, autonomous highways, and formations of unmanned aerial vehicles. In these contexts, agents are governed by vehicle dynamics and often constraints. Constraints can arise in the individual agents, e.g., bounded control inputs for each vehicle. Constraints that couple agents can also be inherent to the system or specified as part of the objective, e.g., collision avoidance constraints between neighboring cars on an automated freeway. By design, such engineered multiagent systems should require little central coordination, as the communication issues in distributed environments often preclude such coordination. Moreover, a distributed control solution enables autonomy of the individual agents and offers scalability and improved tractability over centralized approaches.

Optimization-based techniques for control are well suited to multiagent problems in that such techniques can admit very general cooperative objectives. To be practically implementable in these systems, however, agents must be able to accommodate the computational requirements associated with optimization-based techniques. Receding horizon control in particular is an optimization-based approach that is applicable when dynamics and constraints on the system are present, making it relevant for the multi-vehicle examples discussed above. Moreover, recent experiments reviewed in this dissertation have shown successful real-time receding horizon control of a thrust-vectored flight vehicle.

Several researchers have recently explored the use of receding horizon control to achieve multi-vehicle objectives. In most cases, the common objective is formulated, and the resulting control law implemented, in a centralized way. This dissertation provides a distributed implementation of receding horizon control with guaranteed convergence and performance comparable to a centralized implementation.

To begin with, agents are presumed to have decoupled dynamics, modelled by nonlinear ordinary differential equations. Coupling between agents occurs in a generic quadratic cost function of a single optimal control problem. The distributed implementation is first generated by decomposition of the single optimal control problem into local optimal control problems. A local compatibility constraint is then incorporated in each local optimal control problem. The coordination requirements are globally synchronous timing and local information exchanges between neighboring agents. For sufficiently fast update times, the distributed implementation is proven to be asymptotically stabilizing. Extensions for handling coupling constraints and partially synchronous timing are also explored. The venue of multi-vehicle formation stabilization is used for conducting numerical experiments, which demonstrate comparable performance between centralized and distributed implementations. Other potential multiagent applications, in which agents are not necessarily vehicles, are discussed in the final portion of this dissertation.

1.1 Literature Review

This section provides a review of relevant literature on receding horizon control, multi-vehicle coordination problems, decentralized optimal control and distributed optimization.

1.1.1 Receding Horizon Control

In receding horizon control, the current control action is determined by solving online, at each sampling instant, a finite horizon optimal control problem. Each optimization yields an optimal control trajectory, also called an optimal control plan, and the first portion of the plan is applied to the system until the next sampling instant. The sampling period is typically much smaller than the horizon time, i.e., the planning period. The resample and replan action provides feedback to mitigate uncertainty in the system. A typical source of uncertainty is the mismatch between the model of the system, used for planning, and the actual system dynamics. “Receding horizon” gets its name from the fact that the planning horizon, which is typically fixed, recedes ahead in time with each update. Receding horizon control is also known as model predictive control, particularly in the chemical process control community. “Model predictive” gets its name from the use of the model to predict the system behavior over the planning horizon at each update.

As receding horizon control does not mandate that the control law be pre-computed, it is particularly useful when offline computation of such a law is difficult or impossible. Receding horizon control is not a new approach, and has traditionally been applied to systems where the dynamics are slow enough to permit a sampling rate amenable to optimal control computations between samples, e.g., chemical process plants. These systems are usually governed by strict constraints on states, inputs and/or combinations of both. With the advent of cheap and ubiquitous computational power, it has become possible to apply receding horizon control to systems governed by faster dynamics that warrant this type of solution.

There are two main advantages of receding horizon control:

Generality – the ability to include generic models, linear and nonlinear, and constraints in the optimal control problem. In fact, receding horizon control is the only method in control that can handle generic state and control constraints;

Reconfigurability – the ability to redefine cost functions and constraints as needed to reflect changes in the system and/or the environment.

Receding horizon control is also easy to describe and understand, relative to other control approaches. However, a list of disadvantages can also be identified:

Computational Demand – the requirement that an optimization algorithm must run and terminate at every update of the controller is for obvious reasons prohibitive;

Theoretical Conservatism – proofs of stability of receding horizon control are complicated by the implicit nature of the closed-loop system. As a consequence, conditions that provide theoretical results are usually sufficient and not necessary.

Based on the consideration above, one should be judicious in determining if receding horizon control is appropriate for a given system. For example, if constraints are not a dominating factor, and other control approaches are available, the computational demand of receding horizon control is often a reasonable deterrent from its application. Regarding multiagent systems, it is the generality of the receding horizon approach that motivated this dissertation. The cooperative objective of multiagent systems can be hard to mold into the framework of many other control approaches, even when constraints are not a dominating factor.

Since the results in this dissertation are largely theoretical, a review of theoretical results in receding horizon control is now given. The thorough survey paper by Mayne *et al.* [46] on receding horizon control of nonlinear and constrained systems is an excellent starting point for any research in this area. Therein, attention is restricted to literature in which dynamics are modelled by differential equations, leaving out a large part of the process control literature where impulse and step response models are used.

The history of receding horizon control machinery is quite the opposite of other control design tools. Prior to any theoretical foundation, applications (specifically in process control) made this machinery a multi-million dollar industry. A review of such applications is given by Qin and Badgwell in [61]. As it was designed and developed by practitioners, early versions did not automatically ensure stability, thus requiring tuning. Not until the 1990s did researchers begin to give considerable attention to proving stability.

Receding horizon control of constrained systems is nonlinear, warranting the tools of Lyapunov stability theory. The control objective is typically to steer the state to the origin, i.e., stabilization, where the origin is assumed to be an equilibrium point of the system. The optimal cost function, also known as the value function, is almost universally used as a Lyapunov function for stability analysis in current literature. There are several variants of the open-loop optimal control problem employed. These include enforcing a terminal equality constraint, a terminal inequality constraint, using a terminal cost function alone, and more recently the combination of a terminal inequality constraint and terminal cost function. Enforcing a terminal inequality constraint is equivalent to requiring that the state arrive in a set, i.e., the terminal constraint set, by the end of the planning horizon. The terminal constraint set is a neighborhood of the origin, usually in the interior of any other sets that define additional constraints on the state. Closed-loop stability is generally guaranteed by enforcing properties on the terminal cost and terminal constraint set.

The first result proving stability of receding horizon control with a terminal equality constraint was by Chen and Shaw [9], a paper written in 1982 for nonlinear continuous time-invariant systems. Another original work by Keerthi and Gilbert [38] in 1988 employed a terminal equality constraint on the state for time-varying, constrained, nonlinear, discrete-time systems. A continuous time version is detailed in Mayne and Michalska [45]. As this type of constraint is too computationally taxing and increases the chance of numerical infeasibility, researchers looked for relaxations that would still guarantee stability.

The version of receding horizon control utilizing a terminal inequality constraint provides some relaxation. In this case, the terminal cost is zero and the terminal set is a subset of the state constraint set containing the origin. A local stabilizing controller is assumed to exist and is employed inside the terminal set. The idea is to steer the state to the terminal set in finite time via receding horizon and then switch to the stabilizing controller. This is sometimes referred to as dual-mode receding horizon control. Michalska and Mayne [49] proposed this method for constrained, continuous, nonlinear systems using a variable horizon time. It was found later in multiple studies that there is good reason to incorporate a terminal cost. Specifically, it is generally possible to set the terminal cost to be exactly or approximately equal to the infinite horizon value function in a suitable neighborhood of the origin. In the exact case, the advantages of an infinite horizon (stability and robustness) are achieved in the neighborhood. However, except for unconstrained cases, terminal cost alone has not proven to be sufficient to guarantee stability. This motivated work to incorporate the combination of terminal cost (for performance) and terminal constraint set (for stability).

Most recent receding horizon controllers use a terminal cost and enforce a terminal constraint set. These designs generally fall within one of two categories; either the constraint is enforced directly in the optimization or it is implicitly enforced by appropriate choice of terminal cost and horizon time [32]. The former category is advocated in this dissertation, drawing largely on the results by Chen and Allgöwer [11], who address receding horizon control of nonlinear and constrained continuous time systems.

This dissertation also examines issues that arise when applying receding horizon control real-time to systems with dynamics of considerable speed, namely the Caltech ducted fan flight control experiment. As stated, development and application of receding horizon control originated in process control industries where plants being controlled are sufficiently slow to permit its implementation. This was motivated by the fact that the economic operating point of a typical process lies at the intersection of constraints. Applications of receding horizon control to systems other than process control problems have begun to emerge over recent years, e.g., [73] and [67].

Recent work on distributed receding horizon control include Jia and Krogh [33], Motee and Sayyar-Rodsaru [54] and Acar [1]. In all of these papers, the cost is quadratic and separable, while the dynamics are discrete-time, linear, time-invariant and coupled. Further, state and input constraints are not included, aside from a stability constraint in [33] that permits state information exchanged between the agents to be delayed by one update period. In another work, Jia and Krogh [34] solve a min-max problem for each agent, where again coupling comes in the dynamics and the neighboring agent states are treated as bounded disturbances. Stability is obtained by contracting each agents state constraint set at each sample period, until the objective set is reached. As such, stability does not depend on information updates with neighboring agents, although such updates may improve performance. More recently, Keviczky *et al* [39] have formulated a distributed model predictive scheme where each agent optimizes locally for itself and every neighbor at each update. By this formulation, feasibility becomes difficult to ensure, and no proof of stability is provided. The authors also consider a hierarchical scheme, similar to that in [47], where the scheme depends on a particular interconnection graph structure (e.g., no cycles are permitted).

The results of this dissertation will be of use to the receding horizon control community, particularly those interested in distributed applications. The distributed implementation with guaranteed convergence presented here is the first of its kind, particularly in the ability to handle generic nonlinear dynamics and constraints. The implementation and stability analysis are leveraged by the cooperation between the subsystems. A critical assumption on the structure of the overall system is that the subsystems are dynamically decoupled. As part of our future research, the extension to handle coupled subsystem dynamics will be explored.

1.1.2 Multi-Vehicle Coordinated Control

Multi-vehicle coordination problems are new and challenging, with isolated problems having been addressed in various fields of engineering. Probably the field that contains the most recent research related to the multi-vehicle coordination problem is robotics. An example is the application of hybrid control to formations of robots [20]. In this paper, nonholonomic kinematic robots are regulated to precise relative locations in a leader(s)/follower setting, possibly in the presence of obstacles. Other recent studies that involve coordinating multiple robots include [70], [36], [65], [69], [22]. All of these studies have in common the fact that the robots exhibit kinematic rather than kinetic behavior. Consequently, control and decision algorithms need not consider the real-time update constraint necessary to stabilize vehicles that have inertia.

Space systems design and engineering is another area that also addresses this type of problem. Specifically, clusters of microsatellites when coordinated into an appropriate formation may perform high-resolution, synthetic aperture imaging. The goal is to exceed the image resolution that is possible with current single (larger) satellites. Strict constraints on fuel efficiency and maneuverability, i.e., low-thrust capability, of each microsatellite must be accounted for in the control objectives for the problem to be practically meaningful. There are various approaches to solving variants of the microsatellite formation and reconfiguration problem (see [37, 47] and references therein). In [53], the nonlinear trajectory generation (NTG) software package used in this dissertation is applied to the microsatellite formation flying problem. This paper utilizes differential flatness of the dynamics to directly generate optimal trajectories that account for projected area (imaging) constraints and minimize fuel consumption. The study also includes the dynamical perturbation due to the oblateness of the earth, i.e., the J_2 effect. Issues related to the real-time implementation of this method are also mentioned.

The problem of autonomous intelligent cruise control in automated traffic systems has received much attention. Studies typically consider a platoon of one degree-of-freedom, simplified car models, with controllers that rely upon decision logic and role relations [31]. In air traffic control, the problem of resolution of conflicts involving many aircraft naturally arises. An approach to solving a generalized version of this problem by a combination of convex programming and randomized searches is given in [23]. Collision avoidance is clearly an important issue in multi-vehicle control.

Graph theory methods have proven to be useful for determining properties of a leader/follower architecture in formation flying of multiple spacecraft [47]. The tools of graph theory have also bared relevance in determining stability properties of vehicle formations in [19]. Specifically, Fax and Murray show that the effects of the interconnection topology (i.e., which vehicles are sensed by other vehicles) on vehicle formation stability can be cast into a local Nyquist-like criterion via the Laplacian matrix of a corresponding graph. The authors also investigate transmission of sensed information and its affect on stability in terms of the graph theoretic developments.

The multi-vehicle formation stabilization problem explored in this dissertation is used as a venue for the theory. Two alternative approaches for defining a formation are given. Ultimately, the distributed receding horizon control approach developed here will be applied to the Caltech Multi-Vehicle Wireless Testbed [13], where the individual vehicles have hovercraft dynamics and communicate using wireless ethernet.

1.1.3 Decentralized Optimal Control and Distributed Optimization

Decentralized control is a large subject in the controls community. The large-scale nature of certain systems, particularly power systems and chemical processing systems, motivated controls engineers to attempt to exploit a decentralized structure from traditional control approaches.

For optimal decentralized control, the problem has been thought of as an optimal control problem subject to decentralized information structure constraints, i.e., each local feedback has access only to the local state, for which complete information is available. For coupled LTV stochastic dynamics and decoupled quadratic cost, Savastuk and Siljak give an optimal decentralized control presuming this decentralized information structure and using the method of Lagrange [66].

In other formulations, optimal decentralized controllers have been sought where state information for each subsystem and neighboring subsystems is presumed to be available, by measurement or via communication. This objective is more consistent with cooperative control objectives. It is easy to show that for dynamically decoupled LTI systems with a single quadratic coupling cost on the states in an optimal control problem, the LQR feedback preserves the interconnection structure imposed by the coupling cost. It has also been shown that for stochastic optimal control of a linear system, the structure of information interdependence can determine whether optimal decentralized controls are linear [29] or nonlinear [74]. More recently, Rotkowitz and Lall [64] formulate a decentralized linear control synthesis problem as one of minimizing the closed-loop norm of a feedback system subject to constraints on the (linear) controller structure. When such constraints satisfy a *quadratic invariance* property, the problem becomes a convex optimization problem. In the work in this dissertation, information dependence can be arbitrary, provided it does not change in time and that the dependence is mutual between agents that are coupled. It is in fact straightforward to admit "directed" information flow between agents, and this will be part of our ongoing research. Up to this extension, the results of this dissertation hold for arbitrary information structure as in [64]. The main contribution of this dissertation relative to other approaches is that the subsystems may be governed by heterogeneous nonlinear dynamics, provided again that the subsystem dynamics are decoupled.

Parallel and distributed computation research is of greater relevance to this dissertation than the decentralized control results to date. Motivations for parallel and distributed computation include the need for speed of computations in solving complex problems, e.g., partial differential equations, and the need for scalability of computations in solving large-scale interconnected problems, e.g., queueing systems [3]. In parallelization, a single node decomposes the problem, assigns portions to sub-node processors (often assumed to be homogeneous), and subsequently assembles the solution after an iterative procedure. Communication and synchronization are issues that parallel and distributed numerical methods must address, issues that do not arise in their serial counterparts. In all cases, parallelization seeks to solve the centralized problem in a distributed way. When the computations are solving optimization problems, the book by Bertsekas and Tsitsiklis [3] has an excellent generalization for parallelization based on both algorithm structure and problem structure. As we are more interested in problem structure here, we reference that work now. The primary tool in their work is duality theory. If the dual of an optimization problem can be formulated, it is often more suitable for parallel computation than the original problem. Decomposition methods are tools for breaking up large-scale problems into smaller subproblems [27], and are a good approach when parallel computing systems are available.

In contrast to parallelization methods, the work here does not attempt to solve the original centralized optimization problem, at any receding horizon update. The original centralized problem is used solely to induce distributed problems. While the distributed solution is not the centralized solution of the original problem, it does correspond to the centralized solution of a modified problem. Additionally, the distributed implementation is much more conducive to distributed environments than parallelization, in that the communication requirements are substantially reduced. In particular, at each receding horizon update, agents exchange information used to initialize their distributed optimization problems. Other than this exchange, the agents do not communicate; specifically, they do not communicate while solving their local optimization problems. While the centralized solution of the original problem is not recovered, numerical experiments show that the distributed implementation performs comparably (in terms of closed-loop system performance) to the centralized implementation of the original problem.

1.2 Dissertation Outline

The dissertation begins in Chapter 2 with a review of a receding horizon control law presented in [11] that admits a general nonlinear model, constraints and quadratic cost function. Sufficient conditions for asymptotic stability are stated and issues regarding implementation and relaxations of the assumptions are briefly explored. To date, the predominant number of successful examples of receding horizon control in practice arise in the process control field, where the time scales of the dynamics are sufficiently slow to permit the required online optimization calculations.

A motivation for this dissertation is the application of receding horizon control to multi-vehicle systems. Consequently, it is important to examine the real-time issues that arise when the time scales of system dynamics are much faster than in the applications of process control. Fortunately, the scale of the optimization problem for individual vehicles systems is much smaller than that of process control systems. Additionally, recent computational tools developed at Caltech [52] have made it possible to solve optimal control problems with ample efficiency for real-time trajectory generation and receding horizon control of vehicles. In particular, the successful initial experimental demonstration of receding horizon control of an unmanned flight vehicle is presented in Chapter 3.

Returning to multi-vehicle systems, the generality of receding horizon control makes it easy to formulate a meaningful single optimal control problem for a centralized receding horizon implementation. However, unless the system is restricted to a moderate number of vehicles, the scale of the optimization problem begins to approach that of process control problems. As a result, real-time centralized implementations of many vehicle systems is not possible due to the time scales of vehicle dynamics. This issue motivated the distributed implementation presented in Chapter 4, which is the primary contribution of this dissertation. The implementation is shown to accommodate generic, decoupled nonlinear dynamics and constraints, with coupling in a quadratic cost function. The dynamics need not be vehicle dynamics and examples where the cost function is no longer quadratic are also addressed in a later chapter.

The distributed implementation is generated first by decomposition of the single optimal control problem into local optimal control problems. A local compatibility constraint is then incorporated in each local optimal control problem. Roughly speaking, the constraint ensures that no agent will diverge too far from the behavior that neighbors expect of it. Under stated assumptions, the distributed implementation is proven to be asymptotically stabilizing for sufficiently fast update times.

The distributed implementation and sufficient conditions for stability presented in Chapter 4 are then analyzed in detail in Chapter 5. First, the implementation is interpreted in Section 5.2, giving both qualitative and quantitative comparisons with centralized implementations. In Section 5.3, alternative ways of formulating the distributed implementation, while still preserving closed-loop stability, are explored. This section includes a dual-mode version of the distributed receding horizon control law. Finally, in Section 5.4, specific extensions of the theory, e.g., to handle partially synchronous timing situations, are discussed in detail.

In Chapter 6, the venue of multi-vehicle formation stabilization is used for conducting numerical experiments. The experiments demonstrate comparable performance between centralized and distributed implementations. Consistent with the theory in Chapter 4, the coupling of the vehicles occurs in a quadratic integrated cost function. An alternative approach for accommodating a formation stabilization objective, where the cost is no longer quadratic, is presented at the end of Chapter 6. General extensions of the work in this dissertation, including connections with other relevant fields and potential applications, are discussed in Chapter 7. Finally, Chapter 8 summarizes the main results and future areas of research suggested by this dissertation.

Chapter 2

Receding Horizon Control

2.1 Introduction

In receding horizon control, the current control action is determined by solving online, at each sampling instant, a finite horizon optimal control problem. Each optimization yields an open-loop optimal control trajectory and the first portion of the trajectory is applied to the system until the next sampling instant, when the current state is measured and the optimal control problem re-solved. Application of only a fraction of the open-loop control, followed by resampling and recomputing, results in closed-loop control. In this chapter, we review the receding horizon control law based on the problem formulation and results in the dissertation of Chen [10], which is summarized in [11]. Practical issues that arise from the implementation of receding horizon control are also discussed at the end of this chapter.

2.2 Receding Horizon Control

The dynamics of the system to be controlled are described by an ordinary differential equation as

$$\dot{z}(t) = f(z(t), u(t)), \quad t \geq 0, \quad z(0) \text{ given}, \tag{2.1}$$

with state $z(t) \in \mathbb{R}^n$ and control $u(t) \in \mathbb{R}^m$. The control objective is to drive the state to the equilibrium point z^c. In addition, the state and control are required to be in the constraint sets

$$u(t) \in \mathcal{U}, \quad z(t) \in \mathcal{Z}, \qquad t \geq 0.$$

An *admissible control* is any piecewise, right-continuous function $u(\cdot) : [0, T] \to \mathcal{U}$, for any $T \geq 0$, such that given an initial state $z(0) \in \mathcal{Z}$, the control generates the state trajectory curve

$$z(t; z(0)) = z(0) + \int_0^t f(z(\tau; z(0)), u(\tau))\, d\tau,$$

with $z(t; z(0)) \in \mathcal{Z}$ for all $t \in [0, T]$.

Assumption 2.1 The following holds:

(i) the function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is twice-continuously differentiable, satisfies $0 = f(z^c, 0)$ and f linearized around $(z^c, 0)$ is stabilizable;

(ii) the system (2.1) has a unique, absolutely continuous solution $z(\cdot; z(0))$ for any initial condition $z(0) \in \mathcal{Z}$ and any admissible control;

(iii) the set $\mathcal{U} \subset \mathbb{R}^m$ is compact, convex and contains the origin in its interior, and the set $\mathcal{Z} \subseteq \mathbb{R}^n$ is convex, connected and contains z^c in its interior;

(iv) full state measurement is available and computational time is negligible compared to the evolution of the closed-loop dynamics.

At any time t, given the current state $z(t)$ and fixed horizon time T, the open-loop optimal control problem is

Problem 2.1 Find

$$J^*(z(t), T) = \min_{u(\cdot)} J(z(t), u(\cdot), T),$$

with

$$J(z(t), u(\cdot), T) = \int_t^{t+T} \|z(\tau; z(t)) - z^c\|_Q^2 + \|u(\tau)\|_R^2 \, \mathrm{d}\tau + ||z(t+T; z(t)) - z^c\|_P^2,$$

subject to

$$\left.\begin{aligned} &\dot{z}(s) = f(z(s), u(s)) \\ &u(s) \in \mathcal{U} \\ &z(s; z(t)) \in \mathcal{Z} \end{aligned}\right\} \quad s \in [t, t+T],$$

$$z(t+T; z(t)) \in \Omega(\alpha), \tag{2.2}$$

where

$$\Omega(\alpha) := \{z \in \mathbb{R}^n \ : \ \|z - z^c\|_P^2 \leq \alpha, \ \alpha \geq 0\}.$$

The equation (2.2) is called the *terminal constraint*, as it is a constraint enforced only at the terminal or end time. Let the first optimal control problem be initialized at some time $t_0 \in \mathbb{R}$ and let δ denote the *receding horizon update period.* The closed-loop system, for which stability is to be guaranteed, is

$$\dot{z}(\tau) = f(z(\tau), u^*(\tau)), \quad \tau \geq t_0, \tag{2.3}$$

where the *receding horizon control law* is

$$u^*(\tau) = u^*(\tau; z(t)), \quad \tau \in [t, t+\delta), \quad 0 < \delta \leq T,$$

and $u^*(s; z(t))$, $s \in [t, t+T]$, is the optimal open-loop solution (assumed to exist) to Problem 2.1 with initial state $z(t)$. The receding horizon control law is defined for all $t \geq t_0$ by applying the open-loop optimal solution until each new initial state update $z(t) \leftarrow z(t+\delta)$ is available. The notation above shows the implicit dependence of the optimal open-loop control $u^*(\cdot; z(t))$ on the initial state $z(t)$ through the optimal control problem. The optimal open-loop state trajectory is denoted $z^*(\tau; z(t))$. Since Problem 2.1 is time-invariant, we can set $t = 0$ and solve the optimal control problem at each initial state update over the time interval $[0, T]$.

Proof of asymptotic stability of the closed-loop dynamics under the receding horizon control implementation can be established by taking the optimal cost function $J^*(\cdot)$ as a Lyapunov function.

Definition 2.1 A *feasible control* is any admissible control such that all state constraints in Problem 2.1 are satisfied and the optimal cost function is bounded. Let Z denote the set of states for which there exists a feasible control.

The linearization of the system (2.1) at $(z, u) = (z^c, 0)$ is denoted

$$\dot{z}(t) = Az(t) + Bu(t), \quad A = \frac{\partial f}{\partial z}(z^c, 0), \quad B = \frac{\partial f}{\partial u}(z^c, 0).$$

By assuming stabilizability, a linear feedback $u = K(z - z^c)$ can be found such that $A + BK$ has all eigenvalues in the open left-half complex plane.

Assumption 2.2 The following conditions are satisfied:

(i) the largest constant $\alpha > 0$ in the terminal constraint (2.2) is chosen such that $\Omega(\alpha) \subseteq \mathcal{Z}$ is invariant for the closed-loop nonlinear system $\dot{z} = f(z, K(z - z^c))$, and such that for all $z \in \Omega(\alpha)$, the stabilizing feedback $u = K(z - z^c) \in \mathcal{U}$. In addition, given $Q > 0$ and $R > 0$, the matrices K and $P = P^T > 0$ satisfy

$$(A + BK)^T P + P(A + BK) = -(Q + K^T RK); \tag{2.4}$$

(ii) the optimal solution to Problem 2.1 exists and is numerically obtainable for all $z \in Z$.

We must be sure that there exists an $\alpha > 0$ such that condition (i) above holds. Fortunately, the next result guarantees that such an α does exist.

Lemma 2.1 *[11, Lemma 1] If the linearization of the system (2.1) at* $(z, u) = (z^c, 0)$ *is stabilizable, then there exists a constant* $\alpha \in (0, \infty)$ *such that condition* (i) *in Assumption 2.2 holds.*

The size of the terminal region is determined by either the constraints or the effects of the nonlinearities in the model.

Theorem 2.1 *[11, Theorem 1] Under Assumptions 2.1 and 2.2, for any* $\delta \in (0, T]$, z^c *is an asymptotically stable equilibrium point of the closed-loop system (2.3) with region of attraction* Z*, an open and connected set.*

The stability result in [11] only requires that Problem 2.1 be feasible at initialization, rather than requiring the optimal solution at each update. Also, δ is required to be sufficiently small since the authors consider quantization errors in the numerical implementation of the receding horizon control law.

2.3 Implementation Issues

The receding horizon control law above is conceptual, in that some of the assumptions are valid only conceptually, and not usually valid in practice. In this section, we remark on the degree of validity of such assumptions when implementing receding horizon control in practice. We note first that many theoretical receding horizon control algorithms in the literature are stated in discrete time form, to be consistent with the implemented version in digital computers.

2.3.1 Computational Delay

Process control is by far the industry with the greatest number of implementations of receding horizon control laws. The name model predictive control is more appropriate in process control applications, and the reason is related to the way in which the controller is implemented. Specifically, given the current state $z(t)$ and optimal control $u^*(\tau; z(t))$ to be implemented over the interval $[t, t + \delta]$, the model is used to predict where the state will be at time $t + \delta$ *before* the interval expires. Given the *predicted* state, a new optimal control trajectory is generated, and once time $t + \delta$ occurs, the new control is implemented. The prediction thus permits a margin of computational time in solving Problem 2.1. In the absence of uncertainty, the stability proofs still follow as before, since the predicted state and actual state coincide.

Most theoretical results presume zero computational delay, but we make note of a few that do not make this assumption. Michalska and Mayne [49] give an algorithm for receding horizon control that at any time implements a feasible control, and a more optimal control can be implemented if it is made available in time by the computations. An interesting paper that considers nontrivial computation times in receding horizon control implementation on a flight control experiment, the same experiment in fact that is described in the next chapter, is [51]. The authors sketch the theoretical implications of non-trivial computational delay in a continuous time setting. Another paper that accounts for computational delay is [79], where stability is achieved using a contractive constraint, and the same "prediction + computation" approach used in process control is employed.

2.3.2 Robustness to Uncertainty

The survey paper [46] reviews several studies on the robustness properties of receding horizon control. There are two basic approaches to address robustness. For robustness with respect to exogenous disturbances, most authors formulate a min-max optimization, as in [44]. For robustness with respect to model uncertainty, [49] and [79] take a similar approach. Namely, presuming a Lipschitz bound on the difference between the model of the system and the actual system, both results guarantee convergence to a compact set containing the objective state. We note that this type of robustness analysis is applicable to different variants of receding horizon control, as both receding horizon control laws in [49] and [79] are based on different stability constraints in the optimal control problem.

More recently, the authors in [26] have shown that certain receding horizon control algorithms in the literature exhibit zero robustness, meaning that in particular examples, asymptotic stability is lost by including arbitrarily small disturbances. It is well-known that if a Lyapunov function is used to prove asymptotic stability of a closed-loop system, then the feedback exhibits inherent robustness properties [40], particularly in the form of multiplicative uncertainty on the input. The work in [26] characterizes the loss of robustness by examining when the Lyapunov candidate losses continuously differentiability over the domain, hence losing the inherent robustness properties of a true Lyapunov function. Regarding the conditions on the constraint sets in Assumption 2.1, we note that convexity of both $\mathcal{U}$ and $\mathcal{Z}$ is relevant in guaranteeing that the closed-loop system will have nominal robustness properties [26].

2.3.3 Relaxing Optimality

We briefly state here sufficient conditions which guarantee that the optimal solution exists, referring to section 4.3 of [10] (and references therein) for details. The conditions are stated for a general integrated cost, rather than requiring the integrated cost to be quadratic. Assuming that the feasible control set is nonempty, the integrated cost is a function in $L_1[t, t+T]$, and the conditions in Assumptions 2.1 and 2.2 hold, a sufficient condition for the existence of an optimal solution is that f be affine in u and the integrated cost be a convex function of u on $\mathcal{U}$ (Corollary 4.1 in [10]).

Of course, existence of the optimal control is one issue; computing this control in practice is another. Fundamentally, the distinction has to do with the difference between conceptual algorithms and implementable algorithms in optimization and optimal control [60], a subject beyond the scope of this dissertation. Intuitively, any algorithm that terminates in finite time can at best produce only a good approximate solution to an infinite dimensional problem. Approximations aside, receding horizon control algorithms that require intensive computations at every update have an obvious practical disadvantage. For this reason, several receding horizon control algorithms in the literature relax the optimality requirement by permitting the use of previously available feasible control. For example, the stability results in [11] require only that there exist a feasible control at initialization. In the absence of uncertainty, concatenating the remainder of this control with the terminal controller K defined in Assumption 2.2 results in subsequent feasibility. While stability results still hold, we note that optimality[1] requires optimal trajectories, so the performance of the feasible control is only as good as the initial control. Moreover, if any uncertainty is present, the feasibility argument breaks down. Other studies that include results with relaxation of the optimality requirement include [49, 32].

2.4 Summary

Under stated assumptions, an asymptotically stabilizing receding horizon control law has been defined, based on the formulation and results by Chen and Allgöwer in [11]. While the solution of each optimal control problem results in an open-loop control, employing only the first portion of the control before resampling and recomputing provides a closed-loop control. We have not discussed the important issue of handling only partial state feedback, nor have we discussed the myriad of numerical algorithms available for solving optimal control problems. The purpose of the chapter was solely to introduce notation and the structure of a receding horizon control law that is provably asymptotically stabilizing. In the next chapter, the details of an experimental implementation of receding horizon control are provided. The implementation gives additional insight into the practical issues associated with real-time receding horizon control and, more importantly, demonstrates the success of the receding horizon philosophy and the numerical algorithm employed.

[1]By optimality here, we simply mean the optimal cost, not the resulting closed-loop feedback. As pointed out in the excellent paper by David Mayne [44], receding horizon control is not an optimal feedback in the sense that, from any initial condition, dynamic programming provides an optimal feedback. In fact, the dynamic programming solution is typically the open-loop solution when a terminal constraint is enforced, a result of the principle of optimality.

Chapter 3

Receding Horizon Control of a Flight Control Experiment

3.1 Introduction

This chapter is concerned with the application of receding horizon control to a high-performance flight control experiment shown in Figure 3.1. As stated, receding hori-

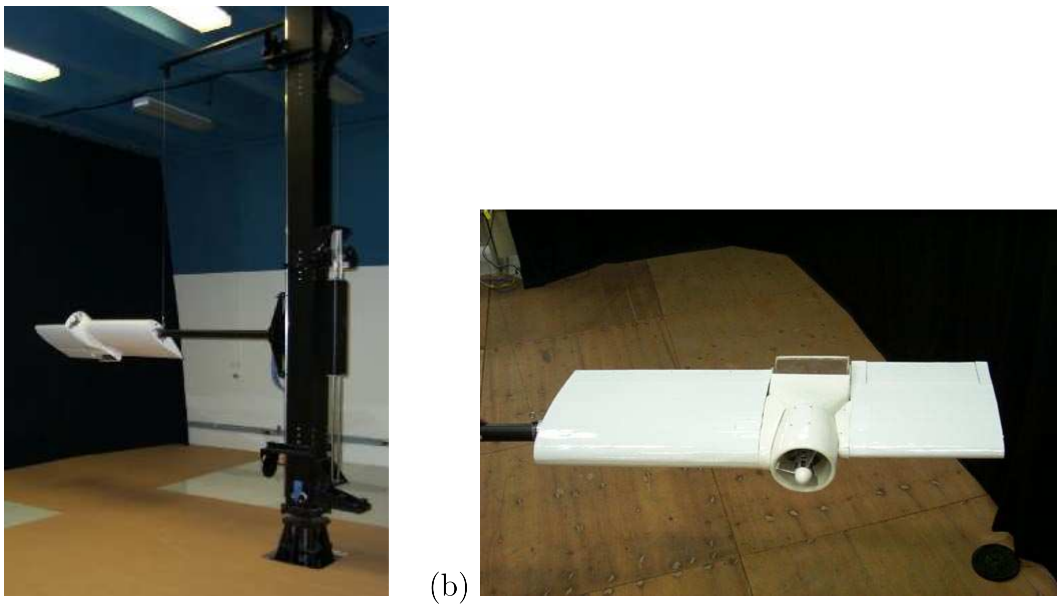
(a)  (b)

Figure 3.1: Caltech ducted fan experiment: (a) full view, (b) close-up view.

zon control is traditionally applied to system with dynamics slow enough to permit computations between samples. It is also one of few suitable methods in applications that can impose constraints on the states and or inputs, as the constraints are directly enforced in the online optimal control problem. With the advent of faster modern computers, it has become possible to extend receding horizon control to systems governed by faster dynamics that warrant this type of solution. An example of such a system is the Caltech ducted fan, a thrust-vectored flight control experiment where actuation and spatial constraints are present. Real-time receding horizon control on this experiment is achieved but using the NTG software package developed at Caltech [52]. Timing issues that arise from real-time receding horizon control computations are here elucidated for this system. A method for applying the generated optimal trajectories while accounting for nontrivial computational time is detailed and implemented. Experimental tests compare various receding horizon control parameterizations and show the success of this methodology for real-time control of the ducted fan. Specifically, the method is compared to a static hover LQR controller and a gain-scheduled LQR controller for stabilization of a step disturbance. Results show that the receding horizon control controllers have a bigger region of attraction than the static hover LQR controller and perform comparably to the gain-scheduled LQR controller.[1]

More recent work on the application of receding horizon control to the Caltech ducted fan has been conducted by Milam *et al.* [51]. Therein, a full nonlinear and aerodynamic model of the ducted fan is used in the optimal control problem. For the results in this chapter, a simpler nonlinear model that does not account for aerodynamics is used. Additionally, several timing approaches are explored in [51] to mitigate the effects of nontrivial computational delay on the closed-loop performance.

3.2 Flight Control Experiment

The Caltech ducted fan is an experimental testbed designed for research and development of nonlinear flight guidance and control techniques for Uninhabited Combat Aerial Vehicles (UCAVs). The fan is a scaled model of the longitudinal axis of a flight vehicle and flight test results validate that the dynamics replicate qualities of actual

[1]The contents of this chapter are summarized in the conference paper [15].

flight vehicles [50].

3.2.1 Hardware

The ducted fan has three degrees of freedom: the boom holding the ducted fan is allowed to operate on a cylinder, 2 m high and 4.7 m in diameter, permitting horizontal and vertical displacements. Also, the wing/fan assembly at the end of the boom is allowed to rotate about its center of mass. Optical encoders mounted on the ducted fan, gearing wheel, and the base of the stand measure the three degrees of freedom. The fan is controlled by commanding a current to the electric motor for fan thrust and by commanding RC servos to control the thrust vectoring mechanism. The sensors are read and control commands sent by a dSPACE multi-processor system. The dSPACE system is comprised of a D/A card, a digital IO card, two Texas Instruments C40 signal processors, two Compaq Alpha processors (500 and 600 MHz), and a ISA bus to interface with a PC. A schematic of the experimental setup is shown in Figure 3.2 Reference state information is sent to the dSPACE system by keyboard commands.

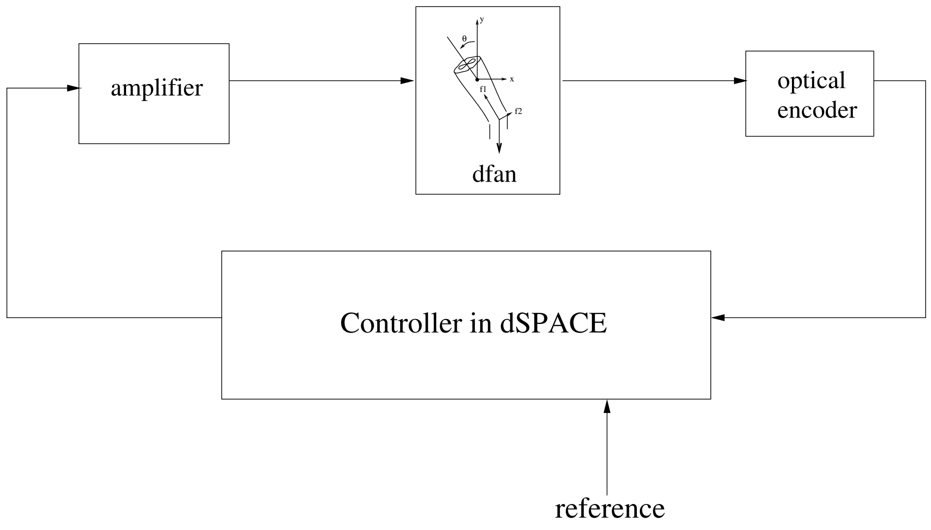


Figure 3.2: Ducted fan experimental setup.

3.2.2 Software

The dSPACE system provides a real-time interface to the 4 processors and I/O card to the hardware. The NTG code resides on the 500 MHz alpha processor. A detailed description of NTG as a real-time trajectory generation package for constrained mechanical systems is given in [52]. The package is based on finding trajectory curves in a lower dimensional space and parameterizing these curves by B-splines. Sequential quadratic programming (SQP) is used to solve for the B-spline coefficients that optimize the performance objective, while respecting dynamics and constraints. The package NPSOL [25] is used to solve the SQP problem.

3.2.3 Model of the Ducted Fan

The full nonlinear model of the fan including aerodynamic and gyroscopic effects is detailed in [50]. For the implementation of the receding horizon approach outlined in this chapter, the planar model of the fan is utilized. A schematic of the planar model is shown in Figure 3.3. The ordinary differential equation that models the planar

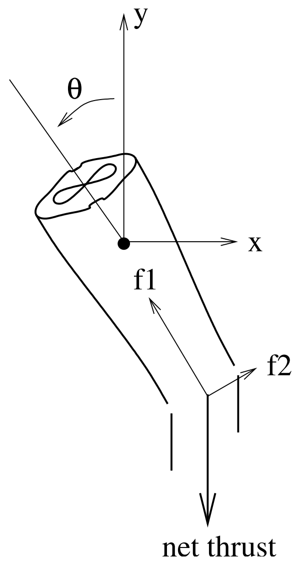


Figure 3.3: Planar model of the ducted fan.

ducted fan is

$$\begin{aligned} m\ddot{x} &= f_2 \cos\theta - f_1 \sin\theta \\ m\ddot{y} &= f_2 \sin\theta + f_1 \cos\theta - mg \\ J\ddot{\theta} &= r f_2. \end{aligned} \tag{3.1}$$

The parameters are the mass m, moment of inertia J, g is the acceleration due to gravity, and r is the distance from force f_2 to the center of mass, i.e., the origin of the x-y frame.

The configuration variables x and y represent, respectively, horizontal and vertical inertial translations of the fan while θ is the rotation of the fan about the boom axis. These variables are measured via the optical encoders, and the derivatives are computed with a FIR filter. The inputs f_1 and f_2 are the body-fixed axial and transverse thrust components, respectively. Consistent with the notation in the previous chapter, we denote the state and input vectors $z = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$ and $u = (f_1, f_2)$ and represent the dynamics in equation (3.1) as $\dot{z} = f(z, u)$ in the optimal control problem defined below.

3.3 Application of Receding Horizon Control

This section outlines the receding horizon control law and a timing method for updating real-time trajectories while accounting for non-negligible computational time.

3.3.1 Receding Horizon Control Formulation

The receding horizon control formulation is restated here for reference. At any time t, the current optimal control $u^*(\tau; z(t)),\ \tau \in [t, t+T]$, for current state $z(t)$ is the solution to the following problem

$$\min_{u(\cdot)} \int_t^{t+T} \|z(\tau; z(t)) - z^c\|_Q^2 + \|u(\tau) - u^c\|_R^2 \, \mathrm{d}\tau + \|z(t+T; z(t)) - z^c\|_{P_0}^2, \tag{3.2}$$

$$\text{s.t.} \quad \left.\begin{aligned} &\dot{z}(s) = f(z(s), u(s)) \\ &u(s) \in \mathcal{U} \\ &z(s; z(t)) \in \mathcal{Z} \end{aligned}\right\} \quad s \in [t, t+T]. \tag{3.3}$$

No terminal constraint is enforced in the experiments that follow. While the stability results in Chapter 2 required a terminal constraint, we note that Jadbabaie *et al.* [32] prove stability in the absence of a terminal constraint, provided the other state and control constraints are absent. For the ducted fan problem, the system dynamics corresponds to equation (3.1) and the equilibrium point of interest is hover, with state and control vectors $z^c = (0, 0, 0, 0, 0, 0)$ and $u^c = (mg, 0)$, and mg is 7.3 N. Note that the cost on the control input in equation (3.2) is modified to reflect that the equilibrium control is not zero.

Regarding the state constraint set, the vertical space in the lab is limited, which translates to the bounds on the vertical position (in meters) $y \in [-1, 1]$. For the tests considered in this chapter, the fan does not hit the boundaries of this constraint, and it is not included in the optimization problem and $\mathcal{Z} = \mathbb{R}^6$. The control constraint set is defined as

$$\mathcal{U} = \left\{ u = (f_1, f_2) \in \mathbb{R}^2 \;\middle|\; f_1 \in [0, F_{\max}],\ f_2 \in [-F_{\max}/2, F_{\max}/2] \right\},$$

where $F_{\max} = 11$ N. In addition to the stated constraints, the control inputs are initially constrained to be within 0.25 N of the previously computed optimal inputs, at the appropriate instant of time. This amount of time will be detailed in the next section that describes how the timing of the receding horizon control process is done. With respect to the cost in equation (3.2), the weighting matrices are defined as

$$Q = \mathrm{diag}(4, 15, 4, 1, 3, 0.3), \quad R = \mathrm{diag}(0.5, 0.5), \quad P_0 = \gamma P,$$

where $\gamma = 0.075$ and P is the unique stable solution to the algebraic Riccati equation corresponding to the linearized dynamics of equation (3.1) at hover and the weights Q and R. We note that when the receding horizon control law is implemented *without* any terminal cost and/or the input constraints in the experiment, *instability* resulted. Also, an inner-loop PD controller on $\theta, \dot{\theta}$ is implemented to stabilize to the open-loop receding horizon control states $\theta^*, \dot{\theta}^*$. The θ dynamics are the fastest for this system and although most receding horizon control laws were found to be nominally stable without this inner-loop controller, small disturbances could lead to instability, hence the use of the PD inner-loop. A schematic of the experimental setup with the details of the controller in dSPACE is shown in Figure 3.4. The controller is the receding

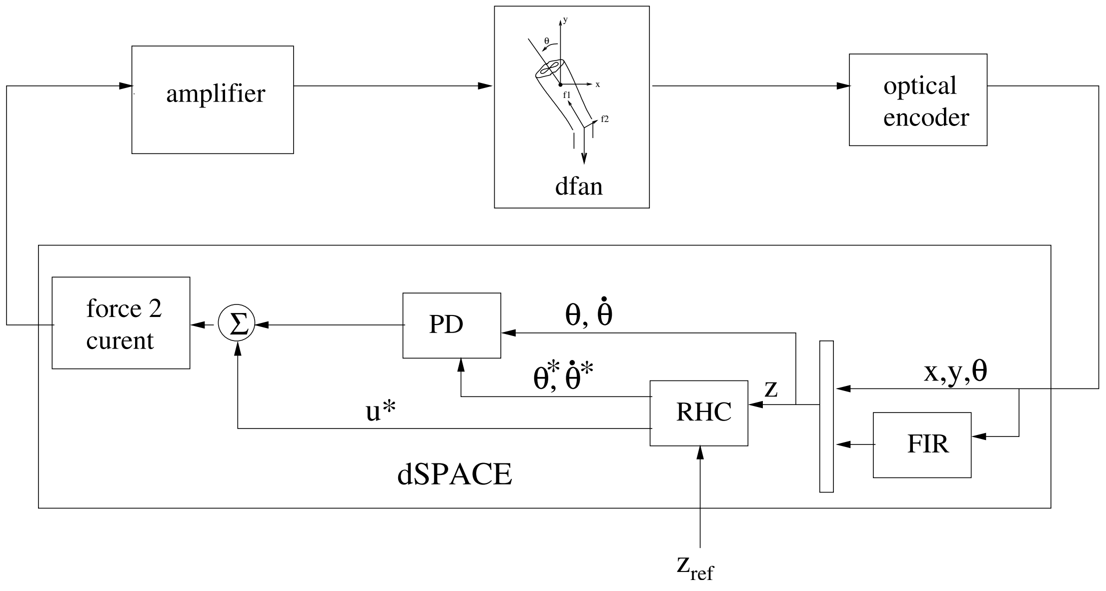


Figure 3.4: Ducted fan experimental setup with receding horizon and inner-loop controller.

horizon controller, implemented with a timing scheme defined in the next section, plus the PD controller. The resultant commanded forces are converted to currents for commanding the motor and RC servos. The reference state z_{ref} is used to explore the response of the closed-loop system to step changes in the value of the objective state. Specifically, we set $z^c = z_{\text{ref}} = (a, 0, 0, 0, 0, 0)$, where a is a constant desired value for the horizontal position.

The optimal control problem is setup in NTG code by parameterizing the three position states (x, y, θ), each with 8 B-spline coefficients. Over the receding horizon time intervals, 11 and 16 breakpoints were used with horizon lengths of 1.0, 1.5, 2.0 and 3.0 seconds. Breakpoints specify the locations in time where the differential equations and any constraints must be satisfied, up to some tolerance. The value of $F_{\max}$ for the input constraints is made conservative to avoid prolonged input saturation on the real hardware. The logic for this is that if the inputs are saturated on the real hardware, no actuation is left for the inner-loop theta controller and the system can go unstable. The value used in the optimization is $F_{\max} = 9$ N.

3.3.2 Timing Setup

Computation time is non-negligible and must be considered when implementing the optimal trajectories. The computation time varies with each optimization as the current state of the ducted fan changes. The following notation facilitates the description of how the timing is setup:

i	:	integer counter of receding horizon control computations,
t_i	:	value of current time when receding horizon control computation i started,
$\delta_c(i)$	:	computation time required for ith computation $(t_{i+1} = t_i + \delta_c(i))$,
$u^*(i)(t)$	:	optimal control from computation i, $t \in [t_i, t_i + T]$.

A natural choice for updating the optimal trajectories for stabilization is to do so as fast as possible. This is achieved here by constantly re-solving the optimization. When computation i is done, computation $i + 1$ is immediately started, so $t_{i+1} = t_i + \delta_c(i)$. Figure 3.5 gives a graphical picture of the timing setup as the optimal control trajectories $u^*(\cdot)$ are updated. As shown in the figure, any computation i for $u^*(i)(\cdot)$ occurs for $t \in [t_i, t_{i+1}]$ and the resulting trajectory is applied for $t \in [t_{i+1}, t_{i+2}]$. At $t = t_{i+1}$ computation $i+1$ is started for trajectory $u^*(i+1)(\cdot)$, which is applied as soon as it is available $(t = t_{i+2})$. For the experimental runs detailed in the results, $\delta_c(i)$ is typically in the range of [0.05, 0.25] seconds, meaning 4 to 20 receding horizon control

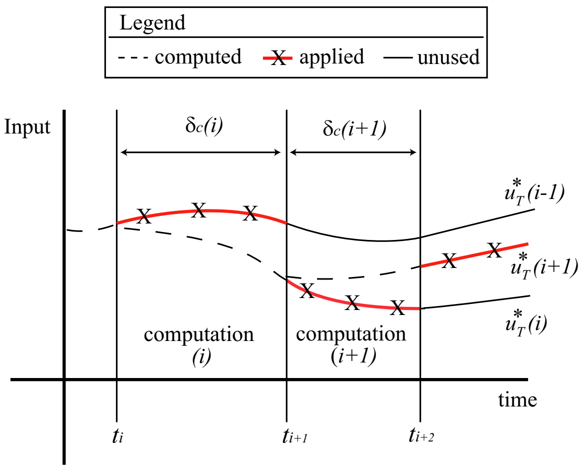


Figure 3.5: Receding horizon input trajectories, showing implementation after delay due to computation.

computations per second. Each optimization i requires the current measured state of the ducted fan and the value of the previous optimal input trajectories $u^*(i-1)$ at time $t = t_i$. This corresponds to, respectively, 6 initial conditions for state vector z and 2 initial constraints on the input vector u. Figure 3.5 shows that the optimal trajectories are advanced by their computation time prior to application to the system. A dashed line corresponds to the initial portion of an optimal trajectory and is not applied since it is not available until that computation is complete. The figure also reveals the possible discontinuity between successive applied optimal input trajectories, with a larger discontinuity more likely for longer computation times. The initial input constraint is an effort to reduce such discontinuities, although some discontinuity is unavoidable by this method. Also note that the same discontinuity is present for the corresponding open-loop optimal state trajectory, again with a likelihood for greater discontinuity for longer computation times. In this description, initialization is not an issue because we assume the receding horizon control computations are already running prior to any test runs. This is true of the experimental runs detailed in the results.

3.4 LQR Controller Design

For comparison, the receding horizon control approach is compared to two LQR designs. One is a simple static LQR controller, designed with the planar ducted fan model equation (3.1) linearized around hover z^c and with the same weights Q and R used in equation (3.2).

The second design is a gain-scheduled LQR. Using the full aero/gyro model of the ducted fan, equilibrium forces and angle of attack for forward flight at constant altitude y and forward velocity $\dot{x}$ are identified. In this case θ is the angle of attack, and it is possible to linearize the full model around the forward flight equilibrium values for $\dot{x}$ and θ, where the value for all other states is zero. The gain-scheduling weights chosen are

$$Q = \text{diag}(1, 1, 15, 30, 4, 0.3), \quad R = \text{diag}(0.5, 0.5),$$

corresponding to the quadratic penalty on the state and input vectors, respectively. In comparison to the previous weighting matrices, the relaxed weights on $(x, \dot{x})$ and increased weights on $(y, \dot{y})$ were chosen specifically to improve stability in the presence of the x disturbance investigated in the results. The LQR gains were scheduled on θ, while scheduling on $\dot{x}$ is also possible.

3.5 Results

The experimental results show the response of the fan with each controller to a 6 meter horizontal offset, which is effectively engaging a step-response to a change in the initial condition for x. The following subsections detail the effects of different receding horizon control parameterizations, namely as the horizon changes, and the responses with the different controllers to the induced offset.

3.5.1 Comparison of Response for Different Horizon Times

The first comparison is between different receding horizon control laws, where time horizon is varied to be 1.5, 2.0, 3.0, 4.0 or 6.0 seconds. Each receding horizon control law uses 16 breakpoints. Figure 3.6 shows a comparison of the average computation time as time proceeds. For each second after the offset was initiated, the data corre-

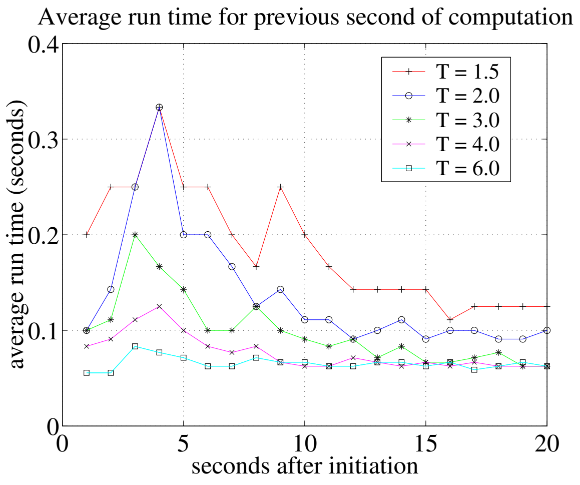


Figure 3.6: Moving one second average of computation time for receding horizon control implementation with different horizon times.

sponds to the average run time over the previous second of computation. There is a clear trend towards shorter average computation times as the time horizon T is made longer. There is also an initial transient increase in average computation time that is greater for shorter horizon times. In fact, the 6 second horizon controller exhibits a relatively constant average computation time. One explanation for this trend is that, for this particular test, a 6 second horizon is closer to what the system can actually do. After 1.5 seconds, the fan is still far from the desired hover position and the terminal cost is large, likely far from its region of attraction. Figure 3.7 shows the measured x response for these different controllers, exhibiting a rise time of 8-9 seconds independent of the controller. So a horizon time closer to the rise time results in a more feasible optimization in this case.

As noted in Section 3.3.2, smaller computation time naturally leads to better tracking performance as shown in Figure 3.8. The figure shows the tracking of θ for the

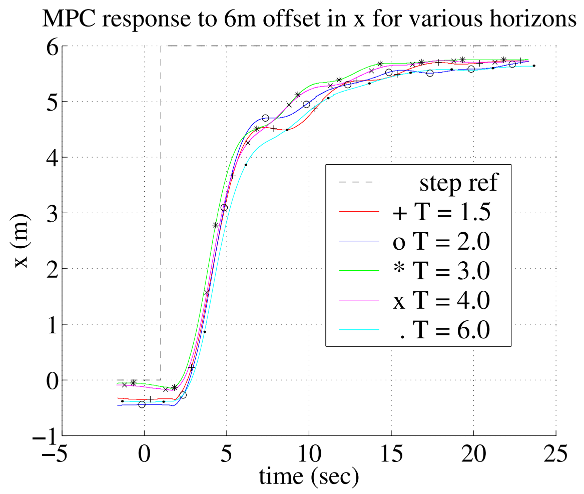


Figure 3.7: Response of receding horizon control laws to 6-meter offset in horizontal position x for different horizon times.

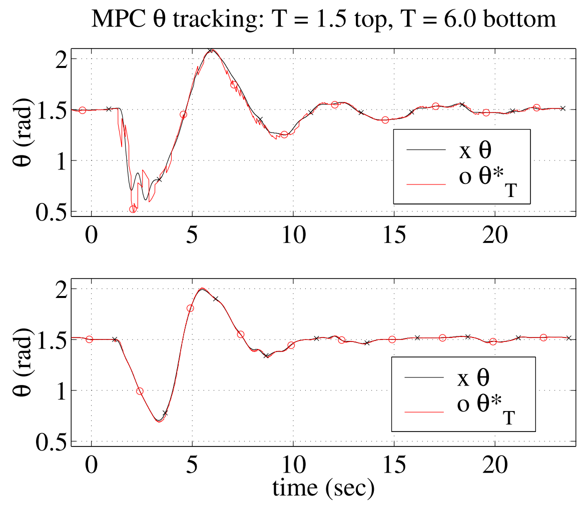


Figure 3.8: Position tracking for receding horizon control law with a 6-second horizon.

6 and 1.5 second horizon controllers, respectively. The dotted lines (o) represent the computed open-loop optimal trajectories θ^* and the solid lines (x) are the measured θ. Discontinuity in θ^* for $T = 1.5$ sec reaches 0.25 radians ($t = 2.5$ sec) while for $T = 6.0$ all states are as smooth as θ^* shown.

3.5.2 LQR vs. Receding Horizon Control

Position responses due to the horizontal offset for the static hover LQR controller, the gain-scheduled LQR controller and two receding horizon control laws are shown in Figure 3.9. The receding horizon control 6.0 second horizon response is the same as in the previous section. The 1.0 second horizon controller uses 11 breakpoints instead of 16, thereby reducing the computational demand. The LQR hover controller

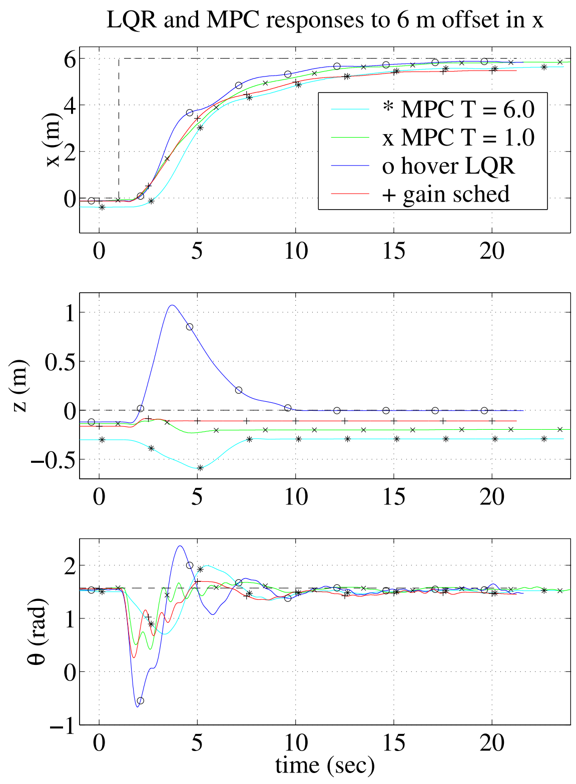


Figure 3.9: Response of LQR and receding horizon control to 6-meter offset in horizontal position x.

could not stabilize the fan for an offset in x bigger than 4 meters. The unstable response to the 6 meter offset is shown in the figure as y exceeds the constraint set of $[-1, 1]$. In this case, the fan crashed and then redirected to the desired position. All other controllers were stabilizing, with remarkable similarity between the gain-scheduled LQR and receding horizon control 1 second horizon controllers. For the receding horizon control laws, only the response of the fan is shown and not the open-loop optimal trajectories that were being tracked. In reference to Figure 3.6, the average computation profile for the 1 second horizon receding horizon control with 11 breakpoints looks like the 3 second horizon controller with 16 breakpoints. By reducing the number of breakpoints, and hence the computational demand, tracking is improved. The fan is thus stabilized along a path much different than the longer horizon path (observe the different θ responses). Clearly, there is a highly coupled interaction between horizon length and the number of breakpoints.

3.6 Summary

The Caltech ducted fan has been successfully realized as an experimental testbed for real-time receding horizon control implementation. The real-time trajectory generation package NTG made it possible to run receding horizon control laws at speeds necessary for closed-loop stabilization of a system with considerable dynamics and strict input constraints. Specifically, the thrust vectoring mechanism dynamics are modelled as a second order filter with frequency 4 Hz and the 6 second horizon receding horizon control ran faster than 10 Hz, a factor of 2.5 times greater than the actuator dynamics. Results show the success of different receding horizon control laws for stabilizing a step offset in x. A timing setup based on “compute as fast as possible” accounts for non-negligible computation time as it affects the application of the repeatedly updated optimal trajectories. The region of attraction of receding horizon control laws is shown to be larger than that of the static hover LQR controller. Moreover, the performance of some receding horizon control laws is close to that of the gain-scheduled LQR controller.

Extensions of this work could include a parametric study to better understand the nontrivial coupled relationship between the horizon length and number of breakpoints. More recent experimental efforts made it possible to remove the inner-loop controller on the θ dynamics [51]. Also, it seems worthwhile to explore the effect of applying a higher density of breakpoints over the time interval for which the optimal trajectories are applied.

Chapter 4

Distributed Receding Horizon Control

4.1 Introduction

Chapter 2 presented a receding horizon control law with sufficient conditions that guarantee closed-loop stabilization. Such receding horizon control laws have traditionally been applied to systems governed by dynamics that are sufficiently slow enough to permit the online optimal control computations. However, the work presented in Chapter 3 demonstrated that receding horizon control is a viable approach for stabilization of faster vehicle dynamics, specifically those of an indoor flight control experiment. The NTG software package is what made such a demonstration possible, by solving the optimal control problems with ample efficiency.

As discussed in the introductory Chapter 1, we are interested in the application of receding horizon control to multiagent systems. While the conditions in Chapter 2 guarantee stabilization, the control law requires centralized computations and information. Specifically, the centralized computation at every receding horizon update requires the initial condition of every agent.

A centralized implementation of receding horizon control to a multiagent system has several disadvantages. For one, multiagent systems usually operate in distributed environments, mandating that the control law require little, if any, central coordination. Also, the control approach should be distributed to enable autonomy among the individual agents. Lastly, the computational and communication cost of implementing a centralized receding horizon control law may be too expensive in many multiagent applications. Thus, we are interested in generating a distributed implementation of receding horizon control. Not only should the distributed implementation enable autonomy of the individual agents, but it should also improve the tractability of the corresponding centralized implementation.

The contribution of this chapter is to define such a distributed implementation. By this implementation, each agent is autonomous, in the sense that each computes a control for itself. Also, the only centralized coordination is that the control updates occur globally synchronously. Relaxation of the synchronous timing requirement is explored in the next chapter. Aside from synchronous timing, all communication exchanges between agents are purely local. An outline of the chapter is now presented to aid in comprehension of the main ideas behind the distributed implementation.

4.2 Outline of the Chapter

The contribution of this chapter is to take the centralized receding horizon control law presented in Chapter 2 and generate a distributed implementation. A distinctive feature of the distributed implementation presented here is that it is *designed for analysis*, i.e., the distributed computations and information exchanges are specifically designed to enable the closed-loop stability analysis.

In Section 4.3, the assumed structure of the centralized problem is presented. Specifically, the overall system dynamics are assumed to be comprised of dynamically decoupled subsystems, an assumption appropriate for many multiagent systems, e.g., multi-vehicle formations. The coupling of the individual agents is assumed to occur in a quadratic integrated cost function in the centralized optimal control problem. The theory admits more general coupling cost functions, but the presentation here is facilitated by this choice and a more general cost is presented in the next chapter. The coupling cost is decomposed to define the distributed integrated costs, one for each agent.

Section 4.4 presents the distributed optimal control problems, one for each agent. Each such problem incorporates the dynamics of the individual agent and the distributed integrated cost. As with the centralized formulation, the distributed terminal cost functions and terminal constraints are designed to ensure stability. In this case, the stability analysis is facilitated by designing the terminal cost functions and terminal constraints to be decoupled. Each optimal control problem is then augmented with an additional *compatibility constraint*. Loosely speaking, the constraint ensures that no agent will diverge too far from the behavior that others expect of it. The compatibility constraint is a central element to the stability analysis. The information exchanged between coupled agents is also defined in this section.

Section 4.6 presents the closed-loop stability analysis. Each distributed optimal control problem has a common horizon time and update period. The main result of this section is that the common update period determines how close the closed-loop system can get to meeting the overall stabilization objective. The smaller the update period, the closer the closed-loop system gets to the desired equilibrium. Only in the limit that the update period shrinks to zero can convergence to the desired equilibrium be guaranteed.

The chapter is concluded with a summary of the results. In the next chapters, the distributed receding horizon control law is analyzed and applied in simulation experiments.

4.3 Structure of Centralized Problem

The notation in Chapter 2 describes the centralized system dynamics as

$$\dot{z}(t) = f(z(t), u(t)), \tag{4.1}$$

with state $z(t) \in \mathbb{R}^{nN_a}$ and control $u(t) \in \mathbb{R}^{mN_a}$, where the dimension of the state and control is redefined here. The dimension notation is useful in assuming that the centralized system decouples into N_a subsystems as $z = (z_1, ..., z_{N_a})$, $u = (u_1, ...u_{N_a})$, and $f = (f_1, ..., f_{N_a})$, with

$$\dot{z}_i(t) = f_i(z_i(t), u_i(t)), \quad \text{for each } i = 1, ..., N_a.$$

Henceforth, each subsystem is referred to as an *agent*, and $z_i(t) \in \mathbb{R}^n$ and $u_i(t) \in \mathbb{R}^m$ are the state and control of agent i, respectively. The results that follow do not require that the dimension of the state (or control) of each agent be the same; this is assumed solely for notational ease. The control objective is to stabilize the agents to the equilibrium point z^c, which we partition as $z^c = (z_1^c, ..., z_{N_a}^c)$. We assume each agent i is subject to decoupled state and input constraints $z_i(t) \in \mathcal{Z} \subseteq \mathbb{R}^n$ and $u_i(t) \in \mathcal{U} \subset \mathbb{R}^m$. Coupling constraints between neighboring states will be explored in Section 5.4. Assumptions on the dynamics and constraints are not stated until the next section.

Remark 4.1 We shall use the norm $\|x\|$ to denote the Euclidean norm of any vector $x \in \mathbb{R}^n$. In cases where x is a curve, we abuse the notation $\|x\|$ to mean $\|x(t)\|$ at some instant of time t.

The assumed structure of the single cost function that couples agents is now described. In Section 5.4.1, a general cost function that couples the states of agents is defined, where the costs on the control of every agents are assumed to be decoupled. There, the coupling cost on the states need not be quadratic or convex. For the presentation in this chapter, we elect to examine the case where the coupling cost on the states is quadratic. One reason is that this choice is consistent with the centralized cost in the optimal control problem of Chapter 2, the problem for which we are generating a distributed implementation in the first place. Other reasons are that a quadratic cost facilitates the analysis that follows and is exactly the cost that arises in the multi-vehicle formation stabilization example explored in Chapter 6.

The centralized integrated cost on the control inputs is assumed to decompose as

$$\|u\|_R^2 = \sum_{i=1}^{N_a} \|u_i\|_{R_i}^2,$$

with $R = \text{diag}(R_1, ..., R_{N_a})$ and each $R_i \in \mathbb{R}^{m\times m}$ is positive definite. The coupling between agents is assumed to occur in the cost function on the state. Specifically, the integrated cost on the state has the general form

$$\|z - z^c\|_Q^2 = \|G(z - z^c)\|^2 \ ,$$

where $Q = G^T G > 0$ is symmetric. The matrix $G \in \mathbb{R}^{nM\times nN_a}$ is partitioned into $n \times n$ block matrices as $G = [G_{i,j}]_{\ i=1,...,M,\ j=1,...,N_a}$, with $G_{i,j} \in \mathbb{R}^{n\times n}$. The dimension M must satisfy $M \geq N_a$ so that $\text{rank}(Q) = nN_a$, and therefore Q is positive definite. We now state an assumption on the blocks of G to simplify the analysis that follows.

Assumption 4.1 Each $n \times n$ block of G satisfies *rank*$(G_{i,j}) \in \{n, 0\}$. That is, each block is either full rank or the $n \times n$ null matrix.

Now, for each agent i, we want to identify the coupling in the cost above between z_i and the states of other agents. If a block row $[G_{j,1} \cdots G_{j,N_a}]$ of G has $G_{j,i} \neq 0$ and $G_{j,k} \neq 0$ for some $k \neq i$, then agents k and i are clearly coupled. Moreover, by Assumption 4.1, $G_{j,i}$ and $G_{j,k}$ have rank n in that block row, so every component of the vector z_i is coupled to at least one component of z_j. To generalize, the nonzero block entries in any block row identifies the coupling of full state vectors between agents. Agents that are coupled in the cost function this way are referred to as *neighbors*. For any agent $i = 1, ..., N_a$, we can identify the set of neighbors of i as

$$\mathcal{N}_i = \Big\{ j \in \{1, ..., N_a\} \setminus \{i\} \ : \ \exists k \in \{1, ..., N_a\} \text{ s.t. } G_{k,i} \neq 0 \text{ and } G_{k,j} \neq 0 \Big\}.$$

Note that the neighbor relationship between agents is mutual by this definition, i.e., $i \in \mathcal{N}_j$ if and only if $j \in \mathcal{N}_i$.

Remark 4.2 Assumption 4.1 is not necessary for the results of this chapter to hold. The assumption merely facilitates the decomposition of the general, quadratic centralized cost above into distributed costs. Without the assumption, it is possible that only components of neighboring agents states are coupled, and the generalization becomes more complicated in that case. Note that in the multi-vehicle formation stabilization example of Chapter 6, neighboring agents do not have full state coupling, and the decomposition is straightforward. Also, one of the centralized costs in that chapter is not even quadratic. So, it is understood that while the results are applicable to a larger class of cost functions, we make assumptions here to mitigate notational complexity that can only get in the way of the underlying concepts.

Ultimately, the goal is to decompose the centralized quadratic cost above into a sum of quadratic costs, one for each agent. The resulting quadratic cost for each agent should depend only upon that agents state and the state of neighboring agents. To that end, we first construct a matrix $\overline{Q}_i$ for each $i = 1, ..., N_a$ as follows:

1. For $j = 1, ..., M$, take each block row $[G_{j,1} \cdots G_{j,N_a}]$ of G that has $G_{j,i} \neq 0$. Append these block rows into a new matrix W_i, with $\dim(W_i) = nM_i \times nN_a$, and partition W_i into $n \times n$ block matrices as $W_i = [W_{i,j}]_{\; i=1,...,M_i, \; j=1,...,N_a}$.

2. For $j = 1, ..., N_a$, remove every block column $[W_{1,j}; \; \cdots ; W_{M_i,j}]$ in W_i that is identically zero. Denote the reduced matrix V_i, with $\dim(V_i) = nM_i \times nN_i$, and partition V_i into $n \times n$ block matrices as $V_i = [V_{i,j}]_{\; i=1,...,M_i, \; j=1,...,N_i}$. Note that $N_i = |\mathcal{N}_i| + 1$.

3. If $i \neq 1$, redefine V_i by replacing the 1st block column $[V_{1,1}; \; \cdots ; V_{M_i,1}]$ with the ith block column $[V_{1,i}; \; \cdots ; V_{M_i,i}]$, and vice versa.

4. For $k = 1, ..., M_i$, rescale each block row $[V_{k,1} \cdots V_{k,N_i}]$ of V_i by dividing by the scalar number of block matrices $V_{k,j}$, $j = 1, ..., N_i$, that are non-zero. Denote the resulting matrix $G_i \in \mathbb{R}^{nM_i \times nN_i}$ and define $\overline{Q}_i = G_i^T G_i \in \mathbb{R}^{nN_i \times nN_i}$.

Remark 4.3 From Assumption 4.1, $rank(G_i) = n \; (min\{M_i, N_i\})$, so if $M_i \geq N_i$ then $\overline{Q}_i$ is positive definite.

Let $z_{-i} = (z_{j_1}, ..., z_{j_{|\mathcal{N}_i|}})$ denote the vector of states of the neighbors of i, i.e., $j_k \in \mathcal{N}_i, \; k = 1, ..., |\mathcal{N}_i|$, where the ordering of the states is consistent with the ordering of coupling in G_i. Also, denote $z^c_{-i} = (z^c_{j_1}, ..., z^c_{j_{|\mathcal{N}_i|}})$. We can now write

$$\|z - z^c\|^2_Q = (z - z^c)^T G^T G(z - z^c) = \sum_{i=1}^{N_a} L^z_i(z_i, z_{-i}) \tag{4.2}$$

$$\text{where} \quad L^z_i(z_i, z_{-i}) = \left\| \begin{bmatrix} z_i - z^c_i \\ z_{-i} - z^c_{-i} \end{bmatrix} \right\|^2_{\overline{Q}_i}.$$

Note that $Q \neq \text{diag}(\overline{Q}_1, ..., \overline{Q}_{N_a})$, and in fact these matrices are not even the same dimension. Step 3 in the construction of G_i is done so that G_i multiplies the vector $(z_i - z^c_i, z_{-i} - z^c_{-i})$ in the definition for L^z_i. The scaling in step 4 is done to preserve the summation in equation (4.2). In words, if a block row of G couples four agents, for example, and each agent accounts for this coupling block row in the construction of their respective G_i matrices, then the row must be multiplied by $\frac{1}{4}$ to preserve the summation. We can now define the distributed integrated cost that will be included in the local optimal control problem of each agent.

Definition 4.1 For each agent $i = 1, ..., N_a$, the *distributed integrated cost* is

$$L_i(z_i, z_{-i}, u_i) = \gamma \left(L^z_i(z_i, z_{-i}) + \|u_i\|^2_{R_i} \right),$$

where L^z_i is given in equation (4.2) and the constant γ common to each cost is chosen such that $\gamma \in (1, \infty)$.

From the definition it follows that

$$\sum_{i=1}^{N_a} L_i(z_i, z_{-i}, u_i) = \gamma \left(\|z - z^c\|^2_Q + \|u\|^2_R \right). \tag{4.3}$$

Remark 4.4 From the definition above, every distributed integrated cost has a common multiplying factor equal to γ, a constant chosen to be larger than one. The purpose of the constant $\gamma > 1$ is that it provides a stability margin for distributed receding horizon control. What we mean by stability margin will be made clear by Lemma 4.4, a result used in the stability analysis of Section 4.6.

To clarify the construction of the distributed integrated costs, an example of generating each L_i^z is now given.

Example 4.1 Consider $N_a = 3$, $z_i \in \mathbb{R}^2$ for each agent, let I denote the 2×2 identity matrix and let $z^c = 0$. Assume a centralized cost with the corresponding G matrix

$$G = \begin{bmatrix} I & 0 & 0 \\ -I & I & 0 \\ 0 & -I & I \end{bmatrix}.$$

The sets of neighbors for each agent are thus $\mathcal{N}_1 = \{2\}$, $\mathcal{N}_2 = \{1, 3\}$ and $\mathcal{N}_3 = \{2\}$. Proceeding with the construction for G_1, we have

$$W_1 = \begin{bmatrix} I & 0 & 0 \\ -I & I & 0 \end{bmatrix} \quad \rightarrow \quad V_1 = \begin{bmatrix} I & 0 \\ -I & I \end{bmatrix} \quad \rightarrow \quad G_1 = \begin{bmatrix} I & 0 \\ -I/2 & I/2 \end{bmatrix}.$$

Proceeding with the construction for G_2, we have

$$W_2 = \begin{bmatrix} -I & I & 0 \\ 0 & -I & I \end{bmatrix} \quad \rightarrow \quad V_2 = \begin{bmatrix} I & -I & 0 \\ -I & 0 & I \end{bmatrix} \quad \rightarrow \quad G_2 = \frac{1}{2} V_2.$$

Similarly, we obtain $G_3 = [I/2 \;\; -I/2]$. To confirm that the decomposition preserves the quadratic summation in equation (4.2), observe that

$$\|z - z^c\|_Q^2 = \|z_1\|^2 + \|z_2 - z_1\|^2 + \|z_3 - z_2\|^2.$$

Also,

$$\begin{aligned} L_1^z(z_1, z_{-1}) &= \|z_1\|^2 + \frac{1}{2}\|z_2 - z_1\|^2 \\ L_2^z(z_2, z_{-2}) &= \frac{1}{2}\left(\|z_2 - z_1\|^2 + \|z_3 - z_2\|^2\right) \\ L_3^z(z_3, z_{-3}) &= \frac{1}{2}\|z_3 - z_2\|^2. \end{aligned}$$

Clearly, the summation is preserved. This concludes the example.

Remark 4.5 By construction, the weighting matrices $\overline{Q}_i$ in equation (4.2) are unique, given Q. However, there are an infinite number of weighting matrices that preserve the summation, specifically by changing the scaling in step 4 of constructing each G_i. The stability results that follow do not depend on equal scaling of terms between neighboring agents, as is done in the construction of $\overline{Q}_i$ above. What is required is that the distributed integrated costs sum up to be the centralized cost, multiplied by a factor (γ) greater than one. The weighting will of course affect the performance of the closed-loop system, so making the scaling lop-sided would result in one agent reacting more to the term than the corresponding neighbors.

The optimal control problems assigned to each agent for a distributed receding horizon implementation are defined in the next section.

4.4 Distributed Optimal Control Problems

Given the decoupled system dynamics (4.1), decoupled constraints and coupled cost function defined in the previous section, we here define the distributed optimal control problems to be implemented in a distributed receding horizon fashion. Assumptions used to prove stability in the next section are also stated.

Let the set $\mathcal{Z}^N$ denote the N-times Cartesian product $\mathcal{Z} \times \cdots \times \mathcal{Z}$, so $z(t) \in \mathcal{Z}^{N_a}$ and $u(t) \in \mathcal{U}^{N_a}$, for all time $t \in \mathbb{R}$. As in Chapter 2, an *admissible control* for any agent $i = 1, ..., N_a$ is any piecewise, right-continuous function $u_i(\cdot) : [0, T] \to \mathcal{U}$, for any $T \geq 0$, such that given an initial state $z_i(0) \in \mathcal{Z}$, the control generates the state trajectory $z_i(t; z_i(0)) \in \mathcal{Z}$ for all $t \in [0, T]$.

Assumption 4.2 The following holds:

(i) for all $i = 1, ..., N_a$, the function $f_i : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is twice-continuously differentiable, satisfies $0 = f_i(z_i^c, 0)$ and f_i linearized around $(z_i^c, 0)$ is stabilizable;

(ii) the system (4.1) has a unique, absolutely continuous solution for any initial condition $z(0) \in \mathcal{Z}^{N_a}$ and any admissible control;

(iii) the set $\mathcal{U} \subset \mathbb{R}^m$ is compact, convex and contains the origin in its interior, and the set $\mathcal{Z} \subseteq \mathbb{R}^n$ is convex, connected and contains z_i^c in its interior for all $i = 1, ..., N_a$;

(iv) for each $i = 1, ..., N_a$, full state measurement $z_i(t)$ is available, the model $\dot{z}_j = f_j(z_i, u_j)$ for each neighbor $j \in \mathcal{N}_i$ is known, and computational time of the optimal control problem is negligible compared to the evolution of the closed-loop dynamics.

Let $u_{\max}$ be the positive scalar

$$u_{\max} = \left\{ \max \|v(t)\| \;\middle|\; v(t) \in \mathcal{U} \text{ is any admissible control, for any } t \in \mathbb{R} \right\},$$

known to exist by the Maximum-Minimum Theorem [43] since $\mathcal{U}$ is compact.

Definition 4.2 The *control objective* is to cooperatively asymptotically stabilize all agents to $z^c = (z_1^c, ..., z_{N_a}^c) \in \mathcal{Z}^{N_a}$.

The cooperation is achieved by the minimization of the distributed integrated cost functions given in Definition 4.1, as incorporated in the optimal control problems defined below.

In the following, we assume that all optimal control problems, one for each agent, will be solved at common instants time, i.e., globally synchronously. The common receding horizon update times are denoted $t_k = t_0 + \delta k$, where $k \in \mathbb{N} = \{0, 1, 2, ...\}$. Due to the coupling in the cost function with neighbors, then, each agent is required to *assume* some state trajectory for each neighbor over the current optimization horizon, since the actual state trajectory for that neighbor is not available at present. To address the need for assumed state trajectories, prior to an update, each agent computes its own *assumed control* trajectory, transmits the trajectory to neighbors and similarly receives assumed control trajectories from neighbors. Given the model and current state for each neighbor, each agent can then compute the required *assumed state* trajectories. To ensure *compatibility* between the assumed and actual state trajectories, in the distributed optimal control problem of each agent, the optimized state trajectory is compared to the assumed state trajectory of the corresponding agent in a constraint. Before defining the optimal and assumed trajectories mathematically, we introduce some notation.

Definition 4.3 Over any interval of time $[t_k, t_k + T]$, $k \in \mathbb{N}$, in the optimal control problem for each agent $i = 1, ..., N_a$, associated with the initial state $z_i(t_k)$ we denote:

applied control	$u_i(\cdot; z_i(t_k))$	the control being optimized and applied to the system over the subinterval $[t_k, t_k + \delta]$;
assumed control	$\hat{u}_i(\cdot; z_i(t_k))$	the control which every neighbor $j \in \mathcal{N}_i$ assumes i is employing over the interval.

The state trajectories corresponding to the applied and assumed controls are denoted $z_i(\cdot; z_i(t_k))$ and $\hat{z}_i(\cdot; z_i(t_k))$, respectively. For each agent i, given the current state $z_j(t_k)$ and assumed control $\hat{u}_j(s; z_j(t_k))$, $s \in [t_k, t_k + T]$, of every neighbor $j \in \mathcal{N}_i$, the assumed state trajectory $\hat{z}_j(s; z_j(t_k))$, $s \in [t_k, t_k + T]$, is computed using the dynamic model $\dot{z}_j = f_j(z_j, u_j)$ for that agent. An important point is that the initial condition of every assumed state trajectory is equal to the actual initial state value of the corresponding agent, i.e.,

$$\hat{z}_i(t_k; z_i(t_k)) = z_i(t_k; z_i(t_k)) = z_i(t_k)$$

for every $i = 1, ..., N_a$. To be consistent with the notation z_{-i}, let $\hat{z}_{-i}(\cdot; z_{-i}(t_k))$ and $\hat{u}_{-i}(\cdot; z_{-i}(t_k))$ be the vector of assumed neighbor states and controls, respectively, of agent i. With consistent initial conditions then, $\hat{z}_{-i}(t_k; z_{-i}(t_k)) = z_{-i}(t_k)$.

The distributed optimal control problems are now defined. Common to each problem, we are given the constant $\gamma \in (1, \infty)$ from Definition 4.1, update period $\delta \in (0, T)$ and fixed horizon time T. Conditions will be placed on the update period δ in the next section to guarantee stability of the closed-loop system. The collection of distributed open-loop optimal control problems is

Problem 4.1 For every agent $i = 1, ..., N_a$ and at any update time t_k, $k \in \mathbb{N}$, given

$z_i(t_k)$, $z_{-i}(t_k)$, and $\hat{u}_i(s; z_i(t_k))$ and $\hat{u}_{-i}(s; z_{-i}(t_k))$ for all $s \in [t_k, t_k + T]$, find

$$J_i^*(z_i(t_k), z_{-i}(t_k), T) = \min_{u_i(\cdot)} J_i(z_i(t_k), z_{-i}(t_k), u_i(\cdot; z_i(t_k)), T),$$

where $J_i(z_i(t_k), z_{-i}(t_k), u_i(\cdot; z_i(t_k)), T)$ is equal to

$$\int_{t_k}^{t_k+T} L_i(z_i(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)), u_i(\tau; z_i(t_k))) \, \mathrm{d}\tau \; + \; \gamma \| z_i(t_k + T; z_i(t_k)) - z_i^c \|_{P_i}^2,$$

subject to

$$\left.\begin{aligned} &\dot{z}_i(s; z_i(t_k)) = f_i(z_i(s; z_i(t_k)), u_i(s; z_i(t_k))) \\ &\dot{\hat{z}}_j(s; z_j(t_k)) = f_j(\hat{z}_j(s; z_j(t_k)), \hat{u}_j(s; z_j(t_k))), \ \forall j \in \mathcal{N}_i \\ &u_i(s; z_i(t_k)) \in \mathcal{U} \\ &z_i(s; z_i(t_k)) \in \mathcal{Z} \\ &\| z_i(s; z_i(t_k)) - \hat{z}_i(s; z_i(t_k)) \| \leq \delta^2 \kappa \end{aligned}\right\} \quad s \in [t_k, t_k + T],$$

$$z_i(t_k + T; z_i(t_k)) \in \Omega_i(\varepsilon_i), \tag{4.4}$$

for a given positive constant $\kappa \in (0, \infty)$, $P_i = P_i^T > 0$ and where

$$\Omega_i(\varepsilon_i) := \left\{ z \in \mathbb{R}^n \ : \ \| z - z_i^c \|_{P_i}^2 \leq \varepsilon_i, \ \varepsilon_i \geq 0 \right\}.$$

As part of the optimal control problem, the optimized state for i is constrained to be at most a distance of $\delta^2 \kappa$ from the assumed state. We refer to this constraint as the *compatibility constraint*. Naturally, the constraint is a means of enforcing a degree of consistency between what an agent is actually doing and what neighbors believe that agent is doing.

Remark 4.6 Instead of communicating assumed controls, neighboring agents could compute and transmit the corresponding assumed state, since that is what each distributed optimal control problem depends upon. That would remove the need for the differential equation of each neighbor, simplifying each local optimization problem.

We shall discuss this further in Section 5.3.

The terminal cost functions and terminal constraints are completely decoupled, which facilitates the proof of stability. While the coupled centralized integrated cost has been accounted for by the decomposition in the previous section, the centralized terminal cost and constraint sets are not incorporated in the distributed optimal problems, although both in general involve inter-agent state coupling as well. This highlights the fact that the centralized problem is *not* begin solved here. Instead, we are formulating an alternative solution to the problem, with a distributed computation and communication structure.

The optimal solution to each distributed optimal control problem, assumed to exist, is denoted

$$u^*_{di}(\tau; z_i(t_k)), \; \tau \in [t_k, t_k + T].$$

The closed-loop system, for which stability is to be guaranteed, is

$$\dot{z}(\tau) = f(z(\tau), u^*_{\text{dist}}(\tau)), \quad \tau \geq t_0, \tag{4.5}$$

where the applied *distributed receding horizon control law* is

$$u^*_{\text{dist}}(\tau; z(t_k)) = (u^*_{d1}(\tau; z_1(t_k)), ..., u^*_{dN_a}(\tau; z_{N_a}(t_k))), \quad \tau \in [t_k, t_k + \delta),$$

for $0 < \delta < T$ and $k \in \mathbb{N}$. As before, the receding horizon control law is updated when each new initial state update $z(t_k) \leftarrow z(t_{k+1})$ is available. The optimal state for agent i is denoted $z^*_{di}(\tau; z_i(t_k))$, for all $\tau \in [t_k, t_k + T]$. The concatenated vector of the distributed optimal states is denoted

$$z^*_{\text{dist}}(\tau; z(t_k)) = (z^*_{d1}(\tau; z_1(t_k)), ..., z^*_{dN_a}(\tau; z_{N_a}(t_k))),$$

for all $\tau \in [t_k, t_k + T]$. Although we denote the optimal control for agent i as $u^*_{di}(\tau; z_i(t_k))$, it is understood that this control is implicitly dependent on the initial state $z_i(t_k)$ and the initial states of the neighbors $z_{-i}(t_k)$.

Before stating the computation of the assumed control $\hat{u}_i(s; z_i(t_k))$ for each agent i, a decoupled terminal controller associated with each terminal cost and constraint set needs to be defined. For the moment, we define not only the terminal controllers, but give the needed assumptions on the controllers as well as the other parameters in Problem 4.1, akin to Assumption 2.2 stated for the centralized implementation. The linearization of the subsystem i in equation (4.1) at $(z_i, u_i) = (z_i^c, 0)$ is denoted

$$\dot{z}_i(t) = A_i z_i(t) + B_i u_i(t), \quad A_i = \frac{\partial f_i}{\partial z_i}(z_i^c, 0), \quad B_i = \frac{\partial f_i}{\partial u_i}(z_i^c, 0).$$

By assuming stabilizability, a linear feedback $u_i = K_i(z_i - z_i^c)$ can be found such that $A_i + B_i K_i$ has all eigenvalues in the open left-half complex plane.

Assumption 4.3 The following conditions are satisfied for every $i = 1, ..., N_a$:

(i) the largest positive constant $\varepsilon_i > 0$ is chosen such that $\Omega_i(\varepsilon_i) \subseteq \mathcal{Z}$ is invariant for the closed-loop nonlinear system $\dot{z}_i = f_i(z_i, K_i(z_i - z_i^c))$, and such that for all $z_i \in \Omega_i(\varepsilon_i)$, the stabilizing feedback is feasible, i.e., $u_i = K_i(z_i - z_i^c) \in \mathcal{U}$. In addition, given $R_i > 0$, the matrices K_i and $P_i = P_i^T > 0$ satisfy

$$(A_i + B_i K_i)^T P_i + P_i(A_i + B_i K_i) = -(Q_i + K_i^T R_i K_i),$$

with $Q_i \in \mathbb{R}^{n \times n}$ chosen such that $\widehat{Q} = diag(Q_1, ..., Q_{N_a})$ satisfies $\widehat{Q} \geq Q$, where Q is the weighting for the centralized integrated cost;

(ii) at any receding horizon update time, the collection of open-loop optimal control problems in Problem 4.1 is solved globally synchronously;

(iii) communication of control trajectories between neighboring agents is lossless.

The receding horizon control law is employed for all time after the initialization and the decoupled linear feedbacks K_i are not employed, even after agent i enters $\Omega_i(\varepsilon_i)$. The value for ε_i simply determines the size of the set over which the conditions in Assumption 4.3 are satisfied. Existence of each $\varepsilon_i \in (0, \infty)$ is guaranteed by Lemma

2.1, as in the centralized case. It may be desired that an agent switch from the receding horizon controller to the decoupled linear feedbacks once inside the terminal constraint set, known as dual-mode receding horizon control in the literature [49]. Generally, ε_i could be chosen to satisfy the conditions in Assumption 4.3 and the additional condition

$$\Omega_i(\varepsilon_i) \bigcap \Omega_j(\varepsilon_j) = \emptyset, \quad \text{for all } i, j = 1, ..., N_a, \ i \neq j,$$

so that once inside $\Omega_i(\varepsilon_i)$, any agent i is closer to its objective state than any other agents objective state. In that case, the agents employ the decoupled feedbacks only if they are close enough to their objective state and far enough from any other agents objective state.

Remark 4.7 Due to condition (ii) above, the distributed receding horizon control laws are not technically decentralized, since a globally synchronous implementation requires centralized clock keeping [3]. A locally synchronous, and consequently decentralized, version will be discussed in Section 5.4.

Although not employed, the decoupled linear feedbacks K_i can asymptotically stabilize each agent to its objective state once the agent enters the decoupled terminal constraint set (4.4). One choice for each Q_i that would satisfy $\widehat{Q} \geq Q$ is $Q_i = \lambda_{\max}(Q) I_{(n)}$, where $\lambda_{\max}(Q)$ is the maximum eigenvalue of the symmetric matrix Q. Define

$$\widehat{K} = \text{diag}(K_1, ..., K_{N_a}), \quad \widehat{P} = \text{diag}(P_1, ..., P_{N_a}),$$

$$A = \text{diag}(A_1, ..., A_{N_a}), \quad B = \text{diag}(B_1, ..., B_{N_a}).$$

As a consequence of assumption (i) above, $\widehat{P}$ is positive definite and symmetric, and satisfies

$$\left(A + B\widehat{K}\right)^T \widehat{P} + \widehat{P}\left(A + B\widehat{K}\right) = -\left(\widehat{Q} + \widehat{K}^T R \widehat{K}\right) \leq -\left(Q + \widehat{K}^T R \widehat{K}\right). \tag{4.6}$$

For the centralized implementation of Chapter 2, it is straightforward to show that if the optimization problem is feasible at initialization, then all subsequent receding horizon optimization calculations have a feasible solution, in the absence of explicit uncertainty. The feasible trajectory at each update is simply the remainder of the trajectory from the previous update, concatenated with the trajectory resulting from the feedback defined inside the terminal constraint set. For the distributed implementation here, initial feasibility likewise ensures subsequent feasibility, a fact proven in the next section. A natural question is the following: how do we initialize the distributed receding horizon control implementation?

Compared to the centralized implementation, initialization of the distributed implementation is more complicated. In the centralized implementation, initialization requires that a feasible solution to the optimization problem be found. In the distributed implementation, initialization requires that each distributed optimization problem have an assumed control for every neighbor, *prior to* solving for an initially feasible solution in each distributed problem. Thus, we are interested in choosing an assumed control for every agent, to be used at the initialization phase of the distributed implementation. The time at which initialization occurs is denoted $t_{-1} = t_0 - \delta$. The assumed control for each agent at initialization is based on the following assumption.

Assumption 4.4 Given an initial state $z(t_{-1})$, the control $u_i(\tau; z_i(t_{-1})) = 0$ for all $\tau \in [t_{-1}, t_{-1} + T]$ is an admissible control for every agent $i = 1, ..., N_a$.

The assumption requires that zero control be admissible from the initial state $z(t_{-1})$. By definition, the admissible control must be such that the decoupled state constraints remain feasible over the specified time domain. It is not required that zero control respect any of the other constraints in Problem 4.1, namely, the terminal and compatibility constraints on the state. Given the assumed control for every agent, the initialization phase for the distributed implementation can now be defined.

Definition 4.4 (*Initialization*) At time t_{-1}, solve Problem 4.1 with initial state $z(t_{-1})$, setting $\hat{u}_i(\tau; z_i(t_{-1})) = 0$ for all $\tau \in [t_{-1}, t_{-1} + T]$ and every $i = 1, ..., N_a$, and also setting $\kappa = +\infty$. The resulting optimal trajectories are denoted $u^*_{di}(\tau; z_i(t_{-1}))$

and $z^*_{di}(\tau; z_i(t_{-1}))$, $\tau \in [t_{-1}, t_{-1}+T]$, for every $i = 1, ..., N_a$. The optimal control $u^*_{\text{dist}}(\tau; z(t_{-1}))$ is applied for $\tau \in [t_{-1}, t_0)$.

At initialization, the compatibility constraints are not necessary. Consequently, the constraints are effectively removed by setting κ to a large number in every distributed optimization problem. The reason that the constraints are not necessary at initialization is that, while the stability analysis presented in the next section depends critically on the compatibility constraints, the closed-loop system behavior is examined for time $t \geq t_0$. Thus, the initialization phase generates optimal controls that are assumed to drive the state to $z(t_0)$, at which time the compatibility constraints are enforced in each distributed optimal control problem, for all subsequent receding horizon updates.

Now that initialization has been defined, the assumed control for each agent can be defined for all receding horizon updates after initialization.

Definition 4.5 (*Assumed Control*) For each agent $i = 1, ..., N_a$ and for any $k \in \mathbb{N}$, the assumed control $\hat{u}_i(\cdot; z_i(t_k)) : [t_k, t_k+T] \to \mathcal{U}$ is defined as follows:

if $z(t_k) = z^c$, then $\hat{u}_i(\tau; z_i(t_k)) = 0$, $\tau \in [t_k, t_k+T]$. Otherwise,

$$\hat{u}_i(\tau; z_i(t_k)) = \begin{cases} u^*_{di}(\tau; z_i(t_{k-1})), & \tau \in [t_k, t_{k-1}+T) \\ K_i\left(z_i^K(\tau - t_{k-1} - T; z_i^K(0)) - z_i^c\right), & \tau \in [t_{k-1}+T, t_k+T] \end{cases},$$

where $z_i^K(0) = z^*_{di}(t_{k-1}+T; z_i(t_{k-1}))$, and $z_i^K(s; z_i^K(0))$ is the closed-loop solution to

$$\dot{z}_i^K(s) = (A_i + B_iK_i)(z_i^K(s) - z_i^c), \quad s \geq 0, \quad \text{given } z_i^K(0).$$

Denote the assumed control vector as

$$\hat{u}(\cdot; z(t)) = (\hat{u}_1(\cdot; z_1(t)), ..., \hat{u}_{N_a}(\cdot; z_{N_a}(t))).$$

Practically, the assumed control for any agent i associate with initial state $z_i(t_k)$ is generated and transmitted to neighbors in the time window $[t_{k-1}, t_k)$, i.e., prior to time t_k. To state Definition 4.5 in words, in Problem 4.1, every agent is assuming all neighbors will continue along their previous optimal path, finishing with the decoupled linear control laws defined in Assumption 4.3, *unless* the control objective is met ($z(t_k) = z^c$) at any update time after initialization. In the latter case, neighbors are assumed to do nothing, i.e., apply zero control.

Remark 4.8 The test of whether $z(t_k) = z^c$ in generating the assumed control is a centralized test. The reason for the test is its use in the proof of Proposition 4.1 in Section 4.6. We note that the asymptotic stability result in the next section guarantees that only in the limit as $t_k \to \infty$ do we have $z(t_k) \to z^c$. Practically then, one could assume $z(t_k) \neq z^c$, which is true for any finite k when $z(t_{-1}) \neq z^c$, and ignore the test completely. Also, if dual-mode receding horizon control is used, the test can be removed, since Proposition 4.1 is not used to prove asymptotic stability in that case. A dual-mode version will be provided in Section 5.3.1.

Now that the distributed optimal control problems and the initialization procedure for the distributed implementation have been defined, the implementation algorithm can be stated.

4.5 Distributed Implementation Algorithm

The distributed optimal control problems are posed in Problem 4.1, with the initialization procedure given in Definition 4.4 and the assumed control given in Definition 4.5. We can now succinctly state the computations and information exchanges that define up the distributed receding horizon control law.

Definition 4.6 (*Distributed Implementation Algorithm*) At time t_{-1}, each agent follows the initialization procedure given in Definition 4.4. Over every receding horizon update interval $[t_{k-1}, t_k)$, $k \in \mathbb{N}$, each agent $i = 1, ..., N_a$:

1. implements the current optimal control trajectory $u^*_{di}(\tau; z_i(t_{k-1}))$, $\tau \in [t_{k-1}, t_k)$,

2. computes $\hat{u}_i(\cdot; z_i(t_k)) : [t_k, t_k + T] \to \mathcal{U}$, and

3. transmits $\hat{u}_i(\cdot; z_i(t_k))$ to every neighbor and receives $\hat{u}_j(\cdot; z_j(t_k))$ from every neighbor $j \in \mathcal{N}_i$.

At every receding horizon update time t_k, $k \in \mathbb{N}$, each agent $i = 1, ..., N_a$:

1. senses its own current state $z_i(t_k)$ and senses or receives the current state $z_j(t_k)$ of each neighbor $j \in \mathcal{N}_i$,

2. computes the assumed state trajectories $\hat{z}_i(\cdot; z_i(t_k))$ and $\hat{z}_j(\cdot; z_j(t_k))$, $j \in \mathcal{N}_i$,

3. computes the optimal control trajectory $u^*_{di}(\tau; z_i(t_k))$, $\tau \in [t_k, t_k + T]$.

Implicit in the algorithm is that the assumed control and state trajectories for each agent i are consistent in every optimization problem in which they occur, i.e., in the optimal control problem for agent i and for each neighbor $j \in \mathcal{N}_i$. The reason is that i is communicating the same assumed control to every neighbor and every neighbor has the same model for the dynamics of i, by Assumption 4.2. Notice that the communication of control trajectories between neighboring agents is not required to happen instantaneously, but over each receding horizon update time interval. Although the algorithm above implies computation of assumed states and optimal controls can happen instantaneously, at each time t_k, a predictive version can also be stated to account for non-trivial computation times, as discussed in Section 2.3.1.

Now that the distributed implementation has been stated, we observe the following about the distributed optimal value function $J^*_i(z_i(t_k), z_{-i}(t_k), T)$ for any $i = 1, ..., N_a$. If $J^*_i(z_i(t_{-1}), z_{-i}(t_{-1}), T) = 0$ at initialization for any agent i, then it can be shown (see proof of Proposition 4.1) that $z_i(t_{-1}) = z^c_i$ and $z_j(t_{-1}) = z^c_j$, for each neighbor $j \in \mathcal{N}_i$, i.e., the local objective has been met. However, even if $J^*_i(z_i(t_{-1}), z_{-i}(t_{-1}), T) = 0$ holds, it may not remain zero for all later time t_k, $k = 1, 2, ...$ An example is where i and all neighbors $j \in \mathcal{N}_i$ are initialized meeting their objective, but some $l \in \mathcal{N}_j$ has not met its objective. Thus, in the subsequent optimizations, j will react to l, followed by i reacting to j, since the coupling cost terms become nontrivial. Consequently, we cannot guarantee that each distributed optimal value function $J^*_i(z_i(t_k), z_{-i}(t_k), T)$ will decrease with each distributed receding horizon update. Instead, we show in the next section that the *sum of the distributed optimal value functions* is a Lyapunov function that does decrease at each update, enabling a proof that the distributed receding horizon control laws *collectively* meet the control objective.

4.6 Stability Analysis

We now proceed with analyzing the stability of the closed-loop system (4.5). At any time t_k, $k \in \mathbb{N}$, the sum of the optimal distributed value functions is denoted as

$$J^*_\Sigma(z(t_k), T) = \sum_{i=1}^{N_a} J^*_i(z_i(t_k), z_{-i}(t_k), T).$$

For stability of the distributed receding horizon control laws, we investigate $J^*_\Sigma(z(t_k), T)$ as a Lyapunov function.

Definition 4.7 Problem 4.1 is *feasible* at time t_k if for every $i = 1, ..., N_a$, there exists a control $u_i(\cdot; z_i(t_k)) : [t_k, t_k + T] \to \mathcal{U}$ such that all the constraints are satisfied and the value function $J_i(z_i(t_k), z_{-i}(t_k), u_i(\cdot), T)$ is bounded. Let $Z_\Sigma \subset \mathcal{Z}^{N_a}$ denote the *set of initial states for which Problem 4.1 is feasible at initialization*, as specified in Definition 4.4.

It is now shown that $z^c \in Z_\Sigma$ is an equilibrium point of the closed-loop system (4.5). For $z(t_{-1}) = z^c$, the feasible and optimal solution to Problem 4.1 is $u^*_{\text{dist}}(\tau; z^c) = 0$, for all $\tau \in [t_{-1}, t_{-1} + T]$. Since $f(z^c, 0) = 0$, subsequent receding horizon updates at any time t_k, $k \in \mathbb{N}$, produce $z(t_k) = z^c$ and $u^*_{\text{dist}}(\tau; z(t_k)) = 0$, for all $\tau \in [t_k, t_k + T]$. Thus, z^c is an equilibrium point of the closed-loop system (4.5).

By definition, if the state $z(t_{-1})$ is in the set Z_Σ, there is a feasible solution to Problem 4.1 at initialization. The next result says that, in that case, there is in fact a feasible solution to Problem 4.1 at every subsequent receding horizon update time t_k, $k \in \mathbb{N}$. In other words, feasibility an initialization implies subsequent feasibility, a property known also to hold for the centralized implementation. Thus, $z(t_{-1}) \in Z_\Sigma$ implies $z(t_k) \in Z_\Sigma$, for all $k \in \mathbb{N}$. The result in fact says something stronger, namely, that Z_Σ is a positively invariant set, i.e, the closed-loop state trajectory remains in Z_Σ for all time $t \geq t_{-1}$.

To state the invariance result, we must modify the definition of the assumed and applied controls after initialization. Aside from the statement below, the applied control is always the optimal control, over the application window $[t_k, t_{k+1})$, and the assumed control is given in Definition 4.5. Instead, for any time $\tau \in [t_k, t_{k+1})$ and any $k \in \mathbb{N}$, set the applied control to be the assumed control as $u_i(\tau; z_i(t_k)) = \hat{u}_i(\tau; z_i(t_k))$, where $\hat{u}_i(\tau; z_i(t_k))$ is redefined by replacing $u^*_{di}(\tau; z_i(t_{k-1}))$ with $u_i(\tau; z_i(t_{k-1}))$ in Definition 4.5. By definition, this simply means that each agent applies the entire open-loop optimal control found at initialization, concatenated with the decoupled feedback controls defined in Assumption 4.3. Since the assumed and applied controls are identical, the compatibility constraint is trivially satisfied in each distributed optimization problem, as are all other constraints. We can now state the invariance result.

Lemma 4.1 *Under Assumptions 4.2–4.4, the set Z_Σ is a positively invariant set with respect to the closed-loop system by redefining the applied and assumed controls as described above. Thus, feasibility at initialization implies subsequent feasibility.*

The proof follows immediately from Definitions 4.4 and 4.5. Note that the assumed control $\hat{u}_i$ is exactly the feasible control trajectory used in Lemma 2 of [11] to show initial feasibility implies subsequent feasibility of the online optimization problem in the centralized case. Also, it is easy to see that if $z(t_k) \in Z_\Sigma$, $z(\tau) \in Z_\Sigma$ for all $\tau \in [t_k, t_k + T]$, with the applied and assumed controls redefined as described above.

Now, we will be exploring the closed-loop behavior for initial states that start in Z_Σ, with the applied control equal to the optimal control, over each application window $[t_k, t_{k+1})$, and the assumed control given by Definition 4.5. For the purposes of numerical computation and for theoretical reasons that follow, it is required that the closed-loop state trajectory, initialized from any $z(t_{-1}) \in Z_\Sigma$, remain bounded. This condition and an optimality condition are stated here as an assumption.

Assumption 4.5 The following conditions are satisfied for any $k \in \mathbb{N}$:

(i) there exists a $\rho_{\max} \in (0, \infty)$ such that for any $i = 1, ..., N_a$,

$$\|z^*_{di}(\tau; t_k) - z^c_i\| \le \rho_{\max}, \quad \forall \tau \in [t_k, t_k + T]; \tag{4.7}$$

(ii) the optimal solution to Problem 4.1 exists and is numerically obtainable for any $z(t_k) \in Z_\Sigma$.

Given the assumptions above, we have the following result.

Lemma 4.2 *Under Assumptions 4.2–4.5, Z_Σ is a positively invariant set with respect to the closed-loop system (4.5). Thus, if $z(t_{-1}) \in Z_\Sigma$, $z^*_{dist}(\tau) \in Z_\Sigma$ for all $\tau \ge t_{-1}$.*

Proof: It is shown by contradiction that each applied piece of any open-loop trajectory (which exists by assumption) that starts in Z_Σ, stays in Z_Σ. Suppose the open-loop initialization trajectory starting from $z(t_{-1}) \in Z_\Sigma$ has left Z_Σ at some time $t' \in (t_{-1}, t_0]$, i.e., $z^*_{\text{dist}}(t'; z(t_{-1})) \notin Z_\Sigma$. A feasible control from that point is to simply continue along the remainder of $z^*_{\text{dist}}(\tau; z(t_{-1}))$, $\tau \in [t', t_{-1} + T)$, and apply the decoupled feedback controllers for time $\tau \in [t_{-1} + T, t' + T]$, which implies that $z^*_{\text{dist}}(t'; z(t_{-1})) \in Z_\Sigma$. By contradiction, then, the open-loop optimal trajectory starting from $z(t_{-1}) \in Z_\Sigma$ cannot leave Z_Σ at any time in the interval $[t_{-1}, t_0]$. Since this holds for any open-loop optimal trajectory over the applied interval $[t_k, t_{k+1}]$, for any $k \in \mathbb{N}$, the closed-loop trajectory satisfies $z^*_{\text{dist}}(\tau) \in Z_\Sigma$ for all $\tau \ge t_{-1}$, if $z(t_{-1}) \in Z_\Sigma$.
■

Lemma 4.1 says that Z_Σ is a positively invariant set, but required redefining the applied and assumed controls. In contrast, the result above says that Z_Σ is a positively invariant set, keeping the applied and assumed controls as defined in terms of the optimal control, but naturally requires the additional assumption that the optimal solution to Problem 4.1 is attained at every receding horizon update.

The next result says that the net objective of the distributed receding horizon control laws is consistent with the control objective.

Proposition 4.1 *Under Assumptions 4.2–4.5, for a given fixed horizon time $T > 0$ and at any time t_k, $k \in \mathbb{N}$,* *1. $J^*_\Sigma(z(t_k),T) \geq 0$ for any $z(t_k) \in Z_\Sigma$, and*

2. $J^*_\Sigma(z(t_k),T) = 0$ if and only if $z(t_k) = z^c$.

In addition, if the optimal control solution $u^*_{dist}(\cdot; z(t))$ to Problem 4.1 is continuous in $z(t)$ at $z(t) = z^c$, then

3. $J^*_\Sigma(z(t_k),T)$ is continuous at $z(t_k) = z^c$.

Proof: The proposition and proof are similar to Lemma A.1 in [10].
1. The non-negativity of $J^*_\Sigma(z(t_k),T)$ follows directly from $L_i(z_i, \hat{z}_{-i}, u_i) \geq 0$ and $P_i > 0$, for all $i = 1, ..., N_a$.
2. ($\Rightarrow$) $J^*_\Sigma(z(t),T) = 0$ implies that for each $i = 1, ..., N_a$,

$$\gamma \| z^*_{di}(t_k + T; z_i(t_k)) - z^c_i \|^2_{P_i} = 0 \quad \text{and}$$

$$\int_{t_k}^{t_k+T} L_i(z^*_{di}(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; \hat{z}_{-i}(t_k)), u^*_{di}(\tau; z_i(t_k)))\, \mathrm{d}\tau = 0.$$

Since the integrand is piecewise continuous in τ on $[t_k, t_k + T]$ and nonnegative, we have that

$$L_i(z^*_{di}(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; \hat{z}_{-i}(t_k)), u^*_{di}(\tau; z_i(t_k))) = 0, \quad \forall \tau \in [t_k, t_k + T],$$

and for every $i = 1, ..., N_a$. From Definition 4.1, this implies $u^*_{di}(\tau; z_i(t_k)) = 0$, for all $\tau \in [t_k, t_k + T]$. Since every P_i is positive definite and γ is positive, $z^*_{di}(t_k + T; z_i(t_k)) = z^c_i$ for every i. Since the dynamics $\dot{z}(t) = f(z(t), u(t))$ are time-invariant and have a unique solution, the reverse time dynamics with initial condition $z(t_k + T) = z^c$ and control $u(s) = 0$, $s \in [t_k + T, t_k]$, yield the solution $z(s) = z^c$ for all $s \in [t_k + T, t_k]$. Consequently, $z^*_{\text{dist}}(\tau; z(t_k)) = z^c$ for all $\tau \in [t_k, t_k + T]$. We must also guarantee that the resulting distributed optimal control and state are feasible. The constraints $z^*_{di}(\tau; z_i(t_k)) \in \mathcal{Z}$ and $u^*_{di}(\tau; z_i(t_k)) \in \mathcal{U}$ are trivial, since z^c_i and 0 are in the interior of $\mathcal{Z}$ and $\mathcal{U}$, respectively, by Assumption 4.2. Also, $z^c_i \in \Omega_i(\varepsilon_i)$ so the terminal constraint is satisfied. Finally, since $z(t_k) = z^c$, by Definition 4.5, $\hat{u}_i(\tau; z_i(t_k)) = 0$ for all $\tau \in [t_k, t_k + T]$ and every i, yielding $\hat{z}_i(\tau; z_i(t_k)) = z_i^c$ for all $\tau \in [t_k, t_k + T]$ and every i, so the compatibility constraint is also satisfied.

2. ($\Leftarrow$) Given $z(t_k) = z^c$, by Definition 4.5 we have that for each $i = 1, ..., N_a$, $\hat{u}_i(\tau; z_i(t_k)) = 0$ for all $\tau \in [t_k, t_k + T]$. Consequently, $\hat{z}_i(\tau; z_i(t_k)) = z_i^c$ for all $\tau \in [t_k, t_k + T]$ and every i. For any agent i, the local cost function becomes

$$\int_{t_k}^{t_k+T} L_i(z_{di}^*(\tau; z_i(t_k)), z_{-i}^c, u_{di}^*(\tau; z_i(t_k)))\ \mathrm{d}\tau \ + \ \gamma \|z_i(t_k + T; z_i(t_k)) - z_i^c\|_{P_i}^2,$$

Given $z_i(t_k) = z_i^c$, the feasible and optimal solution for any i is $u_{di}^*(\tau; z_i(t_k)) = 0$ and $z_{di}^*(\tau; z_i(t_k)) = z_i^c$ for all $\tau \in [t_k, t_k + T]$. Consequently, $J_i^*(z_i(t_k), z_{-i}(t_k), T) = 0$ for every $i = 1, ..., N_a$ and so $J_\Sigma^*(z(t_k), T) = 0$.

3. To show continuity of $J_\Sigma^*(z(t_k), T)$ at $z(t_k) = z^c$, recall first that $u_{\text{dist}}^*(\cdot; z(t_k))$ is continuous in $z(t_k)$ at $z(t) = z^c$, for any fixed horizon time T, by assumption. Since f is C^2, $z_{\text{dist}}^*(\cdot; z(t_k))$ exists and is continuous in initial state $z(t_k)$ at $z(t_k) = z^c$, by simple extension of Theorem 2.5 in [40], utilizing the Gronwall-Bellman inequality. Now, the assumed control is generated by concatenating the optimal control from any $z(t_k)$ with fixed horizon $T - \delta$, by principle of optimality, with the linear feedback specified in Assumption 4.3. Consequently, the full assumed control curve $\hat{u}(\cdot; z(t_k))$ is continuous in $z(t_k)$ at $z(t_k) = z^c$, as is the resulting state trajectory $\hat{z}(\cdot; z(t_k))$. Since every distributed integrated and terminal cost is continuous, we have continuity of $J_\Sigma^*(z(t_k), T)$ at $z(t_k) = z^c$. ■

The continuity assumption used to prove item 3 is typical in the early literature, although it is difficult in general to determine if the condition holds. If the receding horizon control law is continuously updated ($\delta = 0$), as in [48], it is necessary in fact for the optimal control to depend continuously on the associated initial condition *for all* state values in the closed-loop state domain, so that the closed-loop dynamics are continuous. In the centralized case, sufficient conditions that guarantee continuous differentiability of $u_{\text{cent}}^*(\tau; \cdot) : Z \to \mathcal{U}$, i.e., over the domain Z, are stated in [45].

If dual-mode receding horizon control is employed, only item 1 in the proposition is necessary. This removes the need for the continuity condition on the optimal control. The condition *if* $z(t_k) = z^c$ *then* $\hat{u}(\tau; z(t_k)) = 0$ in Definition 4.5 can also be removed, as it was used in showing item 2, i.e., the equivalence of $J^*_\Sigma(z(t_k), T) = 0$ and $z(t_k) = z^c$. So, for a dual-mode implementation, the assumed controls are defined as the remainder of the previous optimal control plus the decoupled feedback *for all* states in Z_Σ. Section 5.3.1 will define a dual-mode distributed receding horizon control implementation.

Our objective now is to show that the distributed receding horizon control law achieves the control objective for sufficiently small δ. We begin with three lemmas that are used to bound the Lyapunov function candidate $J^*_\Sigma(z(t_k), T)$. The first lemma gives a bounding result on the decrease in $J^*_\Sigma(\cdot, T)$ from one update to the next.

Lemma 4.3 *Under Assumptions 4.2–4.5, for a given fixed horizon time $T > 0$, and for the positive constant ξ defined by*

$$\xi = 2\gamma\kappa\rho_{max}\lambda_{max}(Q)T\sum_{i=1}^{N_a} N_i(N_i - 1),$$

the function $J^*_\Sigma(\cdot, T)$ satisfies

$$J^*_\Sigma(z(t_{k+1}), T) - J^*_\Sigma(z(t_k), T) \ \leq \ -\int_{t_k}^{t_k+\delta} \sum_{i=1}^{N_a} L^z_i(z^*_{di}(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)))\mathrm{d}\tau + \delta^2\xi,$$

for any $\delta \in (0, T]$ and for any $z(t_k) \in Z_\Sigma$, $k \in \mathbb{N}$.

Proof: The sum of the optimal distributed value functions for a given $z(t_k) \in Z_\Sigma$ is

$$J^*_\Sigma(z(t_k), T) = \int_{t_k}^{t_k+T} \sum_{i=1}^{N_a} L_i(z^*_{di}(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)), u^*_{di}(\tau; z_i(t_k)))\ \mathrm{d}\tau$$
$$+ \gamma\|z^*_{\text{dist}}(t_k + T; z(t_k)) - z^c\|^2_{\widehat{P}}.$$

Applying the optimal control for some $\delta \in (0, T]$ seconds, we are now at time $t_{k+1} = t_k + \delta$, with new state update $z(t_{k+1})$. A feasible control for the optimal control problem of each agent over the new time interval $\tau \in [t_{k+1}, t_{k+1}+T]$ is $u_i(\cdot; z_i(t_{k+1})) = \hat{u}_i(\cdot; z_i(t_{k+1}))$, given in Definition 4.5. Thus, we have for any $i = 1, ..., N_a$,

$$\begin{aligned} &J_i^*(z_i(t_{k+1}), z_{-i}(t_{k+1}), T) \\ &\qquad = \int_{t_{k+1}}^{t_{k+1}+T} L_i(z_{di}^*(\tau; z_i(t_{k+1})), \hat{z}_{-i}(\tau; z_{-i}(t_{k+1})), u_{di}^*(\tau; z_i(t_{k+1})))\ \mathrm{d}\tau \\ &\qquad\qquad + \gamma \| z_{di}^*(t_{k+1}+T; z_i(t_{k+1})) - z_i^c \|_{P_i}^2 \\ &\qquad \le \int_{t_{k+1}}^{t_{k+1}+T} L_i(\hat{z}_i(\tau; z_i(t_{k+1})), \hat{z}_{-i}(\tau; z_{-i}(t_{k+1})), \hat{u}_i(\tau; z_i(t_{k+1})))\ \mathrm{d}\tau \\ &\qquad\qquad + \gamma \| \hat{z}_i(t_{k+1}+T; z_i(t_{k+1})) - z_i^c \|_{P_i}^2 . \end{aligned}$$

Summing over i, we can upper bound $J_\Sigma^*(z(t_{k+1}), T)$, and comparing to $J_\Sigma^*(z(t_k), T)$ we have

$$\begin{aligned} &J_\Sigma^*(z(t_{k+1}), T) - J_\Sigma^*(z(t_k), T) \\ &+ \quad \int_{t_k}^{t_k+\delta} \sum_{i=1}^{N_a} L_i(z_{di}^*(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)), u_{di}^*(\tau; z_i(t_k)))\mathrm{d}\tau \\ &\qquad \le \quad \int_{t_{k+1}}^{t_k+T} \sum_{i=1}^{N_a} L_i(\hat{z}_i(\tau; z_i(t_{k+1})), \hat{z}_{-i}(\tau; z_{-i}(t_{k+1})), \hat{u}_i(\tau; z_i(t_{k+1})))\ \mathrm{d}\tau \\ &\qquad\quad - \int_{t_{k+1}}^{t_k+T} \sum_{i=1}^{N_a} L_i(z_{di}^*(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)), u_{di}^*(\tau; z_i(t_k)))\ \mathrm{d}\tau \\ &\qquad\quad + \int_{t_k+T}^{t_{k+1}+T} \sum_{i=1}^{N_a} L_i(\hat{z}_i(\tau; z_i(t_{k+1})), \hat{z}_{-i}(\tau; z_{-i}(t_{k+1})), \hat{u}_i(\tau; z_i(t_{k+1})))\ \mathrm{d}\tau \\ &\qquad\quad + \gamma \sum_{i=1}^{N_a} \| \hat{z}_i(t_{k+1}+T; z_i(t_{k+1})) - z_i^c \|_{P_i}^2 - \gamma \| z_{\text{dist}}^*(t_k+T; z(t_k)) - z^c \|_{\widehat{P}}^2 . \end{aligned}$$

Using the notation $\hat{z}(\cdot; z) = (\hat{z}_1(\cdot; z_1), ..., \hat{z}_{N_a}(\cdot; z_{N_a}))$, we observe that

$$\sum_{i=1}^{N_a} \| \hat{z}_i(t_{k+1}+T; z_i(t_{k+1})) - z_i^c \|_{P_i}^2 = \| \hat{z}(t_{k+1}+T; z(t_{k+1})) - z^c \|_{\widehat{P}}^2 ,$$

and also that $\hat{z}(t_k+T;z(t_{k+1})) = z^*_{\text{dist}}(t_k+T;z(t_k))$. From the definition for each $\hat{u}_i(\tau;z_i(t_{k+1}))$ with $\tau \in [t_k+T, t_{k+1}+T]$ and using the notation in equation (4.6), we also have

$$\sum_{i=1}^{N_a} L_i(\hat{z}_i(\tau;z_i(t_{k+1})), \hat{z}_{-i}(\tau;z_{-i}(t_{k+1})), \hat{u}_i(\tau;z_i(t_{k+1}))) = \gamma\|\hat{z}(\tau;z(t_{k+1})) - z^c\|^2_{\widetilde{Q}},$$

where $\widetilde{Q} = Q + \widehat{K}^T R \widehat{K}$. Finally, from the properties of the feedback in Assumption 4.3 and the notation in equation (4.6), we have

$$\begin{aligned}\|\hat{z}(t_{k+1}+T;z(t_{k+1})) - z^c\|^2_{\widehat{P}} - \|\hat{z}(t_k+T;z(t_{k+1})) - z^c\|^2_{\widehat{P}}& \\ = -\int_{t_k+T}^{t_{k+1}+T} \|\hat{z}(\tau;z(t_{k+1})) - z^c\|^2_{Q'} \mathrm{d}\tau,&\end{aligned}$$

where $Q' = \widehat{Q} + \widehat{K}^T R \widehat{K}$ and $Q' \geq \widetilde{Q}$. Thus, we can write

$$\begin{aligned}&\int_{t_k+T}^{t_{k+1}+T} \sum_{i=1}^{N_a} L_i(\hat{z}_i(\tau;z_i(t_{k+1})), \hat{z}_{-i}(\tau;z_{-i}(t_{k+1})), \hat{u}_i(\tau;z_i(t_{k+1})))\ \mathrm{d}\tau \\ +\quad &\gamma\sum_{i=1}^{N_a} \|\hat{z}_i(t_{k+1}+T;z_i(t_{k+1})) - z_i^c\|^2_{P_i} - \gamma\|z^*_{\text{dist}}(t_k+T;z(t_k)) - z^c\|^2_{\widehat{P}} \\ &\qquad = \quad -\gamma\int_{t_k+T}^{t_{k+1}+T} \|\hat{z}(\tau;z(t_{k+1})) - z^c\|^2_{Q'-\widetilde{Q}}\ \mathrm{d}\tau \quad \leq \quad 0.\end{aligned}$$

Now, we have

$$\begin{aligned}&J^*_\Sigma(z(t_{k+1}),T) - J^*_\Sigma(z(t_k),T) \\ +\quad &\int_{t_k}^{t_k+\delta} \sum_{i=1}^{N_a} L_i(z^*_{di}(\tau;z_i(t_k)), \hat{z}_{-i}(\tau;z_{-i}(t_k)), u^*_{di}(\tau;z_i(t_k)))\mathrm{d}\tau \\ &\leq \quad \int_{t_{k+1}}^{t_k+T} \sum_{i=1}^{N_a} L_i(\hat{z}_i(\tau;z_i(t_{k+1})), \hat{z}_{-i}(\tau;z_{-i}(t_{k+1})), \hat{u}_i(\tau;z_i(t_{k+1})))\ \mathrm{d}\tau \\ &\qquad - \int_{t_{k+1}}^{t_k+T} \sum_{i=1}^{N_a} L_i(z^*_{di}(\tau;z_i(t_k)), \hat{z}_{-i}(\tau;z_{-i}(t_k)), u^*_{di}(\tau;z_i(t_k)))\ \mathrm{d}\tau.\end{aligned}$$

By definition, each $\hat{z}_i(\tau; z_i(t_{k+1})) = z^*_{di}(\tau; z_i(t_k))$ and $\hat{u}_i(\tau; z_i(t_{k+1})) = u^*_{di}(\tau; z_i(t_k))$, over the interval $\tau \in [t_{k+1}, t_k + T]$. Consequently, we have from Definition 4.1

$$\begin{aligned}
&\int_{t_{k+1}}^{t_k+T} \sum_{i=1}^{N_a} L_i(\hat{z}_i(\tau; z_i(t_{k+1})), \hat{z}_{-i}(\tau; z_{-i}(t_{k+1})), \hat{u}_i(\tau; z_i(t_{k+1})))\, \mathrm{d}\tau \\
&- \int_{t_{k+1}}^{t_k+T} \sum_{i=1}^{N_a} L_i(z^*_{di}(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)), u^*_{di}(\tau; z_i(t_k)))\, \mathrm{d}\tau \\
&= \gamma \sum_{i=1}^{N_a} \int_{t_{k+1}}^{t_k+T} \left\| \begin{bmatrix} z^*_{di}(\tau; z_i(t_k)) - z^c_i \\ z^*_{-di}(\tau; z_i(t_k)) - z^c_{-i} \end{bmatrix} \right\|^2_{\overline{Q}_i} - \left\| \begin{bmatrix} z^*_{di}(\tau; z_i(t_k)) - z^c_i \\ \hat{z}_{-i}(\tau; z_i(t_k)) - z^c_{-i} \end{bmatrix} \right\|^2_{\overline{Q}_i} \mathrm{d}\tau,
\end{aligned}$$

where $z^*_{-di}(\tau; z_i(t_k))$ is the vector of optimal state trajectories for the neighbors of i, consistent with the ordering of z_{-i}. For the remainder of the proof, we are interested in finding an upper bound for the expression above. To reduce the expression, we first partition each symmetric matrix $\overline{Q}_i \in \mathbb{R}^{nN_i \times nN_i}$ as

$$\overline{Q}_i = \begin{bmatrix} \overline{Q}_{i,1} & \overline{Q}_{i,2} \\ \overline{Q}^T_{i,2} & \overline{Q}_{i,3} \end{bmatrix}, \quad \overline{Q}_{i,1} \in \mathbb{R}^{n \times n}, \quad \overline{Q}_{i,2} \in \mathbb{R}^{n \times n|\mathcal{N}_i|}, \quad \overline{Q}_{i,3} \in \mathbb{R}^{n|\mathcal{N}_i| \times n|\mathcal{N}_i|},$$

where $|\mathcal{N}_i| = N_i - 1$. For the moment, also denote

$$\begin{aligned}
x_i(\tau) &= z^*_{di}(\tau; z_i(t_k)) - z^c_i, \\
y_{-i}(\tau) &= z^*_{-di}(\tau; z_i(t_k)) - z^c_{-i}, \\
w_{-i}(\tau) &= \hat{z}_{-i}(\tau; z_i(t_k)) - z^c_{-i}\ .
\end{aligned}$$

Now, we can write the integrand above as

$$2x_i(\tau)^T \overline{Q}_{i,2}(y_{-i}(\tau) - w_{-i}(\tau)) \; + \; (y_{-i}(\tau) - w_{-i}(\tau))^T \overline{Q}_{i,3}(y_{-i}(\tau) + w_{-i}(\tau)).$$

From Assumption 4.5, $\|x_i\| \leq \rho_{\max}$. For any vector $v = (v_1, v_2)$, $\|v\| \leq \|v_1\| + \|v_2\|$. Thus, we have the bound $\|y_{-i}(\tau) - w_{-i}(\tau)\| \leq \delta^2 \kappa |\mathcal{N}_i|$, for any $i = 1, ..., N_a$. Also, for any two vectors x and y in $\mathbb{R}^n$, we have that $x^T y = \|x\|\ \|y\| \cos(\theta)$, where θ is the angle between the vectors. Consequently, $-\|x\|\ \|y\| \leq x^T y \leq \|x\|\ \|y\|$. Making use of these bounding arguments, we have that

$$\begin{aligned}2x_i(\tau)^T\overline{Q}_{i,2}\Big(y_{-i}(\tau) - w_{-i}(\tau)\Big) \;&\le\; 2\left(\lambda_{\max}(\overline{Q}_{i,2}^T\overline{Q}_{i,2})\right)^{1/2}\|x_i(\tau)\|\;\|y_{-i}(\tau) - w_{-i}(\tau)\|\\ &\le\; 2\delta^2\kappa\rho_{\max}|\mathcal{N}_i|\left(\lambda_{\max}(\overline{Q}_{i,2}^T\overline{Q}_{i,2})\right)^{1/2},\end{aligned}$$

and

$$\begin{aligned}&\Big(y_{-i}(\tau) - w_{-i}(\tau)\Big)^T\overline{Q}_{i,3}\Big(y_{-i}(\tau) + w_{-i}(\tau)\Big)\\ &\qquad\le\; \lambda_{\max}(\overline{Q}_{i,3})\|y_{-i}(\tau) + w_{-i}(\tau)\|\;\|y_{-i}(\tau) - w_{-i}(\tau)\|\\ &\qquad\le\; \delta^2\kappa|\mathcal{N}_i|\lambda_{\max}(\overline{Q}_{i,3})\left(\|y_{-i}(\tau)\| + \|w_{-i}(\tau)\|\right)\\ &\qquad\le\; 2\delta^2\kappa\rho_{\max}|\mathcal{N}_i|^2\lambda_{\max}(\overline{Q}_{i,3}).\end{aligned}$$

From Lemma A.1, $\lambda_{\max}(\overline{Q}_i) \ge \lambda_{\max}(\overline{Q}_{i,3})$ and $\lambda_{\max}(\overline{Q}_i) \ge (\lambda_{\max}(\overline{Q}_{i,2}^T\overline{Q}_{i,2}))^{1/2}$. Combining the terms, the integral expression is bounded as

$$\begin{aligned}&\gamma\sum_{i=1}^{N_a}\int_{t_{k+1}}^{t_k+T}\left\|\begin{bmatrix} z_{di}^*(\tau;z_i(t_k)) - z_i^c\\ z_{-di}^*(\tau;z_i(t_k)) - z_{-i}^c\end{bmatrix}\right\|_{\overline{Q}_i}^2 - \left\|\begin{bmatrix} z_{di}^*(\tau;z_i(t_k)) - z_i^c\\ \hat{z}_{-i}(\tau;z_i(t_k)) - z_{-i}^c\end{bmatrix}\right\|_{\overline{Q}_i}^2 \,\mathrm{d}\tau,\\ &\qquad\le\; \gamma\sum_{i=1}^{N_a}\int_{t_{k+1}}^{t_k+T} 2\delta^2\kappa\rho_{\max}|\mathcal{N}_i|\lambda_{\max}(\overline{Q}_i)(1+|\mathcal{N}_i|)\,\mathrm{d}\tau\\ &\qquad\le\; 2\gamma\delta^2\kappa\rho_{\max}T\sum_{i=1}^{N_a}\left[N_i\cdot|\mathcal{N}_i|\cdot\lambda_{\max}(\overline{Q}_i)\right],\end{aligned}$$

where the last inequality follows by using the bound $T-\delta \le T$. From Lemma A.2, $\lambda_{\max}(Q) \ge \lambda_{\max}(\overline{Q}_i)$ for every $i = 1, ..., N_a$. Finally, the original expression can be bounded as

$$\begin{aligned}&J_\Sigma^*(z(t_{k+1}),T) - J_\Sigma^*(z(t_k),T)\\ &\qquad\le -\int_{t_k}^{t_k+\delta}\sum_{i=1}^{N_a} L_i(z_{di}^*(\tau;z_i(t_k)),\hat{z}_{-i}(\tau;z_{-i}(t_k)),u_{di}^*(\tau;z_i(t_k)))\,\mathrm{d}\tau \;+\; \delta^2\xi\\ &\qquad\le -\int_{t_k}^{t_k+\delta}\sum_{i=1}^{N_a} L_i^z(z_{di}^*(\tau;z_i(t_k)),\hat{z}_{-i}(\tau;z_{-i}(t_k)))\,\mathrm{d}\tau \;+\; \delta^2\xi,\end{aligned}$$

where

$$\xi = 2\gamma\kappa\rho_{\max}\lambda_{\max}(Q)T\sum_{i=1}^{N_a} N_i(N_i - 1).$$

This completes the proof. ■

Ultimately, we want to show that $J_\Sigma^*(\cdot, T)$ decreases from one update to the next along the *actual* closed-loop trajectories. The next two lemmas show that, for sufficiently small δ, the bounding expression above can be bounded by a negative definite function of the closed-loop trajectories. Recall first that every distributed integrated cost defined in Definition 4.1 has a common multiplying factor equal to γ, a constant larger than one. The purpose of this constant, and the reason that it is chosen to be larger than one, is demonstrated in the next two lemmas. Specifically, γ provides a stability margin in that it enables the bounding expression in Lemma 4.3 to be bounded from above by a negative definite function of the closed-loop trajectories, as stated in the following lemma.

Lemma 4.4 *Under Assumptions 4.2–4.5, for any $z(t_k) \in Z_\Sigma$, $k \in \mathbb{N}$, such that at least one agent i satisfies $z_i(t_k) \neq z_i^c$, and for any positive constant ξ, there exists a $\delta(z(t_k)) > 0$ such that*

$$-\int_{t_k}^{t_k+\delta} \sum_{i=1}^{N_a} L_i^z(z_{di}^*(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)))\mathrm{d}\tau \;+\; \delta^2\xi$$
$$\leq \; -\int_{t_k}^{t_k+\delta} \|z_{dist}^*(\tau; z(t_k)) - z^c\|_Q^2 \mathrm{d}\tau,$$

for any $\delta \in (0, \delta(z(t_k))]$. If $z(t_k) = z^c$, then the equation above holds with $\delta(z(t_k)) = 0$.

Proof: At time $\tau = t_k$, $z_{di}^*(t_k; z_i(t_k)) = z_i(t_k)$ and $\hat{z}_{-i}(\tau; z_{-i}(t_k)) = z_{-i}(t_k)$, and so

$$\sum_{i=1}^{N_a} L_i^z(z_{di}^*(t_k; z_i(t_k)), \hat{z}_{-i}(t_k; z_{-i}(t_k))) = \sum_{i=1}^{N_a} L_i^z(z_i(t_k), z_{-i}(t_k)) = \gamma\|z(t_k) - z^c\|_Q^2,$$

where the last equality is from Definition 4.1. Since $Q > 0$, $\gamma > 1$, and at least one agent i satisfies $z_i(t_k) \neq z_i^c$, we have that $z(t_k) \neq z^c$ and

$$\sum_{i=1}^{N_a} L_i^z\left(z_{di}^*(t_k; z_i(t_k)), \hat{z}_{-i}(t_k; z_{-i}(t_k))\right) > \|z(t_k) - z^c\|_Q^2 > 0.$$

Equivalently, we have

$$\sum_{i=1}^{N_a} L_i^z\left(z_{di}^*(t_k; z_i(t_k)), \hat{z}_{-i}(t_k; z_{-i}(t_k))\right) > \|z_{\text{dist}}^*(t_k; z(t_k)) - z^c\|_Q^2. \tag{4.8}$$

Under the assumptions, $z_{di}^*(\tau; z_i(t_k))$ and $\hat{z}_{-i}(\tau; z_{-i}(t_k))$ are absolutely continuous in τ, for any $i = 1, ..., N_a$. Any quadratic function of z_{di}^* and $\hat{z}_{-i}$, including each L_i^z, is therefore absolutely continuous in τ. Thus, for any given $\xi > 0$, we can choose a $\delta(z(t_k)) > 0$ such that for any $\tau \in [t_k, t_k + \delta(z(t_k)))$,

$$\sum_{i=1}^{N_a} L_i^z(z_{di}^*(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k))) \ > \ 2(\tau - t_k)\xi + \|z_{\text{dist}}^*(\tau; z(t_k)) - z^c\|_Q^2,$$

and equality holds when $\tau = t_k + \delta(z(t_k))$. That is, $t_k + \delta(z(t_k))$ is the first time at which the two sides of the inequality are equal. Choosing such a $\delta(z(t_k)) > 0$ is possible even if each L_i^z is a decreasing function of τ and $\|z_{\text{dist}}^*(\tau; z(t_k)) - z^c\|_Q^2$ is an increasing function of τ, because of the initial margin in equation (4.8) and also because the functions have bounded rate. The latter statement follows from the functions being quadratic, therefore the gradients are linear, and from the assumed bounds on the state and control trajectories. Moreover, by integrating both sides in τ over the interval $[t_k, t_k + \delta(z(t_k))]$, the inequality still holds[1]. Thus, for any given

[1]Practically, a larger value of $\delta(z(t_k))$ can be obtained, since we are interested in comparing the integral equations, rather than comparing the expressions over all values of $\tau \in [t_k, t_k + \delta(z(t_k))]$.

$\xi > 0$, there exists a $\delta(z(t_k)) > 0$ such that

$$\begin{aligned}
\int_{t_k}^{t_k+\delta(z(t_k))} \sum_{i=1}^{N_a} L_i^z(z_{di}^*(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)))\ \mathrm{d}\tau & \\
&= \int_{t_k}^{t_k+\delta(z(t_k))} 2(\tau - t_k)\xi \ + \ \|z_{\text{dist}}^*(\tau; z(t_k)) - z^c\|_Q^2\ \mathrm{d}\tau \\
&= \delta(z(t_k))^2\xi \ + \int_{t_k}^{t_k+\delta(z(t_k))} \|z_{\text{dist}}^*(\tau; z(t_k)) - z^c\|_Q^2\ \mathrm{d}\tau. \qquad (4.9)
\end{aligned}$$

Finally, we have

$$\begin{aligned}
\int_{t_k}^{t_k+\delta} \sum_{i=1}^{N_a} L_i^z(z_{di}^*(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k)))\ \mathrm{d}\tau & \\
&\geq \delta^2\xi \ + \int_{t_k}^{t_k+\delta} \|z_{\text{dist}}^*(\tau; z(t_k)) - z^c\|_Q^2\ \mathrm{d}\tau,
\end{aligned}$$

for any $\delta > 0$ no bigger than $\delta(z(t_k))$, i.e., for any $\delta \in (0, \delta(z(t_k))]$. If $z(t_k) = z^c$, then both integrands are identically zero (see proof of Proposition 4.1). As a result, we immediately require $\delta(z(t_k)) = 0$ for the inequality to hold. Reversing the sign in the equation above gives the stated result. ■

Choosing the constant γ larger than one makes the initial margin in equation (4.8) and ultimately the integral bounding expression in equation (4.9) possible. In this way, choosing γ larger than one provides a stability margin, as the lemma above will be used in the proof of the main stability result, Theorem 4.1.

Remark 4.9 For a given value of $\gamma > 1$, as $\|z(t_k) - z^c\|$ decreases, so does the initial margin in equation (4.8). Consequently, as the states $z(t_k)$ approach the objective state z^c, the value of $\delta(z(t_k))$ that satisfies the integral equality in equation (4.9) decreases to zero, i.e., $z(t_k) \to z^c$ implies $\delta(z(t_k)) \to 0$. This corresponds to an increasingly strict compatibility constraint as $\|z(t_k) - z^c\|$ decreases. It also requires that communication of assumed control trajectories must happen at an increasing rate, and with infinite bandwidth, in the limit that $z(t_k) \to z^c$. The reason is that the assumed control trajectories must be communicated between update times, as specified in the distributed implementation algorithm in Definition 4.6. Later, we will construct an update time that is fixed and sufficiently small to guarantee that all agents have reached their terminal constraint sets via the distributed receding horizon control, making it safe to henceforth apply the decoupled linear feedbacks (dual-mode). The sufficiently small value for the update period then serves as an upper bound on how small δ must be for dual-mode distributed receding horizon control.

The lemma above provides a test on the update period that later is used to guarantee distributed receding horizon control stability. It would more be useful to have an analytic expression for the test. Such an expression is difficult to obtain, since the trajectories in the integrals in equation (4.9) are implicity defined and therefore hard to analyze. However, by making an assumption that approximates and simplifies the functions in the integrals, we are able to obtain an analytic bound on the update period.

Assumption 4.6 The following holds:

(i) the interval of integration $[t_k, t_k + \delta]$ for the expressions in equation (4.9) is sufficiently small that first-order Taylor series approximations of the integrands is a valid approximation for any $z(t_k) \in Z_\Sigma$. Specifically, we take

$$\begin{aligned}
&\sum_{i=1}^{N_a} L_i^z(z_{di}^*(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k))) \\
&\qquad \approx \; \gamma \|z(t_k) - z^c\|_Q^2 \\
&\qquad\quad + (\tau - t_k) \sum_{i=1}^{N_a} \Big\{ \nabla_{z_i} L_i^z(z_i(t_k), z_{-i}(t_k))^T f_i(z_i(t_k), u_{di}^*(t_k; z_i(t_k))) \\
&\qquad\qquad\qquad + \Sigma_{j\in\mathcal{N}_i} \nabla_{z_j} L_i^z(z_i(t_k), z_{-i}(t_k))^T f_j(z_j(t_k), \hat{u}_j(t_k; z_j(t_k))) \Big\},
\end{aligned}$$

and

$$\|z^*_{\text{dist}}(\tau; z(t_k)) - z^c\|^2_Q \approx \|z(t_k) - z^c\|^2_Q + (\tau - t_k)2(z(t_k) - z^c)^T Q f(z(t_k), u^*_{\text{dist}}(t_k; z(t_k))),$$

and ignore terms that are $O((\tau - t_k)^2)$ and higher-order;

(ii) for every $i = 1, ..., N_a$, there exists a Lipschitz constant $\mathcal{K}_i \in (0, \infty)$ such that

$$\|f_i(z_i, u_i) - f_i(z'_i, u'_i)\| \leq \mathcal{K}_i\Big(\|z_i - z'_i\| + \|u_i - u'_i\|\Big),$$

for any $z = (z_1, ..., z_{N_a}) \in Z_\Sigma$ and $z' = (z'_1, ..., z'_{N_a}) \in Z_\Sigma$ and any $u_i, u'_i \in \mathcal{U}$. Finally, we define $\mathcal{K} = \max_i \mathcal{K}_i$.

Part (ii) of the assumption is a local Lipschitz requirement, since it is presumed to hold only over the domain Z_Σ. The Lipschitz bound $\mathcal{K}$ is used in a parametric upper bound on the update period that is now defined.

Lemma 4.5 *Under Assumptions 4.2–4.6, for any $z(t_k) \in Z_\Sigma$, $k \in \mathbb{N}$, the margin in Lemma 4.4 is attained with*

$$\delta(z(t_k)) = \frac{(\gamma - 1)\|z(t_k) - z^c\|^2_Q}{\xi + (\gamma + 1)\mathcal{K}\rho_{max}(\rho_{max} + u_{max})\lambda_{max}(Q)\sum_{i=1}^{N_a} N_i^2},$$

given the state and control bounds R and U_{max}, respectively.

Proof: From equation (4.9), for a given $z(t_k)$, we want to choose a δ such that

$$\int_{t_k}^{t_k+\delta} \sum_{i=1}^{N_a} L^z_i(z^*_{di}(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k))) - \|z^*_{\text{dist}}(\tau; z(t_k)) - z^c\|^2_Q \, d\tau \; - \; \delta^2\xi \; \geq 0,$$

with equality when $\delta = \delta(z(t_k))$. Substitution of the Taylor series expression from Assumption 4.6 into the integrals results in

$$\delta\,(\xi - C) \leq (\gamma - 1)\|z(t_k) - z^c\|^2_Q, \tag{4.10}$$

where C combines the first-order terms in the expansions after integration, given as

$$C = \sum_{i=1}^{N_a} \frac{1}{2}\Bigg[(\gamma-1)\nabla_{z_i} L_i^z(z_i(t_k), z_{-i}(t_k))^T f_i(z_i(t_k), u_{di}^*(t_k; z_i(t_k)))$$
$$+ \Sigma_{j\in\mathcal{N}_i} \nabla_{z_j} L_i^z(z_i(t_k), z_{-i}(t_k))^T$$
$$\Big\{\gamma f_j(z_j(t_k), \hat{u}_j(t_k; z_j(t_k))) - f_j(z_j(t_k), u_{dj}^*(t_k; z_j(t_k)))\Big\}\Bigg].$$

Since C is the sum of an inner-product of different vectors, it could be positive or negative. In equation (4.10), $(\gamma-1) > 0$ is given and $\xi > 0$ is given and we are looking for the largest δ such that the inequality holds over the entire domain Z_Σ. Note that if $C \geq \xi$, the inequality holds for any positive δ. In the worst case, the constant C will be a negative number, removing more of the margin for δ. Thus, substituting a negative lower bound for C ensures a sufficient inequality condition on δ. To identify a lower bound for C, we first partition each symmetric matrix $\overline{Q}_i \in \mathbb{R}^{nN_i \times nN_i}$ as

$$\overline{Q}_i = \begin{bmatrix} \overline{Q}_{i,1} & \overline{Q}_{i,2} \\ \overline{Q}_{i,2}^T & \overline{Q}_{i,3} \end{bmatrix}, \quad \overline{Q}_{i,1} \in \mathbb{R}^{n\times n}, \quad \overline{Q}_{i,2} \in \mathbb{R}^{n\times n|\mathcal{N}_i|}, \quad \overline{Q}_{i,3} \in \mathbb{R}^{n|\mathcal{N}_i|\times n|\mathcal{N}_i|},$$

just as in the proof of Lemma 4.3. Now, the gradient functions are written

$$\nabla_{z_i} L_i^z(z_i(t_k), z_{-i}(t_k))^T = 2(z_i(t_k) - z_i^c)^T \overline{Q}_{i,1} + 2(z_{-i}(t_k) - z_{-i}^c)^T \overline{Q}_{i,2}^T, \quad \text{and}$$
$$\nabla_{z_{-i}} L_i^z(z_i(t_k), z_{-i}(t_k))^T = 2(z_i(t_k) - z_i^c)^T \overline{Q}_{i,2} + 2(z_{-i}(t_k) - z_{-i}^c)^T \overline{Q}_{i,3},$$

where $\nabla_{z_j} L_i^z$, $j \in \mathcal{N}_i$, are the components of $\nabla_{z_{-i}} L_i^z$. Using the inner-product property that $x^T y \geq -\|x\|\,\|y\|$ for any vectors x and y, we have that for any i,

$$\frac{1}{2}\nabla_{z_i} L_i^z(z_i(t_k), z_{-i}(t_k))^T f_i(z_i(t_k), u_{di}^*(t_k; z_i(t_k))) \;\geq$$
$$-\left[\|z_i(t_k) - z_i^c\|\lambda_{\max}(\overline{Q}_{i,1}) + \|z_{-i}(t_k) - z_{-i}^c\|\lambda_{\max}^{1/2}(\overline{Q}_{i,2}^T\overline{Q}_{i,2})\right] \|f_i(z_i(t_k), u_{di}^*(t_k; z_i(t_k)))\|.$$

From Lemma A.1, the Lipschitz bound on any f_i and the bounding arguments used in the proof of Lemma 4.3, we have that

$$\frac{1}{2}\nabla_{z_i} L_i^z(z_i(t_k), z_{-i}(t_k))^T f_i(z_i(t_k), u_{di}^*(t_k; z_i(t_k))) \;\geq\; -N_i \mathcal{K} \rho_{\max}(\rho_{\max} + u_{\max})\lambda_{\max}(\overline{Q}_i).$$

By the same tools and using the triangle inequality, we can also bound the other inner-product terms as

$$\begin{aligned}
&\frac{1}{2}\Sigma_{j\in\mathcal{N}_i}\nabla_{z_j} L_i^z(z_i(t_k), z_{-i}(t_k))^T \Big\{\gamma f_j(z_j(t_k), \hat{u}_j(t_k; z_j(t_k))) - f_j(z_j(t_k), u_{dj}^*(t_k; z_j(t_k)))\Big\} \\
&\quad\geq -\Big[\rho_{\max}\lambda_{\max}^{1/2}(\overline{Q}_{i,2}^T\overline{Q}_{i,2}) + \rho_{\max}|\mathcal{N}_i|\lambda_{\max}(\overline{Q}_{i,3})\Big] \times \\
&\qquad\qquad \left[\gamma\sum_{j\in\mathcal{N}_i}\|f_j(z_j(t_k), \hat{u}_j(t_k; z_j(t_k)))\| + \|f_j(z_j(t_k), u_{dj}^*(t_k; z_j(t_k)))\|\right] \\
&\quad\geq -N_i\rho_{\max}\lambda_{\max}(\overline{Q}_i)\left[(\gamma+1)\mathcal{K}|\mathcal{N}_i|(\rho_{\max} + u_{\max})\right].
\end{aligned}$$

Collecting terms and using Lemma A.2, we have

$$\begin{aligned}
C &\geq -\mathcal{K}\rho_{\max}(\rho_{\max} + u_{\max})\lambda_{\max}(Q)\sum_{i=1}^{N_a}\left[(\gamma-1)N_i + (\gamma+1)N_i|\mathcal{N}_i|\right] \\
&\geq -(\gamma+1)\mathcal{K}\rho_{\max}(\rho_{\max} + u_{\max})\lambda_{\max}(Q)\sum_{i=1}^{N_a} N_i^2.
\end{aligned}$$

So, in the worst case, we have that δ must satisfy

$$\delta\left[\xi \;+ (\gamma+1)\mathcal{K}\rho_{\max}(\rho_{\max} + u_{\max})\lambda_{\max}(Q)\sum_{i=1}^{N_a} N_i^2\right] \;\leq\; (\gamma-1)\|z(t_k) - z^c\|_Q^2,$$

or, equivalently,

$$\delta \leq \frac{(\gamma-1)\|z(t_k) - z^c\|_Q^2}{\xi \;+ (\gamma+1)\mathcal{K}\rho_{\max}(\rho_{\max} + u_{\max})\lambda_{\max}(Q)\sum_{i=1}^{N_a} N_i^2}.$$

The inequality becomes an equality when $\delta = \delta(z(t_k))$, completing the proof. ■

Consistent with the observation in Remark 4.9, we see that $\delta(z(t_k)) \rightarrow 0$ as $z(t_k) \rightarrow z^c$. For an analytic test on the update period δ, we can combine the equation for $\delta(z(t_k))$ in Lemma 4.5 and ξ in Lemma 4.3, which results in

$$\delta(z(t_k)) = \frac{(\gamma-1)\|z(t_k) - z^c\|_Q^2}{\left[2\gamma\kappa T \sum_{i=1}^{N_a} N_i(N_i - 1) \ + \ (\gamma+1)\mathcal{K}(\rho_{\max} + u_{\max}) \sum_{i=1}^{N_a} N_i^2\right] \rho_{\max}\lambda_{\max}(Q)}. \tag{4.11}$$

To simplify the expression, we redefine $\delta(z(t_k))$ to be

$$\delta(z(t_k)) = \frac{(\gamma-1)\|z(t_k) - z^c\|_Q^2}{\left[2\kappa T \ + \ \mathcal{K}(\rho_{\max} + u_{\max})\right](\gamma+1)\rho_{\max}\lambda_{\max}(Q) \sum_{i=1}^{N_a} N_i^2}. \tag{4.12}$$

The bound in equation (4.12) is tighter than the bound in equation (4.11), since $\gamma > \gamma - 1$ and $\sum_{i=1}^{N_a} N_i^2 > \sum_{i=1}^{N_a} N_i(N_i - 1)$. Thus, if $\delta \leq \delta(z(t_k))$ using the bound in equation (4.12), then δ also satisfies $\delta \leq \delta(z(t_k))$ using the bound in equation (4.11), and so the results of the previous lemmas can be applied using the bound in equation (4.12). Henceforth, we shall use the bound in equation (4.12).

Remark 4.10 The upper bound on the update period in equation (4.12) has some interesting features. The bound is independent of γ when $\gamma \gg 1$ and shrinks to zero as $\gamma \rightarrow 1$. The latter condition makes sense in light of the previous discussions that γ must be chosen larger than one to provide the stability margin defined by the integral equation in Lemma 4.4. Also, the larger the assumed bounds on the state and control ($\rho_{\max}$ and $u_{\max}$), and the larger the horizon time T, the smaller the required update time. With regard to the compatibility constraint, the update must also be faster for larger values of κ. Given the conservative nature of the proofs, it is not wise to infer too much from the bound in equation (4.12), but it is reassuring to observe such intuitive affects.

We also note that since $\delta(z(t_k))$ depends on $\|z(t_k) - z^c\|_Q^2$, a centralized computation is required to generate equation (4.12) at each receding horizon update. Otherwise, a distributed consensus algorithm, as given in [59] for example, must be run in parallel to determine $\|z(t_k) - z^c\|_Q^2$, or a suitable lower bound on it. In the dual-mode version defined in Section 5.3.1, no such centralized computation is required online, since a fixed bound on the update period is computed offline and applied for every receding horizon update.

The first main theorem of the dissertation is now given.

Theorem 4.1 *For a given fixed horizon time $T > 0$, assume that the optimal control solution $u^*_{dist}(\cdot; z(t))$ to Problem 4.1 is continuous at $z(t) = z^c$ and that Assumptions 4.2–4.6 hold. For any state $z(t_{-1}) \in Z_\Sigma$ at initialization, if the update time δ satisfies $\delta \in (0, \delta(z(t_k))]$, $k \in \mathbb{N}$, where $\delta(z(t_k))$ is defined in equation (4.12), then z^c is an asymptotically stable equilibrium point of the closed-loop system (4.5) with region of attraction Z_Σ, an open and connected set.*

Proof: The assumption that the optimal control solution $u^*_{\text{dist}}(\cdot; z(t))$ to Problem 4.1 is continuous in $z(t)$ at $z(t) = z^c$ is used to qualify part 3 of Proposition 4.1, which is employed below.

If $z(t_k) = z^c$, then $\hat{u}_i(\tau; z_i(t_k)) = 0$ for all $\tau \in [t_k, t_k + T]$ and every $i = 1, ..., N_a$, and the optimal solution to Problem 4.1 is $u^*_{\text{dist}}(\tau; z_i(t_k)) = 0$ for all $\tau \in [t_k, t_k + T]$. This is shown in the proof of Proposition 4.1. Since $f(z^c, 0) = 0$, it is proven that z^c is an equilibrium point of the closed-loop system (4.5).

We observe that $J^*_\Sigma(z(t_k), T)$ has the following properties

- $J^*_\Sigma(z^c, T) = 0$ and $J^*_\Sigma(z(t_k), T) > 0$ for $z(t_k) \neq z^c$,
- $J^*_\Sigma(z(t_k), T)$ is continuous at $z(t_k) = z^c$,
- along the trajectory $z^*_{\text{dist}}(\tau)$ of the closed-loop system starting from $z(t_0)$, where $z(t_0) = z^*_{\text{dist}}(t_0; z(t_{-1}))$ for any initial state $z(t_{-1}) \in Z_\Sigma$,

$$J^*_\Sigma(z(t'), T) - J^*_\Sigma(z(t), T) \leq -\int_t^{t'} \|z^*_{\text{dist}}(\tau) - z^c\|^2_Q \, \mathrm{d}\tau, \tag{4.13}$$

for any times $t,\ t'$ with $t_0 \leq t < t' \leq \infty$.

The first two properties follow from Proposition 4.1. The last property is derived as follows. Combining Lemma 4.3 and Lemma 4.4, we have at any $k \in \mathbb{N}$

$$J^*_\Sigma(z(s), T) - J^*_\Sigma(z(t_k), T) \leq -\int_{t_k}^{s} \|z^*_{\text{dist}}(\tau; z(t_k)) - z^c\|^2_Q \, \mathrm{d}\tau, \quad \text{for any } s \in [t_k, t_{k+1}].$$

Applying this recursively gives the result for any $t \geq t_0$ and any $t' \in (t, \infty]$.

The remainder of the proof follows precisely the steps in the proof of Theorem 1 in [11], and the steps are restated here for completeness. The equilibrium z^c is first proven to be asymptotic stable, and then we show that Z_Σ is a region of attraction for the closed-loop system.

Given $\epsilon > 0$, choose $r \in (0, \epsilon]$ such that the closed ball

$$B(z^c, r) = \{z \in \mathbb{R}^{nN_a} \mid \|z - z^c\| \leq r\}$$

is a small neighborhood of z^c. For the remainder of the proof, we denote $V(z) \triangleq J^*_\Sigma(z, T)$. Since $V(z)$ is continuous at $z = z^c$ and $V(z) > 0$ for all $z \neq z^c$, there exists a $\beta \in (0, \infty)$ such that

$$\beta < \min_{\|z - z^c\| = r} V(z).$$

Define the level set of $V(z)$

$$W_\beta \triangleq \{z \in B(z^c, r) \mid V(z) \leq \beta\},$$

which is a subset contained in the interior of $B(z^c, r)$, a fact that is easy to show by contradiction. By the monotonicity of $V(z(t))$,

$$V(z(t)) \leq V(z(t_0)) \leq \beta, \quad \forall t \geq t_0,$$

so W_β is a positively invariant set for the closed-loop system. Since $V(z)$ is continuous at $z = z^c$ and $V(z^c) = 0$, there exists a constant $\eta \in (0, r)$ such that

$$\|z(t_0) - z^c\| < \eta \;\Rightarrow\; V(z(t_0)) < \beta \;\Rightarrow\; V(z(t)) < \beta \;\Rightarrow\; \|z(t) - z^c\| < \epsilon.$$

Thus, z^c is a stable equilibrium point of the system (4.5).

Asymptotic stability is now proven using equation (4.13). For consistency in notation, let $z(\tau) = z^*_{\text{dist}}(\tau)$. By induction,

$$V(z(t_0)) - V(z(\infty)) \geq \int_{t_0}^{\infty} \|z(\tau) - z^c\|_Q^2 \, \mathrm{d}\tau \;\Rightarrow\; \beta \geq \int_{t_0}^{\infty} \|z(\tau) - z^c\|_Q^2,$$

since $\beta \geq V(z(t_0))$ and $0 \geq -V(z(\infty))$. Therefore, the infinite integral above exists and is bounded. Let $\epsilon_1 < \epsilon$ be such that $z(t)$ belongs to the compact set $\{\|z(t)-z^c\| \leq \epsilon_1\}$ for all $t \in [t_0, \infty)$, known to exist from the strict inequality bound by ϵ shown above. Since $u^*_{\text{dist}}(t)$ is in the compact set $\mathcal{U}^{N_a}$ for all $t \in [t_0, \infty)$ and f is continuous in z and u, we have that $f(z(t), u^*_{\text{dist}}(t))$ is bounded for all $t \in [t_0, \infty)$. From [40], $z(t)$ is uniformly continuous in t on $[t_0, \infty)$. Since $\|z - z^c\|_Q^2$ is uniformly continuous in z on the compact set $\{\|z - z^c\| \leq \epsilon_1\}$, we now have that $\|z(t) - z^c\|_Q^2$ is uniformly continuous in t on $[t_0, \infty)$. Since $Q > 0$, Barbalat's Lemma [40] guarantees that

$$\|z(t) - z^c\| \to 0 \quad \text{as} \quad t \to \infty.$$

Thus, z^c is an asymptotically stable equilibrium point of the system (4.5), with region of attraction W_β.

Now, for any $z(t_0) \in Z_\Sigma$, there exists a finite time T' such that $z(T') \in W_\beta$, which can be shown by contradiction as follows. Suppose $z(t) \notin W_\beta$ for all $t \geq t_0$. From equation (4.13), for all $t \geq t_0$,

$$\begin{aligned} V(z(t+\delta)) - V(z(t)) &\leq -\int_{t}^{t+\delta} \|z(\tau) - z^c\|_Q^2 \, \mathrm{d}\tau \\ &\leq -\delta \cdot \inf\left\{\|z - z^c\|_Q^2 \mid z \notin W_\beta\right\} \leq -\delta \cdot \gamma, \end{aligned}$$

where $\gamma > 0$ since $V(z) > 0$ and $Q > 0$. By induction, $V(z(t)) \to -\infty$ as $t \to \infty$; however, this contradicts $V(z(t)) \geq 0$. Therefore, any trajectory starting in Z_Σ enters W_β in finite time. Finally, Z_Σ is a region of attraction for the closed-loop system (4.5) since it is a positively invariant set by Lemma 4.2. Moreover, for any $z(t) \in Z_\Sigma$, by absolute continuity of $z(t')$ in $t' \geq t_0$, we can always choose a small neighborhood of $z(t)$ in which the optimization problem is still feasible. Therefore, Z_Σ is open and connected. ■

From equation (4.12), we observe that $\delta(z(t_k)) \to 0$ as $z(t_k) \to z^c$. As a consequence, the compatibility constraint gets tighter, and the communication between neighboring agents must happen with increasing bandwidth, as the agents collectively approach the control objective. To mitigate these problems, a *dual-mode* version of the distributed receding horizon control law is formulated in the next chapter. In the dual-mode implementation, the closed-loop system will be equation (4.5) until all agents are in the interior of their terminal constraint sets, at which point each control is *synchronously* redefined to be the decoupled linear feedback defined in Assumption 4.3.

4.7 Summary

In this chapter, a distributed implementation of receding horizon control is formulated. The implementation is based first on a single finite horizon optimal control problem with a specific structure. In particular, a generic quadratic integrated cost function couples the states of a set of dynamically decoupled subsystems. The decoupled dynamics may be nonlinear and heterogeneous. Given this structure, the problem is decomposed into a set of distributed optimal control problems. The implementation requires the addition of a *compatibility constraint* in each distributed optimal control problem. The constraint, a central element in the stability analysis, ensures that actual and assumed responses of each agent are not too far from one another. Asymptotic stability is proven in the absence of explicit uncertainty and for sufficiently fast receding horizon updates. In the next chapter, the distributed implementation is analyzed in detail. Additionally, relaxations of some of the assumptions required in this chapter are explored.

Chapter 5

Analysis of Distributed Receding Horizon Control

5.1 Introduction

This chapter analyzes the distributed receding horizon control implementation presented in Chapter 4 in three different ways. First, the implementation is interpreted in Section 5.2, giving both qualitative and quantitative comparisons with centralized implementations. In Section 5.3, alternative ways of formulating the distributed implementation, that still preserve closed-loop stabilization, are explored. Finally, in Section 5.4, extensions of the theory are discussed in detail, providing an outline of some of the future work to be investigated. A more general discussion of extensions of the theory, in terms of its relevance in other disciplines and potential applications, is given later in Chapter 7.

5.2 Interpretation of the Distributed Receding Horizon Control Law

This section is focused on interpreting the distributed receding horizon control law presented in Chapter 4 in three ways. First, a qualitative comparison, specifically of the cost of computation and communication, between centralized and distributed implementations, is given in Section 5.2.1. Next, the effect of the compatibility constraint on closed-loop performance is explored in Section 5.2.2. Finally, while the distributed implementation does not recover the centralized solution of the original optimal control Problem 2.1 at any receding horizon update, it does correspond to the solution of a different single optimal control problem, defined in Section 5.2.3.

5.2.1 Comparison with Centralized Implementations

In this section, we give a general comparison, of the computational cost and the cost of communication, between centralized and distributed implementations of receding horizon control. While a more quantitative comparison could be made by defining precise complexity bounds, we leave such details for a future work. The purpose of this section is not to give quantitative details of comparison, but to give a qualitative comparison between the two implementations.

For centralized receding horizon control, we mention two possible implementations. In one version, every agent solves a copy of the centralized problem. In the absence of explicit uncertainty, every agent gets the same answer, and so stability follows immediately if the conditions in Chapter 2 are satisfied. The computational cost is that there are N_a copies of the centralized problem, and the size of the centralized problem itself also depends the number of agents N_a. The cost of communication is that, at every receding horizon update, every agent must receive the initial condition *for all agents*. So, even if the agents are sparsely coupled (e.g., each agent is coupled to at most two other agents), all agents must have access to all other agents current state in this implementation, independent of whether or not the agents are neighbors. This implementation is *not scalable*, in terms of the cost of computation or communication. It is not computationally scalable because the centralized computational problem size depends on N_a, and therefore the problem is harder for a larger number of agents. The cost of communication is also not scalable for the same reason.

Another version of the centralized implementation is to assign one agent to solve the centralized problem at each update. That agent must receive the initial state for all agents at each update, as in the previous implementation given above. Further, that agent must disseminate the resulting receding horizon control law to every agent. The individual agents are no longer autonomous in this implementation, since agents are no longer generating their own control locally. The computational cost for the computing agent is the same as the corresponding cost for any agent in the centralized implementation mentioned above. The implementation is, therefore, not computationally scalable. Additionally, the communication cost now involves not only the collection of the current state of all agents at the single computing node, but the transmission of the applied control trajectories to *all other agents* at every update. The latter portion of the communication cost is usually more expensive than just current state information, depending on how the trajectories are parameterized in the numerical method for solving the optimization problem. Therefore, the communication cost in this implementation is not scalable and much more expensive in general than the corresponding cost of the centralized implementation mentioned above.

Now, we consider the cost of computation and communication for the distributed implementation given in Chapter 4. In terms of computational cost, each agent solves a local optimization problem at each update, computing only the optimal control for itself. Since each agent is computing a control for itself, locally, the implementation is a control approach that enables autonomy of the individual agents. Moreover, since each agent is computing a control for *only* itself, the implementation is *computationally scalable*. Specifically, the size of each local optimization problem is independent of the total number of agents N_a. In the local optimization problem for any agent i, a differential equation is integrated for each neighbor $j \in \mathcal{N}_i$. The computational cost of integrating these differential equations is in general negligible compared to the cost of solving for the optimal control of agent i. Therefore, the computational cost for each agent is also independent of $|\mathcal{N}_i|$. *The computational scalability of the distributed implementation is the main advantage over a centralized implementation.*

The cost of communication for the distributed implementation is also scalable. Recall that, at every update time, each agent must communicate initial states only with neighbors. Also, between every subsequent pair of updates, each agent must transmit and receive assumed control trajectories, again only with neighbors. Since this communication is solely between neighboring agents, the associated communication cost is independent of the total number of agents N_a, but dependent on the size of each neighborhood $|\mathcal{N}_i|$ for each agent i. Therefore, the distributed implementation has a total communication cost that is independent of the total number of agents N_a, and is scalable in that sense, but dependent on the number of neighbors $|\mathcal{N}_i|$ for every agent i. Note that the communication cost associated with globally synchronous timing is not accounted for here.

In comparison to the cost of communication for the centralized implementations, an advantage of the distributed implementation is that no single agent needs to communicate with all other agents, except in the case that an agent i is coupled to all other agents ($|\mathcal{N}_i| = N_a - 1$). Still, the distributed implementation does involve the communication of assumed control trajectories. By construction (see Definition 4.5), it is easy to see that the cost of communicating an assumed control trajectory is equivalent to that of an optimal control trajectory. Recall that only the *applied portion* of the optimal control trajectories are communicated to all agents in the centralized implementation with a single computing node. Also, the centralized implementation with all agents solving the centralized problem does not require the transmission of any control trajectories. If the cost of communicating assumed control trajectories is excessive, there may be cases where the communication cost of either centralized implementation is comparable or less than than of the distributed implementation. For example, the cost of communicating the global current state to all agents may be less than communicating states and control trajectories locally, i.e., within neighborhoods. However, if one assumes that agents can communicate only with neighboring agents, either centralized implementation requires multi-hop communication, while the distributed implementation is single-hop. This implies that the distributed implementation has an increasing advantage as the networking environment becomes increasingly distributed and asynchronous, although the analysis in such environments is still to be done. Moreover, given the significant computational savings that the distributed implementation provides over the centralized implementations, moderate gain in communication cost is likely not a sufficient justification for the centralized implementation.

Remark 5.1 The inter-agent communication requirements of the distributed implementation are clearly more intensive than that of other, more traditional, decentralized feedback control [14]. The tradeoff is that the power of an optimization-based approach is available, namely, the ability to address performance in the presence of generic constraints and nonlinearities. If the trajectories are known to be sufficiently smooth, and polynomial-based approximations are valid, the communication requirements need not be substantially worse than that of standard decentralized schemes. The numerical method applied for the simulation results of the next chapter does indeed use polynomial parameterizations for the computed trajectories.

In comparison to the centralized receding horizon control implementation, there are two other important issues elucidated here. Since there are no compatibility constraints in the centralized implementation, the effect of these constraints needs to be determined. For example, the constraints obviously have immediate implications on the difficulty of the optimization problem. However, since the scale of the centralized and distributed problems are so different, it is difficult at this point to make a computational "degree of difficulty" comparison. As an alternative, it is appropriate to examine the implications of the compatibility constraints on closed-loop performance, and to make comparisons based on these implications. This is done in the next section, and also in Chapter 6 via numerical experiments.

Another way of comparing the solution to the centralized problem (Problem 2.1) and the solution to the distributed problems (Problem 4.1) is to realize that the distributed solution actually solves a modified centralized problem! In that way, the two centralized problems can be compared to one another. The centralized problem, to which the distributed implementation is the solution, is formulated in Section 5.2.3.

5.2.2 Effect of Compatibility Constraint on Closed-Loop Performance

In Chapter 4, a compatibility constraint is introduced in every distributed optimal control problem. The constraint is included to mitigate the effect of the discrepancy between what agents plan to do and what their neighbors assume they plan to do. A natural question is "what effect do these constraints have on closed-loop performance?" The theory of Chapter 4 says that, for the agents to converge to the objective state, the update period δ, and consequently the bound on the compatibility constraint $\delta^2\kappa$, must shrink to zero.

Suppose we choose a fixed small update period, such that the bound $\delta^2\kappa$ is also small. One might expect the transient closed-loop response to be sluggish, since the state is not permitted to vary too much from the previous response, from one update to the next. Despite the sluggish transient response, the agents would still eventually come close to converging to the objective state, according to the theory. On the other hand, if the compatibility constraint is relaxed by choosing a larger update time, one might expect the opposite trend, i.e., less sluggish transient response but poor convergence. This is in fact not the case, as the simulation experiments in the next chapter show good convergence even in the absence of the compatibility constraints. Still, sluggish response can result from a sufficiently small bound in each compatibility constraint, an effect that we now qualitatively demonstrate.

Let $N_{\mathrm{RH}} \in \mathbb{N}$ be some number of receding horizon updates after time t_0 such that $N_{\mathrm{RH}} \cdot \delta \approx T$. The state compatibility constraint for every agent $i = 1, ..., N_a$ is

$$\|z_i(t; z_i(t_k)) - \hat{z}_i(t; z_i(t_k))\| \leq \delta^2\kappa, \quad t \in [t_k, t_k + T].$$

At the optimum, the computed $z_i(t; z_i(t_k))$ becomes $z^*_{di}(t; z_i(t_k))$, and the constraint implies

$$\|z^*_{di}(t; z_i(t_k)) - \hat{z}_i(t; z_i(t_k))\| \leq \delta^2\kappa, \quad t \in [t_k, t_k + T].$$

From Definition 4.5, the constraint also implies that, over the subinterval of time

$[t_k, t_{k-1}+T]$,

$$\|z_{di}^*(t; z_i(t_k)) - z_{di}^*(t; z_i(t_{k-1}))\| \leq \delta^2\kappa, \quad t \in [t_k, t_{k-1}+T], \quad k = 1, 2, ...$$

For $k = 1$ and at time $t = t_{N_{\rm RH}}$, where $t_{N_{\rm RH}} := t_0 + N_{\rm RH} \cdot \delta \approx t_0 + T$, we therefore have that

$$\|z_{di}^*(t_{N_{\rm RH}}; z_i(t_1)) - z_{di}^*(t_{N_{\rm RH}}; z_i(t_0))\| \leq \delta^2\kappa.$$

For $k = 2$ and at time $t = t_{N_{\rm RH}}$, we also have that

$$\|z_{di}^*(t_{N_{\rm RH}}; z_i(t_2)) - z_{di}^*(t_{N_{\rm RH}}; z_i(t_1))\| \leq \delta^2\kappa.$$

Applying this recursively up to $k = N_{\rm RH}$, each at time $t = t_{N_{\rm RH}}$, and summing up both sides of the inequalities gives

$$\sum_{k=1}^{N_{\rm RH}} \|z_{di}^*(t_{N_{\rm RH}}; z_i(t_k)) - z_{di}^*(t_{N_{\rm RH}}; z_i(t_{k-1}))\| \leq N_{\rm RH} \cdot \delta^2\kappa \approx T\delta\kappa.$$

Finally, from the triangle inequality, we have that

$$\|z_i(t_{N_{\rm RH}}) - z_{di}^*(t_{N_{\rm RH}}; z_i(t_0))\| \leq T\delta\kappa,$$

where we use the fact that $z_{di}^*(t_{N_{\rm RH}}; z_i(t_{N_{\rm RH}})) = z_i(t_{N_{\rm RH}})$.

The analysis above shows that the current state, after $N_{\rm RH}$ iterations, deviates from the original optimal state, at the appropriate point in time, by at most $T\delta\kappa$. Thus, the deviation of the current state from the original optimal state at time $t_{N_{\rm RH}}$ is bounded and proportional to the update period δ. Therefore, when the update period is small enough to satisfy the theoretical conditions for asymptotic stability, the compatibility constraints imply the closed-loop trajectory must remain relatively close to the trajectory computed at initialization! This is not a desirable property as it implies that the transient response will only be as good as the initial response. In the absence of the compatibility constraints, more optimal responses than continuing along, or close to, the initial response could be chosen as time proceeds. In this sense, the compatibility constraint does imply that the transient response will be more sluggish, particularly for smaller update times.

More generally, the compatibility constraint also takes away from the power of the receding horizon philosophy, namely, the ability to recompute a *new* optimal action based on current conditions, at *every* receding horizon update. Still, larger values for the update period than required by the theory achieve good convergence, as observed in the numerical experiments of Chapter 6. The experiments also indicate that, even if the compatibility constraints are removed, the distributed implementation is more sluggish, albeit only slightly, than the centralized implementations. The reason is that, at each update, agents rely on information that is suboptimal, namely, agents assume their neighbors will keep doing what was previously optimal. Since agents are relying on "old intensions" for their neighbors, the overall closed-loop response is inherently more sluggish than the centralized implementation, even without enforcing the compatibility constraints.

5.2.3 Distributed Implementation Solves a Modified Centralized Problem

In contrast to the parallelization techniques discussed in Section 1.1.3, the distributed implementation does not recover the centralized solution of the original optimal control Problem 2.1. In fact, the original optimal control problem required no additional compatibility constraints, while the distributed optimal control problems do require these constraints in theory. Still, it is possible to define a single modified optimal control problem for which the centralized solution is exactly the distributed solution.

Denote $\hat{u} = (\hat{u}_1, ..., \hat{u}_{N_a})$ where each component is given by Definition 4.5. Collecting the distributed optimal control problems, we have the following modified problem.

Problem 5.1 At any update time t_k, $k \in \mathbb{N}$, given $z(t_k)$ and $\hat{u}(s; z(t_k))$ for all

$s \in [t_k, t_k + T]$, find

$$J^*_\Sigma(z(t_k), T) = \min_{u(\cdot)} J_\Sigma(z(t_k), u(\cdot), T),$$

where $J_\Sigma(z(t_k), u(\cdot), T)$ is equal to

$$\int_{t_k}^{t_k+T} \gamma \left(\sum_{i=1}^{N_a} L_i^z(z_i(\tau; z_i(t_k)), \hat{z}_{-i}(\tau; z_{-i}(t_k))) + \|u(\tau; z(t_k))\|_R^2 \right) \, \mathrm{d}\tau \\ + \gamma \|z(t_k + T; z(t_k)) - z^c\|^2_{\widehat{P}} \ ,$$

subject to

$$\left.\begin{array}{l} \dot{z}(s; z(t_k)) = f(z(s; z(t_k)), u(s; z(t_k))) \\ \dot{\hat{z}}(s; z(t_k)) = f(\hat{z}(s; z(t_k)), \hat{u}(s; z(t_k))), \\ u(s; z(t_k)) \in \mathcal{U}^{N_a} \\ z(s; z(t_k)) \in \mathcal{Z}^{N_a} \\ \max_i \{\|z_i(s; z_i(t_k)) - \hat{z}_i(s; z_i(t_k))\|\} \leq \delta^2 \kappa \end{array}\right\} \quad s \in [t_k, t_k + T],$$

$$z(t_k + T; z(t_k)) \in \Omega(\boldsymbol{\varepsilon}) := \Omega_1(\varepsilon_1) \times \cdots \times \Omega_{N_a}(\varepsilon_{N_a}).$$

By construction, the assumed trajectories also satisfy the constraints $\hat{u}(s; z(t_k)) \in \mathcal{U}^{N_a}$ and $\hat{z}(s; z(t_k)) \in \mathcal{Z}^{N_a}$. Since the compatibility constraint in Problem 4.1 uses the Euclidean norm, we require that the maximum distance between actual and assumed state trajectories, for all agents, be bounded by $\delta^2\kappa$. This is achieved by using the $\max_i$ function in the problem above. If each compatibility constraint in Problem 4.1 used the $\|\cdot\|_\infty$ norm instead, i.e., the component-wise maximum, then the compatibility constraint above would simply be replaced by $\|z(s) - \hat{z}(s)\|_\infty \leq \delta^2\kappa$. Note that the optimal value function $J^*_\Sigma(z(t_k), T)$ for the problem above is exactly the same function used in the stability analysis in the previous sections.

Thus, the distributed receding horizon control law is also the centralized receding horizon control law for the problem above. It might be speculated that one could use alternative choices for $\hat{u}$ and still preserve this problems distributed structure and the stability analysis of the previous sections. However, it was exactly the choice defined in Definition 4.5 for $\hat{u}$ that enabled the stability result. Specifically, the choice enabled the comparison between the net cost of previously optimal plans and the net cost of currently optimal plans. It is precisely this comparison that leverages most stability proofs of receding horizon control, including [11, 32, 49]. In general, therefore, other choices for $\hat{u}$ will not preserve this improvement property in comparing the net cost from one update to the next.

5.3 Alternative Formulations

In this section, alternative formulations of the distributed receding horizon control implementation presented in Chapter 4 are explored. First, a dual-mode version of the distributed receding horizon control law is given in Section 5.3.1. This version alleviates the problem of requiring the update period δ to shrink to zero, a property required in the main stability result Theorem 4.1 that guarantees convergence to the objective state. The dual-mode version permits a fixed, positive upper bound on the update period, whereby the agents can converge to a compact neighborhood of the objective state and henceforth apply decoupled feedback controllers. After the dual-mode version is formulated, exchanging assumed state information between neighbors, as opposed to assumed control information, is discussed. Finally, in Section 5.3.3, the implications of replacing each state compatibility constraints with a *control* compatibility constraint is explored.

5.3.1 Dual-Mode Distributed Receding Horizon Control

To construct the dual-mode version, we will make use of the monotonicity of $J^*_\Sigma(z(t_k), T)$ for guaranteeing invariance properties of the distributed receding horizon control law. In particular, under the conditions of Theorem 4.1, $J^*_\Sigma(z(t_k), T)$ monotonically decreases until $z(t_k) = z^c$. Therefore, the concatenated state $z(t_k)$ is contracting, in some norm-sense, at each receding horizon update. The control switch must therefore rely on a test on the *entire state* $z(t_k)$ to guarantee all agents are in their terminal constraint sets. A sufficient test is to determine if $z(t_k) \in \widehat{\Omega}(\varepsilon_{\min})$, where

$$\widehat{\Omega}(\varepsilon_{\min}) := \left\{ z \in \mathbb{R}^{2nN_a} \ : \ \|z - z^c\|^2_{\widehat{P}} \leq \varepsilon_{\min}, \ \varepsilon_{\min} = \min_i \varepsilon_i \right\}.$$

If this holds, then

$$\|z(t_k) - z^c\|^2_{\widehat{P}} = \sum_{i=1}^{N_a} \|z_i(t_k) - z_i^c\|^2_{P_i} \leq \varepsilon_{\min} \implies \|z_i(t_k) - z_i^c\|^2_{P_i} \leq \varepsilon_{\min}, \ \forall \ i = 1, ..., N_a,$$

guaranteeing all agents are in their terminal constraint sets. Under stated assumptions, we will show that $\|z(t_k) - z^c\|^2_W$ is contracting with each update, where the positive definite, symmetric weighting W will be defined more precisely below. Since the contraction is happening with a different norm-weighting than $\widehat{P}$, we require a sufficient test on the W-weighted quadratic term to guarantee that $z(t_k) \in \widehat{\Omega}(\varepsilon_{\min})$ holds. Recall that $\lambda_{\max}(Q_0)$ is the maximum eigenvalue of any square matrix Q_0, and let $\lambda_{\min}(Q_0)$ denote the minimum eigenvalue of Q_0. Observe that if

$$\|z(t_k) - z^c\|^2_W \leq \frac{\lambda_{\min}(W)\varepsilon_{\min}}{\lambda_{\max}(\widehat{P})}, \tag{5.1}$$

then we have

$$\lambda_{\min}(W)\|z(t_k) - z^c\|^2 \leq \frac{\lambda_{\min}(W)\varepsilon_{\min}}{\lambda_{\max}(\widehat{P})} \quad \implies \quad \|z(t_k) - z^c\|^2_{\widehat{P}} \leq \varepsilon_{\min}.$$

The test on the W-weighted quadratic term is more conservative than testing for $z(t_k) \in \widehat{\Omega}(\varepsilon_{\min})$ directly. However, since the W-weighted term is shown to strictly decrease for the closed-loop system, we are guaranteed invariance, i.e., once equation (5.1) is true, it remains true for all future time. Positive invariance is required as it will take some time for the agents to *agree*, in a distributed way, that they are in the set $\widehat{\Omega}(\varepsilon_{\min})$.

It is required that δ be no larger than a value, denoted $\delta_{\max}$, that guarantees monotonic decrease of the value function $J^*_\Sigma(z(t_k), T)$ so that equation (5.1) holds after some finite time. Now, we assume some quadratic bounds on the function J^*_Σ and show convergence of a sequence such that equation (5.1), and hence $z(t_k) \in \widehat{\Omega}(\varepsilon_{\min})$, is guaranteed to hold after some finite number of iterations, from any feasible state at initialization.

Assumption 5.1 For any $z \in Z_\Sigma$, there exists positive constants $k_1, k_2 \in \mathbb{R}$, $k_2 > k_1$, and positive definite, symmetric matrix $W \in \mathbb{R}^{nN_a \times nN_a}$ such that

$$k_1 \|z - z^c\|^2_W \le J^*_\Sigma(z, T) \le k_2 \|z - z^c\|^2_W,$$

where

$$k_2 > \delta_{\max} \lambda_{\min}(Q)/\lambda_{\max}(W),$$
$$k_2 - k_1 \le \delta_{\max} \lambda_{\min}(Q)/(2\lambda_{\max}(W)),$$

and $\delta_{\max}$ is defined as

$$\delta_{\max} = \frac{(\gamma - 1)c}{[2\kappa T \ + \ \mathcal{K}(\rho_{\max} + u_{\max})]\,(\gamma + 1)\rho_{\max}\lambda_{\max}(Q)\sum_{i=1}^{N_a} N_a^2} \ ,$$

and

$$c \triangleq \frac{\sum_{i=1}^{N_a} N_i^2}{N_a^2} \left(\frac{\lambda_{\min}(W)}{\lambda_{\max}(W)} \right) \frac{\lambda_{\min}(Q)\varepsilon_{\min}}{2\lambda_{\max}(\widehat{P})}.$$

Additionally, we assume that $\mathcal{K}(\rho_{\max} + u_{\max}) \ge 2\kappa T$.

The requirement that $k_2 > \delta_{\max}\lambda_{\min}(Q)/\lambda_{\max}(W)$ is not too restrictive, since the right-hand side is a small number in general. The requirement that $k_2 - k_1 \le \delta_{\max}\lambda_{\min}(Q)/(2\lambda_{\max}(W))$ means k_2 is close to k_1. Equivalently, this means that $J^*_\Sigma(z, T)$ basically exhibits quadratic growth, with a W-weighted norm. In the analysis that follows, it is not required to know the values of k_2 and k_1, so they do not need to be estimated. Since $\delta_{\max}$ is to be computed and used in practice, it will be required to estimate the ratio $\lambda_{\min}(W)/\lambda_{\max}(W)$. More simply, if a reasonable *lower* *bound* on this ratio can be computed, the bound can be used to define c, although this results in more conservatism. Aside from computing the ratio, or a lower bound on it, the weighting matrix W does not need to be computed.

The ratio $\sum N_i^2/N_a^2$ in c is solely an artifact of the bounding argument used in proving the lemma below. The final condition in the assumption, i.e., $\mathcal{K}(\rho_{\max} + u_{\max}) \geq 2\kappa T$, is stated to simplify the analysis below and is not restrictive. The following lemma guarantees that, with the bounding assumptions above, setting $\delta = \delta_{\max}$ results in a desirable invariance property for the closed-loop system.

Lemma 5.1 *Under Assumptions 4.2–5.1, for a given fixed horizon time* $T > 0$ *and for any state* $z(t_{-1}) \in Z_\Sigma$ *at initialization, if the update period* $\delta = \delta_{max}$*, then for the closed-loop system (4.5) there exists a finite integer* $l \in \mathbb{N}$ *at which* $z(t_l) \in \widehat{\Omega}(\varepsilon_{min})$ *and* $z(t_k) \in \widehat{\Omega}(\varepsilon_{min})$ *for all* $k \geq l$*.*

Proof: As in the proof of Theorem 4.1, combining Lemma 4.3 and Lemma 4.4, the monotonicity condition on $J^*_\Sigma(z(t_k), T)$, for any $k \in \mathbb{N}$, is

$$J^*_\Sigma(z(t_{k+1}), T) - J^*_\Sigma(z(t_k), T) \;\leq\; -\int_{t_k}^{t_k+\delta} \|z^*_{\text{dist}}(\tau; z(t_k)) - z^c\|^2_Q \;\mathrm{d}\tau,$$

when $\delta \in (0, \delta(z(t_k))]$, with $\delta(z(t_k))$ defined in equation (4.12). The upper bound $\delta_{\max}$ on δ must have the structure of equation (4.12) for the monotonicity equation above to be valid. From the Taylor series approximation in Assumption 4.6, we have

$$\int_{t_k}^{t_k+\delta} \|z^*_{\text{dist}}(\tau; z(t_k)) - z^c\|^2_Q \;\mathrm{d}\tau$$
$$\approx \delta\|z(t_k) - z^c\|^2_Q + \delta^2 (z(t_k) - z^c)^T Q f(z(t_k), u^*_{\text{dist}}(t_k; z(t_k))).$$

From the assumed bounds, we can bound the second term as

$$\begin{aligned} &-(z(t_k) - z^c)^T Q f(z(t_k), u^*_{\text{dist}}(t_k; z(t_k))) \\ &\qquad \leq\; \lambda_{\max}(Q)\|z(t_k) - z^c\| \; \|f(z(t_k), u^*_{\text{dist}}(t_k; z(t_k)))\| \\ &\qquad \leq\; \lambda_{\max}(Q)(N_a\rho_{\max})\mathcal{K}N_a(\rho_{\max} + u_{\max}). \end{aligned}$$

Substitution into the monotonicity equation gives

$$\begin{aligned}
&J^*_\Sigma(z(t_{k+1}),T) - J^*_\Sigma(z(t_k),T) \\
&\quad\le\ -\delta\|z(t_k)-z^c\|^2_Q + \delta^2\rho_{\max}(\rho_{\max}+u_{\max})\mathcal{K}N_a^2\lambda_{\max}(Q) \\
&\quad\le\ -\delta\frac{\lambda_{\min}(Q)}{\lambda_{\max}(W)}\|z(t_k)-z^c\|^2_W + \delta^2\rho_{\max}(\rho_{\max}+u_{\max})\mathcal{K}N_a^2\lambda_{\max}(Q).
\end{aligned}$$

Using the bounds on $J^*_\Sigma(z(t_k),T)$ from Assumption 5.1 we have

$$\begin{aligned}
&k_1\|z(t_{k+1})-z^c\|^2_W \\
&\quad\le\ \left[k_2-\delta\frac{\lambda_{\min}(Q)}{\lambda_{\max}(W)}\right]\|z(t_k)-z^c\|^2_W + \delta^2\rho_{\max}(\rho_{\max}+u_{\max})\mathcal{K}N_a^2\lambda_{\max}(Q).
\end{aligned}$$

Denoting $y_k = \|z(t_k)-z^c\|^2_W \in [0,\infty)$, and setting $\delta=\delta_{\max}$, we rewrite this as

$$\begin{aligned}
y_{k+1}\ \le\ \rho y_k+\phi, \quad &\text{where} \quad \rho = \frac{k_2-\delta_{\max}\lambda_{\min}(Q)/\lambda_{\max}(W)}{k_1}, \\
&\text{and} \quad \phi = \delta^2_{\max}\rho_{\max}(\rho_{\max}+u_{\max})\mathcal{K}N_a^2\lambda_{\max}(Q)/k_1.
\end{aligned}$$

From Assumption 5.1, $k_2 > \delta_{\max}\lambda_{\min}(Q)/\lambda_{\max}(W)$ and

$$k_2-k_1 \le \delta_{\max}\lambda_{\min}(Q)/(\lambda_{\max}(W)2) < \delta_{\max}\lambda_{\min}(Q)/\lambda_{\max}(W),$$

which implies $0<\rho<1$. Considering the sequence y_{k+1}, which is bounded for each $k\in\mathbb{N}$, by $\rho y_k+\phi$, we observe that

$$y_\infty = \lim_{k\to\infty} y_{k+1} \le \lim_{k\to\infty}\left[\rho^k y_0 + \phi\left(\sum_{i=0}^{k-1}\rho^i\right)\right] = \frac{\phi}{1-\rho}.$$

Also, we can bound the ratio $\phi/(1-\rho)$ as

$$\begin{aligned}
\frac{\phi}{1-\rho}\ &=\ \frac{\delta^2_{\max}\rho_{\max}(\rho_{\max}+u_{\max})\mathcal{K}N_a^2\lambda_{\max}(Q)}{k_1+\delta_{\max}\lambda_{\min}(Q)/\lambda_{\max}(W)-k_2} \\
&\le\ 2\delta_{\max}\rho_{\max}(\rho_{\max}+u_{\max})\mathcal{K}N_a^2\lambda_{\max}(W)\lambda_{\max}(Q)/\lambda_{\min}(Q),
\end{aligned}$$

where the inequality uses the assumed upper bound on $k_2 - k_1$. Substitution for $\delta_{\max}$ gives an expression that can further be bounded as follows. First, observe that $(\gamma - 1) < (\gamma + 1)$ for any $\gamma > 1$. Also, assuming $\mathcal{K}(\rho_{\max} + u_{\max}) \geq 2\kappa T$ implies that

$$\frac{\mathcal{K}(\rho_{\max} + u_{\max})}{2\kappa T \; + \; \mathcal{K}(\rho_{\max} + u_{\max})} \leq 2.$$

Now, we can bound the sequence limit as

$$\frac{\phi}{1-\rho} \;\leq\; \frac{cN_a^2 \lambda_{\max}(W)}{\lambda_{\min}(Q) \sum_{i=1}^{N_a} N_i^2} \;=\; \frac{\lambda_{\min}(W)\varepsilon_{\min}}{2\lambda_{\max}(\widehat{P})},$$

where the last equality follows from the definition of c. Therefore, denoting $y_\infty = \|z(t_\infty) - z^c\|_W^2$, we have

$$\|z(t_\infty) - z^c\|_W^2 \;\leq \frac{\lambda_{\min}(W)\varepsilon_{\min}}{2\lambda_{\max}(\widehat{P})} \quad \Longrightarrow \quad \|z(t_\infty) - z^c\|_{\widehat{P}}^2 \leq \frac{\varepsilon_{\min}}{2},$$

and so $z(t_\infty)$ is in the interior of $\widehat{\Omega}(\varepsilon_{\min})$. Moreover, for any $y_k = \vartheta + \phi/(1-\rho)$, where $\vartheta \in (0, \infty)$, the sequence bound $y_{k+1} \leq \rho y_k + \phi$ guarantees strict monotonic decrease of the sequence. The reason is $\rho y_k + \phi - y_k = -(1-\rho)y_k + \phi = -(1-\rho)\vartheta$, and so $y_{k+1} \leq y_k - (1-\rho)\vartheta$. In particular, once $y_k \leq 2\phi/(1-\rho)$, we are guaranteed that $z(t_k) \in \widehat{\Omega}(\varepsilon_{\min})$ holds, since the factor of 2 simply removes the 1/2 in the implication above.

Now, there is a finite integer $l \in \mathbb{N}$ for which $y_l \leq 2\phi/(1-\rho)$. If this were not the case, $y_{k+1} \leq y_k - (1-\rho)\vartheta$ would hold for all $k \in \mathbb{N}$, which implies $y_k \to -\infty$ as $k \to \infty$. However, this contradicts the fact that $y_k \geq 0$ for all $k \in \mathbb{N}$. Therefore, there exists a finite integer $l \in \mathbb{N}$ for which $y_l \leq 2\phi/(1-\rho)$. Also, since the sequence is required to continue to decrease up to at most the limit bound on $\phi/(1-\rho)$, we have positive invariance as well. This concludes the proof. ■

Remark 5.2 If the update period δ is less than $\delta_{\max}$, then the analysis above still holds, provided the bound on the difference $k_2 - k_1$ in Assumption 5.1 is tightened by replacing $\delta_{\max}$ by δ.

For the control switch to occur synchronously between the agents, we require a distributed means of determining when equation (5.1) holds. Since Lemma 5.1 guarantees monotonic decrease in the W-norm sense, we shall cast the test in terms of the W-norm. Although this incurs more conservatism, it implies that the agents will not come to agreement unless they have all reached the state for which all subsequent receding horizon updates render $\widehat{\Omega}(\varepsilon_{\min})$ a positively invariant set of the closed-loop system.

Distributed Consensus Algorithm for Synchronous Control Switching [59]. The algorithm defined here is a distributed means of determining when equation (5.1) holds. By distributed, we mean each agent communicates only with neighboring agents. For the algorithm to converge, a minimum number of information exchanges is required. We elect to have a separate sample rate for this algorithm. Over the time interval $[t_k, t_{k+1})$, we assume neighboring agents will communicate N_s times, using notation

$$\tau_{k,l} = t_k + \delta(l/N_s), \quad l = 0, 1, ..., N_s - 1,$$

to denote the times at which neighbors exchange information.

To define the dynamics of the switch mechanism, we introduce some notation. At any algorithm update time $\tau_{k,l}$ and for any agent i, let $x_i(\tau_{k,l}) \in \mathbb{R}^+$ be a non-negative scalar value. Also, denote $\varepsilon_0 = \varepsilon_{\min}/\lambda_{\max}(\widehat{P})$ and $\psi = \lambda_{\max}(W)/\lambda_{\min}(W)$, where W is the weighting matrix defined in Assumption 5.1. As with the definition of c in Assumption 5.1, a reasonable upper bound for this ratio, or equivalently a lower bound on $\lambda_{\min}(W)/\lambda_{\max}(W)$, could be used in the place of ψ below. The algorithm is as follows. For every agent $i = 1, ..., N_a$:

1. At any time $\tau_{k,0} = t_k$, set

$$x_i(\tau_{k,0}) = N_a \psi \|z_i(t_k) - z_i^c\|^2,$$

   transmit $x_i(\tau_{k,0})$ and receive $x_j(\tau_{k,0})$ from each neighbor $j \in \mathcal{N}_i$.

2. For each time $\tau_{k,l}$, $l = 1, 2, ..., N_s - 1$,

   (a) set

$$x_i(\tau_{k,l}) = x_i(\tau_{k,l-1}) + \frac{\zeta}{N_s} \sum_{j \in \mathcal{N}_i} \left(x_j(\tau_{k,l-1}) - x_i(\tau_{k,l-1}) \right),$$

   where $\zeta > 0$, and

   (b) transmit $x_i(\tau_{k,l})$ and receive $x_j(\tau_{k,l})$ from each neighbor $j \in \mathcal{N}_i$.

3. Define $x_i(\tau_{k,N_s})$ according to the equation in 2(a) above. If

$$x_i(\tau_{k,N_s}) \leq \varepsilon_0 - \epsilon,$$

   where ϵ is a specified tolerance satisfying $0 < \epsilon \ll \varepsilon_0$, then switch at time t_{k+1} and exit the algorithm. Otherwise, return to 1. **End of algorithm**.

Under the conditions stated in a lemma below, namely if N_s is sufficiently large, then

$$|x_i(\tau_{k,N_s}) - \text{Ave}(x(t_k))| \leq \epsilon, \quad \forall i = 1, ..., N_a,$$

where

$$\text{Ave}(x(t_k)) = \frac{1}{N_a} \sum_{i=1}^{N_a} x_i(t_k) = \psi \sum_{i=1}^{N_a} \|z_i(t_k) - z_i^c\|^2 = \psi \|z(t_k) - z^c\|^2.$$

From this, we have

$$\psi\|z(t_k) - z^c\|^2 - \epsilon \ \leq \ x_i(\tau_{k,N_s}) \ \leq \ \psi\|z(t_k) - z^c\|^2 + \epsilon.$$

Therefore, the test in part 3 of the algorithm guarantees that

$$\|z(t_k) - z^c\|_W^2 \leq \lambda_{\min}(W)\varepsilon_0$$

and equation (5.1) holds.

Lemma 5.2 *For a specified tolerance ϵ, satisfying $0 < \epsilon \ll \varepsilon_0$, the distributed consensus algorithm for synchronous control switching converges in N_s iterations provided that*

- *$N_s > \zeta d_m$, where d_m is the maximum node degree of the graph $\mathcal{G}$, and*
- *denoting $\lambda = 1 - (\zeta/N_s)\lambda_2(L)$, where L is the Laplacian of the graph $\mathcal{G}$ and $\lambda_2(L)$ is the second largest eigenvalue of L,*

$$N_s \geq \frac{\log \epsilon - \log d_0}{\log \lambda}, \quad d_0 = \left[\sum_{i=1}^{N_a} \left(x_i(\tau_{k,0}) - Ave(x(t_k)) \right)^2 \right]^{1/2},$$

where d_0 denotes a measure of initial disagreement.

It is proven in [59] that the eigenvalue $\lambda_2(L)$ bounds the rate of convergence of the average consensus algorithm in continuous time. Converting to discrete time, the convergence bound becomes λ defined above, and the first condition in the lemma implies $0 < \lambda < 1$. The lemma says that if N_s is sufficiently large, every agent will meet the tolerance specified in the algorithm. Therefore, the test in step 3 of the algorithm implies the agents agree to synchronously switch to the linear feedback controllers only if equation (5.1) holds, i.e., if all agents are in the interior of their terminal constraint sets. Note that if $d_0 \leq \epsilon$, consensus has been reached from the initial values for each x_i, and the algorithm would terminate in one iteration. The results of the lemma presumes that initially $d_0 > \epsilon$, as is the case in practice. With $0 < \lambda < 1$ and $d_0 > \epsilon$, the second condition on N_s provides a lower bound that is positive.

Remark 5.3 Since N_s will be large in general, the communication requirements between receding horizon updates will be demanding for the algorithm to converge. To alleviate this, observe that from the invariance property stated in Lemma 5.1, we know that once equation (5.1) holds, it will continue to hold. As such, it is possible to communicate once every receding horizon update. This is done by defining $\tau_{kN_s,l} = t_{kN_s} + \delta l$, so step 1 is entered every δN_s seconds. The tradeoff of course is that the algorithm will take considerably more time to converge. A sample rate between these two extremes could be used, one that is appropriate for the given bandwidth limitations.

We now define the dual-mode distributed receding horizon controller.

Definition 5.1 (*Dual-mode distributed receding horizon controller*)

Data: Initial state $z(t_{-1}) \in Z_\Sigma$, horizon time $T > 0$, update time $\delta = \delta_{\max}$.

Initialization: At time t_{-1}, follow the procedure given in Definition 4.4, yeiding a control for time $t \in [t_{-1}, t_0)$.

Controller:

1. For time $t \in [t_0, t_1)$, employ the distributed receding horizon control law $u^*_{\text{dist}}(t; z(t_0))$.
2. At any time t_k, $k \in \mathbb{N}$:
   (a) If the distributed consensus algorithm has converged (step 3), employ the decoupled linear feedbacks defined in Assumption 4.3 for all time $t \geq t_k$. Else:
   (b) Employ the distributed receding horizon control law $u^*_{\text{dist}}(t; z(t_k))$ for time $t \in [t_k, t_{k+1})$.

Lemma 5.1 guarantees that, under the assumptions, the inequality in equation (5.1) will hold after a finite number $l \in \mathbb{N}$ of receding horizon updates. Under the condition of Lemma 5.2, the agents will agree to switch at time t_{l+1} to the decoupled linear feedback controllers. Since equation (5.1) holds, the agents are known to be in a set for which these feedbacks are asymptotically stabilizing. Therefore, the dual-mode distributed receding horizon controller results in asymptotic stability with region of attraction Z_Σ. Formally, we now state this as the second main theorem.

Theorem 5.1 *Under Assumptions 4.2–5.1 and under the conditions of Lemma 5.2, for a given fixed horizon time $T > 0$ and for any state $z(t_{-1}) \in Z_\Sigma$ at initialization,* *if the update period $\delta = \delta_{max}$, then z^c is an asymptotically stable equilibrium point of the closed-loop system resulting from the dual-mode distributed receding horizon controller, and Z_Σ is a region of attraction.*

Remark 5.4 The distributed receding horizon control law of Theorem 4.1 requires that all agents have the following information available: the horizon time T, the update period δ, and all parameters in the computation for $\delta(z(t_k))$ in equation (4.12), which includes the centralized computation of $\|z(t_k) - z^c\|_Q^2$ at each receding horizon update. The dual-mode distributed receding horizon control law of Theorem 5.1 requires that all agents have the following information available: the horizon time T, all parameters in the computation for $\delta_{\max}$ in Assumption 5.1, and the parameters for the distributed consensus algorithm, satisfying the conditions in Lemma 5.2. Clearly, both controllers require some centralized information; however, the dual-mode version does not require any online centralized computations. We note that for the controller of Theorem 4.1, another consensus algorithm could be incorporated for distributed computation of $\|z(t_k) - z^c\|_Q^2$ at each receding horizon update.

5.3.2 Alternative Exchanged Inter-Agent Information

The distributed optimal control problems here require only the assumed state trajectories from each neighbor. As such, neighboring agents could instead exchange assumed state trajectories, rather than assumed control trajectories. The result would be that agents would then not have to integrate the equations of motion for each neighboring agent, and also not require a separate transmission to obtain the initial condition for each neighboring agent. The result is simplification in the optimal control computations; however, if $n > m$, the communication cost is greater when exchanging assumed state trajectories. To generalize, the tradeoff between exchanging control or state trajectory information, in terms of both communication and computation requirements, should dictate how the needed assumed information should be attained.

If there is uncertainty in the models that agents have of one another, the choice of exchanged information also has implications on robustness. For example, suppose each agent uses a high fidelity model of itself and low fidelity models of its neighbors in each local optimization problem. In that case, the assumed information about a neighbor will be less accurate if the low fidelity model of that neighbor is used to generate the needed trajectories than if the high fidelity trajectories are directly communicated between neighbors. Now, the tradeoff is between the combination of computational cost, communication bandwidth and accuracy of assumed information. As a final remark, we note that the computations required to integrate the model of every neighbor, to generate the needed assumed trajectories, are negligible compared to the computation required to find the optimal control for each agent.

5.3.3 Alternative Compatibility Constraint

We now discuss some implications of replacing each state compatibility constraint with a control compatibility constraint, as done in previous work [17, 18]. The state compatibility constraint is

$$\|z_i(t; z_i(t_k)) - \hat{z}_i(t; z_i(t_k))\| \leq \delta^2 \kappa, \quad t \in [t_k, t_k + T],$$

whereas the control compatibility constraint would be

$$\|u_i(t; z_i(t_k)) - \hat{u}_i(t; z_i(t_k))\| \leq \delta^2 \kappa, \quad t \in [t_k, t_k + T].$$

It is the proof of Lemma 4.3 that requires a bound on the norm of the difference between the assumed state trajectories and the actual state trajectories. Given the control constraint, and by application of Gronwall-Bellman inequality, the difference can be bounded as

$$\begin{aligned}
&\|z_i(t;z_i(t_k)) - \hat{z}_i(t;z_i(t_k))\| \\
&\qquad \le \int_{t_k}^{t} \|f_i(z_i(\tau;z_i(t_k)), u_i(\tau;z_i(t_k))) - f_i(\hat{z}_i(\tau;z_i(t_k)), \hat{u}_i(\tau;z_i(t_k)))\| \,\mathrm{d}\tau \\
&\qquad \le \int_{t_k}^{t} \mathcal{K}\left(\|z_i(\tau;z_i(t_k)) - \hat{z}_i(\tau;z_i(t_k))\| + \|u_i(\tau;z_i(t_k)) - \hat{u}_i(\tau;z_i(t_k))\|\right) \mathrm{d}\tau \\
&\qquad \le \mathcal{K}\delta^2\kappa(t-t_k) + \int_{t_k}^{t} \mathcal{K}\|z_i(\tau;z_i(t_k)) - \hat{z}_i(\tau;z_i(t_k))\| \,\mathrm{d}\tau \\
&\qquad \le \mathcal{K}\delta^2\kappa\left\{(t-t_k) + \mathcal{K}\int_{t_k}^{t} (\tau - t_k)\exp[\mathcal{K}(t-\tau)] \,\mathrm{d}\tau\right\} \\
&\qquad = \delta^2\kappa\left\{\exp[\mathcal{K}(t-t_k)] - 1\right\}.
\end{aligned}$$

The resulting constant ξ is then redefined to be

$$\xi = 2\gamma\kappa\rho_{\max}\lambda_{\max}(Q)T\left\{\frac{\exp[\mathcal{K}T] - (T\mathcal{K}+1)}{\mathcal{K}}\right\}\sum_{i=1}^{N_a} N_i(N_i - 1).$$

With the control compatibility constraint, ξ clearly grows faster with horizon time T, with the same growth relation to the other parameters as before. The upper bound on the update period δ, defined as $\delta_{\max}$ or $\delta(z(t_k))$ in equation (4.12), is proportional to $1/(\xi + c)$, where c is a constant. Therefore, for a given κ and δ that satisfies the theoretical bounds, the compatibility constraint bound $\delta^2\kappa$ could be larger by using the state constraint, rather than the control constraint, when T is large.

From the analysis above, the control compatibility constraint implies

$$\|z_i(t;z_i(t_k)) - \hat{z}_i(t;z_i(t_k))\| \le \delta^2\kappa\left\{\exp[\mathcal{K}(t-t_k)] - 1\right\},$$

while the state compatibility constraint is

$$\|z_i(t;z_i(t_k)) - \hat{z}_i(t;z_i(t_k))\| \le \delta^2\kappa,$$

for all time $t \in [t_k, t_k + T]$. Observe that $\exp[\mathcal{K}(t-t_k)] - 1 > 1$ is equivalent to

$t - t_k > \ln(2)/\mathcal{K}$. Therefore, *the control constraint is more conservative than the state constraint when* $t \in [t_k, t_k + \ln(2)/\mathcal{K})$, in that the computed state is required to be closer to the assumed state over that time interval. The interval $[t_k, t_k + \ln(2)/\mathcal{K})$ is part of the actual closed-loop system, since each optimized trajectory is implemented for $t \in [t_k, t_k + \delta)$. The impact that this has on the evolution of the closed-loop system will be discussed below. Now, *the control constraint is less conservative than the state constraint when* $t \in (t_k + \ln(2)/\mathcal{K}, t_k + T]$. It is likely that $T \gg \ln(2)/\mathcal{K}$, in which case we observe that the state compatibility constraint requires a tighter bound on the state deviation, than does the control compatibility constraint, over *most* of the time interval. A *controllability interpretation* of the equations above is that: 1) the state compatibility constraints imply a fixed limit on controllability, while 2) the control compatibility constraints imply that controllability is initially zero and grows exponentially with time.

Now, consider the impact of the two compatibility constraints on the evolution of the closed-loop system. In Section 5.2.2, the effect of the state compatibility constraint on the closed-loop state response is explored. From that section, we defined $N_{\mathrm{RH}} \in \mathbb{N}$ as the number of receding horizon updates after time t_0 such that $N_{\mathrm{RH}} \cdot \delta \approx T$. At time $t = t_{N_{\mathrm{RH}}}$, where $t_{N_{\mathrm{RH}}} := t_0 + N_{\mathrm{RH}} \cdot \delta \approx t_0 + T$, the state compatibility constraint implies that

$$\|z_i(t_{N_{\mathrm{RH}}}) - z^*_{di}(t_{N_{\mathrm{RH}}}; z_i(t_0))\| \leq T\delta\kappa.$$

In comparison, the control compatibility constraint results in

$$\|z_i(t_{N_{\mathrm{RH}}}) - z^*_{di}(t_{N_{\mathrm{RH}}}; z_i(t_0))\| \leq T\delta\kappa \left\{\exp[\delta\mathcal{K}] - 1\right\}.$$

Now, the current state at time $t_{N_{\mathrm{RH}}}$ deviates from the original optimal state by at most $T\delta\kappa \left\{\exp[\delta\mathcal{K}] - 1\right\}$. If $\delta < \ln(2)/\mathcal{K}$, then the control compatibility constraint requires the closed-loop state trajectory to remain closer to the initialized trajectory than does the state compatibility constraint. With the new version for ξ above, resulting from the control constraint, it turns out that $\delta < \alpha/\mathcal{K}$, with $\alpha \in (0, 1)$ in general, implying $\delta < \ln(2)/\mathcal{K}$ indeed holds. Therefore, obeying the sufficient conditions in the theory implies that the receding horizon controller will resemble the initial open-loop solution *more* when using the control compatibility constraint.

As a final comment for this section, the choice of compatibility constraint should ultimately be dictated by the degree of difficulty each imposes on the optimization algorithm. The state version, for example, must compete with the terminal constraint on the state, as well as the other state constraints, implicit (dynamics) and explicit. The control version, in our formulation, has to be explicitly conducive only with the control bounds, and implicity with constraints on the state through the dynamics. This suggests that the control version may be preferable for numerical reasons. Numerical experiments in the next chapter show that for coupling in the cost function, both compatibility constraints perform comparably, numerically and in terms of closed-loop performance. In the next section, coupling constraints are discussed. It turns out that for inter-agent coupling state constraints, only the state compatibility constraint makes sense in general, to maintain feasibility and stability of the closed-loop system.

5.4 Extensions of the Theory

In this section, we discuss extensions to the theory presented in Chapter 4. First, minimal conditions on the coupling integrated cost function are stated such that the decomposition exists and such that the theoretical properties of Chapter 4 are preserved. The conditions are minimal in the sense that we presume as little as possible about the form of the coupling cost function. Next, the possibility of coupling constraints between neighboring agents is considered. Finally, an implementation with locally synchronous timing requirements is described. A discussion on connections with other fields and future application areas is postponed until Chapter 7.

5.4.1 A General Coupling Cost Function

In the theory of the previous sections, neighboring agents are coupled by terms in the integrated cost function. While the generalization of the form of the integrated cost in Section 4.3 is restricted to quadratic, more general forms of the cost function are possible. For example, a non-quadratic coupling cost that is relevant for multi-vehicle formation stabilization and for which the decomposition is straightforward is given at the end of Chapter 6. In this section, we explore a generalization of the coupling integrated cost.

We begin with a single optimal control problem, for which there is defined an integrated cost function $L : \mathbb{R}^{nN_a} \times \mathbb{R}^{mN_a} \to \mathbb{R}$ that is twice continuously differentiable and satisfies

$$L(z,u) \geq 0,\ \forall (z,u) \in \mathbb{R}^{nN_a} \times \mathbb{R}^{mN_a}, \quad \text{and } L(z,u) = 0 \text{ if and only if } (z,u) = (z^c, 0).$$

As before, we are interested in stabilizing the dynamics to the equilibrium point z^c with $0 = f(z^c, 0)$.

By definition, a function $l(x,y)$ is additively separable (or just separable) in x and y if there exists functions l_x and l_y such that $l(x,y) = l_x(x) + l_y(y)$. A necessary condition for separability of a cost that is twice continuously differentiable is that $\frac{\partial^2 l(x,y)}{\partial x \partial y} = 0$ for all x and y. For simplicity, we assume that the single cost on the control is additively separable, using the notation

$$L(z,u) = L^z(z) + \sum_{i=1}^{N_a} L_i^u(u_i).$$

Consequently, we are interested in generic conditions for decomposing a general state dependent cost L^z. First, we can define what it means for any two agents to be coupled in the state dependent cost. To simplify things, we make the following assumption.

Assumption 5.2 *The cost $L^z(z)$ is the sum of separable and nonseparable terms, each of which is nonnegative.*

Definition 5.2 Agents $i_1, ..., i_M \in \{1, ..., N_a\}$, $2 \leq M \leq N_a$, are *coupled* if there exists a nonnegative term in the cost $L^z(z)$ that depends on at least one component of every vector $z_{i_1}, ..., z_{i_M}$ and the term is not additively separable in any such components.

The definition rules out looking at the zero function as coupling any subset of agents, since zero is additively separable. Consistent with previous notation, let $\mathcal{N}_i$ be the set of *neighbors* of agent i, where each neighbor is coupled to i in the cost function $L^z(z)$ in the sense of Definition 5.2. From the definition, it is clear that $i \in \mathcal{N}_j$ if and only if $j \in \mathcal{N}_i$, for any $i, j \in \{1, ..., N_a\}$. As before, let $z_{-i} = (z_{j_1}, ..., z_{j_{|\mathcal{N}_i|}})$ be the concatenated vector of states of the neighbors of i, and denote $N_i := |\mathcal{N}_i| + 1$. We make another assumption without loss of generality.

Assumption 5.3 *Every agent is coupled to at least one other agent, i.e., for every agent $i \in \{1, ..., N_a\}$, $N_i \geq 2$.*

By definition, every agent $i \in \{1, ..., N_a\}$ is coupled to agents $j_1, ..., j_{|\mathcal{N}_i|} \in \mathcal{N}_i$ by nonseparable terms in $L^z(z)$ and is coupled to no agents in the set $\{1, ..., N_a\} \setminus \mathcal{N}_i$.

Proposition 5.1 *There exists a nontrivial cost function $L_i^z : \mathbb{R}^{nN_i} \to \mathbb{R}$ for every agent $i \in \{1, ..., N_a\}$ such that*

$$L^z(z) = \sum_{i=1}^{N_a} L_i^z(z_i, z_{-i}) \quad \textit{and} \quad L_i^z(z_i, z_{-i}^c) = 0 \;\Rightarrow\; z_i = z_i^c.$$

Proof: For any agent i coupled to agents $j_1, ..., j_{M_i} \in \mathcal{N}_i$, for some M_i with $1 \leq M_i \leq |\mathcal{N}_i|$, denote the nonseparable coupling term as g_i which exists and is nontrivial by Definition 5.2 and the assumption above. Further, there may be terms in L^z that are additively separable and depend only on components of z_i for any agent i. Now, define each L_i^z as the sum of every separable term in L^z that depends solely on z_i (if such terms exist) and every nonseparable coupling term in L^z that depends on z_i and neighbors of i, normalizing each such term by the total number of agents coupled in the term (e.g., $g_i/(M_i + 1)$ for the general coupling term g_i). By construction, the sum of the L_i^z functions recovers the cost L^z.

Now, $L_i^z(z_i, z_{-i}^c) = 0$ always has at least the solution $z_i = z_i^c$. This follows since the L_i^z functions sum to give L^z and $L^z(z^c) = 0$. It remains to show that this is the unique solution. Let $\bar{z}_i$ be the vector of states of all agents that are not i and not in $\mathcal{N}_i$, i.e., not neighbors of i. Next, define the function $\bar{L}_i^z(z_{-i}, \bar{z}_i) := L^z(z) - L_i^z(z_i, z_{-i})$, i.e., $\bar{L}_i^z$ is all terms in L^z excluding those in L_i^z. Clearly, $\bar{L}_i^z$ does not depend on z_i. By assumption, we have $L^z(z_i, z_{-i}^c) = 0$, which holds *for any* $\bar{z}_i$. Setting the state of every agent in $\bar{z}_i$ to its equilibrium value, and denoting the resultant vector $\bar{z}_i^c$, we know that $L^z(z^c) = L_i^z(z_i^c, z_{-i}^c) + \bar{L}_i^z(z_{-i}^c, \bar{z}_i^c) = 0$ and therefore $\bar{L}_i^z(z_{-i}^c, \bar{z}_i^c) = 0$. So, setting $\bar{z}_i = \bar{z}_i^c$ and $z_{-i} = z_{-i}^c$, we have by assumption that $L_i^z(z_i, z_{-i}^c) + \bar{L}_i^z(z_{-i}^c, \bar{z}_i^c) = 0$, which implies that

$$L(z_1^c, ..., z_{i-1}^c, z_i, z_{i+1}^c, ..., z_{N_a}^c) = 0.$$

Since $L(z) = 0$ if and only if $z = z^c$, we have that $L_i^z(z_i, z_{-i}^c) = 0$ implies $z_i = z_i^c$. ■

Proposition 5.1 is an existence statement, and the proof constructs a specific function L_i^z for each i. By construction, each such function is unique. A different choice for the weighting of each nonseparable coupling term, i.e., one other than normalizing by the number of agents coupled in the term, can be chosen, provided the weighting preserves the summation in the proposition. It is also noted that the second part of Proposition 5.1 can be considered a detectability condition.

The decomposition can now be performed in terms of the unique cost functions L_i^z constructed in the proof of Proposition 5.1. The decomposition is stated as a definition, similar to Definition 4.1.

Definition 5.3 The *distributed integrated cost function* for each agent $i \in \{1, ..., N_a\}$ is defined as

$$L(z_i, z_{-i}, u_i) = \gamma \left[L_i^z(z_i, z_{-i}) + L_i^u(u_i)\right],$$

where $\gamma \in (1, \infty)$.

From Proposition 5.1 and the definition above, $L(z_i, z_{-i}^c, u_i) = 0$ if and only if $z_i = z_i^c$ and $u_i = 0$.

Remark 5.5 It may be that an agent i is coupled to all other agents $\{1, ..., N_a\}\backslash\{i\}$. In that case, the communication cost may be prohibitive and multi-hop information exchanges may be required. When accounting for delays and information loss due to communication, however, the multi-hop situation will not in general perform as well as when every agent has a direct link to every neighbor. The effects of real-time communication delays and loss, in single and multi-hop situations, will be part of the ongoing research in this area.

In many multiagent systems, coupling between agents also occurs in constraints. In the next section, we explore the implications of coupling state constraints on the distributed receding horizon control implementation.

5.4.2 Inter-Agent Coupling Constraints

We now discuss an approach for handling constraints that couple states of neighbors. Assume that $\mathcal{Z} \subseteq \mathbb{R}^{nN_a}$ is the centralized state constraint set. The set $\mathcal{Z}$ should first be broken up into constraint sets $\mathcal{Z}_i$, one for each agent. For the moment, let $\bar{z}_i = (z_1, ..., z_{i-1}, z_{i+1}, ..., z_{N_a})$, i.e., $\bar{z}_i \in \mathbb{R}^{n(N_a-1)}$ is the vector z excluding the subvector z_i. Assume that $\mathcal{Z}$ is defined by a set of inequalities, which for any $i = 1, ..., N_a$ can be broken up into two sets of inequalities: those that depend on components of z_i, and those that do not. The two sets are identified by the constraint vector functions $g_i(z_i, z_{-i}) \leq 0$ and $\bar{g}_i(\bar{z}_i) \leq 0$, where the "$\leq$" is here meant to hold for each component of the constraint vectors. As such, we have

$$\mathcal{Z} = \left\{ z \in \mathbb{R}^{nN_a} \ : \ g_i(z_i, z_{-i}) \leq 0 \text{ and } \bar{g}_i(\bar{z}_i) \leq 0, \text{ for any } i = 1, ..., N_a \right\}.$$

Finally, we have the definition for each $\mathcal{Z}_i$ as

$$\mathcal{Z}_i = \left\{ (z_i, z_{-i}) \in \mathbb{R}^{nN_i} \ : \ g_i(z_i, z_{-i}) \leq 0 \right\}.$$

Remark 5.6 By assumption, there are no constraints in $\mathcal{Z}$ that couple the states of agents that are not neighbors, i.e., that are not already coupled in the integrated cost function. This assumption is not required for the results to remain valid. It is acceptable for the "neighbor" relationship to be defined through coupling constraints, rather than through coupling in the cost function.

Inter-agent coupling state constraints are possible by using the state compatibility constraint in each local optimal control problem and when the following assumption holds.

Assumption 5.4 For every $i = 1, ..., N_a$, given $\mathcal{Z}_i$ and the associated constraint vector function $g_i(z_i, z_{-i}) \in \mathbb{R}^{n_i}$, there exists a constant vector $c^i = (c_1^i, ..., c_{n_i}^i) \in \mathbb{R}^{n_i}$ such that the set

$$\widehat{\mathcal{Z}}_i = \{(z_i, z_{-i}) \in \mathcal{Z}_i \ : \ g_i(z_i, z_{-i}) \leq -c^i\}$$

is not empty and

$$(z_i(t; z_i(t_k), \hat{z}_{-i}(t; z_{-i}(t_k))) \in \widehat{\mathcal{Z}}_i \implies (z_i(t; z_i(t_k), z_{-i}(t; z_{-i}(t_k))) \in \mathcal{Z}_i,$$

for all $t \in [t_k, t_k + T]$.

The conditions above presume it is possible to make the constraints *more conservative* such that enforcing them locally, with assumed state trajectories for neighbors, results in constraint satisfaction with the actual state trajectories. The added conservatism, in reducing the upper bound on every g_i function, is required to mitigate the deviation between actual and assumed state trajectories, which still occurs but is bounded by each compatibility constraint. An example is now provided to clarify what the constant c^i can look like.

Example 5.1 For second-order vehicle dynamics, each state is partitioned as $z_i = (q_i, \dot{q}_i)$, and every vehicle has a collision avoidance constraint with every neighboring vehicle, in the form

$$\rho_{\min} \leq \|q_i(t; z_i(t_k)) - q_j(t; z_j(t_k))\|, \quad \text{for each } j \in \mathcal{N}_i, \ \forall t \in [t_k, t_k + T],$$

where $\rho_{\min}$ is the minimum separation distance. Now, define $c^i = (c_1^i, ..., c_{|\mathcal{N}_i|}^i)$ with every $c_j^i = \kappa\delta^2$. In the ith local optimal control problem, for any i, the collision avoidance constraints are incorporated as

$$\rho_{\min} - \|q_i(t; z_i(t_k)) - \hat{q}_j(t; z_j(t_k))\| \leq -c_j^i, \quad \text{for each } j \in \mathcal{N}_i, \ \forall t \in [t_k, t_k + T].$$

The result is that, for each neighbor $j \in \mathcal{N}_i$,

$$\begin{aligned}\rho_{\min} + \kappa\delta^2 &\leq \|q_i(t; z_i(t_k)) - \hat{q}_j(t; z_j(t_k))\| \\ &\leq \|q_i(t; z_i(t_k)) - q_j(t; z_j(t_k))\| + \|\hat{q}_j(t; z_j(t_k)) - q_j(t; z_j(t_k))\| \\ &\leq \|q_i(t; z_i(t_k)) - q_j(t; z_j(t_k))\| + \kappa\delta^2,\end{aligned}$$

which implies that the actual trajectories respect the original collision avoidance constraint. Note that the result still holds for any $c^i_j \geq \kappa\delta^2$, provided the assumption on the vector c^i still holds. This concludes the example.

By assuming that the constraints can be made more conservative by incorporating the negative constant vector c^i, we are not able to handle equality constraints. However, such constraints can sometimes be eliminated by a change of variables, or, if the components c^i_j are small (as they would be in the example above since δ is required to be small), acceptable relaxations on equality constraints can be achieved.

In the simulation results of the next chapter, inter-agent coupling comes only in the integrated cost function, and good closed-loop performance is observed with and without enforcing the compatibility constraint. When coupling constraints are present, however, *compatibility constraints are required to ensure feasibility*, at initialization and all subsequent receding horizon updates. This is an important observation, since feasibility must surely be addressed prior to stability or performance. To elucidate this point, we look at another example.

Example 5.2 [1] Consider the following two agent, centralized optimal control prob-

[1]Thanks to Eric Klavins for inspiring this example.

lem

$$\begin{aligned}
\min_{u_1(\cdot),\ u_2(\cdot)} \int_0^T & \|q_1(\tau)\|^2 + \|q_2(\tau) - d\|^2 \, \mathrm{d}\tau \\
& \dot{q}_1(t) = u_1(t), \quad q_1(0) = d, \\
& \dot{q}_2(t) = u_2(t), \quad q_2(0) = 0, \\
& \|q_1(t) - q_2(t)\| \geq \rho_{\min} \\
& q_1(T) = 0, \quad q_2(T) = d,
\end{aligned}$$

were $q_i(t) \in \mathbb{R}^2$ for $i = 1, 2$, $d = (1, 0) \in \mathbb{R}^2$, $T = 3$ and $\rho_{\min} < 1$. The objective is for the agents to switch positions in time T while not colliding. The distributed optimal control problem for agent 1 is

$$\begin{aligned}
\min_{u_1(\cdot)} \int_0^T & \|q_1(\tau)\|^2 \, \mathrm{d}\tau \\
& \dot{q}_1(t) = u_1(t), \quad q_1(0) = d, \\
& \|q_1(t) - \hat{q}_2(t)\| \geq \rho_{\min} + c_1 \\
& q_1(T) = 0,
\end{aligned}$$

for positive constant c_1 and with the problem similarly defined for agent 2. We assume in the following that $\rho_{\min} + c_i < 1$ for both $i = 1$ and 2. An initially feasible control for the agents is given by

$$u_1(t; q_1(0)) = \begin{cases} (0, -1), & t \in [0, 1) \\ (-1, 0), & t \in [1, 2) \\ (0, 1), & t \in [2, T] \end{cases}, \quad u_2(t; q_2(0)) = \begin{cases} (0, 1), & t \in [0, 1) \\ (1, 0), & t \in [1, 2) \\ (0, -1), & t \in [2, T] \end{cases}.$$

By this control, the agents travel in straight lines, safely avoiding collision and with a minimum separation distance of 2, which occurs at time $T/2$. Now, with $t_{-1} = 0$, define $\hat{u}_i(t; q_i(t_{-1})) = u_i(t; q_i(0))$, for both $i = 1$ and 2. The optimization at initialization *without* the state compatibility constraint results in

$$u^*_{d1}(t; q_1(t_{-1})) = -1/T, \quad \text{and} \quad u^*_{d2}(t; q_2(t_{-1})) = 1/T,$$

for all $t \in [t_{-1}, t_{-1} + T]$. That is, each agent assumes the neighbor will apply the initially feasible control, making it safe (feasible) to apply the simpler control of going in a straight line to the desired location. The resulting actual trajectories, however, are infeasible! Consequently, after application of the first portion of the control, at the next update time $t_0 = \delta$, the *agents cannot use the new assumed control as a feasible control*, as it would result in collision constraint violation. If instead, each local optimal control problem incorporated the additional compatibility constraint

$$\|q_i(t; q_i(t_k)) - \hat{q}_i(t; q_i(t_k))\| \leq \delta^2 \kappa, \quad t \in [t_k, t_k + T], \ \forall k = -1, 0, 1, 2, ...,$$

where $\delta^2 \kappa \leq c_i$, the optimal solution of each problem can be used to generate the next assumed control per Definition 4.5, at every receding horizon update, with guaranteed feasibility. This concludes the example.

5.4.3 Locally Synchronous Timing

In this section, we explore means of removing the requirement that the local optimal control problems be solved globally synchronously. Instead, over any interval $[t_k, t_{k+1})$, the ith optimal control problem is assumed to be solved at time $\tau_{k,i} \in [t_k, t_{k+1})$. To facilitate the analysis, we assume that each agent solves with a fixed update interval δ as before, so $\tau_{k+1,i} = \tau_{k,i} + \delta$, for every $i = 1, ..., N_a$ and all $k \in \mathbb{N}$.

In the absence of uncertainty and ignoring the effects of computation and communication delays, a locally synchronous version is not too difficult to construct. First, the assumed control trajectories are redefined to include all of the optimal control plus the terminal controller, as follows.

Definition 5.4 (*Assumed Control*) For each agent $i = 1, ..., N_a$ and for any $k \in \mathbb{N}$, the assumed control associated with update time $\tau_{k,i}$ is denoted $\hat{u}_i(\cdot; z_i(\tau_{k,i}))$:

$[\tau_{k,i}, \tau_{k+1,i} + T] \rightarrow \mathcal{U}$, defined as follows:

$$\hat{u}_i(\tau; z_i(\tau_{k,i})) = \begin{cases} u^*_{di}(\tau; z_i(\tau_{k,i})), & \tau \in [\tau_{k,i}, \tau_{k,i} + T) \\ K_i \left(z_i^K(\tau - \tau_{k,i} - T; z_i^K(0)) - z_i^c \right), & \tau \in [\tau_{k,i} + T, \tau_{k+1,i} + T] \end{cases},$$

where $z_i^K(0) = z^*_{di}(\tau_{k,i} + T; z_i(\tau_{k,i}))$, and $z_i^K(s; z_i^K(0))$ is the closed-loop solution to

$$\dot{z}_i^K(s) = (A_i + B_i K_i)(z_i^K(s) - z_i^c), \quad s \geq 0, \quad \text{given } z_i^K(0).$$

The corresponding assumed state trajectory is denoted $\hat{z}_i(\tau; z_i(\tau_{k,i}))$, for all $\tau \in [\tau_{k,i}, \tau_{k+1,i} + T]$. The state compatibility constraint in the ith local optimal control problem is kept the same, which in terms of the new notation becomes

$$\|z_i(\tau; z_i(\tau_{k,i}) - \hat{z}_i(\tau + \delta; z_i(\tau_{k-1,i}))\| \leq \delta^2 \kappa, \quad \tau \in [\tau_{k,i}, \tau_{k,i} + T]. \tag{5.2}$$

The locally synchronous implementation logic is now defined.

Definition 5.5 (*Distributed Locally Synchronous Algorithm*) For each agent $i = 1, ..., N_a$, at every receding horizon update time $\tau_{k,i}$, $k \in \mathbb{N}$, the agent:

1. senses its own current state $z_i(\tau_{k,i})$ and senses or receives the current state $z_j(\tau_{k,i})$ of each neighbor $j \in \mathcal{N}_i$,

2. computes the assumed state trajectory $\hat{z}_j(\tau; z_j(\tau_{k,i}))$, $\tau \in [\tau_{k,i}, \tau_{k,i} + T]$, for each neighbor $j \in \mathcal{N}_i$, using the available assumed control as follows:

$$\begin{aligned} &\text{if } \tau_{k,j} \leq \tau_{k,i}, \quad \text{use } \hat{u}_j(\tau; z_j(\tau_{k,j})), \ \tau \in [\tau_{k,i}, \tau_{k,i} + T] \\ &\text{if } \tau_{k,j} > \tau_{k,i}, \quad \text{use } \hat{u}_j(\tau; z_j(\tau_{k-1,j})), \ \tau \in [\tau_{k,i}, \tau_{k,i} + T], \end{aligned}$$

3. computes the optimal control trajectory $u^*_{di}(\tau; z_i(\tau_{k,i}))$, $\tau \in [\tau_{k,i}, \tau_{k,i} + T]$,

4. computes the assumed control and state trajectory $\hat{u}_i(\tau; z_i(\tau_{k,i}))$ and $\hat{z}_i(\tau; z_i(\tau_{k,i}))$, $\tau \in [\tau_{k,i}, \tau_{k+1,i} + T]$, and

5. transmits $\hat{u}_i(\cdot; z_i(t_k))$ to every neighbor $j \in \mathcal{N}_i$.

Between receding horizon update times $[\tau_{k,i}, \tau_{k+1,i})$, for every $k \in \mathbb{N}$, each agent $i = 1, ..., N_a$:

1. implements the current optimal control $u^*_{di}(\tau; z_i(\tau_{k,i}))$, $\tau \in [\tau_{k,i}, \tau_{k+1,i})$, and
2. receives assumed controls from each neighbor $j \in \mathcal{N}_i$ at time $\tau_{k,j}$, if $\tau_{k,j} > \tau_{k,i}$, or at time $\tau_{k+1,i}$ otherwise.

The implicit assumptions that are required to implement the algorithm above are the following:

- computation and communication delays are negligible;
- each agent can receive current state information for every neighbor at the every associated update time.

The timing is locally synchronous in that each agent need only know *when* the latest assumed control trajectory is received from every neighbor, relative to the next update time. Given the assumed trajectory and the time is was received, the agents can generate the appropriate assumed state information for neighbors, according to step 2 in the algorithm procedure at each update time.

Now, the update windows of any two agents overlap each other. Consequently, agents have an assumed trajectory for each neighbor that *matches the actual trajectory* over the overlap time period. More precisely, take agent i updating at time $\tau_{k,i}$, with assumed control from neighbor $j \in \mathcal{N}_i$ received at time $\tau_{k,j} < \tau_{k,i}$. Over the interval $[\tau_{k,i}, \tau_{k+1,j}]$, agent i has the actual state trajectory for agent j. For the remainder of the time interval, i.e., $[\tau_{k+1,j}, \tau_{k,i} + T]$, agent i has the assumed state trajectory for neighbor j, which deviates at most by $\delta^2\kappa$ from the actual state trajectory, according to equation (5.2). Thus, the bounding argument in the stability analysis should follow along the same lines as before. The details of the analysis are left for future work.

5.5 Summary

In this chapter, the distributed receding horizon control implementation is qualitatively and quantitatively compared to a centralized implementation. The main tradeoff is that, while the distributed implementation provides substantial scalable savings in computation over the centralized implementation, the addition of the compatibility constraints results in a more sluggish closed-loop response. Alternative ways of formulating the distributed implementation, while still preserving closed-loop stabilization, are explored in this chapter as well.

Extensions of the theory for handling inter-agent coupling state constraints is also discussed. When the constraints can be made more conservative in the local optimal control problems, to account for discrepancy between assumed and actual trajectories, the state compatibility constraints can guarantee that initial feasibility implies subsequent feasibility of the actual closed-loop trajectories. The initialization phase of the distributed implementation, however, now requires a better guess for the assumed controls, namely, one that is known to be feasible for the coupling constraints. Still, this requirement is no stronger than that imposed on the centralized implementation; *if* the centralized problem has an initial feasible solution, subsequent feasibility can be ensured. If an initially feasible solution to the centralized problem is available, it could be used to define the assumed controls at initialization in the distributed case.

Finally, a distributed implementation with locally synchronous timing requirements is explored. Practically, this is the most likely situation in the general distributed problem. The algorithm stated for this case relied on negligible computation and communication delays. In the extreme, an asynchronous algorithm that accounted for computation and communication delays could be developed. There are receding horizon control formulations that have explored the effects of computational delay (see Section 2.3.1). However, the effect of communication delays over networks on control algorithms, optimization-based or not, poses many interesting challenges that are beginning to be explored only in recent years.

In the next chapter, formations of vehicles are stabilized using the centralized and distributed receding horizon control. The simulations reveal that for a fixed, small value for the update period δ, convergence is obtained with good accuracy for both implementations. Moreover, the closed-loop performance of the distributed implementation is comparable to that of the centralized implementation.

Chapter 6

Receding Horizon Control of Multi-Vehicle Formations

6.1 Introduction

Cooperative receding horizon control of multiple autonomous vehicles is considered in this chapter. The objective of the vehicles is to achieve a specific formation, one that associates a precise location for every vehicle. Here, cooperation has an offline phase and an online phase, for both centralized and distributed implementations of receding horizon control. For both implementations, vehicles are assigned roles in the formation offline. The roles are defined by coupling terms in the cost function of a centralized optimal control problem. For a distributed implementation of receding horizon control, decomposition into local optimal control problems, as well as the inclusion of compatibility constraints, is also done offline. The online phase requires dynamic coordination of vehicles, i.e., exchanging information to realize the control objective. For the centralized implementation, one vehicle must solve the problem at each update, requiring the collection of current state information of every vehicle and the dissemination of the control laws to every vehicle. Alternatively, every vehicle could solve a copy of the centralized problem, in which case every vehicle must attain the current state of all vehicles at each update. For the distributed implementation, coordination only happens between vehicles that are coupled in the local optimal control problems, i.e., vehicles that are neighbors. The specific coordination involves sharing assumed control trajectories, as prescribed by the distributed implementation algorithm in Chapter 4.

An outline of this chapter is as follows. The formation stabilization objective is first defined, which in turn yields the cost function and the centralized optimal control problem. Next, the distributed optimal control problems are defined for the distributed implementation. Numerical experiments are then provided to compare the centralized and distributed implementations. Additionally, simulations with a naive attempt to relax the communication requirements of the distributed implementation, via sharing less information, are examined. An alternative formulation to the multi-vehicle formation problem is discussed and the chapter is concluded with summarizing remarks.

6.2 Formation Stabilization Objective

In this section, we present the system dynamics and constraints and define the control objective. To facilitate analysis, we consider only linear dynamics. For ease of reference, we also restate the relevant definitions and assumptions necessary for the theoretical results on centralized and distributed receding horizon control given in Chapter 2 and Chapter 4, respectively.

We wish to stabilize a group of vehicles toward a common objective in a cooperative way using receding horizon control. Each vehicle is assumed to have dynamics, described by an ordinary differential equation, completely decoupled from all other vehicles. Specifically, for $i = 1, ..., N_a$ vehicles, the state and control of vehicle i are $z_i(t) = (q_i(t), \dot{q}_i(t)) \in \mathbb{R}^{2n}$ and $u_i(t) \in \mathbb{R}^n$, respectively, and the dynamics are given by

$$\dot{z}_i(t) = A_i z_i(t) + B_i u_i(t), \quad t \geq 0, \quad z_i(0) \text{ given}$$

$$\text{where} \quad A_i = \begin{bmatrix} 0 & I_{(n)} \\ 0 & 0 \end{bmatrix}, \quad B_i = \begin{bmatrix} 0 \\ I_{(n)} \end{bmatrix}.$$

The matrix $I_{(n)}$ is the identity matrix of dimension n. Each vehicle i is also subject to the input constraints $u_i(t) \in \mathcal{U}$. The set $\mathcal{U}^N$ is the N-times Cartesian product $\mathcal{U} \times \cdots \times \mathcal{U}$. Concatenating the states and inputs into vectors as $q = (q_1, ..., q_{N_a})$, $\dot{q} = (\dot{q}_1, ..., \dot{q}_{N_a})$, $z = (z_1, ..., z_{N_a})$ and $u = (u_1, ..., u_{N_a}) \in \mathcal{U}^{N_a}$, the dynamics are equivalently

$$\dot{z}(t) = Az(t) + Bu(t), \quad t \geq 0, \quad z(0) \text{ given}, \tag{6.1}$$

where $A = \text{diag}(A_1, ..., A_{N_a})$, $B = \text{diag}(B_1, ..., B_{N_a})$. Define the invertible map $U : \mathbb{R}^{2nN_a} \to \mathbb{R}^{2nN_a}$ as

$$\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = Uz.$$

Note that U is a unitary matrix, so $U^T U = I$.

Definition 6.1 The *control objective* is to cooperatively asymptotically stabilize all vehicles to $z^c = (z_1^c, ..., z_{N_a}^c)$, an equilibrium point of equation (6.1), with equilibrium control equal to zero.

The cooperation is achieved by the minimization of the cost function defined below. The control objective for each vehicle i is thus to stabilize to z_i^c while cooperating with neighboring vehicles. The position values at z^c are denoted $q^c = (q_1^c, ..., q_{N_a}^c)$, and the equilibrium velocity is clearly zero.

Assumption 6.1 The following holds:

(i) $\mathcal{U} \subset \mathbb{R}^n$ is compact, convex and contains the origin in its interior;

(ii) each vehicle i can measure the full state z_i, there is no uncertainty, and computational time is negligible compared to the evolution of the closed-loop dynamics.

In the absence of constraints, linear quadratic optimal control could be used to meet the cooperative control objective.

The multiple vehicle formation is here defined by a set of relative vectors that connect the desired locations of the vehicles. The desired formation can in turn be viewed as a graph, as in [19, 62]. For example, consider a desired formation of vehicles in Figure 6.1, where the position components of q_i are denoted $(x_i, y_i) \in \mathbb{R}^2$. The left figure shows the vector structure associated with the formation. The num-

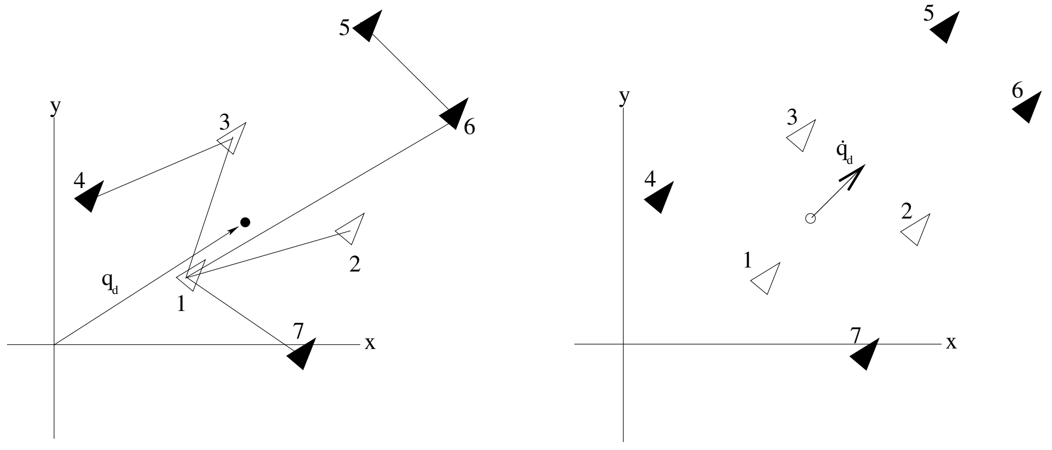


Figure 6.1: Seven-vehicle formation: vector structure on the left, and resulting formation on the right.

bers correspond to vehicle identity and a line segment between two numbers is a two dimensional relative vector. The dot in the center of the figure is the center of geometry of vehicles 1, 2 and 3. Given a desired location for this center of geometry, and the relative vectors between vehicles as shown, this formation designates a globally unique location for each of the vehicles. To generalize, a formation of N_a vehicles is uniquely defined given $N_a - 1$ relative vectors, such that each vehicle is at one end of at least one vector, and 1 vector (denoted q_d) designating a desired center of geometry location for a subset of the vehicles. The vehicles used to relate to q_d are called the *core vehicles*, consistent with the definition in [56]. The figure on the right shows the associated vehicle formation, where the core vehicles are denoted by white triangles, and all other vehicles are denoted by black triangles. The tracking objective is being achieved for some *formation path*, defined by $\mathbb{R} \ni t \mapsto (q_d(t), \dot{q}_d(t)) \in \mathbb{R}^4$.

We now generalize the description of the formation as a graph. The vector formation graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, ..., N_a\}$ is the set of vehicles and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of relative vectors between vehicles, where an edge in the graph is an ordered pair $(i, j) \in \mathcal{E}$ for every relative vector between vehicles $i,\ j \in \mathcal{V}$.

All graphs considered are undirected, so $(i,j) \in \mathcal{E} \Rightarrow (j,i) \in \mathcal{E}$. Two vehicles i and j are called neighbors if $(i,j) \in \mathcal{E}$. The set of neighbors of vehicle i is denoted $\mathcal{N}_i \subset \mathcal{V}$. In addition, although any two core vehicles may not have a relative vector between them in the formation, we consider all core vehicles to be neighbors of one another, as they are all coupled through the tracking objective.

Let $\mathcal{E}_0$ denote an orientation of the set $\mathcal{E}$, where $\mathcal{E}_0 \subset \mathcal{E}$ contains one and only one of the two permutations of (i,j), for all $(i,j) \in \mathcal{E}$. Also, without loss of generality, we take the core vehicles to be 1, 2 and 3, which is the case in Figure 6.1. To reiterate, although it may be that $(2,3) \notin \mathcal{E}$, as in the case of the example in Figure 6.1, vehicle $3 \in \mathcal{N}_2$ and $2 \in \mathcal{N}_3$ since 2 and 3 are core vehicles and thus coupled through the tracking objective.

Assumption 6.2 The undirected vector formation graph $\mathcal{G}$ is connected.

If the formation graph is not connected, there exists a vehicle whose desired location is not uniquely specified by the graph, in addition, the cost function that we define would additively separate into more that one coupled cost function. For a connected graph and resulting cost function defined below, the centralized receding horizon control law involves a single coupled optimal control problem that will be given in the next section. The connectivity assumption clearly holds for the example in Figure 6.1.

Remark 6.1 When the minimal number of relative vectors is used to define the formation, $|\mathcal{E}_0| = N_a - 1$. It may be that more vectors are added to the formation description, provided they are consistent with the existing vectors, as described below. Generally, we shall denote $|\mathcal{E}_0| = M$, $M \geq N_a - 1$. In graph theory, assuming the graph is connected implies $M \geq N_a - 1$ [5].

Let $e_1, ..., e_M$ denote an ordering of the elements of $\mathcal{E}_0$. Also, the tail of the edge e_i, denoted $t(e_i)$, is the first element in the corresponding ordered pair and the head of the vector $h(e_i)$ is the second element. In the case of Figure 6.1, let

$$\mathcal{E}_0 = \{e_1, e_2, e_3, e_4, e_5, e_6\} = \{(1,2), (1,3), (1,6), (1,7), (3,4), (5,6)\}.$$

For example, we have $t(e_3) = 1$ and $h(e_3) = 6$.

Definition 6.2 The *desired relative vector* between any two neighbors i and j is denoted $d_{ij} \in \mathbb{R}^n$, where it is understood that $q_i^c + d_{ij} = q_j^c$. All desired relative vectors are constant vectors, in length and orientation, and satisfy the following consistency conditions:

- For all $(i, j) \in \mathcal{E}$, $d_{ij} = -d_{ji}$.
- When $M > N_a - 1$, if (i, j), (j, l) and (i, l) are in $\mathcal{E}$, then $d_{ij} + d_{jl} = d_{il}$.

Definition 6.3 Given an admissible oriented formation graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_0)$ and a formation path, the *formation vector* $F = (f_1, ..., f_{M+1}) \in \mathbb{R}^{n(M+1)}$ has components $f_l \in \mathbb{R}^n$ defined as

$$f_l = q_i - q_j + d_{ij}, \quad \text{where } i = t(e_l),\ j = h(e_l),\ \forall l = 1, ..., M,$$

$$f_{M+1} = q_\Sigma - q_d, \quad q_\Sigma = \frac{1}{3}(q_1 + q_2 + q_3).$$

For a stabilization objective, $\dot{q}_d(t) = 0,\ \forall t \in \mathbb{R}$, and to be compatible with the control objective, $q_d = (q_1^c + q_2^c + q_3^c)/3$. Clearly, the vehicles are in formation when $F \equiv 0$. Write the linear mapping from q to F as:

$$F = Gq + \hat{d}, \quad G^T = \left[C_{(n)}\ V\right]. \tag{6.2}$$

The vector $\hat{d} = (..., d_{ij}, ..., -q_d)$ has the ordering of the vectors d_{ij} consistent with the definition of F. The matrix $V^T = [V_1\ \dots\ V_{N_a}] \in \mathbb{R}^{n \times nN_a}$ has elements $V_i \in \mathbb{R}^{n \times n}$ defined as

$$V_i = \begin{cases} \frac{1}{3} I_{(n)}, & \text{if i} = 1, 2, 3 \\ 0, & \text{otherwise,} \end{cases}.$$

The matrix $C_{(n)} \in \mathbb{R}^{n(N_a \times M)}$ is related to the *incidence matrix* $C \in \mathbb{R}^{N_a \times M}$, where the elements of $C = [c_{ij}]$ are defined in terms of the elements of the oriented edge set

$\mathcal{E}_0$ as

$$c_{ij} = \begin{cases} +1, & \text{vertex } i = t(e_j) \\ -1, & \text{vertex } i = h(e_j) \\ 0, & \text{otherwise} \end{cases} .$$

The matrix $C_{(n)}$ is defined by replacing each element of C with that element multiplied by $I_{(n)}$. The incidence matrix for the example in Figure 6.1 is

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} .$$

In defining the cost function for the optimal control problem, the following proposition is useful.

Proposition 6.1 *The matrix G in equation (6.2) has full column rank, equal to* $\dim(q) = nN_a$.

Proof: Since the vector formation graph is connected, the incidence matrix C has rank $(N_a - 1)$ [5]. Scaling the entries of C with the identity matrix $I_{(n)}$ implies the rank of $C_{(n)}$ is equal to $n(N_a - 1)$. The matrix V in equation (6.2) has full column rank equal to n, and each column vector is linearly independent from the column vectors in $C_{(n)}$. Thus, G has rank nN_a, which is the column dimension of the matrix. ■

From the definition of the formation vector, we know that $Gq^c = -\hat{d}$ and so

$$\|F\|^2 = (G(q - q^c))^T G(q - q^c) = \|q - q^c\|^2_{G^T G}.$$

In the following we penalize $\|F\|^2$ and $\|\dot{q}\|^2$ in the centralized cost function. To define the objective in terms of the state z, we have that

$$\begin{bmatrix} G(q-q^c) \\ \dot{q} \end{bmatrix} = \widehat{G}U(z-z^c), \quad \text{where} \quad \widehat{G} = \begin{bmatrix} G & 0 \\ 0 & I_{(nN_a)} \end{bmatrix}.$$

As a result, we have

$$\|F\|^2 + \|\dot{q}\|^2 = \|z-z^c\|^2_{\widehat{G}^T\widehat{G}}.$$

In the next section, the optimal control problem associated with the multiple vehicle formation stabilization objective is defined for centralized and distributed receding horizon control.

6.3 Optimal Control Problems

In this section, we define the receding horizon control law that achieves the cooperative control objective, using both centralized and distributed implementations.

6.3.1 Centralized Receding Horizon Control

The centralized integrated cost function of interest is

$$L(z,u) = \sum_{(i,j)\in\ \mathcal{E}_0} \omega\|q_i - q_j + d_{ij}\|^2 + \omega\|q_{\Sigma} - q_d\|^2 + \nu\|\dot{q}\|^2 + \mu\|u\|^2,$$

with positive weighting constants ω, ν and μ. We refer to the term $\omega\|q_{\Sigma} - q_d\|^2$ as the *tracking cost*, although we are concerned with stabilization.

Remark 6.2 For collision avoidance, an appropriate cost function between any two vehicles is defined in [56]. Alternatively, to guarantee avoidance, collision avoidance can be cast as a constraint, as in [63]. We do not incorporate any type of collision avoidance in this chapter, although coupling constraints between neighboring vehicles could be accommodated as discussed in Section 5.4.

From the previous section, we also have that

$$L(z,u) = \|z - z^c\|_Q^2 + \mu\|u\|^2, \quad Q = \begin{bmatrix} \omega G^T G & 0 \\ 0 & \nu I_{(nN_a)} \end{bmatrix}. \tag{6.3}$$

From Proposition 6.1, Q is positive definite and clearly symmetric. The centralized optimal control problem, receding horizon control law, and sufficient conditions for stabilization are stated in Chapter 2.

6.3.2 Distributed Receding Horizon Control

In this section, the distributed receding horizon control law is stated for the multi-vehicle formation objective. In the centralized integrated cost, the non-separable terms $\|q_i - q_j + d_{ij}\|^2$, for all $(i,j) \in \mathcal{E}_0$, as well as the tracking term $\|q_\Sigma - q_d\|^2$, couple the states of neighboring vehicles. Recall that the set of neighbors of each vehicle i is denoted $\mathcal{N}_i$, also referenced by the subscript $-i$, where z_{-i} denote the vector of states of the neighbors of i. Also, let $q_{-i} = (q_{j_1}, ..., q_{j_{|\mathcal{N}_i|}})$ and $u_{-i} = (u_{j_1}, ..., u_{j_{|\mathcal{N}_i|}})$, where the ordering is consistent with z_{-i}.

Definition 6.4 The *distributed integrated cost* in the optimal control problem for any vehicle $i = 1, ..., N_a$ is defined as

$$\begin{aligned} L_i(z_i, z_{-i}, u_i) &= L_i^z(z_i, z_{-i}) + \gamma\mu\|u_i\|^2, \\ \text{where} \quad L_i^z(z_i, z_{-i}) &= \gamma\left[\sum_{j\in\mathcal{N}_i}\left\{\frac{\omega}{2}\|q_i - q_j + d_{ij}\|^2\right\} + \nu\|\dot{q}_i\|^2 + L_d(i)\right], \\ \gamma > 1, \quad L^d(i) &= \begin{cases} \frac{\omega}{3}\|q_\Sigma - q_d\|^2, & i = 1,2,3 \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

Thus, $\sum_{i=1}^{N_a} L_i(z_i, z_{-i}, u_i) = \gamma L(z,u) = \gamma\left[\|z - z^c\|_Q^2 + \mu\|u\|^2\right]$.

From the definition, the distributed integrated cost for any vehicle i includes one-half of each relative vector penalty coupling i with each neighbor $j \in \mathcal{N}_i$, the velocity and control penalties for i, and, if i is one of the three core vehicles (1, 2 or 3), one-third of the tracking penalty. In addition, all terms are multiplied by a common factor γ, a constant greater than one. In the proof of stability, the key structure is that the sum of the distributed integrated costs equals the centralized cost multiplied by γ.

Remark 6.3 The stability results of Chapter 4 do not depend on equal weighting of terms between neighboring vehicles. What is required is that the distributed integrated costs sum up to be the centralized cost, multiplied by a factor (γ) greater than one. The weighting will of course affect the performance of the closed-loop system, so making the weights lop-sided would result in one vehicle reacting more to the term than the corresponding neighbor. Note that in the limit that one vehicle takes the entire term, while the other ignores the term, we have a leader-follower effect.

Given the distributed integrated cost as defined above, the distributed optimal control problems and the distributed implementation algorithm are as stated in Chapter 4. The initialization procedure used in the simulations is given by Definition 4.4, where initially each vehicle assumes that neighbors apply zero control and the compatibility constraint is removed by setting κ to a large number.

6.4 Numerical Experiments

A simulation of a four-vehicle formation is presented in this section. While the scale of the problem is small, the simulation serves primarily to compare the centralized and distributed implementations. The use of the control compatibility constraint, in place of the state compatibility constraint (see Section 5.3.3 for description and discussion) is also explored. Additionally, in an attempt to alleviate the communication demands in the distributed implementation of transmitting assumed control trajectories, we explore the performance when each vehicle assumes that neighbors apply zero control at *every* update, in which case neighbors need only share initial conditions.

The dimension of the position vector for all vehicles is two ($n = 2$). The control constraint set is defined as

$$\mathcal{U} = \left\{ (u_1, u_2) \in \mathbb{R}^2 \ : \ -1 \le u_j \le 1, \ j = 1, 2 \right\}.$$

The objective is a "fingertip formation" that tracks the reference trajectory $(q_{\mathrm{ref}}(t), \dot{q}_{\mathrm{ref}}(t)) \in \mathbb{R}^4$, defined as

$$q_{\mathrm{ref}}(t) = \begin{cases} (t, 0), & t \in [0, 10) \\ (10, 10 - t), & t \in [10, \infty) \end{cases}, \tag{6.4}$$

where $t_0 = 0$ in the notation of the previous sections. The acceleration $\ddot{q}_{\mathrm{ref}}(t)$ is zero for all time except at $t = 10$ seconds. The expression "finger-tip formation" describes the relative location of the four vehicles. Specifically, one vehicle is in front, with two others behind and forming an isosceles triangle with the vehicle in front. The fourth and final vehicle is behind one of the two trailing vehicles, forming a line with both the front vehicle and the vehicle trailing the front vehicle. The reference to "finger-tip" is made because the formation resembles the relative location of finger tips of a hand that is flat with fingers held together. The front vehicle resembles the tip of the middle finger, and so on.

To be consistent with the cooperative *stabilization* objective, we rewrite the system dynamics in error form. In particular, the error system for any vehicle i has state $(q_{e,i}, \dot{q}_{e,i}) \triangleq (q_i - q_{\mathrm{ref}}, \dot{q}_i - \dot{q}_{\mathrm{ref}})$ and dynamics $\ddot{q}_{e,i} = u_i$. The jump in the reference velocity at time $t = 10$ serves to examine how well the error dynamics are stabilized for two different legs of the reference trajectory. Equivalently, the problem is to stabilize the error dynamics from an initial condition at time 0, and then again from the current state at time 10.

To eliminate any offset between the center of geometry of the formation and the reference trajectory, we set the formation path to $q_d(t) = (0, 0)$ for all $t \geq 0$. The vector formation graph is defined by vertices $\mathcal{V} = \{1, 2, 3, 4\}$ and relative vectors $\mathcal{E} = \{(1, 2), (1, 3), (2, 4)\}$. As in the generalization of Section 6.2, the core vehicles associated with the tracking cost are $\{1, 2, 3\}$. The relative vectors are defined for the two legs of the reference trajectory as

$$d_{12} = d_{24} = \begin{cases} (-2, 1), & t \in [0, 10) \\ (1, 2), & t \in [10, \infty) \end{cases}, \quad d_{13} = \begin{cases} (-2, -1), & t \in [0, 10) \\ (-1, 2), & t \in [10, \infty) \end{cases}.$$

The common rotation in the vectors at time $t = 10$ is match the heading of the fingertip formation with the heading of the reference trajectory. The initial conditions for each vehicle are given as $q_1(0) = (-1, 2)$, $q_2(0) = (-4, 0)$, $q_3(0) = (-2, 0)$ and $q_4(0) = (-7, -1)$, with $\dot{q}_i(0) = (0, 0)$ for each vehicle $i \in \mathcal{V}$. In both centralized and distributed receding horizon implementations, a horizon time of $T = 5.0$ seconds is used. Unless stated otherwise, the update period is $\delta = 0.5$ seconds. Also, the following weighting parameter values are consistent in both implementations: $\omega = 2.0$, $\nu = 1.0$ and $\mu = 2.0$. As stated, collision avoidance is not incorporated in the optimal control problems, either by cost or constraint. In all simulation results presented, no collisions were observed to occur.

To solve the optimal control problems numerically, we employ the Nonlinear Trajectory Generation (NTG) software developed at Caltech. A detailed description of NTG as a real-time trajectory generation package for constrained mechanical systems is given in [52]. The package is based on finding trajectory curves in a lower dimensional space and parameterizing these curves by B-splines. Sequential quadratic programming (SQP) is used to solve for the B-spline coefficients that optimize the performance objective, while respecting dynamics and constraints. The package NPSOL [25] is used to solve the SQP problem. In the all simulations that follow, the two position variables for each vehicle are parameterized as output curves in NTG. In particular, every optimal control problem is discretized by 21 breakpoints over two intervals (3 knot points), and the curves are represented as 5th order piecewise polynomials that are twice continuously differentiable at the knot points. Note that the update time in receding horizon implementations should occur at a breakpoint to guarantee that the constraints are satisfied at the new initial condition of each update. This is the case here, as 21 breakpoints over 5 seconds occur every 0.25 seconds, with the update occurring 0.5 seconds into each 5 second trajectory.

6.4.1 Centralized Implementation

For the centralized receding horizon control law, parameter values in the optimal control problem must be chosen to guarantee that Assumption 2.2 is true. For the weights chosen above, K is defined as the linear quadratic regulator and P the corresponding stable solution to the algebraic Riccati equation. Choosing $\alpha = 0.4$ implies that the assumption (i) is true. To prove this, define $y = z - z^c$ and observe that

$$y^T P y \leq \alpha \quad \Leftrightarrow \quad \lambda_{\max}(P) y^T P y \leq \lambda_{\max}(P)\alpha \quad \Rightarrow \quad y^T P^2 y \leq \lambda_{\max}(P)\alpha$$

$$\Rightarrow \quad y^T P B B^T P y \leq \lambda_{\max}(P)\alpha \quad \Leftrightarrow \quad y^T K^T K y \leq \frac{\lambda_{\max}(P)\alpha}{\mu^2}.$$

Choosing $\alpha \leq \mu^2/\lambda_{\max}(P) \approx 0.4$ guarantees that $\|Ky\|^2 \leq 1$. Finally, the latter condition guarantees that $K(z - z^c) \in \mathcal{U}^{N_a}$ for all $z \in \Omega(\alpha)$, since each component of Ky will be between -1 and 1 for all time. The centralized receding horizon control of the fingertip formation is shown in Figure 6.2. The four closed-loop position

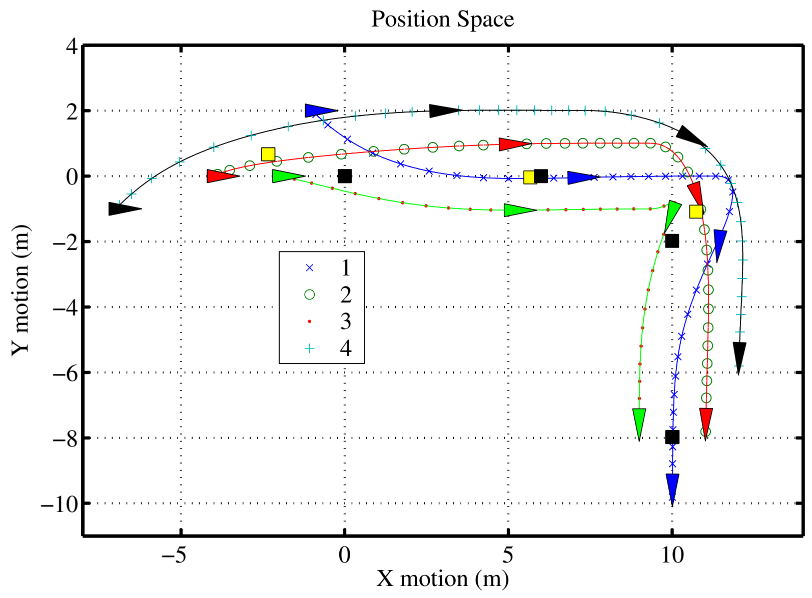


Figure 6.2: Fingertip formation response in position space using centralized receding horizon control.

trajectories of the vehicles are shown in the figure, with each vehicle depicted by a triangle. The heading of any triangle shows the direction of the corresponding velocity vector. The symbols along each trajectory mark the points at which the receding horizon updates occur. The legend identifies a symbol with a vehicle number for each trajectory. The triangles show the position and heading of each vehicle at snapshots of time, specifically at 0.0, 6.0, 12.0 and 18.0 seconds.

Also shown at these instants of time are the reference trajectory position $q_{\mathrm{ref}}(t)$, identified by the black square, and the average position of the core vehicles $q_{\Sigma}(t)$, identified by the yellow square. The tracking part of the cooperative objective is achieved when $q_{\Sigma}(t) = q_{\mathrm{ref}}(t)$, i.e., when the two squares are perfectly overlapping. At time 6.0, the vehicles are close to the desired formation, and the squares are nearly overlapped, indicating that the tracking objective is being reached. After 8.0 seconds, the formation objective has been met to a numerical precision of 0.01, which is the value of the the optimal cost function at that time. At time 12.0, the snapshot shows the formation reconfiguring to the change in heading of the reference trajectory which occurred at time 10.0. At time 18.0, the objective has again been met and the optimal cost function has a numerical value of less that 0.01.

The receding horizon control law time history for vehicle 3 is shown in Figure 6.3. At receding horizon updates, the control is not required to initially match the last control value applied. Consequently, the resulting closed-loop control will be discontinuous in general. The figure shows greater discontinuity during the transient phase of the closed-loop response, with the largest discontinuity occurring at time 0.0 and at time 10.0, when the reference trajectory changed heading.

6.4.2 Distributed Implementation

For the distributed receding horizon implementation, the initial state at time 0.0 is used for initialization, as described in Definition 4.4. In terms of the notation, we thus have $t_{-1} = 0.0$. Regarding the conditions in Assumption 4.3, we first choose $Q_i = \lambda_{\max}(Q) I_{(4)}$, where $\lambda_{\max}(Q) \approx 6.85$. As in the centralized case, K_i is defined as

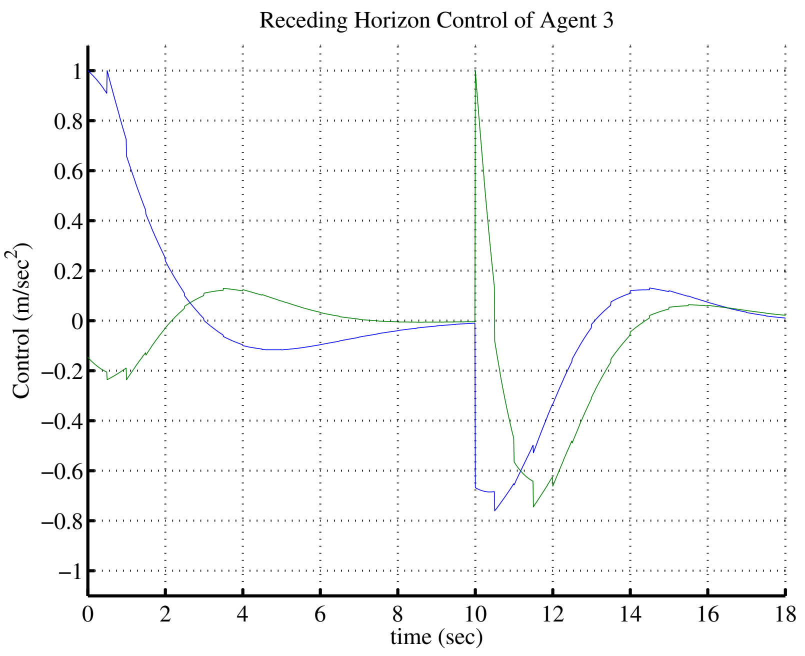


Figure 6.3: Centralized receding horizon control law time history for vehicle 3.

the linear quadratic regulator and P_i the corresponding stable solution to the algebraic Riccati equation. Following the steps above, we can show that $\alpha_i = 0.33$ guarantees that the conditions in the Assumption 4.3 will hold. Finally, we set $\gamma = 2$ in the cost functions of the distributed optimal control problems. The distributed receding horizon controller is applied for all time and switching to the decoupled feedbacks is not employed.

Before employing the distributed receding horizon control law defined in Section 6.3.2, we investigate the performance of the closed-loop system with $\kappa = +\infty$. That is, we have exactly the distributed implementation *except* that the compatibility constraint is not enforced. The closed-loop fingertip formation response is shown in Figure 6.4 and the receding horizon control law time history for vehicle 3 is shown in Figure 6.5. As before, the triangles in Figure 6.4 show the position and heading of each vehicle at the time snapshots of 0.0, 6.0, 12.0 and 18.0 seconds. The performance is very close to that of the centralized implementation. At snapshot time 6.0, the formation is slightly lagging the reference, compared to the centralized version. At time 18.0, the formation objective is close to being met, and for slightly more time

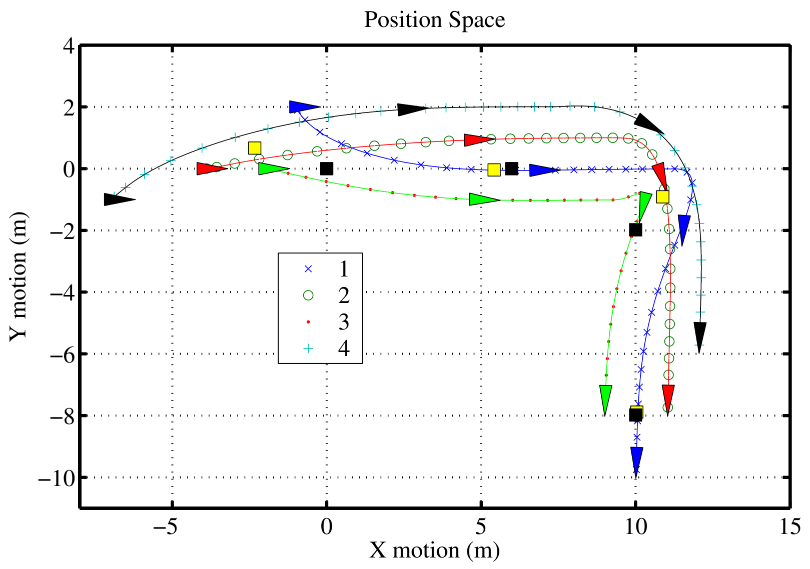


Figure 6.4: Fingertip formation response using distributed receding horizon control without compatibility constraints ($\kappa = +\infty$).

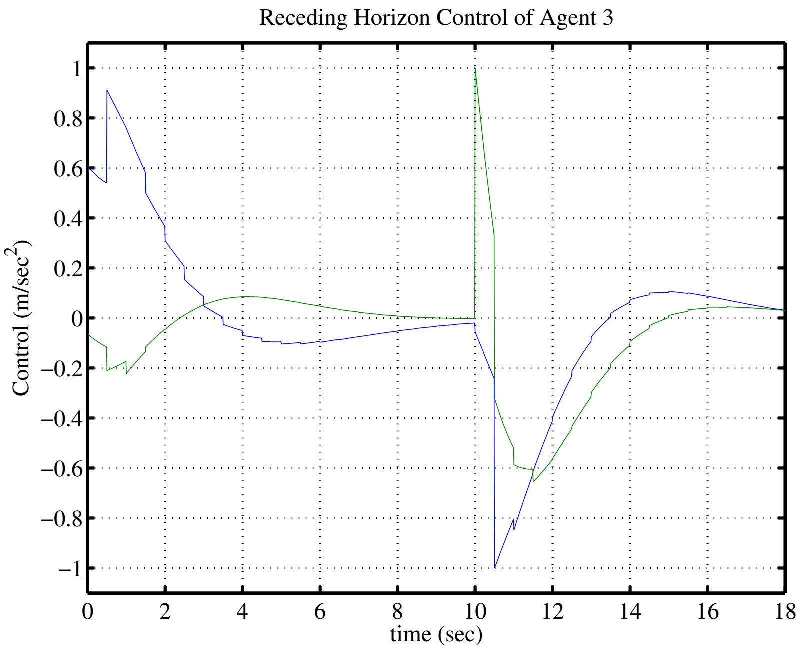


Figure 6.5: Distributed receding horizon control law time history for vehicle 3, without compatibility constraints ($\kappa = +\infty$).

the same precision as the centralized implementation is achieved. In comparing the control time histories for vehicle 3, the centralized controller response is for the most part smoother than the distributed controller response.

Based on the results above, it seems that when the dependence on the assumed information about neighbors comes in the cost function, there is good performance without enforcing compatibility between assumed and actual trajectories. On the other hand, when there are coupling constraints, the need for compatibility cannot be ignored, even in trivial cases (see Section 5.4 on inter-agent coupling state constraints).

Now, we consider the performance of the distributed receding horizon control law defined in Section 6.3.2. After initialization, $\kappa = 2$ and the compatibility constraint becomes

$$\|q_i(s; z_i(t_k)) - \hat{q}_i(s; z_i(t_k))\| \leq \delta^2 \kappa = 0.5, \quad s \in [t_k, t_k + T], \quad i = 1, ..., N_a.$$

Although the norm in each compatibility constraint is defined to be the Euclidean 2-norm, we implement the ∞-norm, since it is a linear constraint and therefore easier for the optimization algorithm. The closed-loop fingertip formation response is shown in Figure 6.6 and the receding horizon control law time history for vehicle 3 is shown in Figure 6.7. From Figure 6.6, the performance is nearly identical to the distributed

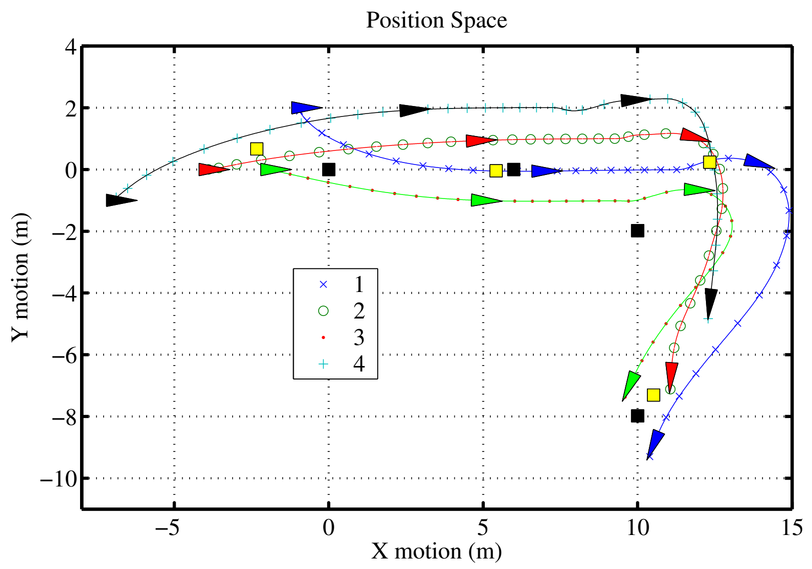


Figure 6.6: Fingertip formation response using distributed receding horizon control and position compatibility constraints.

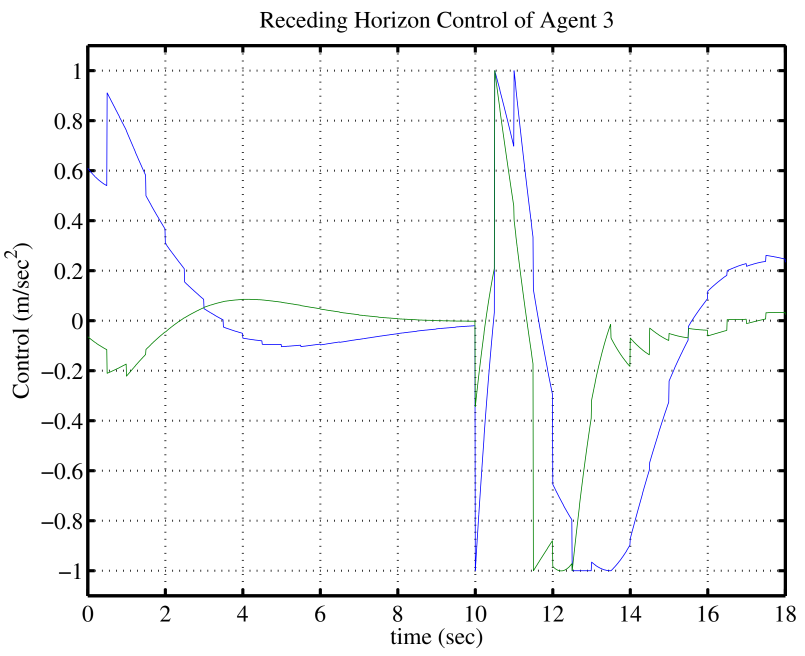


Figure 6.7: Distributed receding horizon control law time history for vehicle 3, using position compatibility constraints.

implementation with $\kappa = +\infty$, until time 10.0 seconds. At time 10.0 seconds, when the reference changes heading, serious difficulties occur. The reason is simple: the distributed optimization problems are no longer feasible at that time, and for several subsequent update times. The infeasibility arises since the terminal constraints are redefined, in the error dynamics, relative to the new location and heading of the reference, while the compatibility constraints will not allow the positions to deviate enough to reach the new terminal constraint sets.

As obvious fix to this problem, which arises only because of the change in the reference heading at time 10.0 seconds, is to set $\kappa = +\infty$ (or a large number, in practice) at time 10.0 seconds and then $\kappa = 2.0$ again for all later times. The resulting response looks *exactly* like the results with $\kappa = +\infty$ for all update times, i.e., the formation response in Figure 6.4 and the control law history for vehicle 3 in Figure 6.5.

Now, since the responses are identical by reinitializing κ in this way, it can be inferred that the position compatibility constraint is never active with $\kappa = 2.0$. Smaller values of κ could be chosen to explore the effect of an active position compatibility constraint on the transient responses, i.e., the response after initialization and after the change in heading of the reference. Instead of exploring smaller values of κ with position compatibility constraints, we explore replacing these constraints in every local optimal control problem with the control compatibility constraint

$$\|u_i(s; z_i(t_k)) - \hat{u}_i(s; z_i(t_k))\| \leq \delta^2 \kappa = 0.5, \quad s \in [t_k, t_k + T], \quad i = 1, ..., N_a.$$

From Section 5.3.3, the resulting deviation between each optimized and assumed state trajectory is given by

$$\|z_i(s; z_i(t_k)) - \hat{z}_i(s; z_i(t_k))\| \leq \delta^2 \kappa \left\{\exp[\mathcal{K}(s - t_k)] - 1\right\} \; = \; 0.5 \left\{\exp[s - t_k] - 1\right\}, \tag{6.5}$$

where $\mathcal{K} = 1$ is the Lipschitz bound for each subsystem in equation (6.1). The bound is initially zero and grows exponentially with time, so it is likely that the control compatibility constraint becomes active during the first part of each update interval, particularly during the initial transient phases (just after time 0.0 seconds and time 10.0 seconds) of the closed-loop response. An interpretation of equation (6.5) is that controllability is initially zero and grows exponentially with time. The closed-loop fingertip formation response is shown in Figure 6.8 and the receding horizon control law time history for vehicle 3 is shown in Figure 6.9. The closed-loop trajectories in Figure 6.8 look much like the other distributed implementations, with a slight overshoot by vehicles 1 and 3 observable at the snapshot time of 12.0 seconds. The overshoot can be attributed to the loss of controllability in the initial portion of optimization window. Specifically, each agent must initially stick close to the previous plan, which during this phase relied on the reference trajectory proceeding with the original heading. The closed-loop control time history of vehicle 3 in Figure 6.9 shows the compatibility constraints becoming active for a few update periods after the change in reference heading at time 10.0 seconds.

Here, $\kappa = 2.0$ for all updates after initialization, and the feasibility problem at time

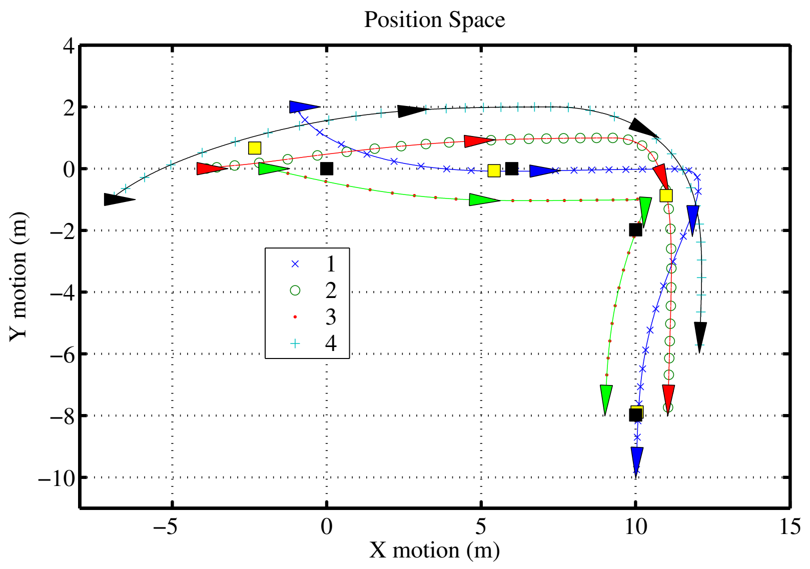


Figure 6.8: Fingertip formation response using distributed receding horizon control and control compatibility constraints.

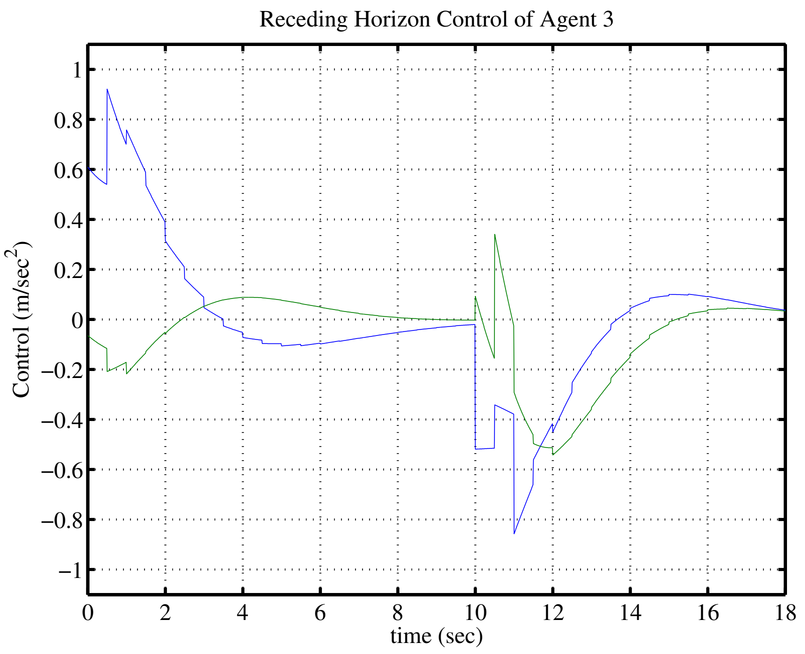


Figure 6.9: Distributed receding horizon control law time history for vehicle 3, using control compatibility constraints.

10.0 seconds is now no longer observed numerically. The reason is that, from equation (6.5), the optimized state is allowed to diverge more from the assumed state when using the control compatibility constraint as time proceeds within each optimization window. Since the original conflict was with the terminal constraints, enforced at the end time in each window, we now observe no feasibility problem for the particular parameter values here (T, δ, κ). It is possible that for a different set of parameters, infeasibility could of course still occur even with the control compatibility constraint.

As a byproduct of the control compatibility constraint, it is also observed that the allowable size of discontinuity in the closed-loop control at receding horizon update times is reduced. In particular, for the other centralized and distributed implementations, the control can jump by as much as 2, i.e., from -1 to 1 or vice versa. However, in the distributed implementation with the control compatibility constraint, since we used the ∞-norm, each component of the control can jump by 0.5 at most. The control in Figure 6.9 shows a few places where such maximal discontinuities occur, specifically just after time 10.0 seconds.

To alleviate the communication demands in the distributed implementations of transmitting assumed control trajectories, we now explore the performance when each vehicle assumes that neighbors apply zero control at *every* update, in which case neighbors need only share initial conditions. No compatibility constraint is enforced in this case ($\kappa = +\infty$). This was explored in simulations in a previous paper [16]. In other words, neighbors are assumed to continue along straight line paths over any optimization horizon. The resulting closed-loop fingertip formation response is shown in Figure 6.10. The receding horizon control law time history for vehicle 3 is shown in Figure 6.11. The response is characterized by overshoot. Although the core vehicles have perfect knowledge about the reference position and velocity trajectory, the overshoot is caused by the vehicles believing that neighbors will continue along vectors tangent the path over the entire optimization horizon at every update. The figure shows the position and heading triangles at the same snapshots of time as before, plus one additional snapshot at time 24.0 seconds. As time grows, the formation is observed to get closer to meeting the formation objective, but only after a long time. Moreover, the formation never meets the objective to acceptable precision.

Interestingly, for this given weights in the cost function, the closed-loop perfor-

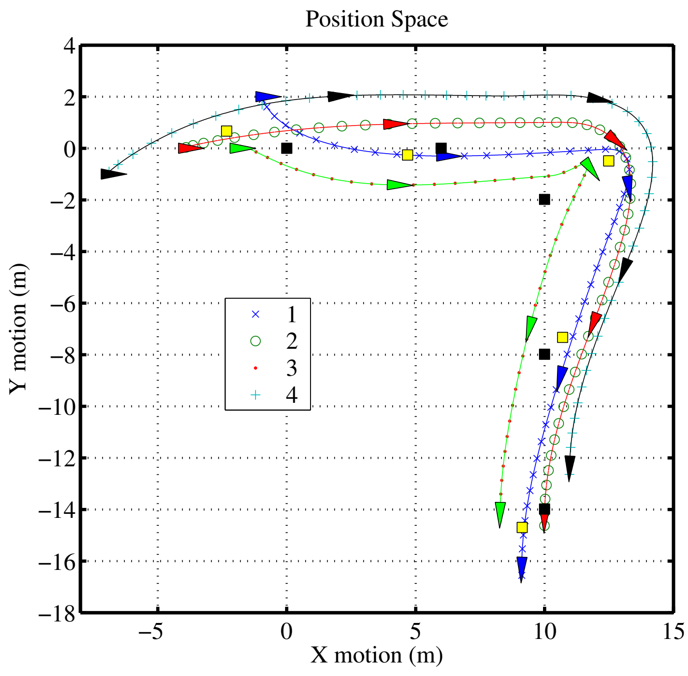
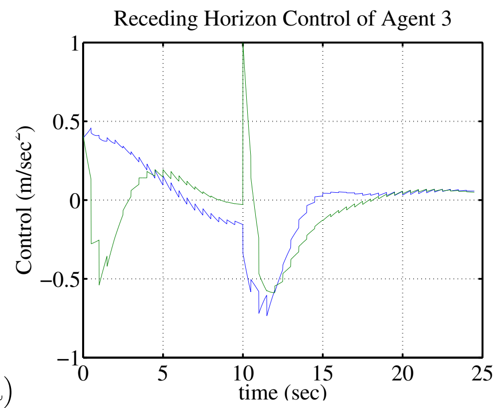


Figure 6.10: Fingertip formation response using distributed receding horizon control, assuming neighbors continue along straight line paths at each update and without enforcing any compatibility constraints ($\kappa = +\infty$).

mance is observed to stay the same even if the update period is decreased. In particular, if δ is reduced to 0.25 or 0.1, the response is nearly the same. For example, the receding horizon control law time history for vehicle 3 with $\delta = 0.1$ is also shown in Figure 6.11. As in the previous paper [16], other parameters were observed to improve the performance in this case. Specifically, increasing the damping weighting ν reduces the overshoot effect, although the response of the formation becomes more sluggish. If the *horizon time T is shortened*, overall *performance improves*, as the assumption becomes more valid. The reason is that a straight line approximation (first-order hold) is generally a valid approximation locally, and shrinking T means the assumption should hold over a more local domain, relative to larger values of T. However, theoretical results indicate that smaller T results in a smaller region of attraction for the closed-loop system.

In the formulation in [39], where each vehicle optimizes for itself as well as for neighboring vehicles, a similar effect is observed. There, vehicles assume that neigh-

(a) 

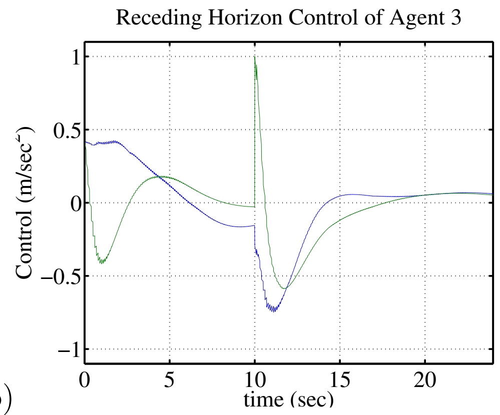
(b) 

Figure 6.11: Distributed receding horizon control law time history of vehicle 3 for update periods: (a) $\delta = 0.5$, (b) $\delta = 0.1$.

bors will react solely with regard to the local cost function and constraints. Apparently, such a self-interested philosophy is not too bad if vehicles are not looking too far into the future, since initial conditions are consistent in all distributed optimization problems at each update. The sensitivity to horizon time, as observed above when neighbors are assumed to continue along straight-line paths, and as observed in the formulation in [39], is not present in the distributed implementations that rely on assumed information and that incorporate compatibility constraints between assumed and actual information.

To compare controllers, the tracking cost with the actual closed-loop trajectories, i.e.,

$$\|(q_1^*(s) + q_3^*(s) + q_3^*(s))/3 - q_{\text{ref}}(s)\|, \quad s \geq 0,$$

is plotted in Figure 6.12 for the centralized and distributed implementations. The distributed implementation from the theory (DRHC 1) is shown to have a slightly longer settling time compared to the centralized implementation. Also, the overshoot in assuming neighbors continue along straight line paths (DRHC 2) is apparent, particularly between 8 and 10 seconds, before the reference changes heading.

Regarding the communication requirements of transmitting assumed controls to neighboring vehicles, in the NTG formulation corresponding to the simulations above, 14 B-spline coefficients specified the two-dimensional assumed control trajectories of

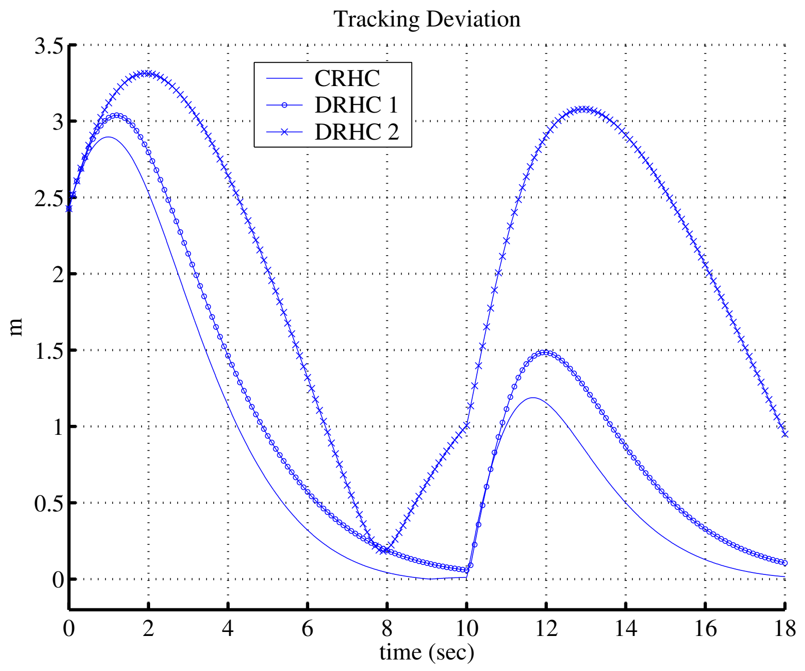


Figure 6.12: Comparison of tracking performance of centralized (CRHC) and two distributed (DRHC) implementations of receding horizon control. DRHC 1 denotes the distributed implementation corresponding to the theory, with control compatibility constraints, and DRHC 2 denotes the implementation with no compatibility constraints and neighbors are assumed to apply zero control.

each vehicle. In comparison, when vehicles assume neighbors continue along straight lines, 4 numbers much be communicated at each update, representing the initial condition of the state at the update time. Thus, such representations of trajectories in the optimization problem can aid in keeping the communication requirements closer to that of other decentralized schemes [14].

6.5 Alternative Description of Formations

We briefly describe an alternative formulation of multi-vehicle formation stabilization, where instead of basing the formation description on relative vectors, it is based on deviations from desired distances. In [57], the notion of formation graphs and their importance in unique representations of multi-vehicle formations is introduced. These graphs are used to obtain bounded and distributed control laws for formation stabilization of vehicles with linear, double-integrator dynamics. In [58], the notion of formations of multiple agents/vehicles and minimal requirements, in terms of the number of edges, for uniquely specifying a formation is formalized. This is done based on the tools from combinatorial graph rigidity and the specification of foldability [57] is added to the definition of a formation graph. These concepts were used to define a cost function for centralized receding horizon control in [56]. The cost function is restated here to give a flavor for this type of formation description.

As before, an undirected graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices, each denoted $v_i \in \mathcal{V}$, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of edges of the graph. Each edge is denoted by $e_{ij} = (v_i, v_j) \in \mathcal{E}$ or $ij \in \mathcal{E}$ for simplicity of notation where $i, j \in \mathcal{I} = \{1, \ldots, n\}$. An orientation of the edges of the graph, $\mathcal{E}_o \subset \mathcal{E}$, is the set of edges of the graph which contains one and only one of the two permutations of $ij \in \mathcal{E}$ (ij or ji) for all the edges $ij \in \mathcal{E}$.

A *triangulated graph* is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ with the set of faces $\mathcal{F} \subset \mathcal{V} \times \mathcal{V} \times V$ with elements $f_{ijk} = (v_i, v_j, v_k)$ or simply ijk $(i, j, k \in \mathcal{I})$ satisfying the following consistency condition:

$$f_{ijk} = (v_i, v_j, v_k) \in \mathcal{F} \rightarrow (v_i, v_j) \in \mathcal{E}, (v_j, v_k) \in \mathcal{E}, (v_k, v_i) \in \mathcal{E}, \forall f_{ijk} \in \mathcal{F}.$$

Similarly, an orientation of the faces of a triangulated graph $\mathcal{G}$ is a set of faces $\mathcal{F}_o \subset F$ that contains one out of the six permutations of each face $ijk \in \mathcal{F}$. Define the *dual graph* $D(\mathcal{G})$ of a triangulated graph $\mathcal{G}$ as a graph with $|\mathcal{F}_o|$ number of nodes, one corresponding to each (oriented) face of $\mathcal{G}$. There is an edge between two distinct faces $f_1, f_2 \in \mathcal{F}_o$ if and only if f_1 and f_2 share a common edge $e \in \mathcal{E}$. A *triangulated formation graph* is a quintuple

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D}, \mathcal{F}, \mathcal{A}),$$

with a connected dual graph $D(\mathcal{G})$. Let $q_i = (x_i, y_i)^T \in \mathbb{R}^2$ denote the position of the node v_i. Here, $\mathcal{D}$ is the set of distances $\|q_i - q_j\|$ and $\mathcal{F}$ is the set of triangular faces with the corresponding set of areas $\mathcal{A} = \{a_{ijk}\}$ defined by

$$a_{ijk} = \det \begin{bmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_k & y_k & 1 \end{bmatrix} = (q_k - q_i)^T S(q_j - q_i), \tag{6.6}$$

where

$$S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

Delaunay triangulation of a set of points is used to obtain the triangulated graphs. The edge and face orientation of the triangulated graph $\mathcal{G}$ is now fixed such that for all the faces $a_{ijk} \geq 0$, i.e., if for the face $ijk \in \mathcal{F}_o \subset \mathcal{F}$, $a_{ijk} < 0$ then replace the triplet $(v_i, v_j, v_k) \in \mathcal{F}_o$ by (v_j, v_i, v_k) to change the sign of the determinant in equation (6.6). The following *edge and face deviation variables* (also known as shape variable [58]) associated with the edges and faces of the triangulated graph $\mathcal{G}$ are defined, respectively, as

$$\begin{aligned} \eta_{ij} &= \|q_j - q_i\| - d_{ij}, & \forall ij \in \mathcal{E}_o \\ \delta_{ijk} &= q_{ik} \otimes q_{ij} := (q_k - q_i)^T S(q_j - q_i) - a_{ijk}, & \forall ijk \in \mathcal{F}_o, \end{aligned}$$

where $q_{rs} := q_s - q_r$ and the tensor product $\otimes$ is defined by $\alpha \otimes \beta := \alpha^T S \beta$ for $\alpha, \beta \in \mathbb{R}^2$.

Let $p_i = \dot{q}_i$ denote the velocity of each node $v_i \in \mathcal{V}$. Then, the *edge and face deviation rate variables* (also known as shape velocities [58]) associated with the set of edges and faces of the graph $\mathcal{G}$ are defined, respectively, as follows:

$$\begin{aligned} \nu_{ij} := \dot{\eta}_{ij} &= \frac{(p_j - p_i)^T \cdot (q_j - q_i)}{\|q_j - q_i\|} = \mathbf{n}_{ij}^T \cdot (p_j - p_i), & \forall ij \in \mathcal{E}_o \\ \xi_{ijk} := \dot{\delta}_{ijk} &= (p_k - p_i)^T S(q_j - q_i) + (q_k - q_i)^T S(p_j - p_i), & \forall ijk \in \mathcal{F}_o, \end{aligned}$$

where $\mathbf{n}_{ij} = q_{ij}/\|q_{ij}\|$ for $q_i \neq q_j$. Using the notation $p_{rs} = p_s - p_r$ and $\alpha^\perp := S\alpha$

(thus $\alpha \otimes \beta = \alpha^T \cdot \beta^\perp$), we can simplify the expression for the shape velocities as

$$\begin{array}{rcll} \nu_{ij} & := & \mathbf{n}_{ij}^T \cdot p_{ij}, & \forall ij \in \mathcal{E}_o \\ \xi_{ijk} & := & p_{ik} \otimes q_{ij} + q_{ik} \otimes p_{ij} = p_{ik}^T \cdot q_{ij}^\perp - p_{ij}^T \cdot q_{ik}^\perp, & \forall ijk \in \mathcal{F}_o. \end{array}$$

The integrated cost for the problem of tracking in formation is constructed by combining costs, one for each task: formation stabilization, collision avoidance, and tracking. The formation cost is restated here, while the other two costs can be found in [56].

Let $\sigma(x) : \mathbb{R} \to \mathbb{R}$ be a continuous and locally Lipschitz function satisfying the following properties: i) $\sigma(0) = 0$, ii) $(x-y)(\sigma(x)-\sigma(y)) > 0, \forall x \neq y$. Then, based on ii), $x\sigma(x) > 0$ and $\phi(x) = \int_0^x \sigma(s)ds$ is a positive definite and convex function which we refer to as a *cost function.* As an example, consider

$$\sigma(x) = \frac{x}{\sqrt{x^2+1}} \to \phi(x) = \sqrt{x^2+1} - 1$$

The *potential-based cost* $V_\mathcal{G}(q)$ and the *kinetic-based cost* $T_\mathcal{G}(q,p)$ associated with the formation graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{D}, \mathcal{F}, \mathcal{A})$ are defined as

$$\begin{array}{rcl} V_\mathcal{G}(q) & := & \sum_{ij\in\mathcal{E}_o} \phi_1(\eta_{ij}) + \sum_{ijk\in\mathcal{F}_o} \phi_2(\delta_{ijk}), \\ T_\mathcal{G}(q,p) & := & \sum_{ij\in\mathcal{E}_o} \phi_3(\nu_{ij}) + \sum_{ijk\in\mathcal{F}_o} \phi_4(\xi_{ijk}), \end{array}$$

where $\phi_i(x) = \int_0^x \sigma_i(s)ds, i = 1,2,3,4$ and the σ_i's satisfy conditions i) and ii). For the special case where all the σ_i's are equal to the identity function, $\phi_i(x) = x^2/2$ and both $V_\mathcal{G}, T_\mathcal{G}$ are quadratic functions of the shape variables and velocities. The formation Hamiltonian [57] given by

$$H_\mathcal{G}(q,p) = T_\mathcal{G}(q,p) + V_\mathcal{G}(q)$$

is the *formation cost* induced by the cost graph $(\mathcal{G}, \Phi_f)$. In the single optimal control problem, the integrated cost is defined as the sum of the formation cost above plus a tracking cost and collision avoidance cost, defined in [56]. The terminal cost is defined as the integrated cost minus the collision avoidance cost, and the dynamics and constraints of the vehicles are defined below.

In the paper [56], simulations are explored with each vehicle as a hovercraft mobile robot. The vehicle dynamics correspond to those of the Caltech Multi-Vehicle Wireless Testbed (MVWT) [13]. The hovercraft dynamics are given by

$$\begin{aligned}
\dot{q}_i &= p_i \\
m\dot{p}_i &= \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix} (u_i^1 + u_i^2) - k_1 \cdot p_i \\
\dot{\theta}_i &= \omega_i \\
I_0 \dot{\omega}_i &= r_0(u_i^1 - u_i^2) - k_2 \cdot \omega_i
\end{aligned}$$

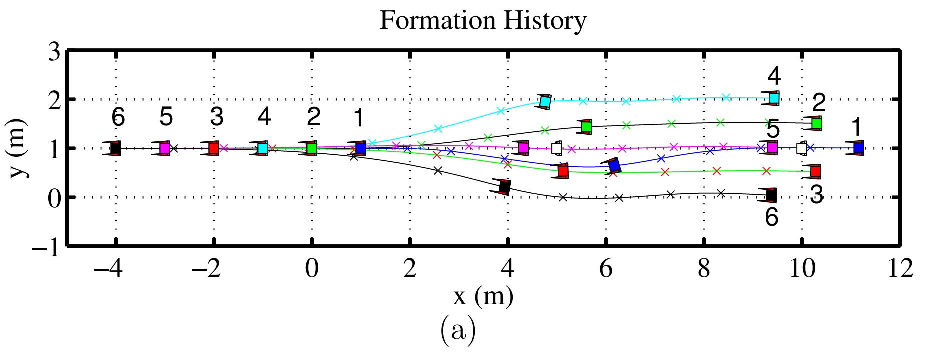
where $0 \leq k_1/m, k_2/I_0 \ll 1$ and $m, I_0, r_0 > 0$ are physical parameters of the vehicle dynamics. In addition, the control inputs of each vehicle are positive and bounded as $0 \leq u_i^1, u_i^2, \leq u_{max}$. In other words, the control $u_i = (u_i^1, u_i^2)$ belongs to a compact set $\mathcal{U}$ as

$$u_i \in \mathcal{U} = [0 \; u_{max}] \times [0 \; u_{max}] \tag{6.7}$$

The system dynamics are underactuated with 3 degrees of freedom and 2 control inputs.

The simulation results for tracking in formation for a group of six hovercraft are reported here. The receding horizon controller is centralized with a horizon time of 6 seconds and an update time of 1 second. Figure 6.13 (a) shows a 6-vehicle formation tracking a reference moving with constant velocity and heading. The vehicles are multi-colored and the reference vehicle is clear (white). The trajectories of each vehicle are denoted by lines with "x" marks for each receding horizon update.

The vehicles are initially lined up with a velocity of 1 in the horizontal direction, equal to the reference velocity. Note that without the collision avoidance cost term, vehicles 4 and 3 collide around 2.5 seconds. Figure 6.13 (b) shows snapshots of the evolution of the formation for the first 5 seconds of tracking. Figures 6.14 (a) and (b) show the control inputs for vehicles 1 through 3 and vehicles 4 through 6, respectively. It can be observed that vehicle 6 in particular performs a rather aggressive maneuver,



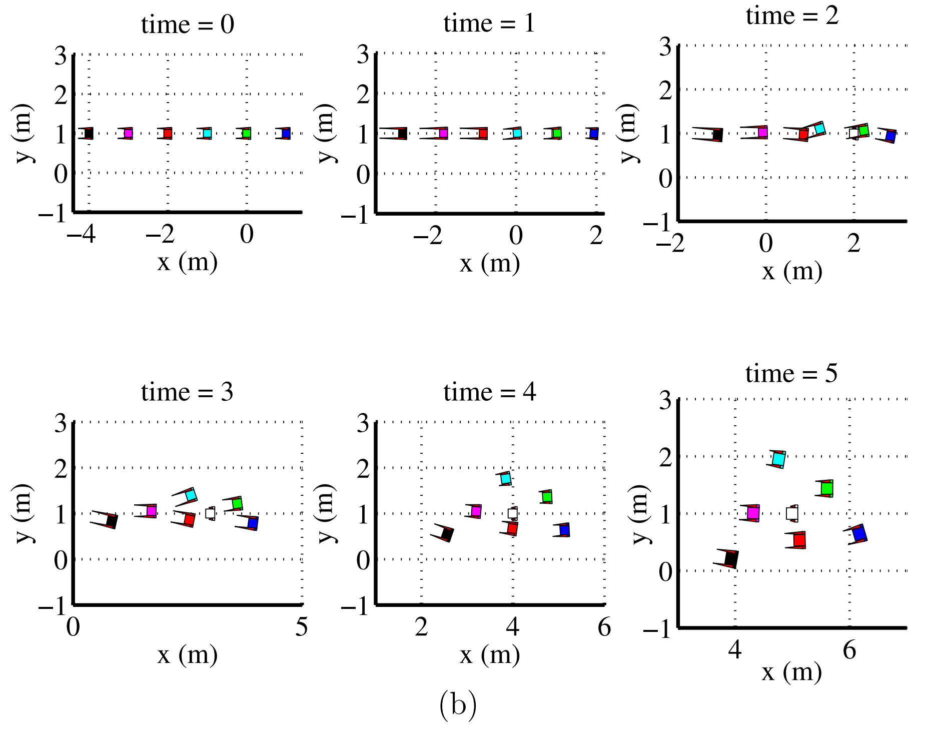
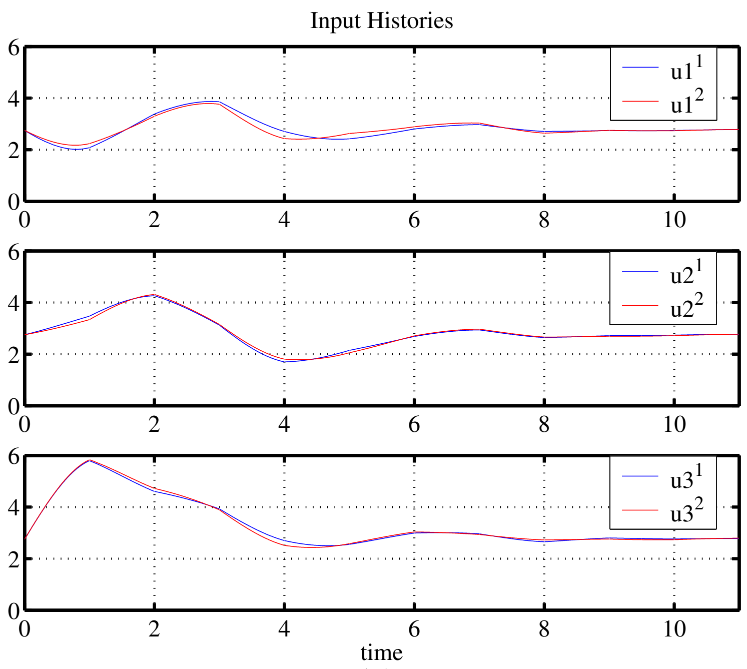


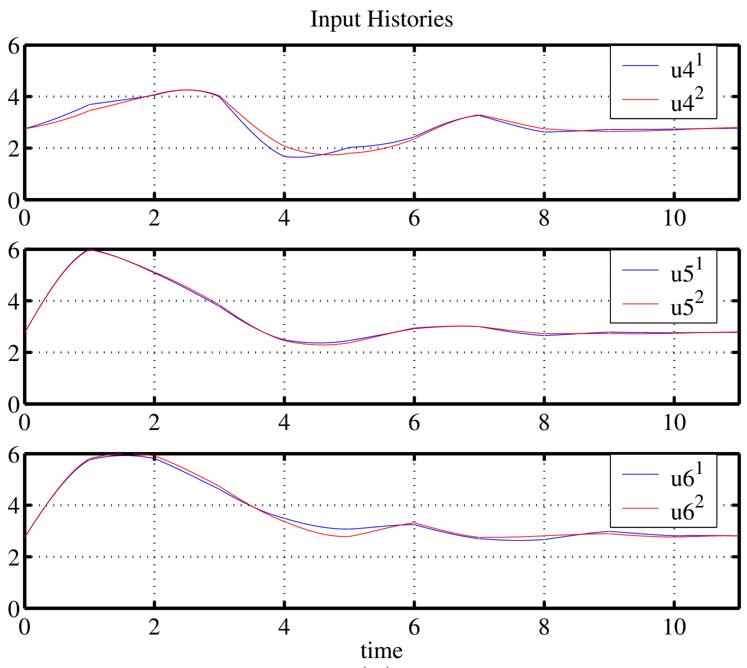
Figure 6.13: Trajectories of a six-vehicle formation: (a) the evolution and the path of the formation, (b) snapshots of the evolution of the formation (note: the two cones at the sides of each vehicle show the magnitudes of the control inputs).

as the control goes to the constraint bounds for nearly 1 second. After 10 seconds, all vehicles reach a steady-state behavior.

In comparison, we observe first that while the relative vector description generates a cost function that is quadratic (convex) in the vehicle states, the relative distance and area description here induces a cost that is non-convex in the vehicle states. For numerical reasons, then, the relative vector description may be preferable. However, a nice property of the relative distance penalties is that they each incorporate a degree of (local) collision avoidance, in the sense that the cost increases when the vehicles



(a)



(b)

Figure 6.14: Control inputs applied by each vehicle for the purpose of tracking in formation: (a) controls of vehicles 1 through 3, (b) controls of vehicles 4 through 6.

get too close from any direction. The relative vector penalties, on the other hand, do not incorporate repulsive type forces and by minimizing this cost alone, there are simple scenarios where vehicles are encouraged to pass through one another to reach the desired relative location.

As for the decomposition, the same procedure used in the first part of the chapter still applies: each local integrated cost is defined by a portion of each edge and area term, as well as a potion of each corresponding rate term, in which the state of that agent occurs. Design of the terminal costs and constraints can still be based on the linearization techniques presented in Chapter 4. Alternatively, the decoupled terminal costs could be based on appropriate control Lyapunov function (CLF) that accounts for the nonlinear dynamics and input constraints, and a level set of the CLF could in turn define each decoupled terminal constraint set. Ultimately, the distributed implementation for receding horizon control will be applied to the MVWT experiment, where the vehicles have the hovercraft dynamics given above.

6.6 Summary

In this chapter, numerical experiments for stabilization of a multi-vehicle fingertip formation were used to compare centralized and distributed implementations of receding horizon control. With the theoretical conditions given in Chapter 4 as a guide, performance of the distributed implementation is close to that of the centralized implementation. In particular, comparable performance is achieved by relying on assumed information about neighbors and for a fixed small update period, without enforcing compatibility constraints. A reason for the success without enforcing the compatibility constraints is the inherent compatibility observed between current optimal trajectories and the (remainder of the) previous optimal trajectories. Specifically, it is often observed in many applications that there is little change from one update to the next in these trajectories during transient response. On the other hand, when coupling occurs in constraints instead of coupling only in the cost function as in the simulations, compatibility constraints are crucial to ensure feasibility, as demonstrated in Section 5.4.

The scale of the simulation example here is small, as it is not intended to demonstrate the improvement in computational tractability over the centralized problem.

Rather, we are interested in comparing performance, viewing the centralized response as a benchmark. It is observed that if the trajectories are known to be sufficiently smooth, and polynomial-based approximations are valid, the communication requirements need not be substantially worse than that of standard decentralized schemes. Finally, we should also emphasize that the multi-vehicle formation stabilization problem is simply a venue. In the next chapter, connections between the theory of Chapter 4 and other fields, as well as other potential venues for the theory, are identified for future research.

Chapter 7

Extensions

7.1 Introduction

The ultimate utility of the theory developed in this dissertation relies upon two things: 1) identifying connections with related existing approaches, particularly those that enjoy success in applications, and 2) determining if relevant engineering problems could benefit by application of the theory or suggest appropriate modifications to the theory. To that end, this chapter motivates potential connections between the theory presented here and other relevant approaches, some of which are outside of control theory. Additionally, potential applications in other domains of engineering are mentioned. This chapter, while not complete, provides a road map for future research opportunities.

7.2 Relevant Areas of Research

In this section, several existing areas of research relevant to distributed optimization-based control are discussed.

7.2.1 Parallel and Distributed Optimization

The work of Bertsekas and Tsitsiklis in [3] is the benchmark for characterizing numerical algorithms that are suited for large-scale parallelization, motivated by the search for solutions to large-scale numerical problems, e.g., optimization problems.

The characterization is in the form of quantifying the effects of communication overhead and delay under different timing structures, i.e., when timing is synchronous or asynchronous. The distributed receding horizon control algorithm, and at a lower level, the algorithms used the solve each optimization problem, should be characterized in the same way. This would provide a means of quantitative comparisons with the centralized implementation, in terms of computation and communication based performance and not just the closed-loop performance explored in the simulation studies. Also, it may be that the principles underlying the relaxation methods can guide in improving the communication overhead of the distributed implementation as it is now.

7.2.2 Optimal Control and Neighboring Extremals

For a finite horizon optimal control problem of type Bolza[1], suppose that $u_{\text{opt}}(t)$ is a control that satisfies all of the first-order necessary conditions. In this problem, the only constraints are the dynamics and a terminal equality constraint on the state. The *neighboring extremal paths*, as explored in Chapter 6 of [35], are determined by linearizing the first-order equations around the extremal path, considering small perturbations in initial condition and terminal constraint function. The resulting equations that give the neighboring extremals correspond to a linear-quadratic problem, for which there are efficient numerical techniques, and occasionally analytic solutions. The purpose of finding the perturbation feedback control $u_{\text{pert}}(t)$, i.e., the control in the vicinity of a nominal path, is that the problem of finding the nonlinear optimal feedback law by dynamic programming is usually computationally intractable. Instead, one can apply $u_{\text{opt}}(t) + u_{\text{pert}}(t)$, which is the sum of the nominal control (pre-computed) and a continuous linear feedback law (cheap to compute), providing a feedback for the original nonlinear system in a neighborhood of the extremal path.

In comparison to the neighboring extremal approach, receding horizon control involves resolving the same optimal control problem at each update, with a planning

[1]Refer to the problem defined in Section 2.5 of [35].

horizon that is a fixed amount of time. An obvious connection between the two approaches is the following: at each receding horizon control update, formulate the linear-quadratic problem by linearizing around the current feasible path, generated by taking the remainder of the previous optimal and concatenating with the terminal control law path. While an optimal control problem is still being solved at each update, inline with the receding horizon control philosophy, the linear-quadratic version alleviates the computational problems of the general nonlinear problem. However, it is likely difficult to extend the neighboring extremal approach when path constraints on the states and controls are present.

Recent work on connecting neighboring extremals and receding horizon control includes the companion papers by Strizzi *et al.* [68] and Yan *et al.* [77]. The authors formulate the computational problem associated with a two degree-of-freedom control approach, where the outer-loop involves trajectory generation for a general nonlinear system [68], and the inner-loop solves the neighboring optimal control problem [77], using the trajectory provided by the outer-loop as the nominal path.

In the distributed implementation of Chapter 4, each local optimal control problem can be solved given initial conditions and assumed controls, for the corresponding agent and neighboring agents. It is assumed that there is no explicit uncertainty in the problem. However, while there is no mismatch in the initial conditions of the assumed and actual state trajectories, there is "uncertainty" in each local optimal control problem in the discrepancy that arises between the assumed and actual state trajectories of neighbors *over time*, all the way to the decoupled terminal constraint sets. The purpose of the compatibility constraint is to mitigate this uncertainty. From the compatibility constraint, actual trajectories remain close to assumed trajectories.

It would be interesting to explore whether the perturbation analysis, used to define the neighboring extremal path problems, could be used to analyze the perturbation effect that neighbors have on agents as the receding horizon control gets updated. It would be particularly useful if the analysis could identify reduced-order versions of the computation and communication requirements between neighboring agents. For example, linearized models around updated nominal paths could be used to generate the needed assumed trajectories of any neighbor.

7.2.3 Multiagent Systems in Computer Science

There is a large body of literature based in the computer science and artificial intelligence communities on intelligent agents and multiagent systems. Wooldridge describes an *agent* as “a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objective” [75]. The first example of an agent cited is “any control system,” using the thermostat as an instance of a control system. Some tools involved in defining properties of agents and, eventually, desired features of agents (including intelligence) include first-order logic, which is used to define abstract architectures, and game-theoretic algorithms to enable decision making. Generally, an agent senses input information and performs output actions in response, and non-determinism in the environment is assumed to incorporate the feature that agents can at best partially influence their environment. Agents are given a set of possible actions available to them with pre-conditions, or predicates, that determine when a particular action could be applied.

Researchers in the multiagent systems community are interested in modelling multiagent interactions, using logic and game theory, as well as designing algorithms for reaching consensus or agreement. Aspects of communication between concurrent systems is also of importance, including language design and synchronization. An intuitive argument is that any two processes need to be synchronized if there is possibility of destructive interference. An obvious example of such a possibility is in the initialization of collision-free trajectories between agents, as discussed in Section 5.4. A description of how agents work together involves dynamic coordination and cooperation [75].

A generic description also in [75] most related to the work in this dissertation is titled cooperative distributed problem solving (CDPS). Cooperations is necessary in multiagent systems of this type, where no single agent has sufficient resources or information to solve the overall system-level problem. A difficulty arises in that the agents most cooperate amidst self-interest, i.e., the agents have personal goals in conjunction with the overall objective. Coherence and coordination are metrics used to measure the success of CDPS. As with the distributed receding horizon control law developed here, CDPS is distinguished from parallel problem solving. A recent paper by Ygge and Akkermans [80] explored a multiagent system approach for climate control in a large building, comparing it to other relevant approaches. Another book worthy of exploration for connection with receding horizon control, and distributed implementation, is by Liu [42].

As the multiagent system community has established itself as an important area of computer science, there is much to be learned by seeking connections with the work presented in this dissertation.

7.2.4 Other Areas

A tool arising in Markov decision problems that bears relevance to receding horizon control is rollout algorithms. For combinatorial optimization problems, dynamic programming can provide an optimal policy, although the computational demands are usually too large in practice. Therefore, heuristics are employed to solve these problems. Rollout algorithms are a way of improving the performance of such heuristics, by sequentially calling them as subroutines and by satisfying an improvement property [4]. A connection between receding horizon control and rollout policies was recently initiated in the work of Chang [7, 8].

Another area of research relevant to receding horizon control is cooperative dynamic games [21]. For distributed problems, the theory behind such games with multiple players may also be related. There appears at first site to also be a connection between the compatibility constraints used in the theory of Chapter 4 and the incentive compatibility constraints defined in the book “The Analytics of Uncertainty and Information” by Hirshleifer and Riley [28]. The constraint in [28] is to ensure that an agent, acting on behalf of but unobservable to a principle, has incentives consistent with that of the principle.

7.3 Potential Future Applications

In this section, potential applications of the distributed receding horizon control approach developed in this dissertation are discussed. Since all such applications involve communication networks, we could divide them into two problem categories: 1) control over networks, and 2) control of networks. By control over networks, we mean that the controller is designed with some objective in mind, subject to the networking issues (topology, routing, protocols, etc.). An interesting research question posed in a recent panel report on future directions in control, dynamics, and systems [55] is the following: "Can one develop a theory and practice for control systems that operate in a distributed, asynchronous, packet-based environment?" Practically, that is the situation in any distributed control environment.

Control of networks, on the other hand, involves controlling some aspect of the network itself, e.g., protocols for minimizing congestion or round-trip time. In this section, examples are given that involve control over networks, while control of networks is also briefly explored. There may be engineering systems in the future where the local control mechanisms and the communication protocols that enable them are jointly synthesized, thus combining the categories. The generality of the receding horizon control mechanism allows us to postulate such combined synthesis[2].

7.3.1 Mobile Sensor Networks

To motivate distributed computations, Bertsekas and Tsitsiklis give a vision of information acquisition, information extraction and control applications within geographically distributed systems [3]. An example is sensor networks, for distributed acquisition and cooperative processing of information. Although the motivation was

[2]In a simple one-dimensional multi-vehicle example, Yan and Bitmead explored the interaction between the control performance of receding horizon control and the information quality of different information flow architectures [78].

a side note in a book primarily dedicated to parallelization of problems, is clearly now becoming an active and growing area of research. When the sensors become mobile, e.g., mobile robots capable of localized sensing, the potential for applications soars. A solid recent work by Cortés *et al.* [12] provides motivation and review of recent literature in this area. The authors address designing control and coordination algorithms for optimal coverage in an asynchronous distributed network, while collision avoidance and nonisotropy of sensors is not considered. To be practically implementable, algorithms for coverage problems must address all of these issues collectively, which is a tall order indeed.

If each local optimization problem can be solved efficiently, the distributed implementation presented here is particularly useful for mobile sensor network problems, in its generality and ability to handle reconfigurations. To address the nonconvexity of collision avoidance, an initialization procedure should be designed to operate on top of each optimization problem.

7.3.2 Control of Networks

The basic problems in control of networks include routing the flow of packets through the network, caching and updating data at multiple locations, controlling congestion across network links, and managing power levels for wireless networks [55]. Congestions control of the Internet has an optimization interpretation, viewed specifically as a asynchronous distributed primal-dual algorithm. Specific protocols, e.g., TCP Reno, correspond to particular utility functions in the optimization problem. In multiagent systems, it is more likely that information exchanges happen over an *ad hoc* wireless network, in which transmission power control becomes an additional issue.

A connection between receding horizon control and rollout policies in mentioned above. Recent work has explored the use of rollout policy for congestion control [76]. A drawback is that the rollout policy requires centralized implementation. Baglietto *et al.* [2] consider the problem of distributed dynamic routing in a network and apply a receding horizon approach. In order to obtain a distributed solution to the centralized problem, stochastic approximation is used to make the gradient algorithm computations tractable, and heuristics are used for step-size selection in the hopes of convergence. It would not be too difficult to formulate a discrete-time version of the distributed implementation in this dissertation and compare to the results in [2].

7.4 Summary

There are many theoretical extensions and application opportunities for distributed receding horizon control. A characterization of the computation and communication overhead should be quantified. The characterization would aid in comparisons with other approaches and in determining for which applications the implementation is viable. After all, demonstrations in such applications will ultimately determine the true utility of the approach.

This chapter sought to identify problems that may benefit from a distributed receding horizon control approach, without considering the more detailed questions related to implementation. There are several problem domains not mentioned above that have large-scale interconnected structure and make use of receding horizon control. One example is a study that employs model predictive control for short term performance maximization and long term degradation minimization in power plants [24]. Another paper that explores the use of model predictive control in power systems is [41]. Given the large-scale, interconnected, constrained and dynamic nature of power systems, distributed implementations of predictive control may be a useful approach, provided the computation and communication requirements can be accommodated. Recently, researchers in supply chain systems have also explored the use of receding horizon control [71, 72, 6]. Individual supply chains are often large-scale and distributed, with (internal) cooperative objectives, although supply chains are competitive with one another in general market situations.

Of course, the success in applications of the theory presented here depends critically on the algorithms available to solve the optimization problem, which in turn depends upon the details of the specific problem in question. There is also obvious dependence on the communication network properties. While implementation issues were discussed briefly in Chapter 2 and experimentally addressed in Chapter 3, there is clearly much to be done, in theory and in practice.

Chapter 8

Conclusions

8.1 Summary of Main Results

We began this dissertation with a review of a receding horizon control law that admits a general nonlinear model, constraints and quadratic cost function. Sufficient conditions for asymptotic stability are stated and issues regarding implementation and relaxations of the assumptions are briefly explored. To date, the predominant number of successful examples of receding horizon control in practice arise in the process control field, where the time scales of the dynamics are sufficiently slow to permit the required online optimization calculations.

A motivation for this dissertation is the application of receding horizon control to multi-vehicle systems. Consequently, it is important to examine the real-time issues that arise when the time scales of system dynamics are much faster than in the applications of process control. Fortunately, the scale of the optimization problem for individual vehicles systems is much smaller than that of process control systems. Recent computational tools developed at Caltech have also made it possible to solve optimal control problems with ample efficiency for real-time trajectory generation and receding horizon control of vehicles. In particular, successful initial experimental demonstration of receding horizon control of an unmanned flight vehicle, the Caltech ducted fan, is reported in Chapter 3.

Returning to multi-vehicle systems, the generality of receding horizon control makes it easy to formulate a meaningful single optimal control problem for a centralized receding horizon implementation. However, unless the system is restricted to a moderate number of vehicles, the scale of the optimization problem begins to approach that of process control problems. As a result, real-time centralized implementations of many vehicle systems is not possible due to the time scales of vehicle dynamics. This issue motivated the distributed implementation presented in Chapter 4, which is the primary contribution of this dissertation. In an attempt to incorporate generality, we refer to the governing system as a multiagent system, where the individual agents have dynamics that may be nonlinear and heterogeneous. Agent dynamics need not be vehicle dynamics; however, the agent dynamics are presumed to be decoupled from one another and modelled as ordinary differential equations. Also, the coupling cost function is posed in a generic, quadratic form.

The distributed implementation is generated first by decomposition of the single optimal control problem into local optimal control problems. A local compatibility constraint is then incorporated in each local optimal control problem. The coordination requirements are globally synchronous timing and local information exchanges between neighboring agents. For sufficiently fast update times, the distributed implementation is proven to be asymptotically stabilizing.

The distributed receding horizon control law is then analyzed in detail in Chapter 5. The implementation is qualitatively compared, in terms of the cost of computation and communication, to two centralized implementations of receding horizon control. While the distributed implementation has a considerable advantage over centralized implementations by being computationally scalable, a tradeoff is that the compatibility constraints result in a more sluggish transient response, as shown quantitatively in that chapter. Also, while the theory of Chapter 4 requires the update period to shrink to zero as the collective objective of the agents is approached, a dual-mode version of the implementation is presented in Chapter 5 to permit the use of a fixed, small update period while still guaranteeing convergence. Extensions for admitting more general coupling cost functions, for handling coupling constraints and partially synchronous timing are also explored.

In Chapter 6, the venue of multi-vehicle formation stabilization is used for conducting numerical experiments. The experiments demonstrate comparable performance between centralized and distributed implementations. The numerical experiments also indicated that the compatibility constraints need not be enforced to achieve good convergence. In other words, for good convergence, it is sufficient to exchange the assumed information that is based on the previous optimal trajectories; an intuitive reason for this it that receding horizon trajectories are often observed to (eventually) change little from one update to the next. The result depended on the fact that coupling occurred in the cost function, and not in constraints. In the case of coupling constraints, compatibility constraints are critical for maintaining feasibility from one receding horizon update to the next. This is shown by simple example in Chapter 5.

Regarding application and theory of the distributed implementation, we can make the following generalizations about what is easy, and what is hard:

Easy: the accommodation of heterogeneous decoupled dynamics and constraints;

Easy: the accommodation of cooperative objectives, in the form of a generic coupled cost function (the coupling cost need not be quadratic, as explored in Section 5.4.1);

Easy: the decomposition of the single optimal control problem into distributed optimal control problems;

Easy: the reconfigurability in local optimal control problems, i.e., the ability to change constraints and costs on-the-fly, provided they happen uniformly[1];

Easy: solving the distributed optimization problems, relative to the scale of the centralized problem, particularly for a large number of agents;

Easy: the ability to respect a communications topology, as agents are required to communicate only with neighboring agents;

[1]Uniform reconfigurations could be performed using the same distributed consensus algorithm approach applied for synchronous control switching in the dual-mode implementation.

Hard: the computational requirements, relative to standard control techniques (this is the case for any implementation of receding horizon control, centralized or distributed);

Hard: the communication requirements, relative to standard decentralized control techniques, i.e., in the distributed receding horizon control implementation, current state information *and* control *trajectories* must be exchanged;

Hard: the compatibility constraint in each local optimization problem, which must be enforced in particular when coupling constraints are present, complicates each local computational problem;

Hard: theoretical results that do not depend on synchronization or lossless delay-free communication (this is hard for *any* control approach).

Finally, potential connections with relevant fields and problem domains were briefly reviewed.

8.2 Summary of Future Research

There are a growing number of problems in engineering that are applying receding horizon control, because of the utility of the predictive mechanism and the ability to incorporate generic constraints and performance objectives. Examples, as cited above, include power systems, supply chain systems, and routing and congestion problems in networks. There are also a growing number of engineering systems that have distributed and interconnected structure, where dynamics and constraints cannot be ignored. In such systems, the challenge of extending the theory and practice of control systems for operation in a distributed, asynchronous, packet-based environment is a very challenging one.

The great potential for receding horizon control is that very general problem descriptions can be addressed, and the control mechanism itself is quite simple to explain and understand. For the distributed and interconnected class of problems mentioned above, the contribution of this dissertation bears relevance. The distributed receding horizon control law has guaranteed convergence properties for a very general class of problems. Moreover, the conditions for stability, while conservative, provide a guide for implementations, shown to be successful in numerical experiments. For future research, it will be important to characterize the cost of implementation, in terms of computation and communication overhead. Practically, extensions for addressing the asynchrony of timing and information must also be developed.

The current information technology revolution will ultimately require convergence of tools from communication, control and computing. By its generality, a distributed receding horizon mechanism, that can utilize efficient computational devices and accommodate asynchrony, may be the best hope as a single unifying tool for control and information management in distributed and networked environments.

Appendix A

Basic Lemmas

From the proof of Lemma 4.3, recall the symmetric positive semi-definite matrix $\overline{Q}_i \in \mathbb{R}^{nN_i \times nN_i}$, partitioned as

$$\overline{Q}_i = \begin{bmatrix} \overline{Q}_{i,1} & \overline{Q}_{i,2} \\ \overline{Q}_{i,2}^T & \overline{Q}_{i,3} \end{bmatrix}, \quad \overline{Q}_{i,1} \in \mathbb{R}^{n \times n}, \quad \overline{Q}_{i,2} \in \mathbb{R}^{n \times n|\mathcal{N}_i|}, \quad \overline{Q}_{i,3} \in \mathbb{R}^{n|\mathcal{N}_i| \times n|\mathcal{N}_i|},$$

where $N_i = |\mathcal{N}_i| + 1$. The lemma below holds for any i, and so we use the notation $\overline{Q} = \overline{Q}_i$, $Q_a = \overline{Q}_{i,1}$, $Q_b = \overline{Q}_{i,2}$ and $Q_c = \overline{Q}_{i,3}$, with

$$\overline{Q} = \overline{Q}^T = \begin{bmatrix} Q_a & Q_b \\ Q_b^T & Q_c \end{bmatrix} \geq 0.$$

Recall that $\lambda_{\max}(M)$ denotes the maximum eigenvalue of any square matrix M. In the proof of Lemma 4.3, the following lemma is utilized[1].

Lemma A.1 *1. $\lambda_{max}(\overline{Q}) \geq \lambda_{max}(Q_c)$ and $\lambda_{max}(\overline{Q}) \geq \lambda_{max}(Q_a)$,*

2. $\lambda_{max}(\overline{Q}) \geq \lambda_{max}^{1/2}(Q_b^T Q_b)$.

Proof: Since $\overline{Q} \geq 0$, we have that $Q_a \geq 0$ and $Q_c \geq 0$. In addition, Q_a and Q_c are symmetric, since $\overline{Q}$ is symmetric. From the Rayleigh-Ritz Theorem [30],

$$\lambda_{\max}(\overline{Q}) = \max_{z^T z = 1} z^T \overline{Q} z.$$

[1]Thanks to Steve Waydo for his help in proving these lemmas.

1. For eigenvalue $\lambda_{\max}(Q_c) \in [0,\infty)$, let $y \in \mathbb{R}^{n|\mathcal{N}_i|}$ be the associated non-zero eigenvector. The eigenvalue is known to be real and nonnegative since Q_c is symmetric and nonnegative. Defining $z_y = (0, y/\|y\|) \in \mathbb{R}^{nN_i}$, we have

$$\lambda_{\max}(\overline{Q}) \geq z_y^T \overline{Q} z_y = \frac{y^T Q_c y}{y^T y} = \lambda_{\max}(Q_c),$$

which proves the first part of item 1. The proof that $\lambda_{\max}(\overline{Q}) \geq \lambda_{\max}(Q_a)$ follows by the same reasoning.

2. Computing the symmetric nonnegative matrix $\overline{Q}^T\overline{Q}$, we have

$$\overline{Q}^T\overline{Q} = \begin{bmatrix} Q_a^2 + Q_b Q_b^T & Q_a Q_b + Q_b Q_c \\ Q_b^T Q_a + Q_c Q_b^T & Q_b^T Q_b + Q_c^2 \end{bmatrix}.$$

By the same reasoning as in the proof of item 1 we have

$$\lambda_{\max}^2(\overline{Q}) = \lambda_{\max}(\overline{Q}^T\overline{Q}) \geq \lambda_{\max}(Q_b^T Q_b + Q_c^2) \geq \lambda_{\max}(Q_b^T Q_b),$$

where the last inequality follows since Q_c^2 is nonnegative, concluding the proof. ■

The following lemma is also used in the proof of Lemma 4.3.

Lemma A.2 *For every $i = 1, ..., N_a$, $\lambda_{max}(Q) \geq \lambda_{max}(\overline{Q}_i)$, where Q is the weighting of the centralized integrated cost and $\overline{Q}_i$ is the weighting in each distributed integrated cost.*

Proof: Recall that

$$\|z - z^c\|_Q^2 = \sum_{i=1}^{N_a} \left\| \begin{bmatrix} z_i - z_i^c \\ z_{-i} - z_{-i}^c \end{bmatrix} \right\|_{\overline{Q}_i}^2 .$$

Leaving out all nonnegative terms in the summation except for the ith term, we have for any $z \in \mathbb{R}^{nN_a}$,

$$\|z - z^c\|_Q^2 \geq \left\| \begin{bmatrix} z_i - z_i^c \\ z_{-i} - z_{-i}^c \end{bmatrix} \right\|_{\overline{Q}_i}^2 .$$

For eigenvalue $\lambda_{\max}(\overline{Q}_i) \in [0, \infty)$, let $y \in \mathbb{R}^{nN_i}$ be the associated non-zero eigenvector, partitioned as $y = (y_i, y_{j_1}, ..., y_{j_{|\mathcal{N}_i|}})$. Defining the vector $z^y = (z_1^y, ..., z_{N_a}^y) \in \mathbb{R}^{nN_a}$ with subvector components

$$z_i^y = y_i/\|y\|, \qquad z_j^y = \begin{cases} y_j/\|y\|, & j \in \mathcal{N}_i \\ 0, & j \in \{1, ..., N_a\} \setminus (\{i\} \bigcup \mathcal{N}_i) \end{cases},$$

we have that z^y is a unit vector. From the Rayleigh-Ritz Theorem, $\lambda_{\max}(Q) \geq (z^y - z^c)^T Q (z^y - z^c)$. From this inequality and the one above, it is clear that $\lambda_{\max}(Q) \geq \lambda_{\max}(\overline{Q}_i)$, for any $i = 1, ..., N_a$, completing the proof. ■

Bibliography

[1] L. Acar. Boundaries of the receding horizon control for interconnected systems. *Journal of Optimization Theory and Applications*, 84(2), 1995.

[2] M. Baglietto, T. Parisini, and R. Zoppoli. Neural approximations and team theory for dynamic routing: A receding horizon approach. In *Proceedings of the IEEE Conference on Decision and Control*, Phoenix, AZ, 1999.

[3] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Athena Scientific, 1997.

[4] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3:245–262, 1997.

[5] B. Bollobás. *Modern Graph Theory.* Graduate Texts in Mathematics, Springer-Verlag, 1998.

[6] M. W. Braun, D. E. Rivera, W. M. Carlyle, and K. G. Kempf. A model predictive control framework for robust management of multi-product, multi-echelon demand networks. In *15th IFAC World Congress*, Barcelona, Spain, 2002.

[7] H. S. Chang. *On-Line Sampling-Based Control for Network Queueing Problems.* PhD thesis, Purdue University, 2001.

[8] H. S. Chang and S. I. Marcus. Two-person zero-sum Markov games: Receding horizon approach. *IEEE Transactions on Automatic Control*, 48(11):1951–1961, 2003.

[9] C. C. Chen and L. Shaw. On receding horizon feedback control. *Automatica*, 18:349–352, 1982.

[10] H. Chen. *Stability and Robustness Considerations in Nonlinear MPC.* PhD thesis, University of Stuttgart, 1997.

[11] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive scheme with guaranteed stability. *Automatica*, 14(10):1205–1217, 1998.

[12] J. Cortés, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *Accepted in IEEE Transactions on Robotics and Automation*, 2003.

[13] L. Cremean, W. B. Dunbar, D. van Gogh, J. Meltzer, R. M. Murray, E. Klavins, and J. Hickey. The Caltech multi-vehicle wireless testbed. In *Proceedings of the Conference on Decision and Control*, Las Vegas, NV, 2002.

[14] R. D'Andrea and G. E. Dullerud. Distributed control design for spatially interconnected systems. *IEEE Transactions on Automatic Control*, 48(9):1478–1495, 2003.

[15] W. B. Dunbar, M. B. Milam, R. Franz, and R. M. Murray. Model predictive control of a thrust-vectored flight control experiment. In *Proceedings of the IFAC World Congress*, Barcelona, Spain, 2002.

[16] W. B. Dunbar and R. M. Murray. Model predictive control of coordinated multi-vehicle formations. In *Proceedings of the IEEE Conference on Decision and Control*, Las Vegas, NV, 2002.

[17] W. B. Dunbar and R. M. Murray. Distributed receding horizon control with application to multi-vehicle formation stabilization. Technical Report 04-001, Control and Dynamical Systems, California Institute of Technology, 2004. Submitted to Automatica, January, 2004.

[18] W. B. Dunbar and R. M. Murray. Receding horizon control of multi-vehicle formations: A distributed implementation. In *Submitted to the IEEE Conference on Decision and Control*, Bahamas, 2004.

[19] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *Submitted to IEEE Transactions on Automatic Control*, April 2004.

[20] R. Fierro, A. K. Das, V. Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. In *Proceedings of the 2001 IEEE International conference on Robotics and Automation*, May 2001.

[21] J. A. Filar and L. A. Petrosjan. Dynamic cooperative games. *International Game Theory Review*, 2(1):47–65, 2000.

[22] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Collaborative multi-robot localization. *Advances in Artificial Intelligence*, 1701:255–266, 1999.

[23] E. Frazzoli, Z.-H. Mao, J.-H. Oh, and E. Feron. Resolution of conflicts involving many aircraft via semidefinite programming. *Journal of Guidance, Control and Dynamics*, 24(1):79–86, 2001.

[24] E. Gallestey, A. Stothert, M. Antoine, and S. Morton. Model predictive control and the optimization of power plant load while considering lifetime consumption. *IEEE Transactions on Power Systems*, 17(1):186–191, 2002.

[25] P. Gill, W. Murray, M. Saunders, and M. Wright. *User's guide for NPSOL 5.0: A fortran package for nonlinear programming.* Systems Optimization Laboratory, Stanford University, Stanford, CA 94305, 1998.

[26] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel. Example of zero robustness in constrained model predictive control. In *Proceedings of the IEEE Conference on Decision and Control*, Maui, HI, 2003.

[27] D. M. Himmelblau, editor. *Decomposition of Large-scale Problems.* North-Holland Pub. Co., 1972.

[28] J. Hirshleifer and J. G. Riley. *The Analytics of Uncertainty and Information.* Cambridge University Press, 1992.

[29] Y-C. Ho and K. C. Chu. Team decision theory and information structures in optimal control problems - part I. *IEEE Transactions on Automatic Control*, 17(1):15–22, 1972.

[30] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[31] S. Huang and W. Ran. Autonomous intelligent vehicle and its performance in automated traffic systems. *International Journal of Control*, 72(18):1665–1688, 1999.

[32] A. Jadbabaie, J. Yu, and J. Hauser. Unconstrained receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 46(5):776–783, 2001.

[33] D. Jia and B. H. Krogh. Distributed model predictive control. In *Proceedings of the American Control Conference*, 2001.

[34] D. Jia and B. H. Krogh. Min-max feedback model predictive control for distributed control with communication. In *Proceedings of the American Control Conference*, 2002.

[35] A. E. Bryson Jr. and Y. Ho. *Applied Optimal Control.* Taylor and Francis, 1975. Revised Printing.

[36] T. Kaga and T. Fukuda. Group behaviour control on dynamically reconfigurable robotic system. *International Journal of Systems Science*, 32(3):353–363, 2001.

[37] W. Kang, A. Sparks, and S. Banda. Coordinated control of multisatellite systems. *Journal of Guidance, Control and Dynamics*, 24(2):360–368, 2001.

[38] S. S. Keerthi and E. G. Gilbert. Optimal, infinite horizon feedback laws for a general class of constrained discrete time systems: Stability and moving-horizon

approximations. *Journal of Optmization Theory and Application*, 57:256–293, 1988.

[39] T. Keviczky, F. Borrelli, and G. J. Balas. Model predictive contorl for decoupled systems: A study of decentralized schemes. In *Submitted to the American Control Conference*, Boston, MA, 2004.

[40] H. K. Khalil. *Nonlinear Systems, Second Edition.* Prentice Hall, 1996.

[41] M. Larsson and D. Karlsson. Coordinated systems protection scheme against voltage collapse using heuristic search and predictive control. *IEEE Transactions on Power Systems*, 18(3):1001–1006, 2003.

[42] J. Liu. *Autonomous Agents and Multi-Agent Systems.* World Scientific, 2001.

[43] J. E. Marsden and M. J. Hoffman. *Elementary Classical Analysis.* W. H. Freeman and Company, 1993. Second Edition.

[44] D. Q. Mayne. Control of constrained dynamic systems. *European Journal of Control*, 7(2-3):87–99, 2001.

[45] D. Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35:814–824, 1990.

[46] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Contrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.

[47] M. Mesbahi and F. Y. Hadaegh. Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching. *Journal of Guidance, Control and Dynamics*, 24(2):369–377, 2001.

[48] H. Michalska and D. Q. Mayne. Receding horizon control of nonlinear systems without differentiability of the optimal value function. *Systems and Controls Letters*, 16:123–130, 1991.

[49] H. Michalska and D. Q. Mayne. Robust receding horizon control of contrained nonlinear systems. *IEEE Transactions on Automatic Control*, 38:1623–1632, 1993.

[50] M. Milam and R. M. Murray. A testbed for nonlinear flight control techniques: the Caltech ducted fan. In *1999 Conference on Control Applications*, August 1999.

[51] M. B. Milam, R. Franz, J. Hauser, and R. M. Murray. Receding horizon control of a vectored thrust flight experiment. *Submitted to IEE Proceedings on Control Theory and Applications*, 2003.

[52] M. B. Milam, K. Mushambi, and R. M. Murray. A new computational approach to real-time trajectory generation for constrained mechanical systems. In *Proceedings of the Conference on Decision and Control*, 2000.

[53] M. B. Milam, N. Petit, and R. M. Murray. Constrained trajectory generation for microsatellite formation flying. In *AIAA Guidance, Navigation and Control Conference*, 2001.

[54] N. Motee and B. Sayyar-Rodsari. Optimal partitioning in distributed model predictive control. In *Proceedings of the American Control Conference*, 2003.

[55] R. M. Murray, editor. *Control in an Information Rich World*. SIAM, 2003.

[56] R. Olfati-Saber, W. B. Dunbar, and R. M. Murray. Cooperative control of multi-vehicle systems using cost graphs and optimization. In *Proceedings of the American Control Conference*, Denver, CO, 2003.

[57] R. Olfati-Saber and R. M. Murray. Distibuted cooperative control of multiple vehicle formations using structural potential functions. In *Proceedings of the 15th IFAC World Congress*, Barcelona, Spain, 2002.

[58] R. Olfati-Saber and R. M. Murray. Graph rigidity and distributed formation stabilization of multi-vehicle systems. In *Proceedings of the IEEE Conference on Decision and Control*, Las Vegas, Nevada, 2002.

[59] R. Olfati Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *Accepted in IEEE Transactions on Automatic Control*, 2004.

[60] E. Polak. *Optimization: Algorithms and Consistent Approximations.* Springer, New York, 1997.

[61] S. J. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. In J. C. Kantor, C. E. Garcia, and B. Carnahan, editors, *Fifth International Conference on Chemical Process Control*, pages 232–256. CACHE, AIChE, 1997.

[62] W. Ren and R.W. Beard. A decentralized scheme for spacecraft formation flying via the virtual structure approach. *Journal of Guidance, Control and Dynamics*, To Appear. Revised Submission: June, 2003.

[63] A. G. Richards, J. P. How, T. Schouwenaars, and E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *AIAA Journal of Guidance Control and Dynamics*, 25(4):755–764, 2002.

[64] M. Rotkowitz and S. Lall. Decentralized control information structures preserved under feedback. In *Proceedings of the IEEE Conference on Decision and Control*, Las Vegas, NV, 2002.

[65] A. K. Sanyal, A. Verma, and J. L. Junkins. Adaptation and cooperation in control of multiple robot manipulators. *Journal of the Astronautical Sciences*, 48(2-3):305–336, 2000.

[66] S. V. Savastuk and D. D. Siljak. Optimal decentralized control. In *American Control Conference*, Baltimore, MD, 1994.

[67] R. C. Scott and L. E. Pado. Active control of wind-tunnel model aeroelastic response using neural networks. *Journal of Guidance Control and Dynamics*, 23(6):1100–1108, 2000.

[68] J. Strizzi, F. Fahroo, and I. M. Ross. Towards real-time computation of optimal controls for nonlinear systems. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Monterey, CA, 2002.

[69] A. Tews and G. Wyeth. Maps: a system for multi-agent coordination. *Advanced Robotics*, 14(1):37–50, 2000.

[70] S. Thrun. A probabilistic on-line mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363, 2001.

[71] W. Wang, D. E. Rivera, and K. G. Kempf. Centralized model predictive control strategies for inventory management in semiconductor manufacturing supply chains. In *American Control Conference*, Denver, CO, 2003.

[72] W. Wang, J. Ryu, D. E. Rivera, K. G. Kempf, and K. D. Smith. A model predictive control approach for managing semiconductor manufacturing supply chains under uncertainty. In *Annual AIChE Meeting*, San Francisco, CA, 2003.

[73] X. G. Wang and C. K. H. Kim. Improved control of pneumatic lumber-handling systems. *IEEE Transactions on Control Systems Technology*, 9(3):458–472, 2001.

[74] H. S. Witsenhausen. A counterexample in stochastic optimum control. *SIAM Journal of Control*, 6(1):131–147, 1968.

[75] M. Wooldridge. *An Introduction to Multi-Agent Systems.* John Wiley & Sons, Ltd., 2002.

[76] G. Wu, E. K. P. Chong, and R. L. Givan. Congestion control using policy rollout. In *Proceedings of the IEEE Conference on Decision and Control*, Maui, HI, 2003.

[77] H. Yan, F. Fahroo, and I. M. Ross. Real-time computation of neighboring optimal control laws. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Monterey, CA, 2002.

[78] J. Yan and R. R. Bitmead. Coordinated control and information architectures. In *Proceedings of the IEEE Conference on Decision and Control*, Maui, HI, 2003.

[79] T. H. Yang and E. Polak. Moving horizon control of nonlinear systems with input saturation, disturbances and plant uncertainty. *International Journal of Control*, 58(4):875–903, 1993.

[80] F. Ygge and H. Akkermans. Making a case for multi-agent systems. Technical report, University of Karlskrona/Ronneby, 1997.