Robust Distributed Control of Constrained Linear Systems

P. A. Trodden

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering, October 2009.

Word count: circa sixty-thousand.
Abstract

This thesis presents new algorithms for the distributed control of a group of constrained, linear time-invariant (LTI) dynamic subsystems. Control agents for subsystems, which are dynamically-decoupled but share coupling constraints, exchange plans to achieve constraint satisfaction in the presence of unknown, persistent, but bounded, state disturbances. Based on the tube model predictive control method for robust control of LTI systems, the distributed model predictive control (DMPC) method guarantees robust feasibility and stability. As agents communicate only after optimizing, the resulting algorithm offers low and flexible communication levels; this is the first work to combine robust feasibility and convergence with flexible communications.

A cooperative form of the distributed MPC is presented, for problems where application of the standard DMPC results in poor performance owing to 'greedy' behaviour. By a local agent designing plans for other subsystems in the problem, cooperative behaviour is promoted by sacrificing local performance. A key contribution is that robust constraint satisfaction, feasibility and stability guarantees are maintained, yet system-wide performance may improve with only partial cooperation.

This thesis includes a formal analysis of cooperation in DMPC. Firstly, under specified assumptions, weaker than those required for robust stability, convergence of the system to a state limit set is shown. By relating game-theoretical concepts to the algorithm at convergence, it is shown the set of limit sets does not enlarge with cooperation; confirmation of an intuitive concept despite the extra conditions imposed for constraint satisfaction. In terms of closed-loop performance, adding cooperation may steer the system to a 'better' outcome. Secondly, cooperation is linked to the coupling structure. It is shown that 'full' cooperation is not always necessary. A new algorithm with adaptive cooperation is proposed, where a local agent searches for paths to other agents in a graph of active couplings; if a path exists to another agent, cooperation with that agent may offer a benefit. Furthermore, it is confirmed that the set of immediately-coupled neighbours, as adopted by previous cooperative DMPC approaches, is not necessarily the optimal cooperating set, and is insufficient to guarantee best distributed performance.

A further contribution is a generalization to permit local optimizations in parallel. The proposed approach further tightens each agent's coupling constraints by some margin; sufficient conditions are developed on the size of margin required to guarantee robust constraint satisfaction. Simulations show the method may not be excessively conservative, with closed-loop performance improving over that of the single-update formulation, yet at the expense of increased communication.

The algorithms are demonstrated throughout by numerical examples. A multi-vehicle applications chapter is also included, applying the DMPC to two problems: search, or coverage, of an area by a team of vehicles, and tracking and observation of dynamic targets by sensing vehicles. Simulations demonstrate the practicality of the proposed algorithms, and, furthermore, the benefits of inter-agent cooperation.
Acknowledgements

I would like to thank my supervisor, Dr Arthur Richards, for his guidance, support and inspiration over the last three and a half years, and for his initial idea in conceiving this project. It has been a most interesting and enjoyable journey.

Dave Nicholson has been very helpful at various points over the duration of this PhD. My thanks to him, and others in the Advanced Information Processing department up the road, for making useful the time spent there over a summer.

May I also offer my appreciation and thanks to Gillian Clare and Alison Eele for taking the time to read, and provide criticism on, parts of this thesis.

Finally, I would like to offer my deepest thanks to my wife, Laura, for her constant support throughout, and for her uncanny ability to grasp some of the more technical concepts in this work.
Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the Regulations of the University of Bristol. The work is original except where indicated by special reference in the text, and no part of the dissertation has been submitted for any other academic award. Any views expressed in the dissertation are those of the Author.

Signature:

Date: 23/10/09
# Contents

1 Introduction .................................................. 1
   1.1 Motivation .............................................. 1
   1.2 Background ............................................. 4
      1.2.1 Model predictive control .......................... 4
      1.2.2 Decentralized, distributed and cooperative control .. 6
   1.3 Outline .................................................. 9
   1.4 Notation and definitions ................................. 13

2 Robust Distributed MPC ....................................... 14
   2.1 Introduction ............................................ 15
   2.2 Problem statement ...................................... 16
   2.3 Tube model predictive control .......................... 18
      2.3.1 Feasibility of tube MPC ............................ 22
      2.3.2 Stability of tube MPC .............................. 26
      2.3.3 Relationship between tube MPC and constraint-tightening MPC 33
   2.4 Robust distributed MPC using tubes ...................... 39
      2.4.1 Feasibility of DMPC ................................. 42
      2.4.2 Stability of DMPC .................................. 43
   2.5 Communication analysis .................................. 51
      2.5.1 Comparison with constraint-tightening DMPC .......... 55
3 Distributed MPC with Cooperation

3.1 Introduction .................................. 70
3.1.1 Motivation: greedy behaviour in DMPC .................. 72
3.2 Distributed MPC with cooperation ................... 75
3.2.1 A local objective function for cooperation ............ 78
3.2.2 Implementation .......................... 82
3.2.3 Cooperative behaviour in DMPC ................ 87
3.3 Stability analysis ................................ 89
3.3.1 Augmented local optimization cost as Lyapunov function .. 90
3.3.2 Stability-constrained DMPC .................. 94
3.4 Numerical examples ........................... 98
3.5 Summary ................................ 105

4 Analysis of Cooperation in Distributed MPC ........ 108

4.1 Introduction ................................ 109
4.2 Preliminaries ............................... 111
4.2.1 Problem statement ........................ 111
4.2.2 Definitions from game theory and graph theory .......... 112
4.3 Effect of cooperation on state convergence ............... 115
4.3.1 Robust convergence to a state limit set ............ 116
4.3.2 Distributed MPC as a game .......................... 119
4.3.3 State convergence properties .................. 132
4.4 Adaptive cooperation in robust DMPC ................. 141
4.4.1 Properties of the coupling graph ................ 141
4.4.2 Adaptive cooperation based on active couplings .... 144
4.4.3 Implementation of active cooperation graph determination .. 146
4.4.4 Numerical examples ....................... 149
5 Distributed MPC with Parallel Updates

5.1 Introduction
5.2 Feasible parallel-update DMPC
  5.2.1 Margins for robust feasibility
  5.2.2 Feasible parallel-update DMPC for polyhedral constraints
5.3 Numerical example
5.4 Summary

6 Multi-Vehicle Applications of Distributed MPC

6.1 Introduction
6.2 Multi-vehicle cooperative search
  6.2.1 Introduction
  6.2.2 Problem statement
  6.2.3 Multi-vehicle cooperative search using distributed MPC
  6.2.4 Numerical examples
6.3 Distributed control of multiple vehicles for dynamic target tracking
  6.3.1 Introduction
  6.3.2 Preliminaries
  6.3.3 Distributed MPC for dynamic target tracking
  6.3.4 Numerical examples
6.4 Summary

7 Conclusions

7.1 Summary of contributions
7.2 Future research directions

Bibliography
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Air traffic over Britain</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Relationships of distributed MPC methods proposed in this thesis.</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>Comparison of tube and terminal sets</td>
<td>25</td>
</tr>
<tr>
<td>2.2</td>
<td>Tightened non-convex sets</td>
<td>31</td>
</tr>
<tr>
<td>2.3</td>
<td>Example of strictly-smaller feasible set for tube MPC compared with CT-MPC</td>
<td>38</td>
</tr>
<tr>
<td>2.4</td>
<td>Trajectories of two constrained masses controlled by DMPC</td>
<td>49</td>
</tr>
<tr>
<td>2.5</td>
<td>Variation of update sequence for DMPC of two masses</td>
<td>50</td>
</tr>
<tr>
<td>2.6</td>
<td>Timing diagram illustrating redundant inter-agent communication</td>
<td>54</td>
</tr>
<tr>
<td>2.7</td>
<td>Timing diagrams for tube DMPC and CT-DMPC</td>
<td>57</td>
</tr>
<tr>
<td>2.8</td>
<td>Computation time per update versus number of subsystems</td>
<td>63</td>
</tr>
<tr>
<td>2.9</td>
<td>Comparison of distributed and centralized terminal sets</td>
<td>64</td>
</tr>
<tr>
<td>2.10</td>
<td>Closed-loop cost versus communication</td>
<td>67</td>
</tr>
<tr>
<td>3.1</td>
<td>Illustration of ‘greedy’ behaviour</td>
<td>74</td>
</tr>
<tr>
<td>3.2</td>
<td>Illustration of cooperation concept</td>
<td>79</td>
</tr>
<tr>
<td>3.3</td>
<td>Illustration of information requirements for cooperative DMPC</td>
<td>86</td>
</tr>
<tr>
<td>3.4</td>
<td>Illustration of cooperative behaviour</td>
<td>87</td>
</tr>
<tr>
<td>3.5</td>
<td>Vehicle plans at time steps $k = {0,1,2,10}$</td>
<td>88</td>
</tr>
<tr>
<td>3.6</td>
<td>Cooperative DMPC for three and four vehicles</td>
<td>100</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Five mass system example: comparison of controllers ............. 139
5.1 Comparison of performance of parallel-update DMPC methods ... 173
6.1 Comparison of stretch metric for search ......................... 192
List of Algorithms

2.1 Centralized MPC using tubes ........................................... 21
2.2 Robust distributed MPC .................................................. 41
3.1 Robust DMPC with cooperation ........................................ 82
4.1 Robust DMPC with adaptive cooperation ............................. 148
5.1 Robust feasible-parallel DMPC ......................................... 169
6.1 Multi-vehicle cooperative search using DMPC ......................... 187
6.2 Robust DMPC for dynamic target tracking ............................. 206
Chapter 1

Introduction

This thesis proposes new algorithms and analyses for the distributed control of linear, dynamic subsystems subject to uncertainty and operating constraints. The key problems associated with the independent decision-making of local control agents are those of ensuring coupling constraint satisfaction, robustness to disturbances, stability, and good closed-loop performance; specifically, how should agents communicate and cooperate to achieve these aims?

This introduction chapter begins by explaining the motivation behind the problem of providing optimal control for distributed, constrained systems. In Section 1.2, an introduction is given to model predictive control, the technique underpinning the algorithms developed in this thesis, and also to the topics of decentralized, distributed and cooperative control. An outline of this thesis is presented in Section 1.3, and, finally, some of the notation and definitions used in subsequent chapters are defined in Section 1.4.

1.1 Motivation

The problem of providing control for real-world systems or plants is defined by one fundamental difficulty that classical control techniques overlook: that real dynamic
systems are subject to constraints, often inherent but also sometimes imposed. For the former, constraints arise from physical limitations: actuators have maximum slew rates or applied forces, aircraft have flight envelopes, and motor vehicles have maximum speeds, accelerations, and minimum turning radii. Safety or efficiency considerations may lead to the latter. In addition to the control system for such plants providing stable control, the designer is often faced with performance considerations: what control will maximize some performance measure while meeting all constraints? The presence of uncertainty or disturbances complicates matters even further; not only should the constraint-satisfying control law be optimal, it should also be robust.

This thesis considers such a control problem for large-scale constrained linear systems, such as chemical plants [1], teams of autonomous vehicles [2], or networks of sensors or cameras [3]. Control of such systems by a single, centralized agent, though perhaps desirable, is often difficult or even impossible to achieve, owing to the complexity or organizational structure of the system—and the resulting communication and computation requirements. A centralized approach is further hindered by reliance on a single processor, rendering the system prone to single-point failure.

Figure 1.1 shows a snapshot of air traffic over Britain [4]. Such a ‘system’ exhibits remarkable complexity, consisting of several layers. On the one hand, each airport on the map provides air traffic decision-making for aircraft within the local airspace; control is centralized and coordinated. The aircraft are dynamically-decoupled, and have their own kinematic constraints (flight envelope). The objective might be for an air traffic controller to maximize throughput for the airport. A decentralized approach, where each aircraft makes decisions locally and autonomously, might require limitless inter-aircraft communication and pre-set coordination rules to achieve satisfactory performance.

On the other hand, this is an example of decentralized control in action. At a higher level, the ‘system’ of several thousand aircraft is controlled by the numerous, independent airports over the world, each making routing and timing decisions
locally. The idea of providing centralized control at this level is inconceivable—air traffic controllers would have an unimaginable task, and a single incident could lead to a spectacular failure of global aircraft operations. By distributing the problem, complexity is made manageable, at least in terms of the capacity of each airport.

The class of problems studied in this thesis is defined by the following characteristics: the overall system is composed of, or may be decomposed to, a number of dynamically-decoupled subsystems. Each has linear, time-invariant dynamics, and is subject to local constraints and unknown, persistent, but bounded, disturbances. The subsystems are coupled through the constraints, and should coordinate decision-making to satisfy these constraints robustly and also to minimize some system-wide performance objective. Although this class is not restricted to aircraft or vehicle control, the air traffic scenario exhibits many of these properties. When considering individual aircraft as the 'subsystems' to control, dynamics are independent, and constraints are both local (flight envelopes) and coupled (collision avoidance, timing). Objectives are system-wide (i.e., maximize throughput, minimize fuel use or flight times), yet locally-made decisions might conflict; promoting one aircraft's objective over others may adversely affect the overall performance.

Figure 1.1: Air traffic over Britain. Source: BBC [4].
1.2 Background

The core technology behind the distributed control methods developed in this thesis is model predictive control. The next section introduces this technique, and briefly surveys the literature relevant to robust forms of MPC. Subsequently, the concept of decentralized or distributed MPC is introduced, and its relationship with cooperative control explained.

1.2.1 Model predictive control

Model predictive control (MPC) [5] is a modern control technique that solves, on-line at each sampling instant, an optimal control problem subject to constraints. Using a model of the system dynamics, the controller outputs a sequence of predicted open-loop controls, calculated to minimize some performance objective and satisfy constraints at the current, measured state. The first control in the sequence is applied to the system, and the process is repeated, introducing feedback.

MPC—often referred to as receding horizon control (RHC)—generally results in sub-optimal closed-loop performance. However, it has two redeeming features that make it well-suited to practical applications, and account for its wide-adopt in the process control industry [6]: (i) constraints are handled naturally by the constrained optimization framework, allowing operation closer to, or up against, boundaries or limits; (ii) the on-line, finite-horizon optimizations are tractable and implementable in real-time, especially given the continual advances in processor technology. This compares favourably with other modern control methods, such as linear quadratic regulation (LQR) or $\mathcal{H}_\infty$, which provide optimal controls (at least with respect to the system model), but lack constraint-handling capabilities. The main challenges for MPC are those of guaranteeing feasibility and closed-loop stability, and many schemes exist; see Mayne et al. [7], for example, for an excellent survey of techniques and results.
A key open problem for MPC, and a currently-active area of research, is robustness to the presence of uncertainty or disturbances [7–9]. Uncertainty can take several forms, including additive state disturbances [10–15], model mismatch or variability [16–20], and state measurement or estimation errors [21, 22]. This thesis considers only additive state disturbances, assumed unknown a priori, but bounded—a common assumption [7]. In this case, it is acknowledged that to account for disturbances, the on-line optimization should provide a feedback policy rather than a sequence of open-loop controls. However, optimization over an arbitrary, and possibly non-linear, set of policies is difficult, and so computational complexity must be reduced by assuming, a priori, some form for the control law. (In one sense, this is comparable to a ‘move blocking’ approach [23], in which the optimized input sequence assumes a certain form by fixing the inputs or its derivatives to be constant over several time steps). One proposal uses min-max optimization to design controls accounting for worst-case disturbances [24]; however, the optimizations scale unfavourably with problem dimension, and quickly become computationally prohibitive.

A more practical proposal uses a pre-determined feedback law, designed off-line, with the on-line MPC optimizations restricted to selecting admissible perturbations to this law. Set invariance concepts [25] are used to ensure robustness of the closed-loop system. The constraints in the optimization are tightened by a sufficient margin—with margin size increasing monotonically over the prediction horizon—so that an allowance is given for future feedback action and disturbances. A key benefit to this approach, first proposed by Gossner et al. [11] and later generalized by Chisci et al. [12], then Richards and How [14], Kuwata et al. [26], is the use of a nominal (i.e., disturbance-free) system model in the optimizations, so that complexity is comparable to standard MPC. A related approach is the tube MPC method proposed by Mayne et al. [27], in which constraints are tightened uniformly over the horizon, and—uniquely—the initial state is a decision variable in the optimization. It is this method on which the work in this thesis is based; the properties of tube MPC are
exploited to produce a robust distributed MPC scheme with communications more flexible than for similar methods.

1.2.2 Decentralized, distributed and cooperative control

The study of decentralized control dates back to the early 1970s [28–30], with interest having grown significantly since then; see Bakule [31] for a comprehensive survey of past and present methods.

Attention has recently been focused on developing decentralized or distributed model predictive control (DMPC) [32], to take advantage of the associated benefits MPC brings for constrained, dynamic systems. Such approaches see control decision-making distributed among agents corresponding to the different subsystems making up the whole. The challenge is then how to coordinate efforts to ensure that the distributed decisions lead to constraint satisfaction, feasibility and stability of the overall closed-loop system.

Distributed or decentralized MPC methods are mostly differentiated by the assumptions made on the system model and on the form interactions between constituent subsystems take. For example, dynamics may be linear or non-linear and continuous- or discrete-time; interactions may be caused by coupling in the dynamics [33–36], constraints [37, 38], or objectives [39]. A recent survey paper [40] has classified the common and different protocols found in the literature, and, furthermore, defines distributed control as a separate entity to decentralized control: regardless of the dynamics and interaction models assumed, distributed agents share information while decentralized agents do not.

Several strategies for DMPC have been presented in the literature: Dunbar and Murray [39] propose a scheme based on exchange of plans for subsystems coupled non-separably through the objective function, but not in the constraints. A similar method applies also to dynamically-coupled subsystems [35]; nominal stability is established by use of a compatibility constraint based on move suppression. Many authors propose iterative optimization or bargaining between subsystem
agents to achieve closed-loop stability, most often for dynamically-coupled subsystems [32, 33, 41, 42], but also for subsystems coupled through the constraints [37, 43] or cost function [44]. Other approaches to the problem include partial grouping of computations [45, 46]; partially-decoupled modelling of a dynamically-coupled system [47]; use of sequential solutions, for both dynamically-coupled systems [48] and for subsystems coupled through the constraints [38, 49]. In Shim et al. [50], Franco et al. [51], collision avoidance between locally-controlled vehicles, and hence coupling, is via potential functions in the local objectives; global stability in the latter is established from bounds derived from local LQR control laws. Input-to-State Stability (ISS) analysis has also been employed to prove stability of non-linear DMPC, both for coupled dynamics with no information exchange [52], and decoupled agents with a common objective sharing delayed state information [53].

Robustness to disturbances is a key challenge in the development of MPC [7], and is harder still for DMPC; few schemes in the literature offer robustness. Richards and How [38] showed that feasibility and stability can be guaranteed by updating each subsystem's plan in a sequence, whilst 'freezing' the plans of others. This has similarities with the 'move suppression' strategy of Dunbar [54]. Alternative approaches include treatment of interconnected subsystems' state trajectories as bounded uncertainties, and using min-max optimization [34]—though the complexity issues with such an optimization method are well documented [7, 15, 24, 55]. Using the recently-developed comparison model approach to robustness [13], another distributed method [56] uses worst-case predictions of state errors, determined based on a robust control Lyapunov function, and tightens constraints accordingly. Magni and Scattolini [57] propose a robust stable decentralized algorithm for non-linear dynamically-coupled systems, with no information exchange between agents, although for an asymptotically-decaying disturbance.

Another relevant approach for uncertain systems, not based on MPC, is distributed reinforcement learning [58–60]. In this on-line, adaptive method, each agent seeks to determine a policy that maximizes a long-term reward, based on
the current knowledge of the states of the world. The basic reinforcement learning model consists of states, actions, and rewards, where an agent’s actions are mapped to the state by a Markov Decision Process (MDP) [61]. An MDP is a discrete-time model with stochastic uncertainty, but observable and controllable discrete states. Further results have been developed for MDPs with partially-observable states (POMDPs) [62, 63]. However, these methods should be regarded as complementary to those of distributed MPC for systems with bounded, rather than stochastic, uncertainty, and where also the state is continuous.

While decentralized or distributed control places no assumptions on whether agents’ objectives are conflicting or not, cooperative control is concerned with the problem of agents sharing information and coordinating behaviour to accomplish common objectives or missions [64]. From the problem of agents reaching a consensus [65, 66] to UAV task-assignment [67] or a surveillance and reconnaissance mission [68], it is agreed that some degree of cooperation between agents must exist, else ‘greedy’ behaviour may lead to poor performance [69]. Where cooperation is encouraged, for example by including the global cost in local optimizations, team performance may improve. In Venkat et al. [1, 36, 70, 71], for dynamically-coupled systems without coupling constraints, at each step all agents optimize, exchange plans and iterate until convergence. Performance nears that of the centralized ideal [36], whereby Pareto-, or team-, optimal solutions are applied at each control update.

Where coupling constraints are present, distributed control methods may no longer necessarily apply Pareto-optimal solutions, despite iteration [69, 72]. Thus, the presence of such constraints has been identified as a key open research problem [72]. Approaches to achieving system-wide cooperation for this problem include a hybrid logic rule-based approach [73], dual decomposition and, subsequently, hierarchical optimizations [44, 74], bargaining [37, 75], and local agents iteratively solving low-order parameterizations of neighbours’ problems where the coupling structure is sparse [43]. In Keviczky et al. [45, 46], all agents solve their respec-
tive problems independently and simultaneously; although consideration is given to a neighbour's objective, coupling constraint satisfaction is not guaranteed [45].

The methods developed in this thesis apply to dynamically-decoupled, linear subsystems, coupled via the constraints. Subsystem objectives may be either decoupled or coupled, and examples of both are presented. The control agents communicate and share information, placing the algorithms within the class of distributed MPC methods. It is assumed that agents must aim to minimize a system-wide objective, but conflict may exist between local objectives. The main challenges addressed are that of achieving robust constraint satisfaction, feasibility and good closed-loop performance.

1.3 Outline

This thesis presents new algorithms for the distributed control of constrained linear systems. A key feature of the problem class under study is the presence of coupling constraints, and a number of new results are developed for this type of interaction between subsystems. Figure 1.2 summarizes the algorithms, spatially placing each according to communication and computation levels available. For example, were communication and computation unlimited, one would choose centralized MPC for best performance. Conversely, non-cooperative DMPC is a method that requires only low levels of both. The detailed contributions of each chapter are summarized in what follows.

Chapter 2 presents a distributed form of MPC for systems with linear, time-invariant, dynamically-decoupled dynamics. The subsystems share coupling constraints. Based on the tube MPC method [27] for robustness, the key features of the formulation are that (i) only one subsystem control agent optimizes at each time step, the rest 'freezing' their current, feasible plans; (ii) control agents exchange plans, but only immediately following optimizations; (iii) constraint satisfaction, feasibil-
Figure 1.2: Relationships of distributed MPC methods proposed in this thesis.


dity and stability of the closed-loop system are guaranteed, despite the presence of persistent disturbances, and for any choice of update sequence.

The chapter includes an analysis of communication and information requirements, and a comparison with the related constraint-tightening method of Richards and How [38]. It is shown that communication is considerably more flexible for this new method. In addition, a numerical exploration of the trade between closed-loop performance and communication is provided, finding that the distributed MPC can achieve lower cost values than for centralized MPC at low levels of communication.

Chapter 3 begins with a vehicle collision avoidance example, showing that the direct application of distributed MPC can lead to 'greedy' behaviour and, as a result, poor performance. A cooperative form of the DMPC is proposed, in which control agents consider a greater proportion of the system-wide objective when making decisions. In the cooperative optimization, a local agent designs—in addition to its own plan—hypothetical plans for other agents in the problem; the basic premise is to promote cooperation by sacrificing local performance.

An important feature of the method is that the robust constraint satisfaction guarantees of the 'greedy' DMPC still hold, by retaining certain constraints in local optimizations. The additional information requirements for the cooperative al-
gorithm are identified, and an analysis of stability is included; robust stability is guaranteed under mild assumptions or by adding a stabilizing constraint to local optimizations. Numerical simulations show that improved performance can result even with only partial cooperation. Furthermore, application of the method to a vehicle deadlock situation shows that cooperation 'breaks' the deadlock where non-cooperative DMPC does not.

Chapter 4 is made up of two parts. The main contribution of the first part is a formal analysis of cooperation in DMPC, examining the effect of cooperation on the convergence outcome of the system. Firstly, it is shown that under conditions milder than those required for asymptotic stability of a neighbourhood of the origin, the controlled system robustly converges to some state limit set. Subsequently, by relating game-theoretical concepts to the distributed algorithm, it is shown that the system is in such a limit set if and only if the control agents are continually playing Nash solutions. Relating the Nash solutions to the cooperation graph, it is proven that increasing inter-agent cooperation does not enlarge the set of Nash solutions. Thence, it follows that increasing cooperation does not enlarge the set of state limit sets for the system. Examples show that cases exist where the convergence outcome is improved by adding cooperation.

The second part of the chapter investigates the effect of the coupling structure on the cooperation required for 'good' performance. Whereas previous approaches have adopted cooperation between either all agents [36] or directly-coupled agents [43, 45], it is shown by analysis that—depending on the coupling structure—the former approach is not necessary, and the latter not sufficient, to obtain best distributed performance. Using these results, an adaptive form of cooperation between agents is proposed, where agents cooperate with others connected by paths in a graph of active coupling constraints. The key contributions, then, are a new distributed MPC algorithm with adaptive levels of cooperation, and confirmation that the set of im-
mediate, coupled neighbours is not necessarily the optimal cooperating set.

Chapter 5 extends the algorithm proposed in Chapter 2 to permit optimizations by local agents in parallel. By further tightening coupled constraints, by some specified margin, constraint violation is avoided even when directly-coupled subsystems modify their plans simultaneously. Furthermore, robust feasibility and stability are guaranteed, in the presence of disturbances, for any choice of updating set. Numerical simulations show that the proposed method may not be excessively conservative, and performance can improve over that of the original DMPC, though for some loss of flexibility in communications.

Chapter 6 applies the distributed MPC algorithms to two multi-vehicle control examples. The first is the problem of cooperative search, or coverage, of an area by a team of controlled vehicles. The aim is to completely search an area of known extent but unknown content, collecting rewards, while avoiding both collision with other vehicles and duplication of efforts. The proposed approach employs a local objective function for each vehicle that maximizes predicted rewards; in practice, a binary flag is associated with each cell, and control agents share information on predicted cell visits. The main contribution is a look-ahead team coverage algorithm that includes dynamic models, kinematic constraints, and collision avoidance. Numerical examples show that by using the cooperative form of the DMPC, performance is better than that for 'greedy' DMPC, searches being completed faster.

The second application is distributed control for dynamic target tracking. The problem statement is for a number of range-only sensing vehicles to track and observe a number of independent dynamic processes. Gaussian uncertainty is associated with both the sensing models and the target dynamics. Sensor platform dynamics are subject to bounded disturbances, as before. An information-theoretic objective function is developed, based on Kalman filtering, and incorporated into the local DMPC optimization problems. Agents determine the controls and trajec-
stories that would maximize predicted information-gathering over the horizon, and subsequently share and fuse plans and observations. Though the resulting optimizations are non-linear and non-convex, the efficacy and practicality of the proposed approach is demonstrated using a commercially-available solver. Furthermore, although the information-theoretic objective is highly-coupled, a performance benefit is seen by using the cooperative form of DMPC.

Chapter 7 summarizes the conclusions of this thesis, and outlines areas for future research.

1.4 Notation and definitions

A scalar, vector, matrix, and set are denoted, respectively, $a, a, A$, and $A$, unless otherwise stated. The relational operator ‘$	riangleq$’ denotes a definition. The double subscript notation $(k + j|k)$ indicates a prediction of a variable $j$ steps ahead from time $k$. The notations $1_{m,n}, 0_{m,n}$ denote, respectively, the $m \times n$ matrix of ones and matrix of zeros, while $I_m$ is the $m \times m$ identity matrix. Let $\mathbb{N} \triangleq \{0, 1, 2, \ldots\}$, and $\mathbb{R}_+$ and $\mathbb{R}_0^+$ denote the sets of positive and non-negative reals respectively.

The matrix mapping of a set is defined as $AB \triangleq \{c : \exists b \in B, c = Ab\}$. The operator ‘$\sim$’ denotes the Pontryagin difference [76], a set-shrinking operation defined as $A \sim B \triangleq \{a : a + b \in A, \forall b \in B\}$. The operator ‘$\oplus$’ denotes the Minkowski sum, defined as $A \oplus B \triangleq \{a + b, a \in A, b \in B\}$. An important consequence of the Pontryagin difference is that if $a \in (A \sim B)$ and $b \in B$, then $(a + b) \in A$ [76]. The function $d(\cdot, \cdot)$ denotes an appropriate norm of the distance of a point from a set, defined by $d(a, B) \triangleq \inf_{b \in B} \|a - b\|$.

A set $\mathcal{X}$ is a positively-invariant (PI) set for a system $x(k + 1) = f(x(k))$ if and only if, for all $x \in \mathcal{X}$, $f(x) \in \mathcal{X}$. A set $\mathcal{R}$ is a robust positively-invariant (RPI) set for a system $x(k + 1) = f(x(k)) + w(k)$ if and only if, for all $x \in \mathcal{R}$, $f(x) + w \in \mathcal{R}, \forall w \in \mathcal{W}$.
A robust, distributed form of MPC for linear subsystems is developed. The method guarantees stability and satisfaction of coupled constraints, despite the action of unknown, persistent, bounded disturbances. The distributed control agents make decisions locally, and communicate plans with each other to achieve coupled constraint satisfaction.

The new formulation is based on the tube MPC [27] method for robust control of linear systems, in which the problem of trajectory design for an uncertain system is solved by designing a 'tube' of trajectories for the perturbed system to follow under feedback. The key feature of the distributed method is greater flexibility in communication and computation over that of existing distributed methods.

Other contributions of the chapter include an investigation of the trade between performance and communication for an example scenario, identifying how to exploit the communication flexibility of the new algorithm; and a comparison with an existing constraint-tightening method.
2.1 Introduction

This chapter presents a distributed form of model predictive control (MPC) [5, 7, 32] that guarantees stability and satisfaction of coupled constraints despite the action of unknown, persistent, bounded disturbances. The distributed control agents communicate plans with each other to achieve constraint satisfaction. Key features of the new formulation are that (i) only one subsystem agent updates its plan at each time step, (though this is relaxed in Chapter 5), (ii) robust stability is guaranteed for any choice of update sequence, and (iii) each agent communicates only after its update. The resulting algorithm offers flexibility in communication and computation. This is the first work to combine guaranteed robust feasibility and convergence, in the presence of a persistent disturbance, with flexible communication. In addition, a thorough investigation of the trade between performance and communication is provided for an example scenario, identifying how to exploit the flexibility of the new algorithm.

The distributed MPC method presented in this chapter achieves robustness to persistent disturbances by use of tube MPC [27], a recently-developed form of robust MPC that guarantees feasibility and stability despite the action of an unknown but bounded disturbance. Similar to the ‘sequential’ DMPC method of Richards and How [38], robust feasibility and stability of the overall system is guaranteed by local agents updating plans one at a time, without iteration. However, this new method permits a flexible order of updating, as opposed to a fixed, pre-determined sequence.

MPC in the presence of bounded disturbances is linked to reachability of a target tube set, and various results exist; see, for example, Blanchini [25], Bertsekas [77], and references therein, for comprehensive surveys. In the tube MPC formulation proposed in Mayne et al. [27], the ‘tube’ is a sequence of robust invariant sets centered on a trajectory for the nominal (i.e., disturbance-free) system; use of feedback ensures that the system remains inside the tube for all possible realizations of the disturbance. A key observation of this new work is that if that feedback uses
only local information, each subsystem can remain within its tube without the need for communication, and exchange of information with other agents is only required when an agent optimizes for a new tube. The algorithm proposed in this chapter exploits this feature to achieve flexibility in communication. An additional advantage of this approach is that the optimization involves only the nominal system dynamics, avoiding the large increase in computational complexity associated with the inclusion of uncertainty in the optimization [24].

The outline of this chapter is as follows. Section 2.2 defines the problem statement for linear, time-invariant (LTI) subsystems with coupled constraints. In Section 2.3, the tube MPC method for robustness is reviewed, then compared with a related, constraint-tightening method. Section 2.4 develops the main result, a robust distributed MPC algorithm, by extending tube MPC to a distributed implementation where only one subsystem agent updates at each time step. The precise communication requirements are identified in Section 2.5. By comparing, in Section 2.5.1, the method to the constraint-tightening DMPC of Richards and How [38] it is seen that communication requirements are more flexible for this new method. Section 2.5.2 then investigates the trades between performance and communication, both for the new method and the constraint-tightening DMPC.

2.2 Problem statement

Consider a system of \( N_p \) linear time-invariant, discrete-time subsystems, the set of which is denoted \( \mathcal{P} = \{1, \ldots, N_p \} \), described by the state equations

\[
x_p(k + 1) = A_p x_p(k) + B_p u_p(k) + w_p(k), \forall p \in \mathcal{P}, k \in \mathbb{N},
\]

where \( x_p \in \mathbb{R}^{N_x,p} \), \( u_p \in \mathbb{R}^{N_u,p} \) and \( w_p \in \mathbb{R}^{N_w,p} \) are, respectively, the state vector, control input vector, and disturbance acting on subsystem \( p \). Assume that each system \( (A_p, B_p) \) is controllable, and that the complete states \( x_p \) are available at each sampling instant. The disturbances are unknown a priori, but are assumed to
lie in known independent, bounded, compact sets that contain the origin:

\[ w_p(k) \in \mathcal{W}_p \subseteq \mathbb{R}^{N_{x,p}}, \forall p \in \mathcal{P}, k \in \mathbb{N}. \]

Each subsystem is subject to local constraints on an output \( y_p(k) \in \mathbb{R}^{N_{y,p}} \):

\[
\forall p \in \mathcal{P}, k \in \mathbb{N} : y_p(k) = C_p x_p(k) + D_p u_p(k) \tag{2.2}
\]

\[
y_p(k) \in \mathcal{Y}_p \subset \mathbb{R}^{N_{y,p}}, \tag{2.3}
\]

where the set \( \mathcal{Y}_p \) is closed and contains the origin, and \( N_c \) coupling constraints exist across multiple subsystems. Each coupling constraint \( c \in \mathcal{C} = \{1, \ldots, N_c\} \) applies to coupling outputs \( z_{cp} \in \mathbb{R}^{N_{z,c}} \), the sum of which must lie in a closed set \( \mathcal{Z}_c \):

\[
\forall c \in \mathcal{C}, p \in \mathcal{P}, k \in \mathbb{N} : z_{cp}(k) = E_{cp} x_p(k) + F_{cp} u_p(k), \tag{2.4}
\]

\[
\sum_{p=1}^{N_p} z_{cp}(k) \in \mathcal{Z}_c \subset \mathbb{R}^{N_{z,c}}. \tag{2.5}
\]

The matrices \( C_p, D_p, E_{cp}, F_{cp} \) and the sets \( \mathcal{Y}_p, \mathcal{Z}_c \) are all chosen by the designer as part of the problem; this is a very general form of constraints, intended to allow coupling between any number of subsystems. The following definitions identify structure in the coupling, and are used later to determine the requirements for communication. Define \( \mathcal{P}_c \) as the set of all subsystems involved in constraint \( c \), and similarly let \( \mathcal{C}_p \) be the set of constraints involving subsystem \( p \):

\[
\mathcal{P}_c \triangleq \{ p \in \mathcal{P} : [E_{cp} F_{cp}] \neq 0 \}, \tag{2.6}
\]

\[
\mathcal{C}_p \triangleq \{ c \in \mathcal{C} : [E_{cp} F_{cp}] \neq 0 \}. \tag{2.7}
\]

Then the set of all other subsystems coupled to \( p \) is

\[
\mathcal{Q}_p = \left( \bigcup_{c \in \mathcal{C}_p} \mathcal{P}_c \right) \setminus \{ p \}. \tag{2.8}
\]
The system-wide objective is assumed to be decoupled, and is a summation of some function of the state and input, given by

$$\min \sum_{p=1}^{N_p} \sum_{k=0}^{\infty} l_p(x_p(k), u_p(k)), \quad (2.9)$$

where the stage cost $l_p : \mathbb{R}^{N_x,p} \times \mathbb{R}^{N_u,p} \to \mathbb{R}_{0+}$.

Such a problem statement captures a broad range of problems, including steering a team of vehicles to their respective targets, while avoiding collision and satisfying the kinematic and dynamic constraints. It is noted here that the decoupled objective assumption is not necessary for robust feasibility of the method, but is merely adopted to follow the convention of coupling and interactions between subsystems being in the constraints.

### 2.3 Tube model predictive control

This section forms a revision of what shall be termed tube MPC, first proposed by Mayne et al. [27]. The approach guarantees robust feasibility and stability for a controlled system, under certain well-known assumptions [7]; here, however, the terminal set assumptions are relaxed to widen the choice of available terminal controllers, a variety of which will be used in this thesis. The tube method forms the basis for the development of the distributed MPC algorithm in Section 2.4, and this preceding section introduces a number of key concepts and results. In addition, tube MPC is formally compared with a related method, the constraint-tightening method for robustness [11, 12, 14, 26].

The following standing assumption is required.

**Assumption 2.1 (Robust positively-invariant set).** There exists a stabilizing controller $K_p$ for each subsystem $(A_p, B_p)$ and also a corresponding robust positively-invariant set $S_p$.
invariant (RPI) set $\mathcal{R}_p$, satisfying

$$(A_p + B_p K_p)x_p + w_p \in \mathcal{R}_p, \forall x_p \in \mathcal{R}_p, w_p \in \mathcal{W}_p,$$

$$(C_p + D_p K_p)\mathcal{R}_p \subseteq \mathcal{Y}_p,$$

$$\bigoplus_{p=1}^{N_p}(E_{cp} + F_{cp} K_p)\mathcal{R}_p \subseteq \mathcal{Z}_c, \forall c \in \mathcal{C}. \tag{2.10}$$

Note that this assumption places an implicit restriction on the ‘size’ of the maximum disturbance set, seen by the latter two conditions. (A mapped RPI set must not be larger than any output constraint set). Thus, the severity of the disturbances that can be handled is relative to the output set sizes. The linear control matrix $K_p$ is known as the tube controller. Remark 2.1 discusses suitable choices and practical considerations for both $K_p$ and $\mathcal{R}_p$.

Tube MPC [27] uses a nominal model of the system dynamics,

$$\dot{x}_p(k+1) = A_p \dot{x}_p(k) + B_p \bar{u}_p(k), \forall p \in \mathcal{P}, k \in \mathbb{N}, \tag{2.11}$$

to design a sequence of disturbance-invariant state sets, centered around a nominal trajectory, for a horizon of $N$ steps. The decision variable includes the sequence of controls over the prediction horizon and, uniquely, the nominal initial state, and is defined as

$$U_p(k) \triangleq \{ \bar{x}_p(k|k), \bar{u}_p(k|k), \ldots, \bar{u}_p(k + N - 1|k) \}, \forall p \in \mathcal{P}, \tag{2.12}$$

where the notation $(k+j|k)$ denotes a prediction for $j$ steps into the future from time $k$. As the optimization involves only nominal terms, complexity is comparable to standard MPC, and robustness to disturbance is guaranteed by use of a feedback law to keep the state around the tube centre, the nominal state. Of course, as is inherent in all MPC approaches [7], the quality of predictions depends on the quality of the model, and model-system mismatch is not just possible, but likely. A simple, though
conservative, method for dealing with such parameter uncertainty is to convert the model uncertainty to an equivalent additive disturbance. In Langson et al. [78], the model mismatch is explicitly considered for the tube MPC method. In this thesis, it will be assumed, as is commonly done, that the model \((A_p, B_p)\) is accurate, and that uncertainty manifests itself only through the additive disturbances.

The *centralized* optimal control problem at a state \(x(k) = \{x_1(k), \ldots, x_{N_p}(k)\}\) is \(\mathcal{P}^C(x_1(k), \ldots, x_{N_p}(k))\), defined as

\[
V^\text{opt}(x_1(k), \ldots, x_{N_p}(k)) = \min_{\{U_1(k), \ldots, U_{N_p}(k)\}} \sum_{p=1}^{N_p} J_p(U_p(k)) \quad (2.13)
\]

subject to \(\forall p \in \mathcal{P}, \forall j \in \{0, \ldots, N - 1\} : \)

\[
\begin{align*}
\tilde{x}_p(k + j + 1|k) &= A_p \tilde{x}_p(k + j|k) + B_p \tilde{u}_p(k + j|k), \\
x_p(k) - \tilde{x}_p(k|k) &\in \mathcal{R}_p, \\
\tilde{x}_p(k + N|k) &\in \mathcal{X}_{F_p}, \\
\bar{y}_p(k + j|k) &= C_p \tilde{x}_p(k + j|k) + D_p \tilde{u}_p(k + j|k) \\
\bar{y}_p(k + j|k) &\in \mathcal{Y}_p, \\
\forall c \in \mathcal{C} : & \quad \tilde{z}_{cp}(k + j|k) = E_{cp} \tilde{x}_p(k + j|k) + F_{cp} \tilde{u}_p(k + j|k), \\
& \sum_{p=1}^{N_p} \tilde{z}_{cp}(k + j|k) \in \mathcal{Z}_c,
\end{align*}
\]

where the cost function is a finite-horizon approximation to (2.9), involving the nominal states and inputs:

\[
J_p(U_p(k)) \triangleq F_p(\tilde{x}_p(k + N|k)) + \sum_{j=0}^{N-1} l_p(\tilde{x}_p(k + j|k), \tilde{u}_p(k + j|k)). \quad (2.15)
\]

The terminal cost \(F_p : \mathbb{R}^{Nz_p} \mapsto \mathbb{R}_{0+}\), is some cost-to-go beyond the end of the horizon. The sets \(\mathcal{Y}_p, \mathcal{Z}_c\) represent the sets \(\mathcal{Y}_p, \mathcal{Z}_c\) tightened by margins to allow for
Algorithm 2.1: Centralized MPC using tubes

1 Design stabilizing controller $K_p$ and RPI set $R_p$ for each $p \in \mathcal{P}$;
2 Tighten sets $\mathcal{Y}_p, \mathcal{Z}_c, \forall p \in \mathcal{P}, c \in C$, via (2.16) and design terminal sets $X_{F_p}$;
3 for $k = 0 : \infty$ do
4 Sample current states $\{x_1(k), \ldots, x_{N_p}(k)\}$;
5 Solve $P_C(x_1(k), \ldots, x_{N_p}(k))$ for $U_p^{opt}(k), \forall p \in \mathcal{P}$;
6 Apply controls (2.17);
7 Wait one time step;
8 end

uncertainty:

\[
\begin{align*}
\dot{\mathcal{Y}}_p &= \mathcal{Y}_p \sim (C_p + D_p K_p) R_p, \quad (2.16a) \\
\dot{\mathcal{Z}}_c &= \mathcal{Z}_c \sim \bigoplus_{p=1}^{N_p} (E_{cp} + F_{cp} K_p) R_p. \quad (2.16b)
\end{align*}
\]

These sets are non-empty by the assumption of (2.10). The sets $X_{F_p}$ are terminal sets; assumptions on these will be introduced shortly. The sets $R_p$ are cross-sections of the tubes and are robust invariant sets, as in (2.10). Then the tubes themselves are given by $\{\hat{x}_p(k|k) \oplus R_p, \hat{x}_p(k + 1|k) \oplus R_p, \ldots, \hat{x}_p(k + N|k) \oplus R_p\}$.

After the optimization is solved at each time step, the following control is to be applied to each subsystem $p \in \mathcal{P}$

\[
u_p(k) = u_p^{opt}(k|k) + K_p (x_p(k) - \hat{x}_p^{opt}(k|k)), \quad (2.17)
\]

Made up of two parts, the feedforward term $u_p^{opt}(k|k)$ moves the state ‘along’ the tube, while the feedback term uses the tube controller $K_p$ to keep the state ‘within’ the tube.

The problem $P_C(x_1(k), \ldots, x_{N_p}(k))$ is employed in Algorithm 2.1, a centralized implementation of tube MPC.
2.3.1 Feasibility of tube MPC

The first results concern feasibility of the tube MPC algorithm, which is guaranteed regardless of objective. Assumptions on the terminal set are now introduced, based on the existence of a set that is, firstly, invariant, and secondly, admissible—a standard notion of an invariant terminal set \cite{7}.

**Assumption 2.2** (Control invariant terminal set). There exist terminal sets \( X_{F_p} \), and terminal control laws \( u_p = \kappa_{F_p}(x_p), \forall p \in P \), so that for all \( x_p \in X_{F_p}, A_p x_p + B_p \kappa_{F_p}(x_p) \in X_{F_p}, \forall p \in P \).

**Assumption 2.3** (Constraint satisfaction in terminal set). For all \( x_p \in X_{F_p}, \forall p \in P \),

\[
C_p x_p + D_p \kappa_{F_p}(x_p) \in \tilde{y}_p,
\]

\[
\sum_{p=1}^{N_p} E_{cp} x_p + F_{cp} \kappa_{F_p}(x_p) \in \tilde{z}_c, \forall c \in C.
\]

Note that this is merely a requirement for nominal control invariance of the terminal set; many other robust MPC methods, e.g., Chisci et al. \cite{12} and Richards and How \cite{14}, require robust control invariance unless certain assumptions are met. This permits a wider range of terminal sets to be employed. Furthermore, this is a relaxation of the assumption of Mayne et al. \cite{27}, in which it is implicitly assumed that \( \kappa_{F_p}(x_p) = K_p x_p \), i.e., the terminal control law is the linear tube controller, which may be restrictive (see Remark 2.1).

If these assumptions, together with Assumption 2.1, hold, then the following robust feasibility result holds for the system controlled by Algorithm 2.1.

**Proposition 2.1** (Robust feasibility of tube MPC). Suppose the sequence of controls \( U^*_p(k_0) = \{ \tilde{x}^*_p(k_0|k_0), \tilde{u}^*_p(k_0|k_0), \ldots, \tilde{u}^*_p(k_0 + N - 1|k_0) \}, \forall p \in P \), exists and is a feasible (but not necessarily optimal) solution to \( P^C(x_1(k_0), \ldots, x_{N_p}(k_0)) \) at some
time step \( k_0 \). Then, for all \( x_p(k_0 + 1) \in A_p x_p(k_0) + B_p u_p(k_0) \oplus W_p, \forall p \in \mathcal{P} \), where 
\[ u_p(k_0) = \bar{u}_p^*(k_0|k_0) + K_p (x_p(k_0) - \bar{x}_p^*(k_0|k_0)), \]
(i) the candidate sequence \( \bar{U}_p(k_0 + 1) \), defined as 
\begin{align*}
\bar{\bar{U}}_p(k_0 + 1) & \triangleq \\
\{ \bar{x}_p^*(k_0 + 1|k_0), \bar{u}_p^*(k_0 + 1|k_0), \ldots, \bar{u}_p^*(k_0 + N - 1|k_0), \kappa_{F_p}(\bar{x}_p^*(k_0 + N|k_0)) \}, \tag{2.19}
\end{align*}
is a feasible solution to \( P^C(x_1(k_0 + 1), \ldots, x_N(k_0 + 1)) \), and (ii) the system controlled by Algorithm 2.1 is robustly-feasible.

Proof. Part (i) follows directly from Mayne et al. [27, Proposition 3], but with the linear terminal control replaced by a general control law, admissibility of which is established by Assumption 2.3. For (ii), robust feasibility of the closed-loop system follows from recursion applied to part (i); given an initial feasible solution at time 0, a feasible solution is guaranteed to exist at every subsequent time step, for any sequence of disturbances \( \{w_p(k)\}_{k}, \forall w_p \in \mathcal{W}_p, p \in \mathcal{P} \).

Robust feasibility holds irrespective of the objective function. Therefore, this robustly-feasible tube MPC applies to a wide class of problems, including coupled objective forms and, for example, objectives for planning in an unknown environment [49]. In the next subsection, stronger stability results are presented, based on those of Mayne et al. [27], which require further assumptions on the form of the objective and the terminal set and control law.

Remark 2.1 (Choice of RPI set \( \mathcal{R}_p \) and tube controller \( K_p \)). As noted by Mayne et al. [27], it is generally desirable that \( \mathcal{R}_p \), which shall form the cross-section of the tube, and thus the bound for the uncertain state evolution, is as small as possible, to reduce conservativeness. The minimal robust positively-invariant (mRPI) set [76] is given by 
\[ \mathcal{R}_{p}^{\text{min}} = \bigoplus_{i=0}^{\infty} (A_p + B_p K_p)^i \mathcal{W}_p, \tag{2.20} \]
where the summation is based on the Minkowski sum [76]. Unfortunately, this set is not necessarily polytopic unless either \( K_p \) is the nilpotent controller [10] or \((A_p + B_pK_p)^s = \alpha I\) for some finite integer \( s \) and \( \alpha \in (0, 1) \) [76]. Nevertheless, it is possible to compute an outer, polytopic approximation—which itself is RPI—to this set [79], which widens the range of systems and feedback gains for which tube MPC may be employed.

Where the tube controller is chosen to be nilpotent, the mRPI set is, as just stated, finitely-determined, i.e.,

\[
R_p^{\text{min}} = \bigoplus_{i=0}^{N_{x,p}} (A_p + B_pK_p)^i W_p,
\]

where \( N_{x,p} \) is the order of the subsystem \( p \), and the number of steps in which the controller steers the subsystem \( x_p(k + 1) = (A_p + B_pK_p)x_p(k) \) to the origin. Such a controller results from placement of closed-loop poles at the origin, and always exists if \((A_p, B_p)\) is controllable.

**Remark 2.2 (Choice of terminal set \( \mathcal{X}_{F_p} \) and terminal control law \( \kappa_{F_p} \)).** The requirements on the terminal set amount to nominal invariance and admissibility under a terminal control law \( \kappa_{F_p}(x_p) \), i.e., \( A_p x_p + B_p \kappa_{F_p}(x_p) \in \mathcal{X}_{F_p}, \forall x_p \in \mathcal{X}_{F_p} \). This is considerably less restrictive than for many other robust MPC methods [24, 80], including the constraint-tightening method [11, 12, 14, 26], in which robust invariance is required unless certain conditions, for example nilpotency of the feedback (tube controller) \( K_p \), are met.

Nominal invariance means much more flexibility for the designer to specify a constraint that suits the problem statement; moreover, many nominal invariant admissible sets may exist where robust invariant sets do not, or are hard to compute. For example, a point, a limit cycle, or another set with no ‘volume’ may be employed. Alternatively, a large terminal set may be desired to enlarge the feasible region and/or minimize the number of prediction steps for the MPC [7]. In such cases, the *maximal control invariant set* [81], \( C^\infty \), is the largest invariant admissible set for any
Figure 2.1: Comparison of tube and terminal sets for $K_p^{LQR}$ and $K_p^{NP}$, for a double-integrator subsystem with constraints $|u_p| \leq 1$, $|0 1|x_p| \leq 2$ and disturbance set $W_p = \{w_p : ||w_p||_\infty \leq 0.1\}$. Note the different scales.

General, non-linear control law $\kappa_{F_p}$. Alternatively, the maximal output-admissible invariant set $O^\infty$, is the corresponding largest set under a pre-specified, linear terminal control law $K_{F_p}$.

It was stated earlier that in Mayne et al. [27], the tube controller and terminal controllers are assumed to be equal. Such a requirement is restrictive and may be contrary to design requirements. For example, suppose the stage cost is the quadratic form $x_p^T Q_p x_p + u_p^T R_p u_p$. A possible choice of terminal controller is the optimal, unconstrained LQR controller, $K_p^{LQR}$, associated with $(A_p, B_p, Q_p, R_p)$. Figure 2.1 shows the sets $R_p^{\text{min}}$ and $O_p^{\infty}$ for an example constrained double-integrator system, with $Q_p = I_2$ and $R_p = 10$, using both $K_p^{LQR}$ and the nilpotent controller, $K_p^{NP}$. Given a constant number of prediction steps, a large terminal set can maximize the feasible region of operation; furthermore, because constraints are tightened by an amount depending on the size of $R_p$, a small tube cross-section can minimize the subsequent contraction of the feasible set. Yet, use of $K_p = K_p^{NP}$ and $\kappa_{F_p}(x_p) = K_p^{NP} x_p$, while delivering a minimal RPI set for the tube, would necessarily limit the size of the terminal set. Conversely, $K_p = K_p^{LQR}$ and $\kappa_{F_p}(x_p) = K_p^{LQR} x_p$ permits a larger terminal set, yet the larger RPI tube set leads to extra conservatism and
tighter constraints. Therefore, it may be more desirable, for example, to choose $K_p = K_p^{NP}$ and $\kappa_{F_p}(x_p) = K_p^{LR} x_p$ to minimize conservativeness but enlarge the terminal set.

### 2.3.2 Stability of tube MPC

In this section, robust stability results are presented. Firstly, the following further assumption on the terminal set and cost is required for monotonocity of the value function.

**Assumption 2.4 (Terminal Cost is Local Lyapunov Function).** For all $x_p \in X_{F_p}$ and $p \in \mathcal{P}$,

$$F_p(A_p x_p + B_p \kappa_{F_p}(x_p)) - F_p(x_p) \leq -l_p(x_p, \kappa_{F_p}(x_p)).$$

Together with Assumptions 2.2 and 2.3, these assumptions represent a specific case of the standard assumptions A1–A4 in Mayne et al. [7] or equivalently A1 and A2 in Mayne et al. [27]. Then the following result holds, based on that of Mayne et al. [27].

**Proposition 2.2 (Monotonicity of the cost).** Suppose the sequence of controls $U_p^*(k_0) = \{\bar{x}_p(k_0|k_0), \bar{u}_p(k_0|k_0), \ldots, \bar{u}_p(k_0 + N - 1|k_0)\}, \forall p \in \mathcal{P},$ exists and is a feasible (but not necessarily optimal) solution to $\mathcal{P}^C(x_1(k_0), \ldots, x_{N_p}(k_0))$ at some time step $k_0$. Then, for all $x_p(k_0 + 1) \in A_p x_p(k_0) + B_p u_p(k_0) \oplus W_p, \forall p \in \mathcal{P},$ where $u_p(k_0) = \bar{u}_p(k_0|k_0) + K_p(x_p(k_0) - \bar{x}_p(k_0|k_0)),$ the upper bound on the cost value decreases monotonically:

$$V^*(x_1(k_0 + 1), \ldots, x_{N_p}(k_0 + 1)) \leq V^*(x_1(k_0), \ldots, x_{N_p}(k_0)) - \sum_{p=1}^{N_p} l_p(\bar{x}_p(k_0|k_0), \bar{u}_p(k_0|k_0)),$$

where $V^*(x_1(k_0), \ldots, x_{N_p}(k_0)) = \sum_{p=1}^{N_p} J_p(U_p^*(k_0))$. 

26
Proof. Directly from Mayne et al. [27, Proposition 3].

The nature of the stability of the system controlled by tube MPC then varies according to the form of the objective (2.15), and further assumptions, if any, on the terminal set and control law. Firstly, asymptotic stability [82] of the RPI set \( R_p \) follows under mild assumptions on the stage cost.

**Theorem 2.1 (Robust asymptotic stability of \( R_p \)).** Suppose \( l_p(x_p, u_p) \geq c \| (x_p, u_p) \| \) for some \( c > 0 \). Then the set \( R_p \) is robust asymptotically stable for the controlled system \( x_p(k+1) = A_p x_p(k) + B_p u_p(k) + w_p(k), \forall p \in \mathcal{P} \), where \( w_p(k) \in \mathcal{W}_p, \forall k \).

**Proof.** Monotonicity of the cost was established in Proposition 2.2. By recursion, because \( V^*(k+1) - V^*(k) \leq - \sum_{p=1}^{N_p} l_p(\bar{x}_p^*(k|k), \bar{u}_p^*(k|k)) \), yet \( V^*(\cdot) \) and the stage cost \( l_p(\cdot, \cdot) \) are both strictly non-negative, it follows that \( V^*(k+1) - V^*(k) \to 0 \) as \( k \to \infty \). In turn, this implies that \( \sum_{p=1}^{N_p} l_p(\bar{x}_p^*(k|k), \bar{u}_p^*(k|k)) \to 0 \); again, as \( l_p(\cdot, \cdot) \geq 0 \), this further implies each \( l_p(\bar{x}_p^*(k|k), \bar{u}_p^*(k|k)) \to 0 \). Because \( l_p(x_p, u_p) \geq c \| (x_p, u_p) \| \) for some \( c > 0 \), and \( l_p(0,0) = 0 \), it must be that the nominal state \( \bar{x}_p^*(k|k) \to 0 \) and the nominal control \( \bar{u}_p^* \to 0 \). Finally, by the fact that \( x_p(k) \in \bar{x}_p(k|k) \subseteq R_p, \forall k \), it follows that the true state \( x_p(k) \to R_p \) as \( k \to \infty \), and, furthermore,

\[
\begin{align*}
  u_p(k) &= \bar{u}_p^*(k|k) + K_p (x_p(k) - \bar{x}_p^*(k|k)) \\
  &= K_p x_p(k)
\end{align*}
\]

as \( k \to \infty \). \( \square \)

If, furthermore, the stage cost is quadratic, then the stronger result of exponential stability [82] holds.

**Theorem 2.2 (Robust exponential stability of \( R_p \)).** Suppose \( l_p(x_p, u_p) \triangleq x_p^T Q_p x_p + u_p^T R_p u_p \) and \( F_p(x_p) \triangleq x_p^T P_p x_p \), where \( Q_p, R_p, P_p \) are positive-definite. Then the set \( R_p \) is robust exponentially stable for the controlled system \( x_p(k+1) = A_p x_p(k) + B_p u_p(k) + w_p(k), \forall p \in \mathcal{P} \), where \( w_p(k) \in \mathcal{W}_p, \forall k \).
Proof. Directly from Mayne et al. [27, Theorem 1].

Remark 2.3 (Alternative stability results). Without loss of generality, the case of steering the nominal system to the origin has been considered, and, correspondingly, the perturbed system to some RPI set $\mathcal{R}_p$. Note that the resulting closed-loop system is therefore controlled according to a dual-mode control law [7, 12].

$$u_p(k) = \begin{cases} 
  c_p(k) + K_p x_p(k), & x_p \notin \mathcal{R}_p \\
  K_p x_p(k), & x_p \in \mathcal{R}_p,
\end{cases}$$

where $c_p(k) = \tilde{u}_p(k|k) - K_p \tilde{x}_p(k|k)$ is chosen by the optimization. However, other types of stability or convergence may be achieved by use of suitable objective functions. For example, by a simple change of coordinates, the developed results also apply to the case of steering the system states to the RPI set around some setpoint state [7].

Also of interest in the literature is the case of steering some tracking output to a compact target set [14].

$$s_p(k) = G_p x_p(k) + H_p u_p(k),$$

$$s_p(k) \xrightarrow{k \to \infty} \mathcal{T}_p \subset \mathbb{R}^{N_p}.$$ Define $l_p(x_p, u_p) \geq d(s_p, T_p)$, where

$$d(a, B) \triangleq \inf_{b \in B} ||a - b||,$$

and the terminal cost $F_p \triangleq 0$. Then, provided the following criterion holds, (an augmentation of Assumption 2.3),

$$G_p x_p + H_p \kappa_F(x_p) \in \tilde{T}_p, \forall x_p \in \mathcal{X}_{F_p}, p \in \mathcal{P},$$
where $\bar{T}_p = T_p \sim (G_p + H_p K_p) R_p \neq \emptyset$, the monotonicity of the cost in Proposition 2.2 is maintained, and $s_p(k) \to T_p$ as $k \to \infty$.

The tracking output may be defined to include any linear combination of state and input, depending on the desired objective. For example, the combination

$$G_p = -K_p, \quad H_p = I, \quad T_p = 0 \quad (2.21)$$

results in the stage cost

$$l_p(\bar{x}_p, \bar{u}_p) = \|\bar{u}_p - K_p \bar{x}_p\|,$$

similar to that employed in Chisci et al. [12], Kerrigan and Maciejowski [80], which—together with the terminal control law being equal to the tube controller, i.e., setting $\kappa_{F_p}(x_p) = K_p x_p$, and a suitable choice of terminal set—again results in convergence of the controls to $u_p = K_p x_p$.

**Remark 2.4 (Implementation of tube MPC for polytopic sets).** If the RPI sets $R_p$, which form the cross-sections of the tubes, are polytopic, and the constraint sets $\mathcal{Y}_p, \mathcal{Z}_c$ are polyhedral, it follows that the tightened sets $\tilde{\mathcal{Y}}_p, \tilde{\mathcal{Z}}_c$ (2.16) are polyhedral also [25]. MATLAB® toolboxes are available for computing the necessary Pontryagin differences, given by (2.16). For example, the Invariant Set Toolbox [83] or the Multi-Parametric Toolbox [84]. The resulting optimization problem may then be a Linear Program (LP) or Quadratic Program (QP), by suitable choice of cost function. For example, consider the tracking output objective of the previous remark, with polyhedral target set

$$T_p = \left\{ t_p : P_p t_p \leq q_p \right\},$$

where $s_p \in \mathbb{R}^{N_x p}, P_p \in \mathbb{R}^{m \times N_x p}, q_p \in \mathbb{R}^m$. The objective function

$$\sum_{j=0}^{N-1} d(\tilde{s}_p(k + j|k), \bar{T}_p) = \sum_{j=0}^{N-1} \inf_{t_p(j) \in T_p} \|\tilde{s}_p(k + j|k) - t_p(j)\|$$
is equivalent to the minimization problem

\[
\min_{\{a_p(0), a_p(1), \ldots \}} \sum_{j=0}^{N-1} a_p(j) \\
\text{s.t. } P_p s_p(k + j|k) \leq q_p + \lambda a_p(j), \\
\quad a_p(j) \geq 0, \\
\forall j \in \{0, \ldots, N - 1\},
\]

or,

\[
\min_{\{a_p(0), a_p(1), \ldots \}} \sum_{j=0}^{N-1} \sum_{i=1}^{m} a_{p,i}(j) \\
\text{s.t. } P_p s_p(k + j|k) \leq q_p + \Lambda a_{p,i}(j), \\
\quad a_{p,i}(j) \geq 0, \\
\forall i \in \{1, \ldots, m\}, \\
\forall j \in \{0, \ldots, N - 1\},
\]

where \(\lambda \in \mathbb{R}_+^m\), \(\Lambda = \text{diag}(\lambda)\) are, respectively, a positive weighting vector and corresponding diagonal matrix, whose elements change the emphasis on each row of the target set constraint, for the purpose of tuning. Insertion of such a cost in the MPC optimization results in an LP.

**Remark 2.5 (Non-convex constraint sets and MIP).** In many applications, the system constraints may be defined by non-convex sets, and so it is necessary to consider how the tightening of such constraints is done in accordance with (2.16). By Kerrigan [81], the Pontryagin difference for a non-convex set \(A\) is

\[A \sim C = (A^c \oplus (-C))^c,\]

where \(A^c\) denotes the complement of \(A\) and \(0 \in C\). Richards [85] applies this result to the case of a set exclusion constraint, i.e., \(a \in A \setminus B\), where \(A \setminus B \triangleq \{\alpha \in A : \alpha \notin B\}\),
Figure 2.2: Non-convex set \((A \backslash B)\) tightened by \(C\) (indicated). The excluded set \(B\) becomes enlarged. Dashed lines correspond to sets prior to tightening.

to give

\[(A \backslash B) \sim C = (A \sim C) \setminus (B \oplus (-C)).\]

Examples of such constraints include obstacle avoidance for vehicles: a vehicle’s position is permitted to lie in \(A\) but must avoid obstacle \(B\). The set tightening for the uncertainty set \(C\) then is effectively an enlargement (dilation) of the obstacle \(B\). If \(B\) is polyhedral, such operations are easily computed using the aforementioned MATLAB\textsuperscript{®} toolboxes [83, 84]; an example is provided in Figure 2.2, computed using the Invariant Set Toolbox [83]. Note that in the case of irregular edge effects arising from the dilation, a simpler, outer-approximation may be used [85].

Of course, such non-convex constraints lead to a non-convex optimization problem. These are notoriously difficult to solve (NP-hard) [86], as multiple feasible regions may exist, with multiple local optima in any region. Fortunately, however, methods exist for dealing with these non-convex constraints, such as mixed integer programming (MIP), and, in particular, its linear (MILP) and quadratic (MIQP) variants [87, 88]. For the non-convex set \((A \backslash B)\), supposing \(B\) is polyhedral, i.e., \(B = \{z : Pz \leq q\}\), where \(z \in \mathbb{R}^n, P \in \mathbb{R}^{m \times n}, q \in \mathbb{R}^m\), exclusion from this set is
given by the \( m \) logical-OR constraints:

\[
P_{1,1}z_1 + P_{1,2}z_2 + \ldots + P_{1,n}z_n \geq q_1,
\]

or \( P_{2,1}z_1 + P_{2,2}z_2 + \ldots + P_{2,n}z_n \geq q_2, \)

\vdots

or \( P_{m,1}z_1 + P_{m,2}z_2 + \ldots + P_{m,n}z_n \geq q_m. \)

The MIP technique associates a binary variable \( b_i \in \{0,1\} \) with each constraint \( i \in \{1, \ldots, m\} \) in the optimization, to act as a switch for that constraint:

\[
P_{1,1}z_1 + P_{1,2}z_2 + \ldots + P_{1,n}z_n \geq q_1 - Mb_1,
\]

\[
P_{2,1}z_1 + P_{2,2}z_2 + \ldots + P_{2,n}z_n \geq q_2 - Mb_2,
\]

\vdots

\[
P_{m,1}z_1 + P_{m,2}z_2 + \ldots + P_{m,n}z_n \geq q_m - Mb_m,
\]

and,

\[
\sum_{i=1}^{m} b_i \leq m - 1,
\]

where \( M \) is a sufficiently large number such that, if \( b_i = 1 \), the \( i \)th constraint is always satisfied [87]. The result is that the logical-OR constraints are converted to the standard logical-AND constraints necessary for implementation. This formulation is used in the multiple-vehicle numerical examples later in this thesis.

A variety of MILP or MIQP solvers are available either freely or commercially, such as CPLEX® [89] or GLPK [90]. Despite this, MIP problems still reside in the class of NP-complete problems [88], meaning an algorithm whose solution time is polynomial in the problem size is unlikely to exist [91]. Thus, it is in the interest of shorter computation times that the number of binary variable constraints is kept small.
2.3.3 Relationship between tube MPC and constraint-tightening MPC

The tube MPC method exhibits similarities to the constraint-tightening (CT-MPC) method for robustness, first proposed by Gossner et al. [11] and later generalized [12, 14, 26]. In that method, the feedback controller is permitted to vary over the prediction horizon, i.e., $K = K(j)$. The result is that constraints are tightened non-uniformly, albeit monotonically, over the horizon, as opposed to the uniform tightening of tube MPC. Specifically,

$$\hat{Y}(0) = Y, \quad \hat{Y}(j + 1) = \hat{Y}(j) - (C + DK(j))L(j)W,$$

where,

$$L(0) = I,$$

$$L(j + 1) = (A + BK(j))L(j).$$

The initial and terminal conditions are

$$\bar{x}(k|k) = x(k),$$

$$\bar{x}(k + N|k) \in \mathcal{X}_F,$$

where $\mathcal{X}_F = \mathcal{R}^c_t \sim L(N - 1)\mathcal{W}$. The set $\mathcal{R}^c_t$ is an RPI set that satisfies,

$$\forall x \in \mathcal{R}^c_t : Ax + B\kappa_F(x) + L(N - 1)w \in \mathcal{R}^c_t, \forall w \in \mathcal{W}$$

$$Cx + D\kappa_F(x) \in \hat{Y}(N - 1).$$

Thus, the terminal set may be required to be robustly-invariant, depending on the choice of $K(j)$, as opposed to the nominal invariance requirement for tube MPC.

In a later section, the distributed forms of CT-MPC and tube MPC are compared, both by an analytical treatment of the communication requirements and by a
numerical demonstration of the performance–communication trades. As a precursor to that, we compare the centralized forms of each approach. It will be shown that the CT-MPC method subsumes tube MPC; hence, the latter is a more conservative method. To facilitate this, the controls applied by both methods may be written as disturbance-, rather than state-, feedback policies [15, 55]. A disturbance-feedback policy has the form,

\[ v = \chi + M\omega \]

where,

\[ v = \begin{bmatrix} u(k)^T & u(k+1)^T & \ldots & u(k+N-1)^T \end{bmatrix}^T, \]

is the stacked vector of controls applied, made up of feedforward and feedback components. The feedforward components,

\[ \chi = \begin{bmatrix} c(k|k)^T & c(k+1|k)^T & \ldots & c(k+N-1|k)^T \end{bmatrix}^T, \]

are deviations—usually optimized online—about a feedback control formed from a strictly lower-triangular feedback matrix

\[
M = \begin{bmatrix}
0 & 0 & \ldots & \ldots & 0 \\
M_{10} & 0 & \ldots & \ldots & 0 \\
M_{20} & M_{21} & \ldots & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
M_{(N-1)0} & M_{(N-1)1} & \ldots & M_{(N-1)(N-2)} & 0
\end{bmatrix}, \quad (2.26)
\]

and the stacked vector of disturbances,

\[ \omega = \begin{bmatrix} w(k)^T & w(k+1)^T & \ldots & w(k+N-1)^T \end{bmatrix}^T, \]
such that the control applied at time \( k + j \) is dependent on the \( j - 1 \) previous disturbances, and given by

\[
u(k + j) = c(k + j|k) + M_{j0}w(k) + M_{j1}w(k + 1) + \ldots + M_{j(j-1)}w(k + j - 1).
\]

In its most general form, the elements of the matrix \( M \) are themselves decision variables to be optimized [15]. In tube- and CT-MPC, those matrices are predetermined offline.

**CT-MPC:** The feedback law may vary over the horizon, and the feedforward controls are determined directly in the optimization, subject to tightened constraints.

\[
M^{\text{ct}} = \begin{bmatrix}
0 & \ldots & \ldots & 0 \\
K(j)L(j) & 0 & \ldots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
K(N-2)L(N-2) & \ldots & K(0)L(0) & 0
\end{bmatrix},
(2.27a)
\]

\[
\chi^{\text{ct}} = \begin{bmatrix}
\tilde{u}^{\text{ct}}(k|k)^T \\
\tilde{u}^{\text{ct}}(k + 1|k)^T \\
\vdots \\
\tilde{u}^{\text{ct}}(k + N-1|k)^T
\end{bmatrix}^T,
(2.27b)
\]

i.e., \( c^{\text{ct}}(k + j|k) = \tilde{u}^{\text{ct}}(k + j|k), \forall j \), and where \( L \) is the state transition matrix previously defined.

**Tube-MPC:** The feedback law is static,

\[
M^{\text{tb}} = \begin{bmatrix}
0 & \ldots & \ldots & 0 \\
KL^j & 0 & \ldots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
KL^{N-2} & \ldots & KL^0 & 0
\end{bmatrix},
(2.28a)
\]

\[
\chi^{\text{tb}} = \begin{bmatrix}
c^{\text{tb}}(k|k)^T \\
c^{\text{tb}}(k + 1|k)^T \\
\vdots \\
c^{\text{tb}}(k + N-1|k)^T
\end{bmatrix}^T,
(2.28b)
\]
where now \( L = (A + BK) \), i.e., is constant, and again feedforward controls are determined in the optimization, subject to now uniformly-tightened constraints. What is unusual is that the feedforward vector \( \chi^{tb} \) itself consists of two parts, owing to the unique initial constraint in the tube MPC optimization problem:

\[
\mathbf{c}^{tb}(k + j|k) = \mathbf{\bar{u}}^{tb}(k + j|k) + KL^j(x(k) - \mathbf{x}^{tb}(k|k)),
\]

where \( \mathbf{\bar{u}}^{tb}(k + j|k) \) is a decision variable in the optimization, and \( x(k) - \mathbf{x}^{tb}(k|k) \in \mathcal{R} \).

It may then be shown that the set of policies generated by tube MPC is a subset of those of static-feedback CT-MPC; that is, for any tube controller \( \mathbf{K} \) and state \( x \), CT-MPC can produce every control policy that tube MPC can produce.

**Proposition 2.3.** The set of \( \mathbf{u} \) from (2.28) is a subset of that from (2.27) with \( \mathbf{K}(j) = \mathbf{K}, \forall j \).

**Proof.** Consider first the feedforward terms \( \mathbf{c}^{ct}(k + j|k) \) and \( \mathbf{c}^{tb}(k + j|k) \). Suppose the input constraint set is \( \mathcal{U} \). For CT-MPC, the tightened set at prediction step \( j \) is given by the recursion

\[
\tilde{\mathcal{U}}^{ct}(0) = \mathcal{U}, \\
\tilde{\mathcal{U}}^{ct}(j + 1) = \tilde{\mathcal{U}}^{ct}(j) + \mathbf{K}(j)L(j)\mathcal{W}, \\
\quad \forall j \in \{0, \ldots, N-2\},
\]

so that

\[
\tilde{\mathcal{U}}^{ct}(j) = \mathcal{U} \sim \mathbf{K} \bigoplus_{i=0}^{j-1} L^i\mathcal{W}, \forall j \in \{0, \ldots, N-1\},
\]

assuming static feedback \( \mathbf{K}(j) = \mathbf{K} \). The predicted input for step \( k+j \) belongs to this set, i.e., \( \mathbf{c}^{ct}(k + j|k) = \mathbf{\bar{u}}^{ct}(k + j|k) \in \tilde{\mathcal{U}}^{ct}(j) \). Now consider the tube MPC predictions. The corresponding prediction, \( \mathbf{c}^{tb}(k + j|k) \), consists of two parts: \( \mathbf{\bar{u}}^{tb}(k + j|k) \) and \( KL^j(x(k) - \mathbf{x}^{tb}(k|k)) \). The former belongs to the set \( \tilde{\mathcal{U}}^{tb} \triangleq \mathcal{U} \sim \mathbf{K}\mathcal{R} \). For the latter, \( x(k) - \mathbf{x}^{tb}(k|k) \in \mathcal{R} \) from the initial constraint. Thus, \( \mathbf{c}^{tb}(k + j|k) \in \tilde{\mathcal{U}}^{tb} \oplus KL^j\mathcal{R} \).
Furthermore, $\mathcal{R} = \sum_{i=0}^{\infty} L^i \mathcal{W}$. Therefore,

$$
\tilde{U}^{tb} \oplus KL^j \mathcal{R} = (U \sim KR) \oplus KL^j \mathcal{R}
$$

$$
= \left( U \sim K \left[ \bigoplus_{m=0}^{j-1} L^m \mathcal{W} \right] \oplus \left[ \bigoplus_{n=j}^{\infty} L^n \mathcal{W} \right] \right) \oplus KL^j \bigoplus_{i=0}^{\infty} L^i \mathcal{W}
$$

$$
= \left( \tilde{U}^{ct}(j) \sim K \bigoplus_{n=j}^{\infty} L^n \mathcal{W} \right) \oplus K \bigoplus_{i=0}^{\infty} L^i \mathcal{W}
$$

$$
= \left( \tilde{U}^{ct}(j) \sim K \bigoplus_{n=j}^{\infty} L^n \mathcal{W} \right) \oplus K \bigoplus_{n=j}^{\infty} L^n \mathcal{W}
$$

$$
\subseteq \tilde{U}^{ct}(j),
$$

where the last line follows from $(A \sim B) \oplus B \subseteq A$ [76]. Then any feasible $c^{tb}(k+j|k)$ is also a feasible $c^{ct}(k+j|k)$, achieved by $\tilde{u}^{ct}(k+j|k) = \tilde{u}^{tb}(k+j|k) + KL^j (x(k) - \tilde{x}^{tb}(k|k))$.

Secondly, consider the feedback terms. For any choice of $K$ in (2.27a), the feedback matrix $M^{ct}$ may be made equal to $M^{tube}$ by the choice $K(j) = K, \forall j$ in (2.28a). Therefore, the set of feasible $u$ from (2.28) is a subset of that from (2.27) with $K(j) = K, \forall j$.

In fact, in many cases there exists a CT-MPC policy that does not exist for tube MPC, even if the sets are convex and compact, because the Pontryagin difference is not an additive inverse [76]. It is simple to construct such a case. For example, consider the system

$$
\mathbf{x}(k+1) = \begin{bmatrix} I_2 & I_2 \\ 0 & I_2 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0.5I_2 \\ I_2 \end{bmatrix} u_p(k) + w_p(k),
$$
where \( w(k) \in \mathcal{W} \). Suppose the input constraint set is ‘square’, but the disturbance set is an \( m \)-sided polyhedron, \( i.e., \)

\[
\mathcal{U} = \{ u : ||u||_\infty \leq U_{\text{max}} \} \\
\mathcal{W} = \{ w : P_w w \leq (0.1U_{\text{max}})1 \}
\]

where \( P_{w,i} = \begin{bmatrix} \cos\left(\frac{2\pi i}{m}\right) & \sin\left(\frac{2\pi i}{m}\right) \end{bmatrix} \) is the \( i \)th row of \( P_w \). Figure 2.3 shows the resulting tightened sets, for the initial prediction step \( j = 0 \), for a nilpotent feedback controller and \( m = 8 \). The CT-MPC constraint set is not tightened at the initial step. The tube MPC set is tightened by a margin \( K\mathcal{R} \), producing a contracted square. Enlarging this tightened set by the same margin, \( K\mathcal{R} \), does not retrieve the original constraint set, but a square with ‘cut-off’ corners—a strict subset of the CT-MPC set. This ‘cut-off’ set is the effective feasible set for tube MPC, once the unique initial constraint, \( x(k) - \bar{x}(k|k) \in \mathcal{R} \), is accounted for.

Therefore, in general, the tube MPC method for robustness is more conservative than static-feedback CT-MPC. However, for the distributed forms, which are compared in Section 2.5.1, other advantages arise from use of the tube method, namely flexibility in communications and computations.
2.4 Robust distributed MPC using tubes

This section extends the tube MPC algorithm to a distributed implementation, and states the main feasibility and stability results. The centralized problem $\mathcal{P}^C$ is distributed amongst subsystem agents as local optimization problems, whereby each control agent makes a decision on the choice of plan, $U_p(k)$, for the local subsystem. The local optimization minimizes an objective function, $V_p(U_p(k))$, equal to $J_p(U_p(k))$, the part of the overall system objective associated with subsystem $p$. To clarify, as this distinction becomes important in later chapters, $V_p$ is the function given to the optimization for $p$ to minimize, while $J_p$ is (the finite-horizon approximation of) the performance measure for $p$.

The result that is crucial to the development of the distributed algorithm is that, by Proposition 2.1, given a feasible solution to $\mathcal{P}^C(x_1(k), \ldots, x_N(k))$, a feasible solution always exists for each subsystem at step $k+1$. The key feature is, then, that at any time step a sole subsystem agent—the optimizing agent—is permitted to optimize locally for a new plan, while all remaining subsystems adopt the feasible candidate plan (2.19):

$$
\hat{U}_p(k + 1) = \{ \hat{x}_p^*(k + 1|k), \hat{u}_p^*(k + 1|k), \ldots, \hat{u}_p^*(k + N - 1|k), \kappa_{F_p}(\hat{x}_p^*(k + N|k)) \},
$$

that is, the tail of the previous feasible solution augmented with a step using terminal controller $\kappa_{F_p}$. Constraint satisfaction and feasibility are maintained, because the candidate plan itself is available to the updating agent as a solution to its optimization. The local problem $\mathcal{P}_p^D(x_p(k); Z_p^*(k))$ for a subsystem $p \in P$ is defined by

$$
V_p^{\text{opt}}(x_p(k); Z_p^*(k)) = \min_{U_p(k)} V_p(U_p(k))
$$

$$
= \min_{U_p(k)} J_p(U_p(k))
$$

(2.29)
subject to $\forall j \in \{0, \ldots, N-1\}$:

\[
\begin{align*}
\dot{x}_p(k+j+1|k) &= A_p x_p(k+j|k) + B_p u_p(k+j|k), \\
x_p(k) - \dot{x}_p(k|k) &\in R_p, \\
\dot{x}_p(k+N|k) &\in X_{F_p}, \\
\dot{y}_p(k+j|k) &= C_p x_p(k+j|k) + D_p u_p(k+j|k), \\
\dot{y}_p(k+j|k) &\in \tilde{Y}_p, \\
\forall c \in C_p : \quad \dot{z}_{cp}(k+j|k) &= E_{cp} x_p(k+j|k) + F_{cp} u_p(k+j|k), \\
\dot{z}_{cp}(k+j|k) + \sum_{q \in P \setminus \{p\}} \dot{z}_{cq}^{*}(k+j|k) &\in \tilde{Z}_c,
\end{align*}
\]

where $Z_p^{*}(k)$ denotes the collection of information about other subsystems' plans that the control agent requires to evaluate the optimization. Specifically, outputs $\dot{z}_{cq}^{*}(|k)$ are required by $p$ to satisfy constraint (2.30g). Note that the collection of (2.30g) over all subsystems $p \in P$ is equivalent to (2.14g); the revised summation removes terms that are identically zero, using the definitions (2.6) and (2.7). It is assumed at this point that the information $Z_p^{*}(k)$ is known and sufficient; in Section 2.5 the communication requirements to obtain $Z_p^{*}(k)$ are identified.

This optimization is included in Algorithm 2.2. Though the algorithm is executed by all agents in parallel, only a single agent, defined as $p_k$, optimizes at each time step $k$, the remainder adopting their feasible candidate plans. The order in which subsystems' plans are optimized is determined by the update sequence, $\{p_1, \ldots, p_k, p_{k+1}, \ldots\}$, to be chosen by the designer. The distributed algorithm requires that a feasible initial plan be made available to each control agent, and this is a common assumption of DMPC methods; for example, see Dunbar [35], Richards and How [38]. A further requirement is that the terminal set $X_{F_p}$ for the local optimization be made available centrally, since coupling constraints must be satisfied therein. However, note that no further centralized processing is required from that point on.
Algorithm 2.2: Robust distributed MPC for a subsystem $p$

1. Design stabilizing controller $K_p$ and RPI set $R_p$;
2. Tighten sets $Y_p, Z_c, \forall c \in C_p$, via (2.16);
3. Wait for feasible solution $U_p^*(0)$, information $Z^*_p(0)$, and terminal set $X_{F_p}$ and control law $\kappa_{F_p}$ from central agent;
4. for $k = 1 : \infty$ do
   5. Sample current state $x_p(k)$;
   6. if $p_k = p$ then
      7. Obtain new plan $U_p(k) = U_p^{opt}(k)$ as solution to $P_p^D(x_p(k); Z^*_p(k))$;
      8. Transmit new plan to agents in $Q_p$;
   9. else
      10. Renew current plan via (2.19): $U_p(k) = \tilde{U}_p(k)$;
   11. end
   12. Apply control (2.17): $u_p(k) = \tilde{u}_p(k|k) + K_p(x_p(k) - \bar{x}_p(k|k))$;
   13. Wait one time step;
5. end

Remark 2.6 (Updating set and updates in parallel). The algorithm developed in this chapter assumes that only one agent $p_k$ may optimize at a step $k$. In fact, a set $P_k \subseteq P$ of agents may optimize simultaneously at a time step, without affecting feasibility, provided that none of the subsystems in the updating set are coupled. Formally, if $p \in P_k$ then $q \in P_k$ only if $q \not\in Q_p$. Such an arrangement moves the algorithm closer towards the class of parallel-update methods, which are popular in the DMPC literature; however, the capacity for parallel updates is heavily dependent on the coupling structure. In Chapter 5, this problem is examined more closely, and a robustly-feasible parallel-update algorithm is proposed, where any number of agents, including coupled ones, may optimize simultaneously. Without loss of generality, this chapter shall use the 'sole agent' form, which allows for the possibility that all subsystems are coupled.

Complementary results to those developed in the previous section for tube MPC then apply to the distributed algorithm. Again, beginning with the assumptions sufficient for robust feasibility, further assumptions on the terminal and stage costs are introduced to develop robust stability results.
2.4.1 Feasibility of DMPC

Under Assumptions 2.2 and 2.3, on the availability of an admissible, nominally-invariant terminal set, the system controlled by Algorithm 2.2 has the properties of robust constraint satisfaction and robust feasibility. It is worth noting once more that this feasibility result holds regardless of the form of the optimization cost function $V_p(U_p(k))$.

**Proposition 2.4 (Robust feasibility of distributed MPC).** Suppose the sequence of controls $U_p^*(k_0) = \{ \tilde{x}_{p}^*(k_0|k_0), \tilde{u}_{p}^*(k_0|k_0), \ldots, \tilde{u}_{p}^*(k_0+N-1|k_0) \}, \forall p \in \mathcal{P}$, exists and is a feasible (but not necessarily optimal) solution to $\mathbb{P}^C(x_1(k_0), \ldots, x_{N_p}(k_0))$ at some time step $k_0$. Then, for all $x_p(k_0 + 1) \in A_p x_p(k_0) + B_p u_p(k_0) \oplus W_p$, $\forall p \in \mathcal{P}$, where $u_p(k_0) = \tilde{u}_{p}^*(k_0|k_0) + K_p(x_p(k_0) - \tilde{x}_{p}^*(k_0|k_0))$, (i) the candidate sequence $\tilde{U}_p(k_0 + 1)$, defined by (2.19), is a feasible solution to $\mathbb{P}^D_p(x_p(k_0 + 1); Z_p^*(k_0 + 1))$, and (ii) subsequently, the resulting closed-loop system controlled by Algorithm 2.2 is robustly-feasible for any choice of update sequence.

**Proof.** For (i), given a feasible solution $U_p^*(k_0), \forall p$, to $\mathbb{P}^C(x_1(k_0), \ldots, x_{N_p}(k_0))$, by Proposition 2.1, $\tilde{U}_p(k_0 + 1)$, defined by (2.19), is a feasible solution to $\mathbb{P}^C(x_1(k_0 + 1), \ldots, x_{N_p}(k_0 + 1))$. $\tilde{U}_p(k_0 + 1)$ is also a feasible solution to $\mathbb{P}^D_p(x_p(k_0 + 1); Z_p^*(k_0 + 1))$, for any $p$; constraints (2.30a) to (2.30f) are satisfied by Proposition 2.1, and constraint (2.30g) is satisfied by $z_{cp}(|k_0 + 1) = \tilde{z}_{cp}(|k_0), \forall c \in \mathcal{C}_p$, so that $\sum_{p \in \mathcal{P}_c} \tilde{z}_{cp}(k_0 + j|k_0) \in Z_c, j \in \{1, \ldots, N\}$, which is then equivalent to constraint (2.14g) in the problem $\mathbb{P}^C(x_1(k_0 + 1), \ldots, x_{N_p}(k_0 + 1))$, (all $c \notin \mathcal{C}_p, p \notin \mathcal{P}_c$, have $\tilde{z}_{cp} = 0$).

Part (ii) follows by applying recursion to (i). By construction, any solution $U_{pk}^*(k)$ to $\mathbb{P}^D_{pk}(x_{pk}(k); Z_{pk}^*(k))$ taken with the candidate solutions $\{ \tilde{U}_p(k) \}, p \neq p_k$, is a solution to $\mathbb{P}^C(x_1(k), \ldots, x_{N_p}(k))$; solving $\mathbb{P}^D_{pk}$ is equivalent to solving $\mathbb{P}^C$ with $p \neq p_k$ constrained to take $U_p(k) = \tilde{U}_p(k)$. A feasible solution to $\mathbb{P}^C(x_1(0), \ldots, x_{N_p}(0))$ then implies all subsequent optimizations $\mathbb{P}^D_p(x_p(k); Z_p^*(k)), k \geq 0$, are feasible, regardless of the choice of update sequence. \[\square\]
2.4.2 Stability of DMPC

The stability results of Section 2.3.2 also have their counterparts for the distributed algorithm. Firstly, again it is assumed that Assumption 2.4 holds; that is,

\[ F_p(A_p x_p + B_p \kappa F_p(x_p)) - F_p(x_p) \leq -l_p(x_p, \kappa F_p(x_p)), \forall x_p \in X_{F_p}, p \in P. \]

Then monotonicity of each local cost is maintained, regardless of update sequence.

**Proposition 2.5** (Monotonicity of the Cost). Suppose the sequence of controls \( U^*_p(k_0) = \{ \bar{u}^*_p(k_0|k_0), \ldots, \bar{u}^*_p(k_0+N-1|k_0) \}, \forall p \in P, \) exists and is a feasible (but not necessarily optimal) solution to \( P^C(x_1(k_0), \ldots, x_{N_p}(k_0)) \) at some time step \( k_0 \). Then, for all \( x_p(k_0 + 1) \in A_p x_p(k_0) + B_p u_p(k_0) + W_p, \forall p \in P, \) where \( u_p(k_0) = \bar{u}^*_p(k_0|k_0) + K_p(x_p(k_0) - \bar{x}^*_p(k_0|k_0)) \), the upper bound on the local cost decreases monotonically:

\[ V_p^*(x_p(k_0 + 1); Z^*_p(k_0 + 1)) \leq V_p^*(x_p(k_0); Z^*_p(k_0)) - l_p(\bar{x}^*_p(k_0|k_0), \bar{u}^*_p(k_0|k_0)), \]

for all \( p \in P, \) where \( V_p^*(x_p(k_0); Z^*_p(k_0)) = J_p(U^*_p(k_0)) \).

**Proof.** At time \( k_0, U^*_p(k_0), \forall p \in P, \) is a feasible solution to \( P^C(x_1(k_0), \ldots, x_{N_p}(k_0)) \). The associated local cost is

\[ V_p^*(x_p(k_0); Z^*_p(k_0)) = J_p(U^*_p(k_0)). \]
At time \( k_0 + 1 \) a non-updating subsystem \( p \neq p_{k_0+1} \) adopts the candidate solution
\[
U_p(k_0 + 1) = U_p(k_0 + 1),
\]
defined by (2.19), with associated cost
\[
\bar{V}_p(x_p(k_0 + 1); Z_p^*(k_0 + 1)) = J_p(\bar{U}_p(k_0 + 1))
\]
\[
= J_p(U_p^*(k_0)) - \ell_p(\bar{x}_p^*(k_0|k_0), \bar{u}_p^*(k_0|k_0))
\]
\[
+ \ell_p\left(\bar{x}_p^*(k_0 + N|k_0), \kappa_{F_p}\left(\bar{x}_p^*(k_0 + N|k_0)\right)\right)
\]
\[
+ F_p\left(A_p\bar{x}_p^*(k_0 + N|k_0) + B_p\kappa_{F_p}\left(\bar{x}_p^*(k_0 + N|k_0)\right)\right)
\]
\[
- F_p\left(\bar{x}_p^*(k_0 + N|k_0)\right).
\]

By Assumption 2.4, the latter three terms sum to less than or equal to zero, leaving
\[
\bar{V}_p(x_p(k_0 + 1); Z_p^*(k_0 + 1)) \leq V_p^*(x_p(k_0); Z_p^*(k_0)) - \ell_p(\bar{x}_p^*(k_0|k_0), \bar{u}_p^*(k_0|k_0)),
\]
for all \( \forall p \neq p_{k_0+1} \).

The optimizing subsystem \( p_{k_0+1} \) obtains \( U_{p_{k_0+1}}(k_0 + 1) \) as the solution to the local optimization \( \mathcal{P} \)
\[
V_{p_{k_0+1}}^*(x_{p_{k_0+1}}(k_0 + 1); Z_{p_{k_0+1}}^*(k_0 + 1)) \leq \bar{V}_{p_{k_0+1}}(x_{p_{k_0+1}}(k_0 + 1); Z_{p_{k_0+1}}^*(k_0 + 1))
\]
\[
= J_{p_{k_0+1}}(\bar{U}_{p_{k_0+1}}(k_0 + 1)).
\]

Thus, for any subsystem \( p \in \mathcal{P} \), it follows that
\[
V_p^*(x_p(k_0 + 1); Z_p^*(k_0 + 1)) \leq V_p^*(x_p(k_0); Z_p^*(k_0)) - \ell_p(\bar{x}_p^*(k_0|k_0), \bar{u}_p^*(k_0|k_0)),
\]
where \( V_p^* \) is the cost of a general feasible solution, and the result is established. \( \square \)

The monotonicity result leads to robust stability results, dependent on the form of the objective. Again, for generality, stability of the origin for the nominal system is considered, leading to robust stability of the RPI set; \( \mathcal{R}_p \) is asymptotically, or
exponentially, stable for the closed-loop system. Of course, Remark 2.3, regarding alternative objective formulations, applies also to the distributed form of the tube MPC algorithm.

**Theorem 2.3 (Robust asymptotic stability of \( \mathcal{R}_p \)).** Suppose \( l_p(x_p, u_p) \geq c \| (x_p, u_p) \| \) for some \( c > 0 \). Then the set \( \mathcal{R}_p \) is robust asymptotically stable for the controlled system \( x_p(k+1) = A_p x_p(k) + B_p u_p(k) + w_p(k), \forall p \in P \), where \( w_p(k) \in \mathcal{W}_p, \forall k \), for any update sequence.

**Proof.** Monotonicity of the sequence \( \{ V_p^*(k) \} \) for any update sequence was established by Proposition 2.5. Because \( V_p^*(k+1) - V_p^*(k) \leq -l_p(x_p^*(k|k), \bar{x}_p^*(k|k)) \), yet \( V_p^*(\cdot) \) and the stage cost \( l_p(\cdot, \cdot) \) are both strictly non-negative, then by recursion it follows that \( V_p^*(k+1) - V_p^*(k) \to 0 \) as \( k \to \infty \). In turn, this implies that \( l_p(x^*_p(k|k), \bar{u}_p^*(k|k)) \to 0 \). Because \( l_p(x_p, u_p) \geq c \| (x_p, u_p) \| \) for some \( c > 0 \), and \( l_p(0, 0) = 0 \), it must be that the nominal state \( \bar{x}_p^*(k|k) \to 0 \) and the nominal control \( \bar{u}_p^* \to 0 \). Finally, by the fact that \( x_p(k) \in \bar{x}_p(k|k) \oplus \mathcal{R}_p, \forall k \), it follows that the true state \( x_p(k) \to \mathcal{R}_p \) as \( k \to \infty \), and, furthermore,

\[
\begin{align*}
  u_p(k) &= \bar{u}_p^*(k|k) + K_p (x_p(k) - \bar{x}_p^*(k|k)) \\
  &\to K_p x_p(k)
\end{align*}
\]

as \( k \to \infty \). \( \square \)

**Theorem 2.4 (Robust exponential stability of \( \mathcal{R}_p \)).** Suppose \( l_p(x_p, u_p) \leq x_p^T Q_p x_p + u_p^T R_p u_p \) and \( F_p(x_p) \leq x_p^T P_p x_p \), where \( Q_p, R_p, P_p \) are positive-definite. Then the set \( \mathcal{R}_p \) is robust exponentially stable for the controlled system \( x_p(k+1) = A_p x_p(k) + B_p u_p(k) + w_p(k), \forall p \in P \), where \( w_p(k) \in \mathcal{W}_p, \forall k \).

**Proof.** The result is adapted from Mayne et al. [27, Theorem 1]. As shown in Mayne et al. [7], given \( l_p(x_p, u_p) \) and the assumptions on \( F_p \) and \( \mathcal{X}_F_p \), there exist constants
$b_p > a_p > 0, \forall p \in \mathcal{P}$, such that, for all $k$

$$
V^*_p(x_p(k), Z^*_p(k)) \geq a_p|\bar{x}_p^*(k|k)|^2, \forall p \in \mathcal{P}, \forall \{x_1(k), \ldots, x_{N_p}(k)\} \in \mathcal{X}
$$

(2.31a)

$$
V^*_p(x_p(k+1), Z^*_p(k+1)) \leq V^*_p(x_p(k), Z^*_p(k)) - a_p|\bar{x}_p^*(k|k)|^2, \\
\forall x_p(k+1) \in A_p x_p(k) + B_p u_p^*(k) \oplus \mathcal{W}_p, p \in \mathcal{P}, \forall \{x_1(k), \ldots, x_{N_p}(k)\} \in \mathcal{X}
$$

(2.31b)

$$
V^*_p(x_p(k), Z^*_p(k)) \leq b_p|\bar{x}_p^*(k|k)|^2, \forall x_p(k) \in \mathcal{X}_p \oplus \mathcal{R}_p,
$$

(2.31c)

where $\mathcal{X}$ is the combined set of states to which a feasible solution exists to the centralized problem $\mathcal{P}^C(x_1(k), \ldots, x_{N_p}(k))$. From Proposition 2.2 and the above, it follows that

$$
V^*_p(x_p(k), Z^*_p(k)) - V^*_p(x_p(k), Z^*_p(k)) \leq -l_p(\bar{x}_p^*(k|k), \bar{u}_p^*(k|k)) \leq -a_p|\bar{x}_p^*(k|k)|^2.
$$

Next, for all $a_p \geq 0$, let $L^\alpha_p \triangleq \{x_p : V^*_p(x_p, Z^*_p) \leq \alpha_p\}$. Then $L^0_p = \mathcal{R}_p$ and there exists an $\alpha_p > 0$ so that $L^\alpha_p \subset \mathcal{X}_p \oplus \mathcal{R}_p$. Consider first the initial states within this set, i.e., $x_p(0) \in L^\alpha_p, \forall p \in \mathcal{P};$ from (2.31) it follows that the value function at a general, subsequent time $k$ is bounded by that at time 0:

$$
V^*_p(x_p(k), Z^*_p(k)) \leq \gamma^k V^*_p(x_p(0), Z^*_p(0))
$$

where $\gamma_p \triangleq (1 - a_p/b_p) \in (0, 1)$. Subsequently, $|\bar{x}_p^*(k|k)| \leq c_p \delta_p^k |\bar{x}_p^*(0|0)|$ for some finite $c_p$, where $\delta_p = \sqrt{\gamma_p}$. Next, consider any states $\{x_1(0), \ldots, x_{N_p}(0)\} \in \mathcal{X}$. From (2.31b) it follows that there exists some finite integer $K_p$ for each $p$ such that, for all $k \geq K_p$, $x_p(k) \in L^\alpha_p$. Consequently, there exists some finite $d_p > c_p$ such that $|\bar{x}_p^*(k|k)| \leq d_p \delta_p^k |\bar{x}_p^*(0|0)|, \forall p \in \mathcal{P}$. Thus, because $\gamma_p \in (0, 1)$, then $\delta_p \in (0, 1)$, and the origin is exponentially stable for the nominal states $\bar{x}_p$, and, as $x_p(k) - \bar{x}_p(k|k) \in \mathcal{R}_p$ for all $k$, the set $\mathcal{R}_p$ is exponentially stable for the perturbed states $x_p$. $\Box$
The feasibility and stability results applicable to the DMPC algorithm rely, as has been shown, on different assumptions; stability requires additional assumptions on the objective function. The algorithm will be applied to a number of examples throughout this thesis, with the different assumptions used interchangeably. Therefore, the algorithm under the basic assumptions for feasibility shall be termed robustly-feasible DMPC, while the algorithm under the additional, stronger assumptions shall be termed robust stable DMPC. As previously stated, the former, though associated with a weaker result of only robust feasibility, is particularly useful for problem classes where convergence of the system to some target is not guaranteed. For example, path planning for multiple vehicles in an unknown environment [49], where the terminal cost is an approximation of the cost-to-go to the goal, and is not a Lyapunov function. Despite the lack of convergence guarantees, further results will be developed in later chapters for this case.

The following example applies the DMPC algorithm to a system of constrained point masses, and demonstrates robust feasibility and stability for different update sequence choices.

**Example 2.1 (DMPC of constrained point masses).** Consider system consisting of $N_p$ identical point masses moving in 1-D, each with double integrator dynamics, discretized using a time step $\delta t = 1$ second:

$$
A_p = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B_p = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}, \quad \forall p \in \mathcal{P}.
$$

Each mass is subject to local constraints on speed and control:

$$
\begin{bmatrix} 0 & 1 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} x_p(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} u_p(k) \leq \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \end{bmatrix},
$$

47
and all pairs are coupled by a constraint to remain 'close', i.e., $|[1 \ 0](x_p - x_q)| \leq \Delta x, \forall p \neq q$, where in this example $\Delta x = 2$. The tube controller $K_p$ is chosen to be the nilpotent controller,

$$K_p = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix},$$

such that $(A_p + B_p K_p)^2 = 0$. Then the sets $R_p$ are finitely-determined, and given by $W_p \oplus (A_p + B_p K_p) W_p$, where $W_p$ in this case is a simple hypercube, $\{ w_p \in \mathbb{R}^2 : \|w_p\|_{\infty} \leq 0.1 \}$.

The stage cost penalizes deviations of the control from $K_p x_p$:

$$l_p(x_p, u_p) = \|u_p - K_p x_p\|_{\infty},$$

and the terminal cost is zero. The $\infty$-norm ensures the resulting optimization problem is an LP. The terminal control law is chosen to be equal to the tube controller $K_p$, and the corresponding terminal set in the optimization for $p$ is the maximal output-admissible set [81], for the stabilized system $x_p(k+1) = (A_p + B_p K_p) x_p(k)$, that satisfies Assumptions 2.2 and 2.3. The prediction horizon is 12 steps.

Figures 2.4 and 2.5 show the results for control of a two-mass system with initial states $x_1(0) = [-5 \ -2]^T$ and $x_2(0) = [-5 \ 0]^T$. In the former figure, for a simple alternating update sequence $\{1, 2, 1, 2, \ldots\}$, the trajectories of the randomly-perturbed system are shown, together with nominal trajectories, tube cross-sections, and terminal sets. The states converge, while respecting the velocity constraints, (indicated by the dotted lines), and remain within the tube at all time steps. Next, as shown in Figure 2.5, two further update sequences are employed: a modification of the alternating sequence, whereby each subsystem performs two consecutive updates, and a zero-update step follows the completion of each cycle; and, a randomly-generated sequence, including zero-update steps. For each of the three sequences, the maximum separation constraints are seen to be satisfied, and the applied controls converge to the stabilizing control law $u_p = K_p x_p$, while keeping within the permitted magnitude.
Figure 2.4: Trajectories of two constrained masses controlled by distributed MPC.
Figure 2.5: Variation of update sequence (rows) for DMPC of two masses, showing (left to right columns) positions and maximum separation bands, controls applied, objective values, and update sequence.
2.5 Communication analysis

It remains to evaluate exactly what information, denoted $Z_p^*(k)$, is required in the local optimization for $p$. In the problem $P^D_p(x_p(k); Z_p^*(k))$, the structure in the coupling constraints, identified in (2.6) and (2.7), has been exploited. Firstly, only constraints $c \in C_p$ are applied, as by definition (2.7), $z_{cp}(k + j|k) = 0$ for all other constraints $c \notin C_p$, so these outputs do not affect the update of subsystem $p$. Secondly, the summation in (2.30g), for each $c$, includes output terms from only those subsystems in $P_c$; by definition (2.6), $z_{cr}(k + j|k) = 0$ for all other subsystems $r \notin P_c$. The coupling terms $z_{cq}^*(k + j|k), \forall q \in P_c \setminus \{p\}$ are not affected by the decision variables $U_p(k)$, so they appear as fixed values in (2.30g), denoted by $\ast$. Using the definition of coupled subsystems (2.8), it follows that to evaluate (2.30g), values for $z_{cq}^*(k + j|k), \forall c \in C_p$, are required from all other subsystems $q$ in $Q_p$.

Define a message vector from subsystem $p$ regarding constraint $c$ at time $k$ as

$$m_{cp}(k) \triangleq \begin{bmatrix} z_{cp}^*(k|k)^T & \ldots & z_{cp}^*(k + N - 1|k)^T & x_p^*(k + N|k)^T \end{bmatrix}^T,$$

which includes the coupling outputs and the terminal state. Again, the $\ast$ superscript denotes a feasible solution. Also, define a propagation matrix,

$$\Pi_{cp} \triangleq \begin{bmatrix} 0 & I & 0 & \ldots & 0 \\ 0 & 0 & I & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & (E_{cp} + F_{cp}K_{Fp}) \\ 0 & 0 & 0 & \ldots & (A_p + B_pK_{Fp}) \end{bmatrix},$$

assuming a linear terminal control law, i.e., $\kappa_{Fp}(x_p) = K_{Fp}x_p$, so that $m_{cp}(k) = \Pi_{cp}m_{cp}(k - 1)$ is the message at time $k$ for a non-updating subsystem $p \neq p_k$. Suppose the last time a subsystem $p$ optimized its plan was at a step $k_p$, before the

51
current step $k$, defined as

$$\hat{k}_p(k) \triangleq \max_{k' \in \{k' < k | p_{k'} = p\}} k'.$$

Then the message at $k$ for a subsystem $p$ that last optimized at $\hat{k}_p$ is $m_{cp}(k) = \Pi_{cp}^{(k-\hat{k}_p)} m_{cp}(\hat{k}_p)$. Relating this back to the information that is required by $p_k$ to evaluate (2.30g), $Z_{pk}^*(k)$ is obtained as

$$Z_{pk}^*(k) = \{J m_{cq}(k)\}_{c \in C_{pk}, q \in Q_{pk}}$$

$$= \{J \Pi_{cq}^{(k-\hat{k}_q)} m_{cq}(\hat{k}_q)\}_{c \in C_{pk}, q \in Q_{pk}},$$

where the matrix operator $J \triangleq \text{diag}(I, I, \ldots, 0)$ removes the terminal states. The inclusion of the terminal state $\bar{x}_p(k+N|k)$ in the message permits the correct propagation for steps $k > \hat{k}_q + N$. This propagation leads to the following requirement.

**Requirement 2.1** (Information requirement for $Z_{pk}^*(k)$). At a time step $k$, the control agent for an optimizing subsystem $p_k$ must have received messages $m_{cq}(\hat{k}_q), \forall c \in C_{pk}$, from all subsystems $q \in Q_{pk}$.

This illustrates a key feature of tube MPC that means it lends itself to distribution; an updating subsystem $p_k$ may obtain $Z_{pk}^*(k)$ by using $\Pi_{cq}$ to propagate previously-communicated data regarding coupled subsystems, with no communication required in the interim. Therefore, to meet Requirement 2.1 it is sufficient for each agent $p$ to transmit the message $m_{cp}, \forall c \in C_p$, after each planning update, as in line 8 of Algorithm 2.2. Alternatively, rather than transmissions taking place immediately following an optimization, an optimizing subsystem agent $p_k$ could request messages from others immediately before its update at step $k$. A token ‘no change’ response could be provided by those agents that have not changed plans in the interim period since agent $p_k$ last optimized. Throughout this thesis, message transmission shall be assumed to occur following update. However, regardless of the communication timing used, instances exist where message transmissions are not
necessary. The remainder of this section identifies these instances, and shows how flexibility in update sequence choice can be exploited to offer a DMPC scheme with low levels of communication.

It is observed that after the optimization at time $k$, the updating system $p_k$ needs to transmit a message if both the following two criteria are met:

C1: The optimized plan differs from the candidate plan, i.e., $U_{p_k}^{opt}(k) \neq \tilde{U}_{p_k}(k)$;

C2: Before subsystem $p_k$ next optimizes, another subsystem in $Q_{p_k}$ will optimize.

Note that the investigations in this section all consider cases in which the updating sequence is predetermined and known by all subsystems. However, the tube DMPC framework can also be extended to cases in which the choice of updating subsystem is made on-line.

The use of these criteria can be demonstrated by counter-examples where a data transmission is not needed. If criterion C1 were not met, the optimization for the updating subsystem will have resulted in a plan identical to the tail of the previous plan, formed by (2.19). As each coupled subsystem already has knowledge of this plan, from a transmission at some earlier step, then there is no need to re-send. Instances also exist where criterion C2 is not met, and Figure 2.6 illustrates such a case. Suppose three subsystems, 1, 2 and 3 share coupling constraints. Subsystem 1 optimizes at time step $k$, followed by 3 at $k+1$, followed by 1 again at $k+2$. Clearly, 2 may not optimize until at least $k+3$, and has no need for 1’s updated plan from step $k$, as this is superseded by its plan from $k+2$. Hence, a message transmission from 1 to 2 at step $k$ is redundant. Similarly, the transmissions to agent 2 at times $k+1$ and $k+2$ could be redundant depending on who optimizes at step $k+3$ and thereafter. Therefore, a priori knowledge of the update sequence may be used to eliminate redundant communication.

The measure of communication that shall be used in the sequel is, with only small loss of generality, the number of data exchanges between any pair of subsystems at a time step, where a data exchange occurs whenever a subsystem agent transmits its
Figure 2.6: Timing diagram illustrating redundant inter-agent communication. Agents measure states (●), form plans, then update controls (♦). Arrows represent transmissions between optimizing agents (shaded boxes) to others. The dash-dot arrow line indicates a redundant transmission. The subsequent transmissions from 1 and 3 to 2 may be redundant depending on the update sequence from step $k + 3$ onwards.

A message is sent to any other subsystem agent. It follows from the preceding discussions that, whenever a subsystem $p_k$ is required to transmit its plan, following an optimization as in Algorithm 2.2, it must transmit to all others in $Q_{p_k}$:

$$N_{data}^D(k) = \begin{cases} n(Q_{p_k}) & \text{if C1 and C2 met,} \\ 0 & \text{otherwise,} \end{cases}$$

where $n(Q_{p_k})$ denotes the cardinality of $Q_{p_k}$.

Similarly, it is possible to establish the communication required for the centralized implementation of the controller (CMPC), Algorithm 2.1. In this case, a central agent must have received current state data from all subsystems at a time step $k$. Following optimization, new plans must be communicated to all subsystems. Assuming that the control agent is located on one of the subsystems $p \in \mathcal{P}$, the minimum number of data exchanges required at an optimization is $2(N_p - 1)$. However, no optimization may take place by all subsystems renewing their plans via (2.19), in
the event of no new plans being transmitted by the control agent.

\[ N_{\text{data}}^C(k) = \begin{cases} 
2(N_p - 1) & \text{if optimization takes place,} \\
0 & \text{otherwise.} 
\end{cases} \]

DMPC therefore requires, at most, \( n(Q_{pk}) \) data exchanges per step, whereas CMPC requires \( 2(N_p - 1) \) exchanges. In the worst case, when coupling constraints exist between all subsystems, subsystem \( p_k \) is coupled to all other subsystems, and \( n(Q_{pk}) = (N_p - 1) \) for any \( p_k \). By definition, \( n(Q_{pk}) \leq (N_p - 1) \); thus, DMPC requires, at most, only half as many data exchanges per optimization as does CMPC.

For centralized MPC, at each time step, a decision is made whether to optimize or not. The resulting number of data exchanges that take place over the length of a simulation is then inextricably linked to the number of updating steps. With the distributed algorithm, we have an extra degree of freedom, in that the decision is not only whether to optimize or not, but also which subsystem is to optimize. For example, the sequence \( \{1, 2, 1, 2, \ldots\} \) requires communication at every step, whereas \( \{1, 1, 2, 2, \ldots\} \) requires communication at alternating steps. There is a many-to-one mapping of update sequences to data exchanges; thus, the link between the number of updating steps and communication is broken. It remains to explore the effect this flexibility has on system-wide performance, and this is done in a later section. Next, however, having established the communication requirements for tube DMPC, a comparison is made with constraint-tightening DMPC [38].

### 2.5.1 Comparison with constraint-tightening DMPC

The DMPC algorithm shares certain similarities with the constraint-tightening (CT-DMPC) method of Richards and How [38], namely achieving robust feasibility and stability for linear, dynamically-decoupled subsystems with coupled constraints. Both methods tighten constraints in the optimization by some margin, and use feedback to maintain feasibility for the perturbed system. The key difference between
the two methods, however, is in the relative timing of optimizations, communications and control updates. Figure 2.7 illustrates the timing of these events; in 2.7(a), the tube DMPC algorithm is shown, in which one agent optimizes while others adopt the feasible candidate solutions (2.19). The case shown assumes that each agent has no knowledge of the future update sequence, so that data transmissions prepare the group of agents for any agent to subsequently optimize. Such an assumption permits both on-line and off-line determination of the sequence, and will also enable the most general comparison between tube DMPC and CT-DMPC to be made.

For tube DMPC, the unique initial constraint, \( x_p(k) - \tilde{x}_p(k|k) \in \mathcal{R}_p, \forall p \), and the uniform tightening of the coupled constraints, starting from the initial step, means that robustness to uncertainty in other agents’ current states is guaranteed, and so no inter-agent exchange of current states is required. In fact, a single communication takes place after the optimization of the sole updating agent.

On the other hand, CT-DMPC assumes a fixed sequence to optimize all subsystems during a time step, with communication immediately after each optimization, as illustrated in Figure 2.7(b), ignoring delays. As a consequence, for a general agent \( p \), the coupling constraint in the local optimization uses the most recent data from upstream agents and predictions for downstream agents based on their updates at the previous time step:

\[
\mathbf{z}_{cp}(k + j|k) + \sum_{q \in \{1, \ldots, p-1\}} \mathbf{z}_{cq}^*(k + j|k) + \sum_{r \in \{p+1, \ldots, N_p\}} \mathbf{z}_{cr}^*(k + j|k - 1) \in \mathcal{Z}_{cp}(j)
\]

The coupling constraints in the local optimizations are then tightened non-uniformly over the horizon, and, additionally, by an extra margin according to the position of
Figure 2.7: Timing diagrams for tube DMPC and CT-DMPC. Shaded boxes indicate local optimizations, which are followed or preceded by communication from transmitting agent (•) to receiving agent, in the direction indicated by an arrow. Clear boxes indicate adoption of the candidate solution. Agents apply control updates (♦) once a new plan is obtained.
an agent in the sequence. Specifically, for all $c \in C, j \in \{0, \ldots, N-2\}$:

\begin{align}
\tilde{Z}_{cNp}(0) &= Z_c, \\
\tilde{Z}_{c(p-1)}(j) &= \tilde{Z}_{cp}(j) \sim (E_{cp} + F_{cp}K_p(j))L_p(j)W_p, \\
\tilde{Z}_{cNp}(j + 1) &= \tilde{Z}_{c1}(j) \sim (E_{c1} + F_{c1}K_1(j))L_1(j)W_1,
\end{align}

where,

\begin{align}
L_p(0) &= I, \\
L_p(j + 1) &= (A_p + B_pK_p(j))L_p(j),
\end{align}

so that the first agent in the sequence has the tightest constraints, associated with the greatest uncertainty about downstream agents’ plans, and the final agent has no extra tightening, given that it will have received the latest plans of upstream agents. In contrast, the constraint margins for tube DMPC, $\tilde{Z}_c$, have no concept of sequence.

We proceed now to examine the scope of the CT-DMPC to provide more flexible update sequences, and derive an information requirement, a counterpart to Requirement 2.1. The key observation here is that, whereas for tube DMPC the candidate plan—and associated coupling data—for a local subsystem is obtained by a simple truncation and augmentation of the previous plan, in CT-DMPC the candidate plan includes a disturbance feedback term. Consequently, propagation of the message vector, a concept introduced earlier in this section for tube DMPC, relies additionally on the last disturbance:

$$m_{cp}(k + 1) = \Pi_{cp}m_{cp}(k) + \Delta_{cp}w_p(k),$$

where $m_{cp}(k)$ is the message vector (2.32), defined in the previous subsection, consisting of predicted coupling outputs made at time $k$; $\Pi_{cp}$ is the propagation matrix.
and

\[
\Delta_{cp} \triangleq \begin{bmatrix}
0 \\
(E_{cp} + F_{cp}K_p(0))L_p(0) \\
\vdots \\
(E_{cp} + F_{cp}K_p(N - 1))L_p(N - 1) \\
L_p(N)
\end{bmatrix}.
\]

The difference between the two methods is now clearer. For tube DMPC, it is possible for an optimizing agent $p_k$—given a message $m_{cq}(k_q)$ from some agent $q$ that last optimized at $k_q$—to locally construct the requisite coupling outputs by matrix multiplication, via (2.34). This is not so for CT-DMPC, as $p_k$ would also require knowledge of the disturbance sequence \{\text{\textit{w}}_q(k_q), \text{\textit{w}}_q(k_q+1), \ldots, \text{\textit{w}}_q(k-2)\}. Therefore, we conclude that the coupling data required by an updating agent at $k$ must originate from \textit{no earlier} than the previous step $k - 1$. (This assumption is met implicitly by CT-DMPC in its standard form \[38\], wherein all agents optimize and communicate at each step).

**Requirement 2.2 (Information requirement for CT-DMPC).** At any time step $k$, the control agent for an optimizing subsystem $p$ must have received messages

1. $m_{cq}(k)$, $\forall c \in C_p$, from all upstream subsystems $q \in Q_p \cap \{1, \ldots, p - 1\}$;
2. $m_{cq}(k-1)$, $\forall c \in C_p$, from all downstream subsystems $q \in Q_p \cap \{p + 1, \ldots, N_p\}$.

Now suppose CT-DMPC were operated with the same optimization timing as for tube DMPC, \textit{i.e.}, with a sole—but not necessarily known \textit{a priori}—agent optimizing at each time step, the remaining agents adopting their respective candidate plans. It is clear that similar communication timings to those for tube DMPC, shown in Figure 2.7(a), are not sufficient for CT-DMPC. One sufficient means of meeting Requirement 2.2 is to maintain the communication arrangement of the standard, sequential CT-DMPC; this is illustrated in Figure 2.7(c), where an update sequence identical to that of Figure 2.7(a) is employed. The result is that communication is invariant with update sequence.
An improved proposal, shown in Figure 2.7(d), has the sole optimizing agent request and receive plans *immediately prior* to its update, a more *ad hoc* mode of communication. As Requirement 2.2 states, information from upstream agents must date from the current time step, while downstream agent information must date from, at the earliest, the previous step. The figure shows the optimizing agent gathering all required data at the current time step, yet—crucially—only after the upstream agents have adopted their candidate plans, based on the latest disturbance. Thus, though communication has been reduced, the timing of transmissions is sequence-dependent. Furthermore, the communication arrangements for CT-DMPC—single-update or otherwise—have a crucial reliance on *instantaneous* data exchanges between agents during a time step. Tube DMPC allows the entire remainder of the time step, following the optimization, for transmission of the new plan. Together, these points illustrate the key contribution of tube DMPC: despite the extra conservatism of this method for robustness, as shown in Section 2.3.3, greater flexibility exists in the relationship between computation and communication.

### 2.5.2 Performance versus communication

The purpose of this section is twofold: by simulation, to show how computation time scales with problem size for the new algorithm, and to compare the performance of DMPC with that of CMPC, and also CT-DMPC, by investigating the trade between performance and communication. It is shown that the flexibility in communication can be exploited to obtain better performance for DMPC with low levels of communication.

The simulations in this section use the 1-D system of Example 2.1, with the ‘remain close’ coupling constraints. Apart from the objective functions, which shall vary, all other parameters, including constraint limits, disturbance sets, and controllers, are the same.
Example 2.2 (Scaling of computation). The tracking output form of the objective is used, with the aim of steering the state to a compact set around the origin. That is, \( s_p = x_p \), with a target set

\[
T_p = \left\{ x_p : \|x_p\|_\infty \leq 0.5 \right\},
\]

which is subsequently tightened for robustness. The distance metric, \( d(s_p, T_p) \), for this objective is implemented as the 1-norm measure (2.23), so then the DMPC optimization is a Linear Program. For an initial state \( x_p(0) = [100\ 0]^T, \forall p \in P \), the number of subsystems \( N_p \) in the problem is incremented, and corresponding means and standard deviations of the computation time per update measured. All simulations were performed on a Pentium 4 HT 3.2 GHz with 2,048 MB RAM, using CPLEX 10.1 as the LP solver. To ensure that computation times are dominated by optimization times, and not overheads, a long horizon of 100 steps is set.

The metric of interest here is the computation time per optimization, distributed or centralized. This metric gives an indication of the computational effort that would be required by a local agent compared with a central control agent, and has implications for the ‘size’ of computer required for each method.

For this example, there exists pair-wise coupling across the whole system, so that the number of inequality coupling constraints \( N_c \) increases as \( 2^{(N_p^2)} = N_p(N_p - 1) \). Hence, for the centralized algorithm, the number of constraints in the optimization grows quadratically with \( N_p \). Also, the number of constraints in the optimization grows linearly with \( N_p \). Conversely, for DMPC, the number of coupling constraints \( C_p \) involving a subsystem \( p \) increases as \( N_p - 1 \) and the number of local decision variables is independent of \( N_p \).

Smoothed analysis of complexity of the Simplex algorithm for linear programming has shown that, although worst-case running time is exponential, the expected running time is is polynomial [92]. (In fact, in practice the performance of the Simplex algorithm usually exceeds that predicted by complexity analysis [93]). That is,
for a linear program

$$\min \mathbf{c}^T \mathbf{x} \text{ subject to } \mathbf{A} \mathbf{x} \leq \mathbf{b}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, the expected running time is polynomial in the dimension of $\mathbf{A}$, i.e., $m \times n$. Therefore, one would expect the computation time for centralized MPC to scale with $O(N_p \times N_c) = O(N_p^3)$, while the time for distributed MPC to scale with only $O(1 \times n(C_p)) = O(N_p)$.

Figure 2.8 shows means and standard deviations of computation time versus number of subsystems, for both CMPC and DMPC. As expected, computation time for DMPC scales much more favourably. Best fit analysis suggests that computation time for CMPC tends to grow with $(N_p)^{2.7}$ while DMPC tends to grow with $(N_p)^{1.2}$. These scalings match the predictions. Note that for small values for $N_p$, measured times are slightly higher than predicted, possibly due to overheads, for example, the passing and receiving of data to and from the solver. Also note that the variability of computation time is much smaller for DMPC than for CMPC. The immediate conclusion is that the distributed algorithm offers better scalability with the number of subsystems.

The purpose of the next example is to compare performance of the distributed method against that of centralized, and, additionally, CT-DMPC.

Example 2.3 (Performance versus communication). The problem setup is changed to represent a more commonplace constrained optimal control problem [7], in which the objective function is the quadratic form

$$l_p(x_p, u_p) = x_p^T Q x_p + u_p^T R u_p,$$

$$F_p(x_p) = x_p^T P x_p,$$

where $Q = I_2$, $R = 0.01$, and

$$P = \begin{bmatrix} 2.0066 & 0.5099 \\ 0.5099 & 1.2682 \end{bmatrix}$$
Figure 2.8: Computation time per update versus number of subsystems, for CMPC and DMPC. (Top), mean and standard deviations on a linear scale; (bottom) means and empirical relationships on log-log scale.
Figure 2.9: Comparison of distributed and centralized terminal sets. The terminal set is the maximal output-admissible invariant set for, respectively, \((A_p, B_p, K_{LQR}^p)\) and \((A, B, K_{LQR})\), where \(A = \text{diag}(A_1, \ldots, A_N)\), etc. The distributed sets are truncated by the requirement for independent coupling constraint satisfaction.

is the terminal cost matrix associated with the optimal, nominal, unconstrained LQR problem \((A_p, B_p, Q, R)\). The tube controller \(K_p\) is again the nilpotent controller, and this equally applies to the feedback controller for the CT-DMPC formulation, i.e., \(K_p(j) = K_p, \forall j\). Conversely, the terminal control law is chosen as the LQR controller, i.e., \(\kappa_{F_p}(x_p) = K_{LQR}\), where

\[
K_{LQR} = \begin{bmatrix}
-0.6609 & -1.3261
\end{bmatrix}.
\]

Subsequently, the terminal sets \(X_{F_p}\) for the distributed algorithm are the maximal output-admissible invariant sets \(O_{P}^\infty\) associated with this control, in which coupling constraints are satisfied in a decoupled manner; i.e., \(x_{p,1} \leq 0.5 \Delta x\) for each \(p\). Note that the nilpotent feedback controller means such a nominally-invariant set is also a valid choice for CT-DMPC [14]. However, the centralized algorithm is provided with a centralized version of this set, in which coupling constraint satisfaction is achieved in a centralized, rather than decoupled, sense. Figure 2.9 compares these terminal sets.
A number of simulations were performed, varying the number of subsystems, the update sequence and the maximum separation distance, $\Delta x$, with an initial state $x_p(0) = [20 \ 0]^T \forall p \in \mathcal{P}$, and a horizon of 20 steps. The update sequence was varied in a different manner for CMPC, DMPC and CT-DMPC. For CMPC, a simple mark-space scheme was employed, where a mark represents an updating step and a space represents a zero-update step. The resulting sequence is repeated periodically to form the update sequence for the simulation. For example, for a mark value of 3 and a space value of 2, the resulting sequence is $\{c, c, c, 0, 0, c, c, c, 0, 0, \ldots\}$, where $c$ denotes a centralized optimization.

For DMPC, a similar mark-space scheme is used, but with an additional degree of freedom. It is assumed that the subsystems optimize in a cyclical manner. Then, $n_1$ denotes the number of repetitions of update steps per subsystem (marks), $n_2$ denotes the number of zero-update steps (spaces), and $n_3$ denotes the number of extra zero-update steps that follow the completion of a cycle. For example, with $n_1 = 2, n_2 = 3, n_3 = 4$:

$$\left\{c, \underbrace{1, 1, 0, 0, 0, 2, 2, 0, 0, \ldots}_{n_1}, \underbrace{N_p, N_p, 0, 0, 0, 0, 0, 0, \ldots}_{n_2}, \underbrace{0, 0, 0}_{n_3}\right\},$$

where $c$ denotes the initial centralized step.

Finally, the update sequence for CT-DMPC was chosen to resemble to centralized sequence, but where a mark step corresponds to all agents updating in the preset sequence $\{1, 2, \ldots, N_p\}$. This amounts to employing the algorithm in its originally-intended, sequential manner [38], yet permitting the communication levels to vary by introducing zero-update steps where all agents adopt the candidate plans. Each algorithm is initialized with an optimal centralized plan at $k = 0$.

Figure 2.10 shows plots of closed-loop cost against communication, in which a 'good' controller is one whose data point lies close to the bottom left of the graph. Results are shown as the convex hulls of points obtained for each controller by varying the update sequence, and as (i) the number of subsystems varies (left to
right), (ii) the separation distance $\Delta x$ increases (top to bottom). The measure of performance in this instance is the value of the stage cost, summed over the duration of the simulation and all subsystems. As discussed in Section 2.5, the measure of communication is the number of data exchanges between subsystems. As expected, all the graphs for both DMPC and CMPC show a trade: better performance can be achieved by using more communication.

Firstly, on the comparison between centralized and distributed tube MPC, in the majority of cases, the plots show regions where the closed-loop objective values for tube DMPC are lower than the corresponding CMPC values for the same level of communication. Predictably, at very high levels of communication, CMPC has better performance than DMPC. This is intuitive since DMPC solves the same optimization but in a more constrained manner. However, at low levels of communication, DMPC can perform better. This is enabled by the extra degree of freedom in the DMPC update sequence, breaking the link between computation and communication levels. In tube DMPC, it is possible to construct an update sequence in which a subsystem replans at every step, but communication is required far less frequently. Furthermore, the range of communication for which DMPC outperforms CMPC can be seen to increase as either $\Delta x$ increases or $N_p$ decreases. These movements correspond to making the optimization less tightly coupled, thus giving more flexibility for local decision making.
Comparing these results to those obtained for CT-DMPC, that method obtains, in the majority of cases, better performance than tube DMPC at all levels of communication. Furthermore, in most cases performance is better than for centralized tube MPC. The CT-MPC, even with static feedback, achieves robustness with less conservativeness than for tube MPC, as discussed in Section 2.3.3. Although communication for CT-DMPC can scale poorly as the number of subsystems increases, even then instances exist where performance at low communication levels is better than any tube MPC implementation. However, and crucially, the CT-DMPC algorithm relies on instantaneous inter-agent transfers of data during a time step, while none of the tube DMPC exchanges require this. The number of instantaneous exchanges increases linearly with the number of subsystems.

Recalling the introduction to this chapter, the fundamental reason for implementing distributed control methods is some inherent limitation in available communication, computation, or both; it is generally accepted that where neither is a limiting factor, centralized MPC shall always provide better performance. The remainder of this thesis will seek to answer the important question of how to more closely match that centralized performance. In fact, it will be shown that the algorithms presented so far may lead to poor closed-loop performance, brought about by ‘greedy’ decisions being made by non-cooperative agents; a cooperative form of the DMPC shall be presented, which seeks to promote better system-wide performance.

2.6 Summary

A new form of robust distributed model predictive control of linear, time-invariant subsystems coupled through the constraints has been presented, in which the order of optimization for each subsystem is unrestricted and communication between subsystems is required only when relevant updates are performed. The new formulation extends the tube MPC concept to a distributed implementation and inherits its property of robust stability despite persistent disturbances. While a formal com-
parison of the tube and constraint-tightening methods for robustness has shown the
former to be more conservative, it has been shown that for a distributed implementa-
tion, the tube method leads to greater flexibility in communications. Furthermore,
by exploiting this greater flexibility, better performance can be achieved than for
centralized MPC when communication is limited.
Chapter 3

Distributed MPC with Cooperation

The distributed MPC algorithm is extended to promote cooperation between subsystem agents. In many situations, conflict naturally arises between agents' objectives, leading to 'greedy' behaviour and 'deadlocks'. By a local DMPC agent designing hypothetical trajectories for other subsystems in some cooperating set, a greater portion of the system-wide objective is considered than was previously, and local performance is sacrificed to benefit system-wide, or team, performance. Robust feasibility and stability is maintained for the cooperative form of the algorithm, as is the flexibility in communications, and for any choice of cooperating set. By simulation it is shown that performance improves over the 'greedy' performance of non-cooperative DMPC.

3.1 Introduction

In the previous chapter, a distributed MPC algorithm was developed that guaranteed feasibility and stability despite uncertainty, and allowed greater flexibility
in communication than existing methods. By only decoupled agents optimizing simultaneously, and communicating their new plans to others, the control agents coordinate to satisfy coupling constraints.

An important challenge remaining, however, is that of obtaining good system-wide performance when agents' objectives are conflicting in some way. Here, cooperation is promoted by choice of an optimization objective function that includes the local objectives of other subsystems in a cooperating set of the updating subsystem, assuming the system-wide objective is a summation of local objective functions. A local agent now designs not only its own trajectory, but also hypothetical trajectories for agents in this set, so that a combined cost is minimized. It is shown, by simulation, that global performance can be improved over the simple, 'greedy' implementation of the previous chapter, while robust feasibility and stability guarantees are maintained. Because only one subsystem—or only non-coupled subsystems—optimizes at a time step, subject to previously-published plans of others, conflicting solutions to neighbouring problems never arise, and coupling constraint satisfaction is assured. Furthermore, no iteration takes place. Therefore, this work differs from the work of Keviczky et al. [45], which does not guarantee coupled constraint satisfaction, and also the iterative cooperative schemes of Venkat et al. [36], Waslander et al. [37], Inalhan et al. [75].

The organization of the chapter is as follows. Firstly, in Section 3.1.1, the development is motivated by an example showing 'greedy' behaviour for a pair of vehicles. Subsequently, the cooperative form of DMPC is derived: an embedded optimization, for use as the local cost function, is proposed in Section 3.2.1; implementation considerations are provided in Section 3.2.2; and, in Section 3.2.3, the vehicles example is revisited to show how the cooperative method improves on previous performance. Stability conditions for the new algorithm are presented in Section 3.3. Then, Section 3.4 includes a number of numerical simulations, comparing cooperative DMPC with its non-cooperative counterpart, and the chapter is summarized in Section 3.5.
3.1.1 Motivation: greedy behaviour in DMPC

The cooperative method to be developed in this chapter is perhaps best motivated by an example, in which it is shown that using the DMPC algorithm can lead to undesirable closed-loop behaviour when conflict exists in agents’ objectives.

Example 3.1 (Greedy behaviour in DMPC). Consider a set $\mathcal{P}$ of homogeneous vehicles, modelled by the point mass dynamics

$$
\begin{align*}
\dot{r}_p &= v_p, \\
m_p\dot{v}_p &= f_p + d_p,
\end{align*}
$$

where $r_p \in \mathbb{R}^2$, $v_p \in \mathbb{R}^2$ represent, respectively, the position and velocity of vehicle $p \in \mathcal{P}$, which has mass $m_p$, and $d_p$ is an additive disturbance to the control force $f_p$. These dynamics are discretized with a time step of 1.5 seconds to provide the linear state-space model (2.1), with state $x_p = [r_p^T \quad v_p^T]^T \in \mathbb{R}^4$. The output constraints take the form of local speed and applied force limits:

$$
\begin{align*}
\|v_p\|_2 &\leq V_{\text{max}}, \\
\|f_p\|_2 &\leq F_{\text{max}},
\end{align*}
$$

and the disturbance is limited to 10% of the maximum control force, i.e., $\|d_p\|_\infty \leq 0.1F_{\text{max}}$. Note that these speed and force limits are based on the 2-norm, and are consequently non-linear constraints. However, polyhedral constraints are desirable for implementation, as, for a suitable choice of objective function, they may lead to the on-line MPC optimization problem being a Linear or Quadratic Program. These are ‘standard’ problems, easier to solve than are general non-linear convex problems [86, 91]. Fortunately, 2-norm constraints may be approximated by polyhedra—a linear approximation to the circle—with only small errors [94].
Coupling between vehicles arises from collision avoidance constraints, expressed as a minimum separation distance $L$ between each pair of vehicles:

$$\| r_p - r_q \|_\infty \geq L, \forall p \in \mathcal{P}, q \in \mathcal{P} : p \neq q.$$ 

In practice, these constraints are implemented using the ‘big-$M$’ approach (Remark 2.5, Chapter 2), applied at each step of the prediction horizon. The $\infty$-norm is used to keep the number of binary decision variables smallest, resulting in a square exclusion region around each vehicle.

The objective for a vehicle $p$ is to be steered close to a target state, a position $t_p$ where the velocity is zero. The nominal stage cost of interest is

$$\| \dot{r}_p - t_p \|_2.$$ 

Again, a polyhedral approximation to this 2-norm function is used, rendering the optimization objective (2.15) linear.

$$l_p(x_p, u_p) = \| P_t (r_p - t_p) \|_\infty, \quad (3.1)$$

where the $i^{th}$ row of $P_t$ is

$$P_{t,i} = \begin{bmatrix} \cos(2\pi i / m) \\ \sin(2\pi i / m) \end{bmatrix}, \quad (3.2)$$

such that the linear form is an $m$-sided polyhedral approximation to the circular 2-norm.

The feedback matrix $K_p$ is the nilpotent controller for the system $(A_p, B_p)$, and the set $\mathcal{R}_p$ is the corresponding mRPI set. Constraints are tightened accordingly, including the non-convex avoidance constraints by the method of Remark 2.5. The terminal set $\mathcal{X}_{F_p}$ for each vehicle is equal to the target state, and the terminal cost $F_p = 0$. Under these conditions, asymptotic convergence of the perturbed vehicles
Figure 3.1: Illustration of ‘greedy’ behaviour for a pair of vehicles. The centralized controller (left) produces a cooperative response, whereas for distributed MPC (right), the low-cost, straight-line trajectory assumed by one vehicle forces the other to adopt a deviated route.

is guaranteed to the RPI set around the target state; that is, $[t_p^T \quad 0^T]^T + \mathcal{R}_p$ (Theorem 2.3).

Two such vehicles, to be controlled by DMPC, are required to traverse a 5 m diameter circle from opposing ends. A straight line path for both would lead to a collision. The initialization solution is sub-optimal, in the sense that one of the vehicles is provided a straight-line plan, whilst for the other a deviated plan is formed to avoid collision.

The centralized and distributed algorithms were implemented with a horizon length of 25 steps, and $V_{\text{max}} = 0.225 \text{ m/s}$, $F_{\text{max}} = 0.08 \text{ N}$, $m_p = 1 \text{ kg}$, $L = 1 \text{ m}$. The results are shown in Figure 3.1 for each control algorithm. Each vehicle is subjected to a sequence of random disturbances over the duration of the simulation. The update sequence employed is the simple alternating sequence, so that vehicle agents optimize plans in sequence. For DMPC, the vehicle travelling from North to South (‘vehicle 1’) follows a desirable straight line path, leaving the other vehicle (‘vehicle 2’) to deviate to avoid collision in the centre. Vehicle 1 has no incentive at any point to adopt a higher cost plan than the one it is following, or to make
any allowances for vehicle 2. For centralized MPC, the solution is cooperative: both vehicles deviate equally and oppositely to avoid collision.

This example—which shall be revisited later in this chapter—illustrates the so-called ‘greedy’ behaviour that DMPC may be prone to, where it is not in the interests of a local control agent to sacrifice local performance for the good of system-wide, or team, performance. Many other examples exist, from deadlock situations for multiple robots [95] to the theory of non-cooperative games and the Nash equilibrium concept [96]. In fact, the areas of distributed optimal control and non-cooperative game theory are closely related, and this will be examined in more depth in a later chapter. For now, however, it is simply noted that without any incentive for a local agent to adopt more altruistic strategies, cooperation and its associated advantages will not be assured. The development of the cooperative form of the DMPC algorithm proceeds by providing such an incentive, by modification of the local objective function in the local optimization to encourage local, non-cooperative agents to behave in a cooperative manner.

3.2 Distributed MPC with cooperation

In this section, the main contribution of the chapter is developed, a cooperative form of the robust DMPC algorithm from Chapter 2. The key observation is that robust feasibility was there guaranteed for any choice of objective function. The development proceeds, therefore, by identifying a cooperation-promoting objective function to be employed in the local optimization problems.

The problem statement follows that of the previous chapter. The system-wide, infinite-horizon objective function (2.9) is assumed to be decoupled; specifically, a summation of individual, local objectives:

\[
\min_{p=1}^{N_p} \sum_{k=0}^{\infty} l_p(x_p(k), u_p(k)).
\]
Recall that a local agent in DMPC seeks to minimize a finite-horizon approximation to its individual share of that objective. That is, the local problem \( P_p^D(x_p(k); Z^*_p(k)) \) was defined as

\[
V_p^{opt}(x_p(k); Z^*_p(k)) = \min_{U_p(k)} V_p(U_p(k))
\]

\[
= \min_{U_p(k)} \left\{ F_p(x_p(k + N|k)) + \sum_{j=0}^{N-1} l_p(x_p(k + j|k), u_p(k + j|k)) \right\},
\]

subject to constraints (2.30).

The decision variable for this optimization is the collection of initial state and sequence of predicted control inputs over the horizon:

\[ U_p(k) = \{ \bar{x}_p(k|k), \bar{u}_p(k|k), \bar{u}_p(k + 1|k), \ldots, \bar{u}_p(k + N - 1|k) \}. \]

Whereas in the previous chapter no such distinction was required, though it was mentioned, here and throughout this chapter two different local objectives will be referred to: (i) the local optimization objective, \( V_p \), defined as the objective function that subsystem agent \( p \) attempts to minimize in its local optimization, and (ii) the local subsystem objective, \( J_p \), defined as the finite-horizon approximation to the objective for the subsystem \( p \), pertaining to, for example, its mission, task, or performance index. Specifically, if the performance measure for the system is the decoupled function

\[
\sum_{k=1}^{\infty} \sum_{p=1}^{N_p} l_p(x_p(k), u_p(k)),
\]

then the part corresponding to a subsystem \( p \) is the summation

\[
\sum_{k=1}^{\infty} l_p(x_p(k), u_p(k)).
\]
The local subsystem objective is then defined as the finite-horizon approximation to this:

\[ J_p(U_p(k)) = F_p(\hat{x}_p(k + N|k)) + \sum_{j=0}^{N-1} l_p(\hat{x}_p(k + j|k), u_p(k + j|k)), \]

based on predicted (nominal) states and inputs. Thus, in the formulation of the previous chapter, the local optimization attempts to minimize the local subsystem objective function:

\[ V_p(U_p(k)) = J_p(U_p(k)). \]

Suppose this local optimization objective \( V_p \) were extended to include a larger portion of the system-wide objective, with the aim of promoting cooperation and a consideration of others by a local agent; that is, \( V_p \) is no longer necessarily equal to \( J_p \). To facilitate this, define the cooperating set of an optimizing agent \( p \), at time \( k \), as some arbitrary set \( N_p(k) \subseteq P \setminus \{p\} \), to be chosen by the designer. Then we proceed by choosing \( V_p(U_p(k)) \) to itself be an optimization of the local function \( J_p \) for \( p \) plus some weighting of these local functions for those subsystems in the cooperating set. The decision variable for this optimization includes a hypothetical (nominal) trajectory for each of the cooperating set subsystems that, when taken with the predicted (nominal) trajectory for \( p \), would minimize the combined costs. The basic premise is that, by \( p \) considering other agents’ objectives, the new plan for \( p \) may leave a better option for another subsystem, and so may permit an improvement to the global behaviour; unlike in the non-cooperative optimization, there is some allowance to sacrifice local cost in favour of improving global, system-wide performance. However, subsystems in the cooperating set of \( p \) are in no way bound to follow these trajectories, which are never communicated.
3.2.1 A local objective function for cooperation

Define the objective function $V_p$ for the local optimization $P_p^D(x_p(k); z_p^*(k))$ itself to be an optimization problem

$$V_p(U_p(k)) = \min_{U_{N_p}(k)} \left\{ J_p(U_p(k)) + \sum_{q \in N_p(k)} \alpha_{pq} J_q(U_q(k)) \right\} \quad (3.3)$$

subject to, $\forall j \in \{1, \ldots, N-1\}, q \in N_p(k)$:

$$\dot{x}_q(k+j+1|k) = A_q \dot{x}_q(k+j|k) + B_q \dot{u}_q(k+j|k), \quad (3.4a)$$

$$\dot{x}_q(k|k) = x_q^*(k|k-1), \quad (3.4b)$$

$$\dot{u}_q(k|k) = u_q^*(k|k-1), \quad (3.4c)$$

$$\dot{x}_q(k+N|k) \in X_{F_q}, \quad (3.4d)$$

$$\dot{y}_q(k+j|k) = C_q \dot{x}_q(k+j|k) + D_q \dot{u}_q(k+j|k), \quad (3.4e)$$

$$\dot{y}_q(k+j|k) \in Y_q, \quad (3.4f)$$

$$\forall c \in C_q : \quad \dot{z}_{cq}(k+j|k) = E_{cq} \dot{x}_q(k+j|k) + F_{cq} \dot{u}_q(k+j|k), \quad (3.4g)$$

and $\forall c \in C_{N_p(k)} = \bigcup_{i \in N_p(k)} C_i$:

$$\ddot{z}_{cp}(k+j|k) + \sum_{q \in N_p(k)} \dot{z}_{cq}(k+j|k) + \sum_{r \in P_c \setminus (P_{N_p(k)})} \ddot{z}_{cr}^*(k+j|k) \in \dot{Z}_c, \quad (3.4h)$$

where $\ddot{z}$ denotes a predicted, hypothetical value for a subsystem in the cooperating set. The decision variable set for this optimization is the collection of nominal trajectories for all in cooperating set: $\hat{U}_{N_p}(k) = \{ \hat{U}_q(k) \}_{q \in N_p(k)}$, where

$$\hat{U}_q(k) = \{ \dot{x}_q(k|k), \dot{u}_q(k|k), \ldots, \dot{u}_q(k+N-1|k) \}.$$
Figure 3.2: Illustration of cooperation concept. Agent 1, optimizing, designs its own plan $\mathbf{U}_1^{\text{opt}}$, and a hypothetical plan $\tilde{\mathbf{U}}_2^{\text{opt}}$ for agent 2 to minimize a combined objective. The latter plan may deviate from the previously-published $\mathbf{U}_2^*$ only after the second prediction step. The initial mismatch between $\mathbf{U}_1^*$ and $\mathbf{U}_1^{\text{opt}}$ arises because of the constraint $x_p(k) - \bar{x}_p(k|k) \in \mathcal{R}_p$. The plan $\tilde{\mathbf{U}}_2^{\text{opt}}$ must be consistent, in terms of satisfying coupling constraints, with both $\mathbf{U}_1^*$ and $\tilde{\mathbf{U}}_2^{\text{opt}}$. This is indicated by dotted lines for the penultimate point of each plan.

The result is that the local optimization minimizes a further optimization, involving these hypothetical plans for cooperating subsystems, while still subject to the original constraints (2.30). The presence of both sets of constraints is crucial. Effectively, two different representations of a plan for a cooperating subsystem $q \in \mathcal{N}_p$ appear in the local optimization for $p$: firstly, a previously-published plan, $\mathbf{U}_q^*$, originating from the last time that $q$ optimized, and the plan that subsystem is currently following; secondly, a hypothetical plan, $\tilde{\mathbf{U}}_q$, designed locally by agent $p$, based on minimizing the combined cost (3.3). Figure 3.2 illustrates this concept. Precise details and implications of the coupling constraints applied will be discussed in the next section. The initial constraints (3.4b) and (3.4c) provide the starting point of the hypothetical trajectory for each $q \in \mathcal{N}_p(k)$. These constraints act on the assumption that any cooperating subsystem $q$ can not optimize its own plan until, at the earliest, the next time step $k+1$. Hence, these predicted trajectories...
shall only begin to diverge from the previously-published trajectories at the $k+1$ prediction step. Note, however, that these hypothetical plans are internal to $p$'s local decision-making and are not communicated to other agents. Moreover, there is no obligation for a cooperating set subsystem to itself optimize at this next step or indeed ever adopt this plan. The main point is that the optimizing subsystem $p$ is allowed to change its own plan by considering what others may be able to achieve should it do so.

**Remark 3.1 (Initial constraints and the update sequence).** The assumption that no other agent may optimize until the next time step appears logical, given the single-update nature of the DMPC. However, two further considerations could be applied. Firstly, in Remark 2.6, it was identified that non-coupled subsystems may optimize simultaneously; therefore, there may be a justification to omit the constraint (3.4c) for these agents, thus permitting deviations of the hypothetical plan from the current plan immediately after the current state. This is implemented in Example 4.3 in Chapter 4. In this way, a local agent $p$ may attempt to forecast what an agent $q$ is currently planning. On the other hand, an opposing argument is that although $q$ could be planning now, it will not learn of $p$'s revised intentions until the next step.

Secondly, if an optimizing agent $p_k$ at a step $k$ has full knowledge of the future update sequence $\{p_{k+1}, p_{k+2}, p_{k+3}, \ldots\}$, then it may constrain the hypothetical plan for an agent $p_{k+j} \in \mathcal{N}_{p_k}$ up to the step $k+j$, at which that agent is due to optimize, as any deviations before then will certainly not be adopted. However, here we avoid specific assumptions on a priori knowledge of the update sequence, and, in keeping with the communication analysis in the previous section, choose to leave open the possibility of any agent optimizing at the next step.

The cooperating set $\mathcal{N}_p(k)$ and the scalar weightings $\alpha_{pq}$ are essentially flexible tuning parameters for the level of cooperation. The size of the cooperating set maps to what portion of the system-wide objective is considered in the local optimization. If $\mathcal{N}_p(k)$ is empty, the objective $V_p(\mathbf{U}_p(k))$, defined by (3.3) subject to (3.4), reverts simply to the function $J_p(\mathbf{U}_p(k))$, as in the standard, non-cooperative DMPC intro-
duced in the previous chapter. Conversely, as $\mathcal{N}_p(k) \rightarrow \mathcal{P}\setminus\{p\}$, the local optimization attempts to solve a problem more closely resembling the centralized problem $\mathcal{P}^C(x_1(0), \ldots, x_{N_p}(0))$. Of course, if the distributed optimization is of comparable size to the centralized optimization, the obvious question is why not have each agent just solve the centralized optimization locally? This is comparable to the approach of Keviczky et al. [45], only in that work optimizations were only partially grouped. The answer is, firstly, that each agent would, prior to optimizing, require a current measure of state from all other agents. Secondly, and perhaps more importantly, such an approach relies on all agents optimizing simultaneously and arriving at the same solution; mismatch can lead to infeasibility or instability [45].

The effects of cooperation will be illustrated by examples later in the chapter. At this point, it is merely noted that the choice of cooperating set is chosen by the designer and unrestricted. In Section 3.4, the effect on performance of different choices is explored for an example multiple vehicle system. Then, in the next chapter, a formal analysis of cooperation is provided, and an adaptive algorithm proposed whereby agents make the cooperating set decision on-line, based on currently-active constraints.

The parameter $\alpha_{pq} \geq 0$ is the weighting applied to the local subsystem objective $J_q$ for $q \in \mathcal{N}_p(k)$; smaller values ($\alpha_{pq} < 1$) place more emphasis on $p$'s own objective and self-interest, whilst larger values ($\alpha_{pq} > 1$) have the opposite effect.

Finally, because robust feasibility of the DMPC algorithm is guaranteed for any choice of optimization objective function (Proposition 2.4), the adoption of such an embedded optimization as the objective function maintains this result, regardless of the choice of $\mathcal{N}_p(k)$. Furthermore, the communication framework and timing developed in that chapter is unchanged, as hypothetical plans are never communicated—though, as the next section shows, the set of agents with whom an optimizing agent must communicate may be enlarged. Stability is a separate matter, and will be addressed later in the chapter.
Algorithm 3.1: Robust DMPC with cooperation, for a subsystem $p$

1. Design stabilizing controller $K_p$ and RPI set $R_p$;
2. Tighten sets $\mathcal{Y}_p, \mathcal{Z}_c, \forall c \in \mathcal{C}_p$, via (2.16);
3. Wait for feasible solution $U_p(0)$, information $\hat{Z}_p^*(0)$, and terminal set $\mathcal{X}_{F_p}$ and controller $\kappa_{F_p}$ from central agent;
4. for $k = 1 : \infty$
   
   5. Sample current state $x_p(k)$;
   6. if $p_k = p$ then
      
      7. Choose cooperating set $\mathcal{N}_p(k)$ and weightings $\alpha_{pq}, \forall q \in \mathcal{N}_p(k)$;
      8. Obtain new plan $U_p(k) = U_p^{opt}(k)$ as solution to $p_p^{D, \mathcal{N}_p(k)}(x_p(k); \hat{Z}_p^*(k))$;
      9. Transmit new plan to other agents;
   10. else
       
       11. Renew current plan via (2.19): $U_p(k) = U_p(k)$;
   12. end
   13. Apply control (2.17): $u_p(k) = u_p(k|k) + K_p(x_p(k) - x_p(k|k))$;
   14. Wait one time step;
   15. end

3.2.2 Implementation

For practical purposes, when the optimization (3.3) subject to (3.4) is incorporated as the objective function for the DMPC optimization problem $p_p^D(x_p(k); Z_p^*(k))$, defined in Section 2.4, the combined form is a single cooperative optimization problem, denoted $p_p^{D, \mathcal{N}_p(k)}(x_p(k); \hat{Z}_p^*(k))$, and defined as

$$V_{p}^{opt}(x_p(k); \hat{Z}_p^*(k)) = \min_{U_p(k)} V_p(U_p(k))$$

$$= \min_{\{U_p(k), \hat{U}_{\mathcal{N}_p(k)}\}} \left\{ J_p(U_p(k)) + \sum_{q \in \mathcal{N}_p(k)} \alpha_{pq} J_q(\hat{U}_q(k)) \right\}$$

subject to (2.30) and (3.4). This cooperative form of the optimization is employed in Algorithm 3.1.

Suppose an agent $p_k$ is to solve this problem at time step $k$. In the previous chapter, it was identified that information from other subsystems was required to evaluate each coupling constraint $c \in \mathcal{C}_{p_k}$ in the local optimization. A message for
an agent $p$ involved in a coupling constraint $c$ was defined as

$$m_{cp}(k) = \begin{bmatrix} z^*_c(k|k)^T & \ldots & z^*_c(k+N-1|k)^T & x^*_p(k+N|k)^T \end{bmatrix}^T,$$

where $^*$ again denotes an argument of a feasible solution. That is, the collection of coupling outputs for constraint $c$ and the terminal state. Requirement 2.1 stated that $p_k$ must have received messages $m_{cq}(\hat{k}_q)$, for all $c \in C_{pk}$, from all coupled subsystems $q \in Q_{pk}$ since their respective last update times $\hat{k}_q$. Subsequently, a local agent is able to propagate a message to the current time step, without requiring additional information.

Inspection of the cooperative optimization, however, shows that the local agent for $p_k$ may require information additional to this. Firstly, a cooperating subsystem $q \in N_{pk}(k)$ may not be coupled to $p_k$, i.e., $q \notin Q_{pk}$, so that $p_k$ has received no information from this other subsystem. Secondly, the additional set of coupling constraints to be evaluated is given by the union $C_{N_{pk}(k)} \triangleq \bigcup_i \in N_{pk}(k) C_i$; it is possible that $C_{N_{pk}(k)} \supset C_{pk}$ even if $N_{pk}(k) \subseteq Q_{pk}$.

The remainder of this section will define precisely what information is required. The collection of information that the control agent for $p_k$ requires is denoted $Z^*_{pk}(k)$ to distinguish it from the $Z^*_k(k)$ of problem $D_{pk}(x_{pk}(k); Z^*_p(k))$. Firstly, the following standing assumption shall apply to the subsequent analysis.

**Assumption 3.1 (Construction of outputs from $U_q(k)$).** Each agent $p \in P$ has a priori knowledge of static model parameters for all other subsystems, including dynamics $(A_q, B_q)$, terminal controller $\kappa_F_q$, and constraint sets $C_q, D_q, \tilde{Y}, E_q, F_q, \tilde{V}$, so that, given the plan $U_q(k)$, all predicted states and outputs may be constructed.

Examining the cooperative optimization, to evaluate the initial constraints (3.4b) and (3.4c), the local agent $p_k$ must have knowledge of $\tilde{x}^*_q(k|k-1)$ and $\tilde{u}^*_q(k|k-1)$ for each $q \in N_{pk}(k)$. The last update time $\hat{k}_q$ for a subsystem $q$ was defined by (2.33). Suppose that the plan $U^*_q(\hat{k}_q)$ for a subsystem $q \in N_{pk}(k)$, from this latest update
step, has been made available to the agent $p_k$:

$$U_q^*(\hat{k}_q) = \{ \bar{x}_q^*(\hat{k}_q|\hat{k}_q), \bar{u}_q^*(\hat{k}_q|\hat{k}_q), \bar{u}_q^*(\hat{k}_q + 1|\hat{k}_q), \ldots, \bar{u}_q^*(\hat{k}_q + N - 1|\hat{k}_q) \}.$$  

Then $p_k$ may construct values as required:

$$\bar{u}_q^*(k|k - 1) = \bar{u}_q^*(k|\hat{k}_q),$$
$$\bar{x}_q^*(k|k - 1) = A_q^{(k-\hat{k}_q)} x_q^*(\hat{k}_q|\hat{k}_q) + \sum_{i=0}^{k-\hat{k}_q-1} A_q^i B_q u_q^*(\hat{k}_q + i|\hat{k}_q),$$  

for all $k \leq \hat{k}_q + N - 1$. For greater values of $k$, states and inputs may be constructed using the terminal control law $\kappa_{F_q}(x_q)$.

At this point, it is worth noting the difference between a message and a plan. From a plan, a message may be constructed given knowledge of the system and constraint matrices. However, a plan may not, in general, be reconstructed from only a message. Therefore, this initial state and control information for a subsystem $q \in \mathcal{N}_{p_k}$ will not necessarily have been made available under the message-based communication scheme that satisfies Requirement 2.1, even if $q \in Q_{p_k}$. In many instances, a message is a smaller representation of a plan, and in a convenient format that aids direct evaluation of the coupling constraints without further matrix operations. For example, for vehicle path-planning with collision avoidance, a message might consist of predicted positions only, while a plan includes velocities, steering angles, and other states. The initial states and inputs (3.6) may not be determined from a message $m_{cq}(\hat{k}_q)$, but may be determined from a plan $U_q^*(\hat{k}_q)$.

Returning to the cooperative optimization, the problem includes two sets of coupling constraints.

1. By constraint (2.30g), the coupling outputs $\bar{z}_{\mathcal{C} p_k}(\cdot|k)$ of the optimizing subsystem $p_k$ satisfy the coupling constraints $c \in \mathcal{C}_{p_k}$ when taken with the previously-published outputs $\bar{z}_{cq}(\cdot|k)$ of subsystems $\forall q \in \mathcal{P}_c \setminus \{ p_k \}$, which may include some $q \in \mathcal{N}_{p_k}(k)$. (Even if $p_k$ includes subsystem $q$ in its cooperating set, any
shared coupling constraints are still evaluated with the previously-published plans from $q$).

2. By constraint (3.4h), the sum of hypothetical coupling outputs $\hat{z}_{cq}(\cdot |k)$ over all $q \in \mathcal{N}_{p_k}(k)$ must be consistent with the coupling outputs of $p_k$, and also with the previously-published outputs of all other subsystems coupled to any $q \in \mathcal{N}_{p_k}(k)$. That is, the collection of $\hat{z}_{cr}^*(\cdot |k)$, $\forall c \in \mathcal{C}_{\mathcal{N}_{p_k}(k)}$, where

$$\mathcal{C}_{\mathcal{N}_{p_k}(k)} \triangleq \bigcup_{i \in \mathcal{N}_{p_k}(k)} C_i,$$

is required from all $r$ in the union

$$Q_{\{p_k, \mathcal{N}_{p_k}(k)\}} \triangleq \bigcup_{i \in \mathcal{N}_{p_k}(k)} Q_i \setminus \{p_k, \mathcal{N}_{p_k}(k)\}. \quad (3.7)$$

Motivated by these requirements for information to evaluate initial and coupling constraints, the information requirement for evaluation of the constraints in the cooperative problem $\mathbb{F}^{D, \mathcal{N}_{p}(k)}_p(x_p(k); Z_p^*(k))$ is now stated, and is a modified version of Requirement 2.1.

**Requirement 3.1** (Information requirement for $Z_{p_k}^*(k)$). At a time step $k$, the control agent for an optimizing subsystem $p_k$ must have received

1. plans $U_q^*(k_q)$ from all $q \in \mathcal{N}_{p_k}(k)$;

2. messages $m_{cq}(k_q)$, $\forall c \in \mathcal{C}_{p_k}$, from all $q \in Q_{p_k}$;

3. messages $m_{cr}(k_r)$, $\forall c \in \mathcal{C}_{\mathcal{N}_{p_k}(k)}$, from all $r \in Q_{\{p_k, \mathcal{N}_{p_k}(k)\}}$.

The first part ensures all that initial constraints can be evaluated. Satisfaction of the second part means $p_k$ can evaluate all its coupling constraints, with respect to the previously-published plans of coupled subsystems. The final part means that the coupling constraints for every cooperating subsystem may be evaluated, using the hypothetical plans for $q \in \mathcal{N}_{p}(k)$ with previously-published plans for any
Figure 3.3: Illustration of information requirements for cooperative DMPC. The optimizing subsystem $p_k$ is coupled to $q$ and $r$ (solid lines). The cooperating set $\mathcal{N}_{p_k}$ contains subsystem $r$ only (dashed line). Agent $p_k$ receives a plan from $r$ but only a message from $q$. In addition, $p_k$ receives a message from $s$, which is coupled to $r$, in order to evaluate the coupling constraints (3.4h).

Note that if $\mathcal{N}_{p_k}(k)$ is empty, $p_k$ does not require plans $U_q^\ast(\hat{k}_q)$ from any other agent (part 1), and the union set $\mathcal{Q}_{\{p_k, N_{p_k}\}}$ (3.7) of part 3 becomes empty; hence, Requirement 3.1 reduces to Requirement 2.1. Conversely, if $\mathcal{N}_{p_k}(k) = P\setminus\{p_k\}$, $p_k$ requires plans from all others, but each term $\mathcal{Q}_i\setminus\{p_k, N_{p_k}(k)\}$ of the union in part 3 is empty, and no previously-published outputs appear in constraint (3.4h). In between these extremes, $p_k$ requires information from (i) those in the cooperating set, (ii) those $p_k$ is coupled to, and (iii) those any $q \in N_{p_k}(k)$ is coupled to.

It will be assumed that the communication scheme is sufficient to meet the information requirement. Thus, the communication step in Algorithm 3.1 is modified from that of Algorithm 2.2 to conservatively specify transmission to all other subsystems following update. However, despite the fact that communication may be required between all agents, and not just coupled subsystem agents, many of the flexibility properties identified in the previous chapter are retained. As before, it is sufficient for one agent to transmit its plan to others only after that plan has changed, i.e., as a result of optimization. Thus, data exchanges need not occur at every time step, and the update sequence may be tailored to exploit this flexibility, as was shown by the investigations in Section 2.5.
3.2.3 Cooperative behaviour in DMPC

The following example revisits Example 3.1, where it was shown that two vehicles controlled by DMPC are non-cooperative, and can exhibit poor performance. With the vehicles now controlled by cooperative DMPC, the 'greedy' behaviour previously seen is eradicated.

Example 3.2 (Cooperative behaviour in DMPC). The two vehicles of Example 3.1 are to be controlled by cooperative DMPC (Algorithm 3.1). Given that there are only two vehicles in the problem, the natural choice of each vehicle's cooperating set $\mathcal{N}_p$ is the other vehicle, i.e., $\mathcal{N}_p(k) = \{q\}, \forall p \in \{1,2\} : q \neq p, \forall k$. The weighting $\alpha_{pq}$ in each local optimization is chosen as unity, to place equal emphasis on the local vehicle's objective $J_p$ and the other vehicle's objective $J_q$. All other parameters are unchanged.

Figure 3.4 shows the results, alongside a reproduction of those for the non-cooperative DMPC controllers from Example 3.1. Again the initial plans provided to each vehicle control agent are sub-optimal, favouring the vehicle heading from North to South ('vehicle 1'). The cooperative control scheme delivers a more equal response, overcoming this sub-optimality. Figure 3.5 illustrates the decision-making process that led to this improved performance. The vehicles are shown at time
Figure 3.5: Vehicle plans at time steps $k = \{0, 1, 2, 10\}$, computed with cooperative DMPC. Shown, at each step, are the new local plan (\(\circ\)) and the hypothetical plan (\(\triangle\)) computed by the optimizing agent, and the candidate plan (\(\star\)) of the non-optimizing agent. At $k = 1$, vehicle 1 adopts a more cooperative plan than the previous.

Steps $k = \{0, 1, 2, 10\}$, and current positions and plans are indicated, together with position histories. At $k = 0$, the 'greedy' initialization is evident. The next step, $k = 1$, is crucial: vehicle 1 optimizes, solving the cooperative local optimization; the result is a new plan, depicted, that now deviates from the straight line, together with a hypothetical plan for vehicle 2 that improves on the previously-published plan. Thus, the control agent has sacrificed its own local performance to permit a future benefit for the system as a whole. At the next step, $k = 2$, vehicle 2 takes advantage of this favourable decision to improve on its previous, large-deviation plan. By $k = 10$, the vehicles are successfully executing a now symmetric, 'roundabout'
manoeuvre. Note that following the cooperative decision at \( k = 1 \), all hypothetical plans match the candidate plans, implying no further system-wide improvement is possible.

Further examples using the cooperative form of the DMPC algorithm are provided in Section 3.4, including multiple subsystem examples and consideration of the extra computational load the method imposes.

### 3.3 Stability analysis

In this section, the stability of the cooperative DMPC method is investigated. A key observation in the development of the cooperative algorithm was that, for the DMPC scheme introduced in Chapter 2, robust feasibility is guaranteed for any choice of local optimization objective function, depending only on satisfaction of the constraints. Thus, the local optimization objective function was chosen to be itself an optimization of plans for neighbouring subsystems in some cooperating set, leading to a robustly-feasible cooperative DMPC algorithm. However, whereas stability of the DMPC method of Chapter 2 is assured there upon certain sufficient conditions on each local optimization objective, \( V_p \), being met, such guarantees do not necessarily hold for the new optimization. This is seen in the following. Given a set of feasible solutions \( \{U_1^*(k_0), U_2^*(k_0), \ldots, U_N^*(k_0)\} \) at some \( k_0 \), and assuming that Assumptions 2.2–2.4 are met, it may be shown that

\[
V_p^*(x_p(k_0 + 1); z_p(k_0 + 1)) \leq J_p(U_p^*(k_0)) + \sum_{q \in \mathcal{N}_p(k_0+1)} \alpha_{pq} J_q(U_q^*(k_0))
\]

\[
- \left[ l_p(x_p^*(k_0|k_0), u_p^*(k_0|k_0)) + \sum_{q \in \mathcal{N}_p(k_0+1)} \alpha_{pq} l_q(x_q^*(k_0|k_0), u_q^*(k_0|k_0)) \right],
\]

for any \( p \), using the fact that the candidate solution

\[
\bar{U}_p(k_0+1) = \{ x_p^*(k_0 + 1|k_0), u_p^*(k_0 + 1|k_0), \ldots, u_p^*(k_0 + N - 1|k_0), \kappa_{F_p}(x_p^*(k_0 + N|k_0)) \}
\]
is a feasible solution for \( p \) itself, and the corresponding candidate solution \( \hat{U}_q(k_0 + 1) \) is a feasible choice for the hypothetical plan \( \hat{U}_q(k_0 + 1) \), for each \( q \in \mathcal{N}_p(k_0 + 1) \).

The first difficulty arises because it is possible, in general, for \( J_p(U_p^{\text{opt}}(k_0 + 1)) \geq J_p(\hat{U}_p(k_0 + 1)) \), as in Figure 3.5 (top right), so that the monotonic descent of the local subsystem objective value sequence, \( \{J_p(k)\} \), is not assured. Secondly, \( \mathcal{N}_p(k_0 + 1) \) is not necessarily equal to \( \mathcal{N}_p(k_0) \). Suppose \( \mathcal{N}_p(k_0 + 1) = \{\mathcal{N}_p(k_0), n\} \) for some \( n \in \mathcal{P} \). Then

\[
V_p^*(x_p(k_0 + 1); \bar{Z}_p^*(k_0 + 1)) \leq V_p^*(x_p(k_0); \bar{Z}_p^*(k_0)) + \alpha_{pn} J_n(U_n^*(k_0))
- \left[ l_p(x_p^*(k_0|k_0), \bar{u}_p^*(k_0|k_0)) + \sum_{q \in \mathcal{N}_p(k_0 + 1)} \alpha_{pq} l_q(x_q^*(k_0|k_0), \bar{u}_q^*(k_0|k_0)) \right],
\]

so that it is possible that \( V_p^*(x_p(k_0 + 1); \bar{Z}_p^*(k_0 + 1)) \geq V_p^*(x_p(k_0); \bar{Z}_p^*(k_0)) \), especially for \( \alpha_{pn} \geq 1 \). Intuitively, increasing the size of the cooperating set at each \( k \) may lead to an increasing local optimization cost, rather than a decreasing one.

In this section, therefore, stability of cooperative DMPC is studied, and guarantees are established by two alternative methods: (i) directly by monotonicity of the sequences of an augmented local cost, based on the value functions for the optimization, and (ii) by use of an additional, stability constraint [97] in the local optimization, based on a Lyapunov-like constraint on the value of each local subsystem objective \( J_p \). The first, direct approach relies on the usual assumptions [7] on each of the terminal sets and costs (here Assumptions 2.2–2.4). The second approach relaxes assumptions on the optimization objective required for stability, broadening the class of valid functions to include, for example, products or non-linear combinations of local subsystem costs.

### 3.3.1 Augmented local optimization cost as Lyapunov function

Define a local augmented optimization cost as the sum of local optimization cost \( V_p \) and local subsystem costs \( J_r \) for subsystems \( r \notin \{p, \mathcal{N}_p\} \), i.e., those not in the
cooperating set:

\[ \mathcal{A}(k) = \sum_{q \in \mathcal{N}_p(k)} \alpha_{pq} \mathcal{J}_q(\hat{U}_q(k)) + \sum_{r \notin \{p, \mathcal{N}_p(k)\}} \alpha_{pr} \mathcal{J}_r(\hat{U}_r^*(k)), \]

where the fixed, previously-published plan \( U_r^*(k) \) is used for each \( r \). Then each of these augmented costs has a constant number of terms and is monotonically decreasing in value, regardless of the choice of cooperating sets \( \mathcal{N}_p(k) \) and weightings \( \alpha_{pq}, \forall p \in \mathcal{P} \). The following result assumes that Assumptions 2.2–2.4 are met; namely, the existence for each \( p \in \mathcal{P} \) of a suitable admissible, invariant terminal set, under some terminal control law, and the requirement therein for the terminal cost to be a control Lyapunov function.

**Proposition 3.1 (Monotonicity of the local augmented cost).** Suppose the sequence of controls \( U_p^*(k_0) = \{ \bar{x}_p^*(k_0|k_0), \bar{u}_p^*(k_0|k_0), \ldots, \bar{u}_p^*(k_0+N-1|k_0) \}, \forall p \in \mathcal{P}, \) exists and is a feasible (but not necessarily optimal) solution to \( \mathcal{P}^C(\bar{x}_1(k_0), \ldots, \bar{x}_{N_p}(k_0)) \) at some time step \( k_0 \). Then, for all \( x_p(k_0 + 1) \in \mathcal{A}_p \bar{x}_p(k_0) + \mathcal{B}_p \bar{u}_p(k_0) \oplus \mathcal{W}_p, \forall p \in \mathcal{P}, \) where \( \bar{u}_p(k_0) = \bar{u}_p^*(k_0|k_0) + K_p(x_p(k_0) - \bar{x}_p^*(k_0|k_0)) \), the upper bound on the augmented optimization cost for each subsystem \( p \in \mathcal{P} \) decreases monotonically:

\[ \mathcal{J}_p^*(k_0 + 1) \leq \mathcal{J}_p^*(k_0) - \sum_{i=1}^{N_p} \alpha_{pi} \mathcal{H}_i(\bar{x}_i^*(k_0|k_0), \bar{u}_i^*(k_0|k_0)), \]

where \( \alpha_{pp} = 1 \) and \( \alpha_{pi} \in [0, \infty), \forall i \neq p, \) for any choices \( \mathcal{N}_p(k_0 + j) \subseteq \mathcal{P} \setminus \{p\}, j \in \{0, 1\} \).

**Proof.** Suppose \( U_p^*(k_0) \) is a feasible, but not necessarily optimal, choice of \( U_p(k_0) \) for every \( p \in \mathcal{P} \) at time \( k_0 \), i.e., the collection \( \{ U_1^*(k_0), \ldots, U_{N_p}^*(k_0) \} \) is a feasible solution to \( \mathcal{P}^C(\bar{x}_1(k_0), \ldots, \bar{x}_{N_p}(k_0)) \). Consider the problem \( \mathcal{P}_p^D(\bar{x}_p(k_0), \hat{Z}_p(k_0)) \) associated with each \( p \in \mathcal{P} \), irrespective of whether that \( p \) solves this problem at \( k_0 \). The decision variable consists of the local \( U_p(k_0) \) and the hypothetical \( \hat{U}_q(k_0) \) for each \( q \in \mathcal{N}_p(k_0) \). It is simple to show that, in addition to \( U_p^*(k_0) \) being a
feasible choice of \( U_p(k_0) \), the solution \( U^*_q(k_0) \) is a feasible choice for \( \hat{U}_q(k_0) \) for each \( q \in N_p(k_0) \). The value of the local augmented cost associated with this optimization is then

\[
J^*_p(k_0) = J_p(U^*_p(k_0)) + \sum_{q \in N_p(k_0)} \alpha_{pq} J_q(U^*_q(k_0)) + \sum_{r \notin \{p, N_p(k_0)\}} \alpha_{pr} J_r(U^*_r(k_0))
\]

\[
= J_p(U^*_p(k_0)) + \sum_{i \neq p} \alpha_{pi} J_i(U^*_i(k_0)),
\]

regardless of the choice of \( N_p(k_0) \), and for all \( p \in \mathcal{P} \).

At time \( k_0 + 1 \), consider a problem \( P_p^{D, N_p(k_0+1)}(x_p(k_0+1); \tilde{Z}_p(k_0+1)) \) for any \( p \in \mathcal{P} \). The candidate solution \( \tilde{U}_p(k_0 + 1) \), defined by (2.19), and given by

\[
\tilde{U}_p(k_0+1) = \{ \tilde{x}_p^*(k_0+1|k_0), \tilde{u}_p^*(k_0+1|k_0), \ldots, \tilde{u}_p^*(k_0+N-1|k_0), \kappa_F(\tilde{x}_p^*_p(k_0+N|k_0)) \},
\]

is available to \( p \) as a feasible choice for \( U_p(k_0 + 1) \), and—similarly—for each \( q \in N_p(k_0 + 1) \), a corresponding solution \( \tilde{U}_q(k_0 + 1) \) is a feasible choice for the hypothetical \( \hat{U}_q(k_0 + 1) \). The associated value of the local augmented cost is

\[
J^*_p(k_0 + 1) = J_p(\tilde{U}_p(k_0 + 1)) + \sum_{q \in N_p(k_0+1)} \alpha_{pq} J_q(\tilde{U}_q(k_0 + 1))
\]

\[
+ \sum_{r \notin \{p, N_p(k_0+1)\}} \alpha_{pr} J_r(\tilde{U}_r(k_0 + 1)).
\]

Rewriting, combining with (3.9), and using Assumption 2.4,

\[
\tilde{J}_p(k_0 + 1) \leq J^*_p(k_0) - \sum_{i=1}^{N_p} \alpha_{pi} l_i(\tilde{x}_i^*(k_0|k_0), \tilde{u}_i^*(k_0|k_0)),
\]

where \( \alpha_{pp} = 1 \) and \( \alpha_{pi} \in [0, \infty), \forall i \neq p \).

At this step \( k_0 + 1 \), only one agent, \( p_{k_0+1} \), solves its local optimization problem \( P_p^{D, N_p(k_0+1)}(x_{p_{k_0+1}}(k_0+1); \tilde{Z}_{p_{k_0+1}}(k_0+1)) \), while all \( p \neq p_{k_0+1} \) adopt their respective candidate plans. The (non-augmented) optimization cost associated with choosing
the feasible candidate plans for \( p_{k_0 + 1} \) and each \( q \in \mathcal{N}_{p_{k_0 + 1}} \) is

\[
\tilde{V}_{p_{k_0 + 1}}(x_{p_{k_0 + 1}}(k_0 + 1); \tilde{z}^*_{p_{k_0 + 1}}(k_0 + 1)) \triangleq J_p(U_p(k_0 + 1)) + \sum_{q \in \mathcal{N}_{p_{k_0 + 1}}(k_0 + 1)} \alpha_{pq} J_q(U_q(k_0 + 1)).
\]

It follows that this cost forms an upper bound on the optimal optimization cost:

\[
V^\text{opt}_{p_{k_0 + 1}}(x_{p_{k_0 + 1}}(k_0 + 1); \tilde{z}^*_{p_{k_0 + 1}}(k_0 + 1)) \triangleq J_p(U^\text{opt}_p(k_0 + 1)) + \sum_{q \in \mathcal{N}_{p_{k_0 + 1}}(k_0 + 1)} \alpha_{pq} J_q(U_q^\text{opt}(k_0 + 1)) \leq \tilde{V}_{p_{k_0 + 1}}(x_{p_{k_0 + 1}}(k_0 + 1); \tilde{z}^*_{p_{k_0 + 1}}(k_0 + 1)).
\]

Subsequently, because the local augmented cost is formed simply by adding constant terms, i.e.,

\[
\beta_{p_{k_0 + 1}}(k_0 + 1) = V_{p_{k_0 + 1}}(x_{p_{k_0 + 1}}(k_0 + 1); \tilde{z}^*_{p_{k_0 + 1}}(k_0 + 1)) + \sum_{r \notin \{p_{k_0 + 1}, \mathcal{N}_{p_{k_0 + 1}}(k_0 + 1)\}} \alpha_{p_{k_0 + 1}r} J_r(U_r(k_0 + 1)),
\]

it follows immediately that \( \beta_{p}(k_0 + 1) \) is an upper bound on \( \beta^\text{opt}_p(k_0 + 1) \), for any \( p \), so that

\[
\beta^\text{opt}_p(k_0 + 1) \leq \beta^*_p(k_0 + 1) \leq \beta_{p}(k_0 + 1) \leq \beta^*_p(k_0 + 1) - \sum_{i=1}^{N_p} \alpha_{pi} l_i(x^*_i(k_0|k_0), u^*_i(k_0|k_0)),
\]

where \( \beta^*_p(k_0 + 1) \) corresponds to any general feasible solution, and the result is established.

Of course, having established monotonicity of each local augmented cost, stability results may be derived similar to those in the previous chapter. The nature of the stability again depends on the form of the stage and terminal costs, and the terminal set.

93
3.3.2 Stability-constrained DMPC

In this subsection, the standard DMPC optimization problem is modified to include a stability constraint, based on the approach of Scokaert et al. [97]. The development of the stability-constrained MPC arose because of practical considerations: the on-line provision of globally-optimal solutions to non-linear, non-convex optimal control problems is a considerable challenge, and necessarily—and severely—limits the problem size for viable implementation. The finding of a merely feasible solution is far easier. Thus, the consequence of the approach proposed by Scokaert et al. [97] is that feasibility implies stability.

In the context of the problem statement of this thesis, in the previous chapter it was established that recursive, robust feasibility of the DMPC is guaranteed irrespective of objective function, depending only on constraint satisfaction. It follows that should robust feasibility be maintained despite the addition of a stability constraint, then stability shall be guaranteed for any optimization objective. Subsequently, a valid choice for a function is the embedded optimization proposed in Section 3.2.1. However, the result is not restricted to this objective, and a far wider class of optimization objectives may now be permitted. For example, whereas the 'cooperative' objective of (3.3) was defined by the summation of local costs

\[ V_p(\mathbf{U}_p(k)) = \min_{\mathbf{U}_{N_p}} \left\{ J_p(\mathbf{U}_p(k)) + \sum_{q \in N_p} \alpha_{pq} J_q(\mathbf{U}_q(k)) \right\}, \]

subject to (3.4), it may be of more interest to the designer to consider the product,

\[ V_p(\mathbf{U}_p(k)) = \min_{\mathbf{U}_{N_p}} \left\{ J_p(\mathbf{U}_p(k)) \times \prod_{q \in N_p} \alpha_{pq} J_q(\mathbf{U}_q(k)) \right\}, \]

subject to (3.4). Such an objective is not without precedence: see, for example, Waslander et al. [37]. Therefore, a stability-constrained cooperative DMPC might be useful when the designer is interested in minimizing something other than a summation of local cost functions.
Recall that the local, finite-horizon subsystem objective is given by

\[ J_p(U_p(k)) \triangleq F_p(\bar{x}_p(k \mid N | k)) + \sum_{j=0}^{N-1} l_p(\bar{x}_p(k+j \mid k), \bar{u}_p(k+j \mid k)). \]

Here it is assumed that the terminal cost is zero, \( F_p \triangleq 0 \), and the local stage cost for each \( p \in \mathcal{P} \) is a function \( l_p(x_p, u_p) : \mathbb{R}^{N_z \times N_u \times N_r} \mapsto \mathbb{R}_{0+} \) such that

\[ l_p(x_p, u_p) \begin{cases} \geq d(x_p, X_{F_p}), & \forall x_p \notin X_{F_p}, \\ = 0, & \forall x \in X_{F_p}, u_p = \kappa_{F_p}(x_p). \end{cases} \quad (3.10) \]

As noted in Remark 2.3, minimizing such a function in the local optimization controls the system according to the dual-mode control law, and guarantees asymptotic convergence to a terminal set wherein the control is known. However, here it is proposed, following Scokaert et al. [971], that, rather than this function being the optimization objective function, instead it appears as an additional constraint included in the optimization, enforcing its monotonic descent. The original local optimization problem \( \mathbb{P}^D_p(x_p(k); Z^*_p(k)) \) becomes

\[ V^\text{opt}_p(x_p(k); Z^*_p(k)) = \min_{U_p(k)} V_p(U_p(k)) \quad (3.11) \]

subject to (2.30) and,

\[ J_p(U_p(k)) \leq J_p(U^*_p(k - 1)) - \lambda_p l_p(\bar{x}^*_p(k - 1 \mid k - 1), \bar{u}^*_p(k - 1 \mid k - 1)), \quad (3.12) \]

where, as usual, the superscript * denotes a feasible solution. The scalar \( \lambda_p \in (0, 1] \) is to be chosen by the designer; remarks will be made on the selection of this parameter after the following stability result.

**Proposition 3.2** (Feasibility implies stability). Suppose the sequence of controls \( U^*_p(k_0) = \{x^*_p(k_0 \mid k_0), \bar{u}^*_p(k_0 \mid k_0), \ldots, \bar{u}^*_p(k_0 + N - 1 \mid k_0)\}, \forall p \in \mathcal{P} \), exists and is a feasible (but not necessarily optimal) solution to \( \mathbb{P}^C(x_1(k_0), \ldots, x_{N_r}(k_0)) \) at some
time step \( k_0 \). Then, for all \( x_p(k_0 + 1) \in A_p x_p(k_0) + B_p u_p(k_0) \oplus W_p, \forall p \in \mathcal{P} \), where \( u_p(k_0) = \hat{u}_p^*(k_0 | k_0) + K_p(x_p(k_0) - \hat{x}_p^*(k_0 | k_0)) \), (i) the candidate sequence \( \hat{U}_p(k_0 + 1) \), defined by (2.19), is a feasible solution to \( \mathbb{P}_p^D(x_p(k_0 + 1); Z_p^*(k_0 + 1)) \) with the additional constraint (3.12), and (ii) subsequently, the resulting closed-loop system controlled by Algorithm 2.2 is robustly-feasible and the states \( x_p(k) \to \mathcal{X}_{F_p} \oplus \mathcal{R}_p \) as \( k \to \infty \), for any choice of update sequence.

Proof. Feasibility of \( \mathbb{P}_p^D(x_p(k_0 + 1); Z_p^*(k_0 + 1)) \) with the solution \( \hat{U}_p(k_0 + 1) \) was proven in Proposition 2.4. It remains to shows that \( \hat{U}_p(k_0 + 1) \) satisfies the additional constraint (3.12). For any \( p \in \mathcal{P} \), the value of the Lyapunov-like function \( J_p \) at \( k_0 \) is

\[
J_p(\hat{U}_p^*(k_0)) = \sum_{j=0}^{N-1} l_p(\bar{x}_p^*(k_0 + j|k_0), \bar{u}_p^*(k_0 + j|k_0)).
\]

Subsequently, at the next time step, adoption of the candidate solution by any \( p \in \mathcal{P} \) achieves a value for \( J_p \) of

\[
J_p(\hat{U}_p(k_0 + 1)) = \sum_{j=1}^{N} l_p(\bar{x}_p^*(k_0 + j|k_0), \bar{u}_p^*(k_0 + j|k_0)) \\
= J_p(\hat{U}_p^*(k_0)) - l_p(\bar{x}_p^*(k_0|k_0), \bar{u}_p^*(k_0|k_0)) \\
+ l_p(\bar{x}_p^*(k_0 + N|k_0), \kappa_{F_p}(\bar{x}_p^*(k_0 + N|k_0))).
\]

The final term is zero, as \( \bar{x}_p^*(k_0 + N|k_0) \in \mathcal{X}_{F_p} \) by (3.10). Thus, a solution always exists that satisfies constraint (3.12) for all \( \lambda_p \in (0, 1] \), and feasibility is established.

For (ii), robust feasibility follows immediately by recursion. For convergence, all non-optimizing agents \( p \neq p_{k_0 + 1} \) at \( k_0 + 1 \) adopt the candidate solution \( \hat{U}_p(k_0 + 1) \), with value (3.13). The value for the optimizing agent \( p_{k_0 + 1} \) is constrained thus:

\[
J_{p_{k_0 + 1}}(\hat{U}^*_{p_{k_0 + 1}}(k_0 + 1)) \leq J_{p_{k_0 + 1}}(\hat{U}^*_{p_{k_0 + 1}}(k_0)) \\
- \lambda_{p_{k_0 + 1}} l_{p_{k_0 + 1}}(\bar{x}_{p_{k_0 + 1}}^*(k_0|k_0), \bar{u}_{p_{k_0 + 1}}^*(k_0|k_0)).
\]
Therefore, for any subsystem $p$ at time $k_0 + 1$, the value of $J_p$ for a general feasible solution $U_p^*(k_0 + 1)$ is bounded as

$$J_p(U_p^*(k_0 + 1)) \leq J_p(U_p^*(k)) - \epsilon_p l_p(x_p^*(k|k), u_p^*(k|k)),$$

where $\epsilon_p$ is strictly positive. By construction, all three terms are non-negative. Applying recursively, in the limit $k \to \infty$, $J_p(k + 1) - J_p(k) \to 0$, implying that $l_p(x_p^*(k|k), u_p^*(k|k)) \to 0$. By the definition of $l_p$, this means $x_p^*(k|k) \to x_{FP}$ and $u_p^*(k|k) \to \kappa_{FP}(x_p^*(k|k))$ as $k \to \infty$, which in turn means the states of the perturbed system $x_p(k) \to x_{FP} \oplus R_p$.

The consequence of Proposition 3.2 is that now stability is enforced through the constraints, leaving the choice of optimization objective $V_p(U_p)$ open. It follows directly that, by choosing $V_p(U_p)$ to be the embedded optimization defined by (3.3) subject to (3.4), robust asymptotic stability of each set $x_{FP} \oplus R_p$ is guaranteed, and, moreover, for any choices of cooperating set $N_p$ and associated weightings $\alpha_{pq}$.

Now we turn to the scalar $\lambda_p$ in the stability constraint, which is to be chosen by the designer. In Scokaert et al. [97] this parameter served a purpose of allowing the designer to trade optimality against feasibility; selecting $\lambda_p = 1$ lowers the upper bound on the value of a new solution, leading to nearer-optimal solutions, at the expense of computation time and the risk of failing to find a feasible solution. Conversely, $\lambda_p \to 0$ enlarges the set of feasible solutions [97]. Here the parameter takes on a different purpose, being a means of tuning the degree of flexibility a local agent has in sacrificing local performance when deciding on a new solution. More specifically, suppose an agent $p_k$ is updating at step $k$. The candidate plan $\bar{U}_{pk}(k)$ is available, based on the feasible plan $U_{pk}^*(k)$ from the previous step. The associated value of $J_{pk}$ is, as we have just seen,

$$J_{pk}(\bar{U}_{pk}(k)) = J_{pk}(U_{pk}^*(k - 1)) - l_{pk}(x_{pk}^*(k - 1|k - 1), u_{pk}^*(k - 1|k - 1)).$$
However, by optimization, agent $p_k$ selects a plan $U_{p_k}^{opt}(k)$, with value

$$J_{p_k}(U_{p_k}^{opt}(k)) \leq J_{p_k}(U_{p_k}^{*}(k - 1)) - \lambda_{p_k} t_{p_k}(\bar{x}_{p_k}^{*}(k - 1|k - 1), \bar{u}_{p_k}^{*}(k - 1|k - 1)).$$

Therefore, $\lambda_{p_k} = 1$ implies $J_{p_k}(U_{p_k}^{opt}(k)) \leq J_{p_k}(\hat{U}_{p_k}(k))$, whereas $\lambda_{p_k} \rightarrow 0$ permits $J_{p_k}(U_{p_k}^{opt}(k)) \geq J_{p_k}(\hat{U}_{p_k}(k))$, yet—crucially—because $\lambda_{p_k} > 0$, the monotonicity required for convergence is maintained. Consequently, a larger set of plans may be available to $p_k$—it may select a plan with a higher local cost than for the candidate plan, leaving an option for others to later adopt plans that improve global performance. The stability constraint ensures that, despite what actions $p_k$ may take in a bid to improve global performance, robust stability is guaranteed.

To summarize this section on stability, two alternative methods of guaranteeing closed-loop stability have been presented. The first defined an augmented optimization cost for each local agent, monotonicity of which was shown. The approach is similar to using the system-wide cost, $\sum_{p=1}^{N_p} J_p$, as a Lyapunov function. The result, however, does depend on each local subsystem objective $J_p$ meeting the usual assumptions [7], and the local optimization objective being a summation of these functions.

The second method employs a stability constraint in the local optimization, forcing the local subsystem cost $J_p$ to decrease monotonically. Feasibility then implies stability, and for any choice of optimization objective, including the cooperative optimization for any choices of cooperating set and weightings. The advantage of this approach is that the class of valid optimization objectives is unrestricted, with stability being guaranteed without requiring further analysis for a particular choice of function.

### 3.4 Numerical examples

This section includes a number of further numerical examples using the cooperative form of the DMPC algorithm. In Example 3.3, the two-vehicle system of Example 3.1
is extended to more vehicles. Performance versus computation time is investigated for a number of different choices of cooperating set, and compared with that of non-cooperative DMPC and centralized MPC. Secondly, the algorithm is used for the problem of vehicles in a 'deadlock', with non-cooperative DMPC unable to offer a resolution. It is shown that switching to cooperative DMPC resolves the conflict.

**Example 3.3** (Multi-vehicle collision avoidance). The two-vehicle system is extended to include more vehicles: $N_p$ vehicles are spaced equally around the perimeter of the 5 m–diameter circle, with respective targets at opposite endpoints of each diameter chord. The minimum separation distance, $L$, is reduced to 0.5 m, to make finding an initial feasible solution easier as the number of vehicles, and thus congestion, increases; all other parameters are held constant.

Figure 3.6 shows resulting trajectories for three and four vehicles, when controlled by non-cooperative and cooperative DMPC. In the latter case, the cooperating set choice is, for now, the next vehicle in line to optimize. (The update sequence is again the simple alternating sequence, e.g., \{1,2,3,4,\ldots,1,2,3,4\} for $N_p = 4$). As observed in the previous examples, by employing the cooperative form of DMPC, the vehicles follow paths with more equally shared deviations from the straight line paths; a 'roundabout' manoeuvre. The sub-optimality of the 'greedy' initialization has been reduced. Furthermore, note that these cooperative trajectories have resulted from each local agent considering only one other vehicle in its optimizations; it appears that to achieve a cooperative outcome it is not necessary for local problems to resemble the centralized problem in size.

Next, the number of vehicles is varied, and the resulting performance compared against computation time. The following schemes were used

- **DMPC$^0$**: $N_p = \emptyset$, $\forall p \in \mathcal{P}$, non-cooperative DMPC;

- **DMPC$^1$**: $N_p = 1 + (p \mod N_p)$, cooperation with next-in-line in the update sequence;

- **DMPC$^2$**: $N_p = 1 + (p - 2 \mod N_p)$, previous-in-line in the update sequence;
Figure 3.6: Cooperative DMPC (right column) for three and four vehicles, compared with ‘greedy’ DMPC (left column).
Figure 3.7: Trade between performance and computation time for two, three, and four vehicles.

- DMPC$^3$ : $N_p = \{1 + (p - 2 \mod N_p), 1 + (p \mod N_p)\}$, previous- and next-in-line, for $N_p \geq 3$.

- DMPC$^4$ : $N_p = \{1 + (p - 2 \mod N_p), 1 + (p \mod N_p), 1 + (p + 1 \mod N_p)\}$, previous-, next-, and next-but-one in-line, for $N_p \geq 4$.

Additionally, centralized MPC is implemented to provide benchmark measures of performance and computation.

For each control scheme, the measure of performance is the stage cost for the perturbed system, summed over the simulation, i.e., the closed-loop cost. This measure then corresponds to a cumulative sum, over time, of distances from the target. The computation time at each step of each simulation is measured, and means and standard deviations calculated; these values are averaged over 10 simulations per control scheme. All simulations were performed on a Pentium 4 HT 3.2 GHz with 2,048 MB RAM, using CPLEX 10.1 as the MILP solver.
Figure 3.8: Computation times for DMPC applied to different numbers of vehicles.

Figure 3.7 shows the resulting trades between performance and mean computation time, for two, three and four vehicles. The performance values are normalized with respect to each centralized benchmark, by expressing as percentage increases in cost over the centralized MPC closed-loop costs. In addition, Figure 3.8 shows the ranges and means of computation times for each control scheme. Two general observations may be made: firstly, as expected, non-cooperative DMPC and centralized MPC provide worst and best performance respectively, corresponding with (on average) shortest and longest computation times. Secondly, performance of the DMPC methods is worse as the number of vehicles, and therefore congestion, increases.
For each of the scenarios \((N_p = 2, 3 \text{ and } 4)\), performance is improved over that of non-cooperative DMPC by selecting a cooperative scheme. It is interesting to note that scheme DMPC\(^1\) shows a marginal improvement in performance over that of DMPC\(^2\)—except when \(N_p = 2\) and the two are identical—but for similar computation, indicating that the best choice of cooperating set is dependent on the update sequence. In the cases where the cooperating sets (where possible) include two or three vehicles, rather than one, performance improves further, at the expense of increased computation time, albeit again marginally. Despite now planning hypothetical trajectories for two other vehicles in the cooperating set, only one of those may update at the subsequent time step, because of the single-update nature of the algorithm. The next chapter more formally analyses the 'best' choices of cooperating set.

**Example 3.4** (Deadlock resolution). Consider once more the two-vehicle system, now placed in a planar field of obstacles. Two rectangular obstacles ('walls') are placed so that they form a corridor, centrally from North to South; a vehicle is permitted to pass either between the walls, or outside of them by deviating. However, the spacing between the walls is such that two vehicles may not cross within the corridor without colliding. The upper-left plot of Figure 3.9 shows the arrangement, and also the vehicles' initial positions, \(r_1 = [0 \ 0.75]^T, r_2 = [0 \ -0.75]^T\).

The objective is, from these starting positions, to reach respective targets \(t_1 = [0 \ -2.5]^T, t_2 = [0 \ 2.5]^T\); thus, the vehicles must pass each other to reach their goals. The terminal set \(X_{F_p}\) for each is chosen to be any position with zero velocity. That is,

\[
X_{F_p} = \{x_p \in \mathbb{R}^4 : Sx_p \leq 0\},
\]

where

\[
S = \begin{bmatrix}
0_2 & I_2 \\
0_2 & -I_2
\end{bmatrix}.
\]
To ensure admissibility of this terminal set, the optimization constraints are extended to additionally cover the terminal prediction step \( j = N \), using \( \tilde{u}_p(k+N|k) \) as the control. The terminal set is then a safety set [49], a set in which the vehicle may remain indefinitely under some known terminal control law—a suitable terminal control law for this example is \( \kappa_{F_p} = 0 \). Such a terminal constraint is of particular interest in path planning for vehicles in unknown environments, where the success of the mission is not guaranteed, yet feasibility—in this case, collision avoidance—is required at all times [49]. This formulation also permits a shorter horizon, here \( N = 10 \), to be used, as predicted trajectories need not end in the target set, but only in a safety set. However, the mission objective is to steer both vehicles to their targets. Hence, the local subsystem objective \( J_p \) in each case again penalizes deviations from the target state, as in (3.1), but now with non-zero terminal cost.

\[
F_p(\tilde{x}_p) = l_p(\tilde{x}_p, \tilde{u}_p) = \|P_t(\tilde{r}_p - t_p)\|_\infty,
\]

where \( P_t \) is defined by (3.2). All other parameters are as in Example 3.1.

Because convergence to the target is not assured, the closed-loop performance may be extremely undesirable. Figure 3.9 shows such an outcome. At \( k = 0 \), each control agent is initialized with a stationary plan that maintains the vehicle at its initial position, and subsequently the vehicles are controlled by non-cooperative DMPC. Again, the update sequence is alternating. By the end of time step \( k = 3 \), both vehicles have optimized since initialization, yet neither has a plan that deviates significantly away from the initial position. At \( k = 9 \) little has changed; the vehicles are still seen to be in regions close to their respective initial positions (perturbed by the disturbance), a ‘deadlock’ situation where it is in neither agent’s interest to concede ground. A deadlock is an example of a Nash equilibrium [96]; briefly, an outcome in which no player can unilaterally improve his cost or payoff. (In Chapter 4, the Nash solution concept is more formally linked to distributed
MPC). Thus, without incentive to change, the vehicles will remain in these states indefinitely.

At $k = 10$, however, the control agents switch to using the cooperative form of DMPC. The choice of cooperating set for each is, naturally, the other vehicle, and a unity weighting ($\alpha_{pq} = 1$) is applied in the optimization objective. Following its optimization at $k = 10$, the southern-most vehicle forms a new plan whereby it will back out of the corridor, making way for the other. Subsequently, this other vehicle is now able to plan a route towards its target, as indeed it does at $k = 11$. Later, at $k = 24$, one vehicle has reached its target, while the vehicle that elected to leave the deadlock has a feasible plan that ends at its target, ensuring an eventually successful outcome.

Note that at certain points of the simulation, vehicles have `cut corners' of obstacles, even though the obstacles have been `enlarged' for robustness (Remark 2.5). This is owing to the discretization of the dynamics; obstacle avoidance constraints are enforced only at discrete intervals, so that while sampled points lie outside of obstacles, interpolated sections between may transgress. This is a well-known occurrence [94, 98], and may be prevented by either further enlarging the obstacles in the optimization, or increasing the sampling frequency. However, this is not a matter of primary concern here. The key point is that by agents switching to cooperative DMPC, a deadlock situation has been resolved, without either collision or constraint violation, or resorting to a rule-based approach [99].

3.5 Summary

A cooperative form of the robust distributed MPC algorithm has been presented. Motivated by the observation that local, non-cooperative decision-making can lead to ‘greedy’ behaviour and poor system-wide performance, cooperation between sub-system agents is promoted by each local agent considering a greater portion of the system-wide objective in its optimization. This is achieved by locally designing hy-
Figure 3.9: Deadlock resolution using cooperation. Two vehicles, with previous positions (solid lines), plans (*), targets (×), and obstacles.
pothetical trajectories for others in some cooperating set, such that the combined cost is minimized; local agents may thus sacrifice local performance for a potential improvement in global performance. Robust feasibility is guaranteed for the cooperative algorithm for any choices of cooperating sets, and conditions for stability have been identified. Furthermore, the flexibility in communications has been retained, in that communication between agents occurs only after a new plan is formed. By simulation, it has been shown that the cooperative method can improve upon poor performance, and break deadlocks, yet without necessarily requiring the same level of computation as for centralized MPC.
Chapter 4

Analysis of Cooperation in Distributed MPC

A formal analysis of the cooperative DMPC algorithm is presented, using tools from game theory and graph theory. By relating game-theoretical concepts to the distributed problem, it is shown that, in a state limit set, control agents are playing Nash solutions. Then, because a Nash strategy for the game with a given cooperation graph is also a Nash strategy for a more sparsely-connected graph, the set of closed-loop state limit sets is a subset of those sets associated with the smaller graph. Examples illustrate that an improvement in the convergence outcome can be seen by increasing cooperation.

Secondly, an adaptive form of cooperation for DMPC is proposed. In this new algorithm, an optimizing local control agent determines the existence of paths in a graph representing currently-active coupling constraints. Where such paths exist, cooperation is promoted by including those connected subsystems in the cooperating set. By simulation, performance is shown to better that of both the non-cooperative algorithm and a scheme using cooperation with only immediately-adjacent agents, rivalling that of a 'fully-cooperative' implementation.
4.1 Introduction

The previous chapter confirmed that where conflict exists between agents, cooperation may provide a means to avoid poor system-wide performance. The method proposed promotes inter-agent cooperation by a local control agent designing, in addition to its own trajectory, hypothetical trajectories for other agents in some cooperating set, the premise being that local performance may be sacrificed to permit an improvement in system-wide performance.

The question remains, however, of which other agents to cooperate with, and when; choices explored in the literature include all other agents [36] and those others to which a local agent is currently directly-coupled [43, 45]. That is, neighbours in an active coupling graph, the graph consisting of subsystems (nodes) and active coupling constraints (edges). Such rules, however, have no formal basis. This chapter seeks to provide such a basis on which to make this cooperation decision, and consists of two parts.

The first part considers the state convergence of the distributed algorithm. Under suitable assumptions, the system controlled by DMPC is shown to robustly converge to some closed-loop state limit set. This set is not necessarily a neighbourhood of the origin, and may depend on the cooperation graph; that is, the graph representing choices of cooperating sets for each agent. By posing the cooperative algorithm as a non-cooperative game [96] between agents, the limit sets for different cooperation graphs may be compared.

A game is defined by the problem of a number of agents trying to minimize their cost functions by choosing some action or strategy. Generally, the cost of an agent’s strategy depends on the actions of the other agents in the game; thus, the difficulty comes in determining an agent’s best strategy given what other agents might do. Unique team-, or Pareto-, optimal solutions are generally desired, because they represent best choices in terms of the overall system; a consequence of sub-optimal strategies is that the outcome of the game may be poor, or may result in the agents
in a 'deadlock'. A coupled decision space makes the game problem harder [100]; normal uniqueness guarantees for equilibrium solutions are no longer valid, and a continuum of sub-optimal Nash equilibrium strategies may exist.

By posing the local optimizations as a game between subsystem agents, where only a subset of ‘free’ agents may play at each time step, the DMPC algorithm is made amenable to game-theoretic analysis. It is shown that a Nash strategy for the game with a given cooperation graph is also a Nash strategy for a more sparsely-connected graph, and, by implication, the set of state limit sets the closed-loop system may converge to is a subset also of the set associated with the smaller graph. Consequently, increasing cooperation between agents will not adversely affect the quality of closed-loop limits.

In the second part of the chapter, an adaptive form of cooperation for DMPC is proposed. Whereas previous cooperative DMPC methods have looked at cooperation between either all agents [1] or adjacent, coupled subsystems [43, 45], in this chapter it is shown that an improvement to system-wide performance may be obtained by also cooperating with agents for subsystems connected by a path in the coupling graph.

By analysis, it is shown that cooperation with those not connected via the coupling graph makes no difference, providing an upper bound on the size of the graph for best performance. Then, an adaptive algorithm is proposed based on estimating the current coupling graph, including only active constraints, and cooperating with connected subsystems. Finally, this argument is used to show that performance can be improved by adding cooperation with subsystems that are not directly-coupled, but connected via paths in the coupling graph. Thus, a key contribution is the confirmation that the set of immediate, coupled subsystems, as used in Kuwata and How [43], Keviczky et al. [45], is not necessarily the optimal cooperating set.

The chapter is structured as follows. Section 4.2 outlines preliminary details, including the problem statement and definitions from game theory and graph theory. In Section 4.3, the closed-loop limit sets of the DMPC algorithm are related to the
cooperation graph, by analysing the game between agents in a limit set. Numerical examples show that, although it is not possible in a general case to prove that strictly better convergence outcomes arise from increasing cooperation, instances do exist. The cooperation graph is linked to the coupling graph in Section 4.4, wherein it is proven that cooperation with a disconnected subsystem brings no benefit. Subsequently, the adaptive form of the cooperative DMPC algorithm is described, and examples provided. The chapter is summarized in Section 4.5.

### 4.2 Preliminaries

#### 4.2.1 Problem statement

The problem statement follows that of Chapter 2, for uncertain constrained subsystems, described by the dynamics,

\[
x_p(k + 1) = A_p x_p(k) + B_p u_p(k) + w_p(k), \forall p \in \mathcal{P}, k \in \mathbb{N},
\]

with coupling between subsystems via the constraints. Of the definitions introduced in that chapter, it is worth reiterating that \( \mathcal{P}_c \) and \( \mathcal{C}_p \) are, respectively, the set of all subsystems \( p \in \mathcal{P} \) involved in a constraint \( c \in \mathcal{C} \), and the set of all constraints \( c \in \mathcal{C} \) involving a subsystem \( p \in \mathcal{P} \). Subsequently, the set of all other subsystems coupled to a subsystem \( p \) is

\[
\mathcal{Q}_p = \left( \bigcup_{c \in \mathcal{C}_p} \mathcal{P}_c \right) \setminus \{p\}.
\]

The system-wide objective will be defined according to need in later sections. Again the existence is assumed of RPI sets \( \mathcal{R}_p \), and associated linear feedback con-
trollers $K_p$, such that, for all $p$,

$$(A_p + B_pK_p)x_p + w_p \in \mathcal{R}_p, \forall x_p \in \mathcal{R}_p, w_p \in \mathcal{W}_p,$$

$$(C_p + D_pK_p)x_p \leq \mathcal{Y}_p,$$

$$\bigoplus_{p=1}^{N_p}(E_{cp} + F_{cp}K_p) \mathcal{R}_p \subseteq \mathcal{Z}_c, \forall c \in \mathcal{C},$$

We also assumed the existence of terminal sets $\mathcal{X}_{F_p}$ and terminal control laws $\kappa_{F_p}$, such that, $\forall x_p \in \mathcal{X}_{F_p}$.

$A_p x_p + B_p \kappa_{F_p}(x_p) \in \mathcal{X}_{F_p},$

$C_p x_p + D_p \kappa_{F_p}(x_p) \in \mathcal{Y}_p,$

$$\sum_{p=1}^{N_p} E_{cp} x_p + F_{cp} \kappa_{F_p}(x_p) \in \mathcal{Z}_c, \forall c \in \mathcal{C}.$$ 

Together, these conditions represent Assumptions 2.1–2.3 from Chapter 2.

### 4.2.2 Definitions from game theory and graph theory

Some definitions from both game theory [96] and graph theory [101] are now introduced. These will facilitate the later analyses and developments.

**Game theory**

Consider the game in which each agent $p \in \mathcal{P}$ aims to minimize a cost function $V_p$ by choosing some strategy or action $\theta_p$, so that

$$\{\theta_1, \ldots, \theta_{N_p}\} \in \Theta,$$
where this set $\Theta$ of feasible strategies may be coupled. Each local cost may also depend on the strategies of others, i.e., $V_p = V_p(\theta_p; \theta_{-p}^*)$, where

$$\theta_{-p}^* \triangleq \{\theta_1^*, \ldots, \theta_{p-1}^*, \theta_{p+1}^*, \ldots, \theta_{N_p}^*\}$$

is the collection of assumed values, denoted by $*$, for all agents bar $p$. The following definitions are taken from Başar and Olsder [96].

**Definition 4.1** (Normal-form game). The normal-form representation of such a game is given by

$$\mathcal{G} = \{\Theta; V_1, \ldots, V_{N_p}\}.$$

**Definition 4.2** (Nash solution). For the game $\mathcal{G}$, the solution $\theta^{Ne} = \{\theta_1^{Ne}, \ldots, \theta_{N_p}^{Ne}\}$ is a Nash solution if and only if

$$V_p(\theta_p^{Ne}; \theta_{-p}^{Ne}) \leq V_p(\theta_p; \theta_{-p}^{Ne}), \forall \theta_p \in \{\theta_p, \theta_{-p}^{Ne}\} \in \Theta, \forall p \in \mathcal{P}.$$

That is, no agent $p$ can do any better than a cost value $V_p(\theta_p^{Ne}; \theta_{-p}^{Ne})$ given that the strategies of the other agents are $\theta_{-p}^{Ne}$, no agent can unilaterally decrease its cost.

When agents share a common objective, i.e. the case where $V_1 = V_2 = \ldots = V$, a team-optimal solution is the minimizer of that objective over the entire decision space. That is, a team-optimal solution $\theta^{opt} = \{\theta_1^{opt}, \ldots, \theta_{N_p}^{opt}\}$ satisfies a single inequality

$$V(\theta_1^{opt}, \ldots, \theta_{N_p}^{opt}) \leq V(\theta_1, \ldots, \theta_{N_p}), \forall \{\theta_1, \ldots, \theta_{N_p}\} \in \Theta.$$

The team-optimal solution is a Pareto-optimal outcome, one that would result from a centralized agent acting for the whole team. It follows that a team-optimal solution is a Nash solution, but the opposite is not true [96]. In fact, the Nash solution is a weaker solution concept, and multiple, even infinite, such solutions may exist to a game [100]. A ‘deadlock’ is an example of a Nash solution—a sub-optimal outcome.
for the team, but inescapable by unilateral action. Thus, a Nash outcome is by no means always a good outcome.

This common objective assumption does not preclude the type of system-wide objective studied in this thesis. For example, suppose

\[
V(\theta_1, \ldots, \theta_{N_p}) = \sum_{p=1}^{N_p} J_p(\theta_p).
\]

Then, the theory of potential games [102] tells us

\[
\arg\min_{\theta_p} J_p(\theta_p) = \arg\min_{\theta_p} J_1(\theta_1^*) + \ldots + J_p(\theta_p) + \ldots + J_{N_p}(\theta_{N_p}^*)
\]

\[= \arg\min_{\theta_p} V(\theta_1^*, \ldots, \theta_p, \ldots, \theta_{N_p}^*)
\]

where all \(\theta_q^*, q \neq p\), are fixed. The important point is that the concept of a team objective, and a team-optimal solution, does not necessarily require all agents to minimize the same function.

**Graph theory**

The following two graphs help describe the distributed problem structure, and shall be referred to extensively.

**Definition 4.3 (Coupling graph).** The coupling graph, \( \mathcal{G}^x = \{\mathcal{P}, \mathcal{E}^x\} \), is the set of vertices (subsystems) \( \mathcal{P} \) and edges \( \mathcal{E}^x \subset \mathcal{P} \times \mathcal{P} \), where an edge \( \{p, q\} \in \mathcal{E}^x \) if and only if \( p \) shares any coupled constraint with \( q \):

\[
\{p, q\} \in \mathcal{E}^x \iff q \in \mathcal{Q}_p.
\]

The coupling graph is an undirected graph. In the cooperative DMPC method developed in the preceding chapter, a local agent \( p \in \mathcal{P} \) promotes cooperation by designing, in addition to its own trajectory, hypothetical trajectories for other agents.
in some designer-specified cooperating set $\mathcal{N}_p$. The cooperation graph is based on the choices of such cooperating sets in the local optimizations:

**Definition 4.4** (Cooperation graph). The cooperation graph is the directed graph $\mathcal{G}^N = \{\mathcal{P}, \mathcal{E}^N\}$, where an edge $\{p, q\} \in \mathcal{E}^N$ if and only if $q$ is in the cooperating set $\mathcal{N}_p$:

$$\{p, q\} \in \mathcal{E}^N \iff q \in \mathcal{N}_p.$$  

With some abuse of notation, the relation $\mathcal{G} \subseteq \mathcal{G}$ shall mean that $\mathcal{G} = \{\mathcal{P}, \mathcal{E}\}$ is the graph with the same vertices as $\mathcal{G} = \{\mathcal{P}, \mathcal{E}\}$, but only a subset of the edges, i.e., $\mathcal{G}^N \subseteq \mathcal{G}^N$ is equivalent to $\mathcal{E}^N \subseteq \mathcal{E}^N$. Such a subset graph shall be referred to as a ‘smaller’ graph.

Finally, a path on a graph is described by the linking of two distinct vertices by a sequence of edges [101].

**Definition 4.5** (Path on $\mathcal{G}$). A path on graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ from $p_0 \in \mathcal{V}$ to $p_n \in \mathcal{V}$ is an ordered set of distinct vertices $\{p_0, \ldots, p_i, \ldots, p_n\}$ such that $\{p_i, p_{i+1}\} \in \mathcal{E}, \forall i \in [0, n - 1]$.

This definition applies to both undirected and directed graphs. In the latter case, for example, $\{a, b, c\}$ is a path only if $\{a, b\}$ and $\{b, c\}$ exist in the set of edges.

### 4.3 Effect of cooperation on state convergence

In this section, closed-loop convergence of the system is considered, primarily by casting the DMPC algorithm as a game at each time step between subsystem agents and applying concepts from game theory. Firstly, given that the concern is with the convergence outcome, the problem statement studied in this section assumes local objective functions that meet only relaxed versions of the assumptions in previous chapters. There, convergence was guaranteed to a neighbourhood of the origin. In the next section, however, it will be shown that convergence is guaranteed to some general robust state limit set, and it is the quality of this limit that is under
investigation. To this end, it is shown that in such a limit set, each agent is continually playing a Nash strategy. It is proven that a Nash solution for the game with one cooperation graph is also a Nash solution for the game associated with any smaller graph. Subsequently, it is shown that the set of closed-loop limit sets for a given choice of cooperation graph is a subset of the corresponding set for a smaller cooperation graph.

4.3.1 Robust convergence to a state limit set

Suppose the system-wide objective remains decoupled, yet is some general positive-definite function

\[
\sum_{k=0}^{\infty} \sum_{p=1}^{N_p} \Psi_p(x_p(k)).
\]  

(4.1)

A finite-horizon approximation to (4.1) is

\[
\sum_{p=1}^{N_p} J_p(U_p(k)) = \sum_{p=1}^{N_p} \left[ F_p(x_p(k+N|k)) + \sum_{j=0}^{N-1} l_p(x_p(k+j|k), u_p(k+j|k)) \right],
\]

(4.2)

where \( F_p \triangleq a_p \Psi_p, a_p > 0, \) and \( U_p(k) \) is, as in previous chapters, the collection of initial state and sequence of controls for \( p: \)

\[
U_p(k) = \{ x_p(k|k), u_p(k|k), u_p(k+1|k), \ldots, u_p(k+N-1|k) \}.
\]

The stage cost \( l_p(x_p, u_p) \geq 0 \) is intended to stabilize the system, and—together with the terminal cost—satisfies the following assumption.

Assumption 4.1. For all \( p \in \mathcal{P}, \)

\[
l_p(x_p, u_p) \geq d(x_p, \mathcal{X}_{F_p}), \forall x_p \notin \mathcal{X}_{F_p},
\]

\[
l_p(x_p, u_p) = 0 \iff x_p \in \mathcal{X}_{F_p}, u_p = \kappa_{F_p}(x_p),
\]

\[
F_p(A_p x_p + B_p \kappa_{F_p}(x_p)) \leq F_p(x_p), \forall x_p \in \mathcal{X}_{F_p}.
\]
Thus, $F_p$ is not required to be a local Lyapunov function in the terminal set, as is commonly assumed [7] and was assumed for the stability results in previous chapters, but only non-increasing. To reiterate, this weaker condition admits a wider class of problems, including, for example, safe path planning [103] in unknown environments. The stage cost is zero in the terminal set, and not only at the origin, when under the terminal control.

The DMPC algorithm, and its extension to a cooperative form, was defined in Chapters 2 and 3. Briefly, at a time step $k$, a sole agent $p_k$ optimizes its plan, while all other agents renew their previous plans; i.e., given $U_p^*(k)$,

$$U_p(k + 1) = \{ \bar{x}_p^*(k + 1|k), \bar{u}_p^*(k + 1|k), \ldots, \bar{u}_p^*(k + N - 1|k), \kappa_{F_p}(\bar{x}_p^*(k + N|k)) \},$$

is the renewed plan for all $p \neq p_k$. Under Assumptions 2.1–2.3, reproduced at the outset of this chapter, the system controlled by this algorithm is robustly-feasible and, with the weaker Assumption 4.1 replacing Assumption 2.4, converges asymptotically to some state limit set.

**Proposition 4.1** (Robust convergence to state limit set). *If every $p \in P$ is included in the update sequence $\{p_k\}$ infinitely many times as $k \to \infty$, there exists some limit set $\bar{X}^e \subseteq \bar{X}_{F_1} \times \ldots \times \bar{X}_{F_N}$ for the nominal states $\bar{x}_p(k|k)$ as $k \to \infty$, and, correspondingly, the set $X^e = \bar{X}^e \oplus \{ R_1 \times \ldots \times R_N \}$ is robust asymptotically-stable for the controlled system $x_p(k + 1) = A_p x_p(k) + B_p u_p(k) + w_p(k), \forall p \in P$, where $w_p(k) \in W_p, \forall k$.*

**Proof.** Based on Proposition 3.1, it is straightforward to show monotonic descent of the augmented value function, which was defined in the previous chapter and consists of the local optimization cost augmented with subsystem costs for those not in the cooperating set. Specifically, for any $p$, given a feasible $U_p^*(k_0)$ at $k_0$, and
candidate plan $\tilde{U}_p(k_0 + 1)$ for $k_0 + 1$,

$$
\tilde{J}_p(k_0 + 1) = J_p(\tilde{U}_p(k_0 + 1)) + \sum_{q \in \mathcal{N}_p(k_0 + 1)} \alpha_{pq} J_q(\tilde{U}_q(k_0 + 1))
+ \sum_{r \notin \{p, \mathcal{N}_p(k_0 + 1)\}} \alpha_{pr} J_r(\tilde{U}_r(k_0 + 1))
\leq J_p^*(k_0) - \sum_{i=1}^{N_p} \alpha_{pi} l_i(x_i^*(k|k_0), u_i^*(k|k_0))
$$

using Assumption 4.1, where $\alpha_{pp} = 1$ and $\alpha_{pi} \in [0, \infty), \forall i \neq p$, for any choices $\mathcal{N}_p(k_0 + j) \subseteq \mathcal{P}\setminus\{p\}, j \in \{0, 1\}$. This provides an upper bound on the value obtained by optimization, so that, for any $p \in \mathcal{P}$,

$$
J_p^{opt}(k_0 + 1) \leq J_p^*(k_0 + 1) \leq \tilde{J}_p(k_0 + 1) \leq J_p^*(k_0) - \sum_{i=1}^{N_p} \alpha_{pi} l_i(x_i^*(k|k_0), u_i^*(k|k_0))
$$

where $J_p^*(k_0 + 1)$ is the cost of a general feasible solution at $k_0 + 1$.

Next, as $F_p(\cdot) \geq 0$ and $l_p(\cdot, \cdot) \geq 0$, with $l_p(x_p, u_p) \geq d(x_p, \mathcal{X}_F), \forall x_p \notin \mathcal{X}_F$, then $J_p^*(k) \geq 0$ and $J_p^*(k+1) - J_p^*(k) \to 0$ as $k \to \infty$. It follows that $l_p(x_p^*(k|k), u_p^*(k|k)) \to 0, \forall p$, implying, by Assumption 4.1, that $x_p^*(k|k) \to \mathcal{X}_F$ and $u_p^* \to \kappa_F(x_p^*(k|k))$.

Furthermore, if each subsystem $p$ is included in the update sequence infinitely many times as $k \to \infty$, and because $J_p^*$ is lower-bounded and decreasing, there must exist some limit for the sequences $\{J_p^*(k)\}_{k}, \forall p$, and a corresponding limit set $\tilde{X}_e \subseteq \mathcal{X}_F = \mathcal{X}_{F1} \times \ldots \times \mathcal{X}_{FN_p}$, to which the nominal states converge, invariant under $\kappa_F$. A (trivial) example of such a set is $\mathcal{X}_F$ itself. Finally, because $x_p(k) - x_p^*(k|k) \in \mathcal{R}_p, \forall k$, the perturbed states converge to some $\mathcal{X}_e = \tilde{X}_e \oplus \{\mathcal{R}_1 \times \ldots \times \mathcal{R}_{N_p}\}$.

Later, it will be shown that the nature of this state limit set depends on the cooperation graph, and that by 'increasing' cooperation the attained limit may be no worse in terms of cost value. The next section compares the distributed MPC algorithm to a non-cooperative game between the control agents, and shows that, upon the states reaching a limit set, each agent is continually playing a Nash solution.

118
4.3.2 Distributed MPC as a game

Model predictive control approximates a dynamic problem as a series of overlapping static problems; at a time step \( k \), with the system at a state \( x(k) \), the infinite-horizon optimal control problem is generally intractable. Instead, a sequence of open-loop controls is determined for \( N \) steps into the future. The first of those inputs is applied, the states evolve according to the dynamics, and the process is repeated at the next time step. In Figure 4.1, the arc (a) illustrates this approximation.

Where a number of dynamic subsystems and control agents are involved, as in a distributed setting, the centralized MPC problem may be decomposed further into a number of sub-problems, as indicated by the arc (b) in the figure. The DMPC algorithm developed in Chapter 2 is an example of such a decomposition.

Again, where a system may be decomposed into a number of subsystems, the infinite-horizon optimal control problem may be alternatively described as an infinite-horizon dynamic game, denoted \( S^\infty(x(0)) \), and defined by a number of agents or

\[ \min \sum_{k=0}^{\infty} l(x(k), u(k)) \]

\( (a) \rightarrow \{ P^C(x(0)), P^C(x(1)), \ldots, P^C(x(k)), \ldots \} \)

\( (c) \rightarrow \{ P^D_P(x_1(0), Z_1(0)), \ldots, P^D_P(x_1(k), Z_1(k)), \ldots \} \)

\( S^\infty(x(0)) \rightarrow \{ S(x(0)), S(x(1)), \ldots, S(x(k)), \ldots \} \)

Figure 4.1: Relationship between MPC and games.

\(^1\)Exactly how those sub-problems are formed and in what order they are solved is, broadly, what differs between the numerous DMPC methods in the literature.
players, each with some cost function to minimize by choosing some strategy or action. Determination of team-optimal, or even person-by-person optimal, solutions requires the capturing of all interactions between subsystems’ dynamics and strategies for all steps into the future. Such an approach is generally intractable. However, it is possible, in a way analogous to the concept of MPC, to approximate this infinite-horizon dynamic game as a series of overlapping static games. At this point, a link may be made between games and distributed MPC.

Consider the collection of \( N_p \) static DMPC problems at a single time step, i.e., the collection of local optimization problems, with no assumption at this point as to which optimizations are actually solved. Each agent must determine its own strategy over the horizon, taking into account the local dynamics model, constraints, and the plans of others. For the chosen local solution, \( U_p \), the associated value of the cost (or the payoff) for each \( p \) is known directly, by evaluation of the local optimization cost \( V_p(U_p) \). Furthermore, over the prediction horizon, no exchange of information or iteration takes place. This corresponds to the open-loop information pattern for dynamic games described in Başar and Olsder [96], wherein treatment of such as a static game is justified. Thus, the DMPC algorithm at each time step may be viewed as a static game (arc (e)), and the true, infinite-horizon dynamic game is approximated by a series of static games (arc (d)), with each game depending on the state, i.e. \( G = G(x(k)) \) is the game at time \( k \) with the state at \( x(k) \); at the next time step, a new game \( G(x(k+1)) \) is posed.

For the specific DMPC algorithm in this thesis, at each time step \( k \) a static optimization problem is solved for a sole ‘free’ agent \( p_k \)—or, as identified by Remark 2.6, a number of non-coupled agents in some updating set \( P_k \). These agents optimize for a new plan, i.e.,

\[
U^\text{opt}_{p_k}(k) = \{ x^\text{opt}_{p_k}(k|k), u^\text{opt}_{p_k}(k|k), \ldots, u^\text{opt}_{p_k}(k + N - 1|k) \},
\]
while all other agents $q \neq p_k$ renew previous plans via (2.19). That is, given some feasible plan

$$U_q^*(k-1) = \{ \bar{x}_q^*(k-1|k-1), \bar{u}_q^*(k-1|k-1), \ldots, \bar{u}_q^*(k+N-2|k-1) \},$$

at step $k-1$, the feasible plan for step $k$ consists of the tail of the previous solution and a step of the terminal control law:

$$\bar{U}_q(k) = \{ \bar{x}_q^*(k|k-1), \bar{u}_q^*(k|k-1), \ldots, \bar{u}_q^*(k+N-2|k-1), \kappa_{F_p}(\bar{x}_q^*(k+N-1|k-1)) \}.$$

Once each control agent has applied the first control of the sequence, a new set of problems, or game, is posed at the next time step, though again with a limited set of free agents. However, this section is concerned with proving properties for closed-loop state limits, and so it is postulated that the games played in some closed-loop limit set are unchanging; at some state $x \in X^e$, a limit set, the outcome of the game at each time step is for the system to remain in that set. (Proposition 4.4 will formalize this notion.)

Some further notation is now introduced, to simplify exposition. Subsequently the local (cooperative) optimization problem $\mathcal{P}^{D_p N_p}(k)(x_p(k); \mathcal{Z}_p(k))$, which was defined in the previous chapter, is re-written in a new form, to assist the development of the results to follow. In the sequel, the local action $\theta_p$ to be chosen by a player of the game is equal to the local control problem decision variable $U_p$:

$$\theta_p(k) = U_p(k) = \{ \bar{x}_p(k|k), \bar{u}_p(k|k), \bar{u}_p(k+1|k), \ldots, \bar{u}_p(k+N-1|k) \},$$

i.e., the collection of initial state and control sequence. This value $U_p(k)$ is also now defined as the message each agent will transmit following optimization, so that a simple mapping exists between the decision variables and the message communicated. Furthermore, the use of such notation makes more transparent the link between the game theory definitions in Section 4.2.2 and the DMPC algo-
rithm. It will be assumed that, from a received $U_p(k)$, every other agent can construct the necessary fixed values for that $p$ in order to evaluate the constraints in the problem $P_p^{D,N_p}(k)(x_p(k); \tilde{Z}_p^*(k))$. Thus, a mapping exists between $\tilde{Z}_p^*(k)$, the previously-published information required for constraint evaluation, and $U^*_{-p}(k) \triangleq \{ U_1^*(k), \ldots, U_{p-1}^*(k), U_{p+1}^*(k), \ldots, U_{N_p}^*(k) \}$, the collection of feasible plans from all agents bar $p$. Note that $U_p(k)$ is implicitly a function of the local state and the information about others, $U_p = U_p(x_p(k), \tilde{Z}_p^*(k)) = U_p(x_p(k), U^*_{-p}(k))$. In much of what follows, for clarity of presentation, the time indices shall be omitted from these decision variables.

Next, Algorithm 3.1 with a time-invariant cooperation graph $\mathcal{G}^N$ shall be referred to as DMPC$^N$; similarly, DMPC$^0$ shall denote non-cooperative DMPC, i.e., that with an empty cooperation graph. Also define the following,

- $U \triangleq \{ U_1, \ldots, U_{N_p} \}$ is the collection of all decision variables.
- $\hat{U}_{N_p} \triangleq \{ \hat{U}_q \}_{q \in N_p}$ is the collection of hypothetical variables for subsystems in $p$'s cooperating set, computed by $p$.
- $U^*_{\{p,N_p\}} \triangleq \{ U^*_r \}_{r \notin \{p,N_p\}}$ is the collection of fixed decisions for non-members of the cooperating set of $p$.

The choices of these decision variables are made from feasible sets, described by the constraints in the optimization problem $P_p^{D,N_p}(k)(x_p(k); \tilde{Z}_p^*(k))$. Firstly, the set of feasible $U$ is defined as $U = U(x(k))$, and is closed and bounded, coupled, and
defined by the constraints of the centralized optimization problem:

\[
\mathbf{U}(x(k)) \triangleq \left\{ \mathbf{U}(k) : \begin{array}{l}
\bar{x}_p(k + j + 1|k) = A_p \bar{x}_p(k + j|k) + B_p \bar{u}_p(k + j|k), \\
x_p(k) - \bar{x}_p(k|k) \in \mathcal{R}_p, \\
\bar{x}_p(k + N|k) \in \mathcal{X}_p, \\
\bar{y}_p(k + j|k) = C_p \bar{x}_p(k + j|k) + D_p \bar{u}_p(k + j|k), \\
\bar{y}_p(k + j|k) \in \bar{y}_p, \\
\bar{z}_{cp}(k + j|k) = E_{cp} \bar{x}_p(k + j|k) + F_{cp} \bar{u}_p(k + j|k), \forall c \in \mathcal{C}, \\
\sum_{i=1}^{N_p} \bar{z}_{ci}(k + j|k) \in \bar{z}_c, \forall c \in \mathcal{C}, \\
\forall p \in \mathcal{P}, j \in \{0, \ldots, N - 1\}.
\end{array} \right\}
\]

(4.3)

The set of feasible \( \mathbf{U}_p \) for a subsystem \( p \), given the fixed feasible solutions \( \mathbf{U}^*_p \) of all other subsystems, and for the system at a state \( x \), is denoted \( \mathbf{U}_p(x; \mathbf{U}^*_p) \), and is defined directly from (4.3) as

\[
\mathbf{U}_p(x; \mathbf{U}^*_p) \triangleq \left\{ \mathbf{U}_p : \{\mathbf{U}_p, \mathbf{U}^*_p\} \in \mathbf{U}(x) \right\}.
\]

(4.4)

A combined feasible set for \( \{\mathbf{U}_p, \mathbf{U}_q\} \), subject to fixed \( \mathbf{U}^*_\{p,q\} \), is given by

\[
\mathbf{U}_{\{p,q\}}(x; \mathbf{U}^*_\{p,q\}) \triangleq \left\{ \{\mathbf{U}_p, \mathbf{U}_q\} : \{\mathbf{U}_p, \mathbf{U}_q, \mathbf{U}^*_\{p,q\}\} \in \mathbf{U}(x) \right\}.
\]

(4.5)

In writing these sets, the coupling constraint structure is neglected—with no loss of generality—to permit dependency of the decision variable for \( p \) on that of any other subsystem \( q \neq p \); for example, we write \( \mathbf{U}_p(x; \mathbf{U}^*_p) \) rather than \( \mathbf{U}_p(x; \mathbf{U}^*_\mathcal{Q}_p) \), where \( \mathcal{Q}_p \) is set of subsystems coupled to \( p \).

These definitions then permit us to define a constrained feasible set for a general local optimization for a local agent \( p \) with cooperating set \( \mathcal{N}_p \). Noting that an agent minimizes its local optimization cost by not only manipulating \( \mathbf{U}_p \) subject to fixed \( \mathbf{U}^*_p \), but also by manipulating variables \( \mathbf{U}_\mathcal{N}_p \) while subject to fixed \( \mathbf{U}^*_{\{p,\mathcal{N}_p\}} \), the feasible set for such an optimization is based on the definitions (4.4) and (4.5), and
is given by

\[ U_p^{N_p}(x; U^*_p) \triangleq \left\{ U_p, \hat{U}_{N_p} \right\} \]

\[ \left\{ U_p, \hat{U}_{N_p} \right\} \in U_{\{p,N_p\}}(x; U^*_{\{p,N_p\}}), \]

\[ \bar{x}_q(k|k) = \bar{x}^*_q(k|k - 1), \]

\[ \bar{u}_q(k|k) = \bar{u}^*_q(k|k - 1), \]

\[ \forall q \in N_p. \]

which now includes the constraints (3.4) for members of \( p \)'s cooperating set, \( N_p \).

Note that this set becomes equal to the set \( U_p(x; U^*_{-p}) \) if \( N_p = \emptyset \), which is consistent with a similar observation made in the previous chapter. Furthermore, note that \( U_p^{N_p}(x; U^*_p) \subseteq U_{\{p,N_p\}}(x; U^*_{\{p,N_p\}}) \). This is an important point. This mismatch in size arises from the inclusion of the coupling constraints (2.30g) in the cooperative optimization, implicit in the expression \( \left\{ U_p, U^*_{-p} \right\} \in U(x) \). The presence of these constraints, as explained in the previous chapter, maintains coupling constraint satisfaction and feasibility, though perhaps at the expense of conservatism. Figure 4.2 illustrates this observation and the relationships between the sets defined thus far.

The local optimization problem \( P_{D,N_p}^D(x_p(k); \hat{Z}_p^*(k)) \) may now be restated in a more concise form as

\[
\min_{\left\{ U_p, \hat{U}_{N_p} \right\}} \left\{ V_p(U_p, \hat{U}_{N_p}) : \left\{ U_p, \hat{U}_{N_p} \right\} \in U_p^{N_p}(x; U^*_p) \right\}. \tag{4.7}
\]

where the local optimization cost is a summation of local subsystem costs:

\[ V_p(U_p, \hat{U}_{N_p}) = J_p(U_p) + \sum_{q \in N_p} \alpha_{pq} J_q(\hat{U}_q). \]

In the remainder of this section, and without loss of generality, unity weightings will be assumed for all subsystems, i.e., \( \alpha_{pq} = 1, \forall \{p,q\} \), so that all agents are seeking to place equal emphasis on their own objectives and other agents' objectives.
Figure 4.2: Illustration of different feasible sets, ignoring initial constraints. Given a current feasible solution \((U_1^*, U_2^*)\), agent \(p = 2\) optimizes to provide the solution \((\tilde{U}_1^*, \tilde{U}_2^*) \in U_2(U_1^*)\); constraints ensure that the point \((U_1^*, U_2^*)\) remains feasible. However, the choice \((\hat{U}_1^*, \hat{U}_2^*) \in U_{(1,2)}\), though hypothetically feasible, renders the resulting \((U_1^*, U_2^*)\) infeasible.

Furthermore, if all cooperating sets are 'full', i.e., \(N_p = \mathcal{P} \setminus \{p\}, \forall p \in \mathcal{P}\), then all agents share the common goal of minimizing the system-wide cost.

**Remark 4.1** (Comparison with Keviczky et al. [45]). This concise format for the local optimizations permits a clearer comparison to be made with the distributed MPC method of Keviczky et al. [45], also for dynamically-decoupled subsystems with coupling constraints. Rewriting the formulation in terms of the notation introduced here, a local optimization in the algorithm of Keviczky et al. [45] is given by

\[
\min_{\{U_p, U_{Q_p}\}} \left\{ V_p(U_p, \hat{U}_{Q_p}) : \{U_p, \hat{U}_{Q_p}\} \in \hat{U}_{\{p, Q_p\}}(x) \right\},
\]

where the effective 'cooperating set', \(N_p\), is equal to \(Q_p\), the set of subsystems coupled to \(p\), as opposed to the unrestricted choice proposed here. Therefore, the
minimum of $V_p$ is sought by manipulation of $U_p$ and $\hat{U}_q, q \in Q_p$, over the combined feasible set. More important, however, is the construction of that feasible set, where two key observations may be made regarding coupling constraints: firstly, only those constraints involving $p$, i.e., $c \in C_p$, are evaluated. That is, for feasibility of $\{U_p, \hat{U}_q\}$, the fixed outputs $U_r, r \notin \{p, Q_p\}$ are not considered, hence the combined set $U\{p, Q_p\}(x)$ in (4.8) may differ from the combined set $U\{p, Q\}(x; U^*\{p, N_p\})$.

Secondly, no consideration is given of the current plans for subsystems $q \in Q_p$, i.e., the constraint $U_p \in U_p(x; U^*\{p\})$ is omitted. As a consequence of these two details, feasibility is not guaranteed in that work [45].

A different viewpoint on the DMPC methods proposed in this thesis is now offered. Starting from a non-cooperative, but guaranteed feasible, method in Chapter 2, cooperation was promoted by additionally designing trajectories for others. Alternatively, starting from the cooperative formulation of Keviczky et al. [45], feasibility may be guaranteed by including extra constraints based on previously-published plans of others. The results developed in this section imply that inclusion of such constraints is not harmful to cooperation.

Before considering the properties of games played at a converged state, we note that the optimization problem (4.7) is the problem applicable to any agent $p$ at a general time step, with the system at a state $x$, whether or not that agent optimizes or adopts the candidate plan. The game at a time step $k$ and state $x(k)$ shall be denoted $G^N(x(k))$, where the superscript indicates the cooperation graph of choice. By considering the solutions available to a game at a general state, the results will be useful later when considering convergence to a given limit state. The next Proposition relates the Nash solutions available at a constant state to the choice of cooperation graph, and shows that a Nash solution for a given cooperation graph is also a Nash solution for any smaller cooperation graph.

**Proposition 4.2 (Inclusion of Nash solutions).** Suppose the game $G^N(x)$ associated with the DMPC$^N$ algorithm at a state $x$ has a Nash equilibrium solution $U^{Ne}$. 

126
Then $U^{Ne}$ is also a Nash solution for the game $G^N(x)$ associated with a DMPC\(^N\) algorithm, where $G^N \subseteq G^N$.

**Proof.** A solution $U^{Ne} = \{U^{Ne}_1, \ldots, U^{Ne}_N\}$ to $G^N(x)$, by definition, must satisfy the following inequalities to be a Nash solution. For all $p \in \mathcal{P}$,

\[
g_{Ne}^p \triangleq J_p(U^{Ne}_p) + \sum_{q \in \mathcal{N}_p} J_q(\hat{U}^{Ne}_q) + \sum_{r \notin \{p, \mathcal{N}_p\}} J_r(U^{Ne}_r) \\
\leq g_p = J_p(U_p) + \sum_{q \in \mathcal{N}_p} J_q(\hat{U}_q) + \sum_{r \notin \{p, \mathcal{N}_p\}} J_r(U^{Ne}_r), \forall \{U_p, \hat{U}_{\mathcal{N}_p}\} \in U^{Ne}_p(x; U_{-p}^{Ne}).
\]

(4.9)

Alternatively,

\[
g_{Ne}^p = \min_{\{U_p, \hat{U}_{\mathcal{N}_p}\}} \left\{ J_p(U_p) + \sum_{q \in \mathcal{N}_p} J_q(\hat{U}_q) + \sum_{r \notin \{p, \mathcal{N}_p\}} J_r(U^{Ne}_r) : \right. \\
\left. \{U_p, \hat{U}_{\mathcal{N}_p}\} \in U^{Ne}_p(x; U_{-p}^{Ne}) \right\}.
\]

By constraining $U_p$ to be equal to the minimizer $U^{Ne}_p$, it is implied that

\[
g_{Ne}^p = \min_{U^{Ne}_p} \left\{ J_p(U^{Ne}_p) + \sum_{q \in \mathcal{N}_p} J_q(\hat{U}_q) + \sum_{r \notin \{p, \mathcal{N}_p\}} J_r(U^{Ne}_r) : \right. \\
\left. \{U^{Ne}_p, \hat{U}_{\mathcal{N}_p}\} \in U^{Ne}_p(x; U_{-p}^{Ne}) \right\} \\
= \min_{U^{Ne}_p} \left\{ J_p(U^{Ne}_p) + \sum_{q \in \mathcal{N}_p} J_q(\hat{U}_q) + \sum_{r \notin \{p, \mathcal{N}_p\}} J_r(U^{Ne}_r) : \right. \\
\left. \{U^{Ne}_p, \hat{U}_{\mathcal{N}_p}\} \in U^{Ne}_p(x; U_{-p}^{Ne}) \right\}.
\]
It follows that, for any \( p \in \mathcal{P} \),

\[
\exists U_{opt}^p : U_{opt}^p \neq U_{opt}^{Ne} , \{ U_{opt}^p , U_{opt}^{Ne} \} \in \mathcal{U}(x),
\]

\[
\min_{\hat{U}_{N_p}} \left\{ J_p(U_{opt}^p) + \sum_{q \in N_p} J_q(\hat{U}_q) + \sum_{r \notin \{p,N_p\}} J_r(U_{opt}^{Ne}) : \{ U_{opt}^p , \hat{U}_{N_p} \} \in U_{opt}^{N_p}(x; U_{opt}^{Ne}) \right\}
\]

\[
< \min_{\hat{U}_{N_p}} \left\{ J_p(U_{opt}^{Ne}) + \sum_{q \in N_p} J_q(\hat{U}_q) + \sum_{r \notin \{p,N_p\}} J_r(U_{opt}^{Ne}) : \{ U_{opt}^{Ne} , \hat{U}_{N_p} \} \in U_{opt}^{N_p}(x; U_{opt}^{Ne}) \right\},
\]

(4.10)

else (4.9) would not hold.

Now consider the ‘smaller’ cooperation graph \( \mathcal{G}^N \). The decision space for an agent \( p \) with \( N_p \) is equal to the space for the same \( p \) with \( N_p \), with additional constraints on the ‘removed’ cooperating agents \( i \in \{ N_p \backslash N_p \} : \)

\[
U_{opt}^{N_p}(x; U_{opt}^{Ne}) = \left\{ U_{opt}^{N_p}(x; U_{opt}^{Ne}) : \hat{U}_i = U_{opt}^{Ne} , \forall i \in \{ N_p \backslash N_p \} \right\}
\]

\[
\subseteq U_{opt}^{N_p}(x; U_{opt}^{Ne}).
\]

The solution \( (U_p, \hat{U}_{N_p}) = (U_{opt}^{Ne}, U_{opt}^{Ne}) \) lies in both sets. By (4.10), the cost of adopting this solution is always lower than that associated with adopting any \( U_p \neq U_{opt}^{Ne} \). It follows that \( U_{opt}^{Ne} = \{ U_{opt}^{Ne} , \ldots , U_{opt}^{Ne} \} \) is a Nash solution to \( \mathcal{G}^N(x) \) \( \square \)

This result illustrates an intuitive concept: that increasing the ‘amount’ of cooperation shall not enlarge the set of Nash solutions available to the game at a state. Alternatively, it is well-known that the optimum over a set is also the optimum over a subset of that set, if still feasible. However, this result is non-trivial in that it has established that this principle still holds despite the addition of constraints required for robust feasibility.

Having now established a relationship between Nash solutions for different degrees of cooperation, the following result shows that a solution to the centralized optimization problem is a Nash solution for the game with any cooperation graph.
This confirms that a solution originating from a centralized optimization may not be bettered by distributed optimization.

**Proposition 4.3** (Centralized solution is Nash solution for $G^N$). Suppose $U^\text{opt}$ is an optimal solution to the centralized optimization $P^C(x)$. Then $U^\text{opt}$ is a Nash solution to the game $G^N(x)$ for all cooperation graphs $G^N$.

**Proof.** The solution $U^\text{opt}$ satisfies the inequality

$$V^\text{opt} = \sum_{i=1}^{N_p} J_i(U^\text{opt}) \leq \sum_{i=1}^{N_p} J_i(U_i), \forall U \in U(x).$$

Rewriting, noting that the point $U^\text{opt} = \{U_1^\text{opt}, \ldots, U_{N_p}^\text{opt}\} \in U_p^{\{P \setminus \{p\}\}}(x; U_{-p}^\text{opt})$ for all $p \in \mathcal{P}$, where the superscript $\{P \setminus \{p\}\}$ refers to the local cooperating set choice of all agents bar $p$,

$$V^\text{opt} \leq J_p(U_p) + \sum_{q \neq p} J_q(U_q), \forall \{U_p, U_{-p}\} \in U_p^{\{P \setminus \{p\}\}}(x; U_{-p}^\text{opt}) \subseteq U(x).$$

But this is the condition for $U^\text{opt}$ to be a Nash solution to $G_{\text{max}}^N(x)$, the game associated with a maximal, all-pairs cooperation graph $G_{\text{max}}^N \triangleq \{P, \mathcal{P}^{(2)}\}$, i.e., where $N_p = P \setminus \{p\}, \forall p$. Thus, $U^\text{opt}$ is a Nash solution to $G_{\text{max}}^N(x)$, and, by successive application of Proposition 4.2, all games corresponding to cooperation graphs with edge sets $E^N \subseteq \mathcal{P}^{(2)}$.

If $U_{\text{Nash}}^N$ is the union of all Nash solutions for $G^N$ over all admissible states, and $U^C$ is the corresponding union of all centralized optima, Propositions 4.2 and 4.3 imply

$$U^C \subseteq U_{\text{Nash}}^{N_{\text{max}}} \subseteq \ldots \subseteq U_{\text{Nash}}^N \subseteq \ldots \subseteq U_{\text{Nash}}^\emptyset.$$

Although it would be desirable to have a result built on strict inclusion of sets, the following counter-example confirms it is not possible to prove strict inclusion for the general case. In particular, for the example given, the set of Nash solutions for the non-cooperative game coincide with the set of centralized optima, i.e., $U_{\text{Nash}}^\emptyset = U^C$. 129
Example 4.1 (Non-strict Nash equilibria). Consider the memoryless system

\[ x_p(k + 1) = u_p(k), \quad p \in \{1, 2\}, \]

where \( x_p \in \mathbb{R} \), \( u_p \in \mathbb{R} \), constrained such that \( x_p \in [-1, 1] \), \( u_p \in [-1, 1] \), and \( |x_1 + x_2| \leq 1 \). The cost to minimize is

\[
\sum_{p=1}^{2} \sum_{k=0}^{\infty} x_p(k)^2,
\]

and so the local, finite-horizon cost for an agent \( p \) is chosen as

\[
J_p(U_p(k)) = \sum_{j=0}^{N} \bar{x}_p(k+j|k)^2 = \bar{x}_p(k)^2 + \bar{u}_p(k|k)^2 + \ldots + \bar{u}_p(k+N-1|k)^2
\]

\[= \|U_p(k)\|_2^2.\]

Note that the terminal cost is, implicitly, \( \bar{x}_p(k+N|k)^2 \). Suppose no terminal sets are included, but constraints are applied over the full horizon range \( j \in \{0, \ldots, N\} \) so that constraint satisfaction is assured for the terminal state.

Figure 4.3 shows the feasible states, denoted \( \mathcal{X} \), and contours of \( x_1^2 + x_2^2 \) for this problem. Though the relation between state convergence and Nash solutions has not yet been discussed, the memoryless system here is, in effect, state-less. In fact, because of the unity mapping between states and inputs, the feasible set of decision variables, \( \mathcal{U}(x) \) at a state \( x \), is easily determined as the Cartesian product \( x \times \mathcal{X} \times \mathcal{X} \times \ldots \times \mathcal{X} \). Therefore, the values of decision variables over the horizon may be projected as trajectories on the two-dimensional plot of Figure 4.3, rather than requiring multi-dimensional plots.

Three controllers are available: centralized MPC, non-cooperative DMPC, and cooperative DMPC. In this case, the optimal solution for the centralized optimization, for any initial state, is to proceed directly to the origin and remain there,
Figure 4.3: Non-strict Nash equilibria. Set of feasible states, \( \mathcal{X} \), contours of \( x_1^2 + x_2^2 \), and reaction curves for each agent, here with a point intersection. Given the candidate plans \( \{ U_1^*, U_2^* \} \), indicated by '*', the optimization for \( p = 1 \) moves all points bar the initial state to the line of best reaction, resulting in the solution indicated by 'o'.

because the state evolution has no memory:

\[
U_{\text{opt}}(x) = \{ x, 0, \ldots, 0 \} = \arg \min_{U \in U(x)} \| U_1(k) \|_2^2 + \| U_2(k) \|_2^2,
\]

so that the set of centralized optima, over all initial states, is

\[
U^C = \bigcup_{x \in \mathcal{X}} U_{\text{opt}}(x) = \mathcal{X} \times 0 \times \ldots \times 0.
\]

For the distributed schemes, consider the game posed between the two agents.

The reaction curve \([96]\) for an agent is the best response of that agent to each action of other agents in the game. For example, for two agents, \( p \) and \( q \),

\[
R_p(U_q^*) = \arg \min_{U_p} \left\{ \| U_p \|_2^2 : U_p \in U_p(x; U_q^*) \right\},
\]

131
represents, for the non-cooperative algorithm, agent $p$'s best reaction to the $U_q^*$ played by $q$. For this example, the reactions may be obtained by inspection, and are shown in Figure 4.3. The reaction curves are useful in determining Nash solutions, which lie at the intersection of the curves \[96\]. Here the intersection is a point, the origin, implying the Nash solution for the game at any state is unique. In a similar manner, it would be possible to determine reactions and Nash solutions for the cooperative algorithm. However, by noting that the non-cooperative Nash solution is—crucially—equal to the centralized optimum, Proposition 4.2 may be invoked, i.e., $U_{Nash}^N \subseteq U_{Nash}^\emptyset$. Consequently,

$$U^C = U_{Nash}^{\max} = U_{Nash}^\emptyset.$$  

Therefore, this counter-example has confirmed that it is not possible, in a general sense, to prove a strict inclusion $U^C \subset U_{Nash}^{\max} \subset \ldots \subset U_{Nash}^\emptyset$.

The next section shall use these Nash inclusion results to derive further results regarding convergence of the closed-loop system under different cooperation graphs. It is shown, by simulations of particular systems, that convergence to 'better' state limit sets can be achieved by increasing cooperation.

### 4.3.3 State convergence properties

In Section 4.3.1 it was proven that the closed-loop system, controlled by $\text{DMPC}^N$, converges to some unspecified state limit set $X^e$. For certain applications, the properties of this limit may be critical to the performance, or even success or failure, of the control. For example, for path-planning for teams of vehicles in a uncertain environment \[49\], a desirable state limit is for each vehicle to reach its goal, yet members of the team might easily end up in an undesirable limit, such as a 'deadlock', (such a situation was encountered in Example 3.4). Hence, the remainder of this section is dedicated to analysing the 'quality' of state limit sets associated with different
cooperation graphs. The first result shows that, in such a limit set, the agents are continually playing Nash strategies.

**Proposition 4.4** (Game in a State Limit Set). Consider the system controlled by Algorithm 3.1, with cooperation graph $G^N$, and suppose each agent $p$ is included in the update sequence infinitely many times as $k \to \infty$. The set $\mathcal{X}^e$ is a limit set if and only if, at every $x \in \mathcal{X}^e$, $U^e = \{U^e_1, \ldots, U^e_{N_p}\}$, where $U^e_p = \{x_p, \kappa_{F_p}(x_p), \kappa_{F_p}(A_p x_p + B_p \kappa_{F_p}(x_p)), \ldots\}$, and $x_p - \bar{x}_p \in \mathcal{R}_p$, is a Nash solution to $G^N(x)$.

**Proof.** (i) **Sufficiency:** assume that $\mathcal{X}^e$ is a limit set, but, for some $x \in \mathcal{X}^e$, $U^e$, is not a Nash solution. Further assume that $x(k_0) \in \mathcal{X}^e$ at some $k_0 \in [0, \infty)$. Then, by Definition 4.2, because $U^e$ is not a Nash solution,

$$\exists x \in \mathcal{X}^e, p \in \mathcal{P}, U^0_p \in U_p(x; U^e_{-p}) : V_p(U^0_p, U^e_{-p}) < V_p(U^e_p, U^e_{-p}). \quad (4.11)$$

Adoption of this solution at some time $k_0 + \alpha$ where $\alpha \in (0, \infty)$ implies $V_p(k_0 + \alpha) < V_p(k_0)$, which contradicts Proposition 4.1, wherein it is established, for constant $A_p(k), V_p(k) = V^e_p, \forall x(k) \in \mathcal{X}^e, \forall k$. Thus, we have a contradiction, and so (4.11) can not hold. Consequently, $\mathcal{X}^e$ being a limit set is sufficient for $U^e$ to be a Nash solution for all $x \in \mathcal{X}^e$.

(ii) **Necessity:** assume that $\mathcal{X}^e$ is not a limit set, but, for any $x \in \mathcal{X}^e$, $U^e$ is a Nash solution. Thus, for any $x \in \mathcal{X}^e$, the solution

$$U^e_p = \{x_p, \kappa_{F_p}(x_p), \kappa_{F_p}(A_p x_p + B_p \kappa_{F_p}(x_p)), \ldots\},$$

is optimal for each $p$. However, by Proposition 4.1, $\tilde{\mathcal{X}}^e$ is invariant under the (nominal) control $\tilde{u}_p = \kappa_{F_p}(\tilde{x}_p), \forall p$, and $\mathcal{X}^e = \tilde{\mathcal{X}}^e \oplus \{\mathcal{R}_1 \times \ldots \times \mathcal{R}_{N_p}\}$ is therefore robustly-invariant under the control $u_p = \kappa_{F_p}(x_p) + K_p(x_p - \bar{x}_p)$, where $x_p - \bar{x}_p \in \mathcal{R}_p$. Thus, we have a contradiction, and conclude that $\mathcal{X}^e$ being a limit set is necessary for $U^e$ to be a Nash solution for all $x \in \mathcal{X}^e$. \qed

133
The consequence of this result is that, upon convergence to a limit set $X^e$, every
game played thereafter is unchanging in the sense that the favoured solution for
each $p$, at any state in $X^e$, is to adopt a constant control policy. This policy, based
on $u_p = \kappa_{F_p}(x_p)$, maintains the nominal system in $X^e$, and hence the true system
in $X^e = \tilde{X}^e \oplus \{R_1 \times \ldots \times R_{N_p}\}$, with a constant cost. Furthermore, as this result
establishes an equivalence between the limit set and the games therein being in
equilibria, it permits, via Propositions 4.2 and 4.3, a link to be made between limit
sets and the cooperation graph. This is achieved in the following Theorem.

**Theorem 4.1 (Inclusion of Closed-Loop Limit Sets).** Suppose $X^e_N$ is the set of
closed-loop limit sets $X^e$ that the system controlled by the DMPC$^N$ algorithm may
converge to, and $X^e_c$ is the corresponding set for system controlled by DMPC$^c$,
where $G^c \subseteq G^N$. Then $X^e_N \subseteq X^e_c$.

**Proof.** Let $X^e$ be a limit set for DMPC$^N$. By Proposition 4.4, the control law
associated with remaining in $X^e$ is a Nash solution to every game $G^N(x), \forall x \in
X^e$. By Proposition 4.2, a Nash solution $G^N(x)$ is also a Nash solution to $G^N(x)$.
Therefore, by Proposition 4.4, $X^e$ is also a limit set for DMPC$^c$, which further
implies that $X^e_N \subseteq X^e_c$. \hfill \Box

By extension, this result implies that $X^e_N \subseteq X^e_0$, i.e., the set of closed-loop limit
sets for DMPC$^N$ is a subset of those for non-cooperative DMPC. Furthermore,
because a team-optimal strategy, i.e., a solution to a centralized optimization, is
always a Nash solution (Proposition 4.3), it follows that $X^c_c \subseteq X^e_c$, where $X^c_c$ is the
set of closed-loop limit sets for a centralized implementation.

Note that the theorem applies to the set of closed-loop limit sets rather than a
sole limit set. For the system at a particular initial state, in general many possible
state limits may exist. The convergence to a particular limit set out of many is de-
pendent on the sequence in which distributed optimizations are performed, i.e., the
update sequence.
Theorem 4.1 is demonstrated in the next example, where a collection of constrained masses converges to 'better' state equilibria by increasing cooperation.

**Example 4.2** (Convergence of constrained masses). Consider again the system of identical point masses from Example 2.1, with the double-integrator dynamics

\[
x_p(k + 1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_p(k) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u_p(k) + w_p(k).
\]

Now each mass is subject to local velocity and control input constraints \(|x_{p,2}| \leq 0.2\) and \(|u_p| \leq 0.15\) respectively, where \(x_{p,2}\) denotes the second component of the state \(x_p\). For clarity and simplicity, the dynamics here are assumed to be disturbance-free, i.e., \(\mathcal{W}_p = \{0\}, \forall p\).

The objective is to control the system so that the infinite-horizon function

\[
\sum_{k=0}^{\infty} \sum_{p=1}^{N_p} \Psi_p(x_p(k))
\]

is minimized, where \(\Psi_p(x_p(k)) = x_p(k)^T x_p(k)\). The local subsystem objective is therefore chosen so that the distance of the terminal state from the origin is penalized, while the stage cost ensures convergence to a terminal set where the mass is stationary and the constraints are satisfied:

\[
l_p(x_p, u_p) = x_p^T Q x_p + u_p R u_p,
\]

\[
F_p(x_p) = x_p^T P x_p,
\]

\[
\mathcal{X}_{F_p} = \{x_p : S x_p \leq 0\},
\]

where the weighting matrices are chosen as

\[
Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 1, \quad P = 10I_2,
\]

135
To ensure admissibility of this terminal set, the optimization constraints are extended to additionally cover the terminal prediction step \( j = N \), using \( u_p(k+N|k) = \kappa_{F_p}(x_p(k+N|k)) \) as the control; Assumption 4.1 is then satisfied for \( \kappa_{F_p} = 0 \). Note the positive semi-definite nature of \( Q \) promotes stabilization to zero velocity states, but does not penalize positional deviations from the origin.

In the first instance, consider the case where \( N_p = 2 \). The two masses are subject to a coupled constraint that sets a lower bound on the sum of their positions, \( x_{1,1} + x_{2,1} \geq 1 \), where \( x_{p,1} \in [0,1] \) denotes the first component of the state \( x_p \). Figure 4.4 shows the results for a simulation with initial state \( x_{p,1} = 1, \forall p \in \{1,2\} \), with all velocities initially zero. The prediction horizon is 6 steps, and the update sequence

\[
S = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}.
\]
employed is the simple alternating sequence, \{1, 2, \ldots, 1, 2\}. The figure shows the convergence of the positions of the masses to a limit state for three controllers:

1. DMPC\(^\Phi\): non-cooperative algorithm of Chapter 2;

2. DMPC\(^{N^1}\), where cooperation graph \(G^{N^1}\) is such that \(N_p = \{1 + (p \bmod N_p)\}, \forall p;\)

3. CMPC: centralized MPC.

The CMPC algorithm converges to a team-optimal position, \(x_{1,1} = 0.5, x_{2,1} = 0.5\), with associated cost \(x^T x = 0.5\). For DMPC\(^\Phi\), a preference is shown for mass 1, which updates first; convergence is seen to the point \(x_{1,1} = 0.35, x_{2,1} = 0.65\), with a higher cost value of \(x^T x = 0.545\). (Note that had the update sequence been reversed, the preference would be for mass 2, with final convergence to the opposite point \(x_{1,1} = 0.65, x_{2,1} = 0.35\)). For DMPC\(^{N^1}\), however, the masses again converge to the optimal point \(x_{1,1} = 0.5, x_{2,1} = 0.5\).

Extending the problem to the \(N_p = 5\) case, subject to the constraint \(\sum_{p=1}^{5} x_{p,1} \geq 1\), two additional controllers are employed:

1. DMPC\(^{N^2}\), where cooperation graph \(G^{N^2}\) is such that \(N_p = \{1 + (p \bmod N_p), 1 + (p + 1 \bmod N_p)\}, \forall p, \text{i.e., the next two masses in line to update, and,}\)

2. DMPC\(^{N^1*}\): similar to DMPC\(^{N^1}\), but with \(N_p = \emptyset\) for masses \(p \in \{4, 5\}\).

Figure 4.5 shows the positions of the point masses against time when controlled by each of the four distributed algorithms, and in each case compared with the results of CMPC. It can be seen that under CMPC, as expected, the system converges to the team-optimal state—an equal share such that \(x_{p,1} = 0.2, \forall p, \text{ and } \sum_p x_{p,1} = 1\). The same outcome follows for DMPC\(^{N^2}\) and DMPC\(^{N^1}\). Note that neither of these schemes employs a full cooperation graph, implying that ‘full’ cooperation is not necessary for team-optimal convergence. Conversely, an empty cooperation graph
Figure 4.5: Convergence of five masses to different limits. Shown are the position histories of the five masses, comparing, on each plot, the distributed controller (solid line) with the centralized controller (dashed). All cooperative controllers avoid the 'unfair' outcome of non-cooperative DMPC. In (d), the partially-cooperative DMPC$^{1*}$ breaks the DMPC$^0$ deadlock.

is not sufficient, as indicated by Figure 4.5(a), where the system converges to a sub-optimal limit state.

One of the distributed control schemes, DMPC$^{N/1*}$, is used only after a number of steps of non-cooperative control, and the results are shown in Figure 4.5(d). By the time immediately before the switch to cooperative DMPC is made, the masses have again converged to the 'greedy' limit states. This partially-cooperative scheme, in which no other control agent attempts cooperation with agent 5, breaks the 'greedy' deadlock, the cooperating masses 1 to 4 converging to a common point, leaving only mass 5 unchanged. This confirms that a limit state for non-cooperative DMPC$^0$ is
Table 4.1: Five mass system example: comparison of controllers

<table>
<thead>
<tr>
<th>Controller</th>
<th>DMPC₀</th>
<th>DMPCᵦ¹</th>
<th>DMPCᵦ²</th>
<th>CMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>x(70)ᵀx(70)</td>
<td>0.3264</td>
<td>0.2003</td>
<td>0.2000</td>
<td>0.2000</td>
</tr>
<tr>
<td>∑ₖ x(k)ᵀx(k)</td>
<td>52.77</td>
<td>46.24</td>
<td>45.81</td>
<td>37.97</td>
</tr>
<tr>
<td>Decision Variables</td>
<td>29</td>
<td>58</td>
<td>87</td>
<td>145</td>
</tr>
<tr>
<td>Constraints</td>
<td>67</td>
<td>135</td>
<td>191</td>
<td>287</td>
</tr>
<tr>
<td>Data Exchanges</td>
<td>272</td>
<td>272</td>
<td>272</td>
<td>552</td>
</tr>
</tbody>
</table>

not a limit state for DMPCᵦ¹⁺, and, also, that all non-empty cooperation graphs do not necessarily imply convergence to a team-optimal state.

Table 4.1 compares the different controllers for the five mass example. Communication between subsystems is measured as the number of data exchanges over the duration of the simulation. In this case, all masses are coupled, so use of a cooperative cost incurs no extra communication. The size of the respective optimization problems is represented by the number of decision variables and constraints per optimization. CMPC provides the best performance at the expense of a large optimization problem and a high level of communication. For DMPC, performance is seen to improve with increasing optimization size, but, crucially, without requiring any additional communication.

The main contribution of this section, and of the first part of this chapter, is the rigorous confirmation of the intuitive concept that, by increasing the level of inter-agent cooperation, the state convergence outcome is no worse in terms of system-wide performance. Examples have shown that cases do exist where, by employing even partially-cooperative DMPC instead of non-cooperative DMPC, the system states converge to a team-optimal limit—the same as that for centralized—breaking an otherwise sub-optimal, deadlock outcome.

The second part of this chapter seeks to answer the question of how the cooperation graph should be chosen to offer best performance when no longer restricted to convergence outcome; i.e., transient performance is also considered. An adaptive
cooperation algorithm is proposed, based on the currently-active edges in the coupling graph. A key finding is that the set of immediate, coupled subsystems is not necessarily the optimal cooperating set.
4.4 Adaptive cooperation in robust DMPC

In this section, it is shown how the structure of the coupling may be exploited to best choose the cooperation graph, and subsequently the cooperating sets in the local optimizations. The DMPC algorithm of previous chapters is modified to include choosing on-line the cooperating set that is expected to promote best performance.

The next subsection examines the effect of the coupling graph on the set of feasible decisions available to control agents. Then, in Section 4.4.2, a time-varying cooperation graph, based on active coupling constraints, is introduced. In Section 4.4.3, an adaptive DMPC algorithm is proposed, where decisions on cooperating sets in local optimizations are made on-line. Finally, numerical examples are provided.

4.4.1 Properties of the coupling graph

In this subsection, it is shown that the existence of paths, and not just adjacent nodes, in the coupling graph is important in determining the cooperation graph, $\mathcal{G}^N$; specifically, we confirm the intuitive concept that a local agent $p$ cooperating with some agent $q$ to which no ‘chain’ of couplings exists offers no benefit. It follows that, if such a ‘chain’ of coupling constraints does exist, that is, a path $\{p, \ldots, p_i, \ldots, q\}$ exists in $\mathcal{G}_*$, it may be beneficial to system-wide performance for the edge $\{p, q\}$ to exist in $\mathcal{G}^N$. Consequently, whereas previous work, e.g. Kuwata and How [43], Keviczky et al. [45], has typically adopted cooperation only between immediately-adjacent agents, i.e., $\mathcal{G}^N \subseteq \mathcal{G}_*$, it may be desirable to cooperate with agents not immediately coupled, allowing $\mathcal{G}^N \supset \mathcal{G}_*$.

As in the previous section, the local decision variable shall be denoted $U_p(k)$, the collection of initial state and sequence of controls. In that section, a number of feasible sets were defined

1. the global set $U(x)$ for variables $U = \{U_1, \ldots, U_{N_p}\}$;
2. the local set $U_p(x; U^*_{-p})$ for $U_p$, the decision variable for $p$;
3. a combined set \( U_{\{p,q\}}(x; U^*_{\{p,q\}}) \) for \( \{U_p, U_q\} \);

4. finally, a constrained set \( U^N_p(x; U^*_{-p}) \) for the local optimization for \( p \) with a cooperating set \( N_p \). Note \( U^N_p(x; U^*_{-p}) = U_p(x; U^*_{-p}) \) if \( N_p = \emptyset \).

Using these definitions, the first result concerns edges in the coupling graph, and formalizes the fact that the combined decision space of a pair of agents is coupled only if the subsystems are coupled; otherwise, it is the product set. In other words, the choice of a decision variable \( U_p \) may only depend on another variable \( U_q \) if they share a coupling constraint. Therefore, if no such constraint exists, the choices may be made independently:

\[
\min \left\{ J_p(U_p) + J_q(U_q) : \{U_p, U_q\} \in U_{\{p,q\}}(x; U^*_{\{p,q\}}) \right\} = \\
\min \left\{ J_p(U_p) : U_p \in U_p(x; U^*_{-p}) \right\} + \min \left\{ J_q(U_q) : U_q \in U_q(x; U^*_{-q}) \right\}.
\]

**Proposition 4.5 (Edges in \( G^* \)).** Suppose \( U_p(x; U^*_{-p}) \) is the feasible set for the decision variable \( U_p \) for subsystem \( p \), and \( U_q(x; U^*_{-q}) \) is the corresponding set for \( U_q \). Then (i) \( U_{\{p,q\}}(x; U^*_{\{p,q\}}) \subseteq U_p(x; U^*_{-p}) \times U_q(x; U^*_{-q}) \) only if \( \{p,q\} \) is an edge in \( G^* \), and (ii) \( U_{\{p,q\}}(x; U^*_{\{p,q\}}) = U_p(x; U^*_{-p}) \times U_q(x; U^*_{-q}) \) if \( \{p,q\} \) is not an edge in \( G^* \).

**Proof.** For (i), suppose \( U_{\{p,q\}}(x; U^*_{\{p,q\}}) \subset U_p(x; U^*_{-p}) \times U_q(x; U^*_{-q}) \). Then there exists some \( U_p^0 \in U_p(x; U^*_{-p}) \) and some \( U_q^0 \in U_q(x; U^*_{-q}) \) such that \( \{U_p^0, U_q^0\} \notin U_{\{p,q\}}(x; U^*_{\{p,q\}}) \). Given that \( U_p(x; U^*_{-p}) \) and \( U_q(x; U^*_{-q}) \) are defined by local and coupling constraints, and only the latter involve the decision variables of other subsystems, the infeasibility must arise from violation of some coupling constraint \( c \in C_p \cap C_q \). Thus, \( p \in Q_q \) and \( q \in Q_p \), so \( \{p,q\} \in E^* \). For part (ii), suppose \( \{p,q\} \notin E^* \). Then \( q \notin Q_p \) and \( p \notin Q_q \), so \( \#\{U_p^1, U_q^1\} \in U_p(x; U^*_{-p}) \times U_q(x; U^*_{-q}) \) such that \( \{U_p^1, U_q^1\} \notin U_{\{p,q\}}(x; U^*_{\{p,q\}}) \). Then \( \{p,q\} \notin E^* \) implies \( U_{\{p,q\}}(x; U^*_{\{p,q\}}) = U_p(x; U^*_{-p}) \times U_q(x; U^*_{-q}) \).
Though this result is not important in itself, it leads to the following, the main result of this subsection. The addition to a cooperating set $\mathcal{N}_p$ for agent $p$ of a subsystem $n$, to which no path in the coupling graph exists, leaves the result of the optimization unchanged. This implies that there is nothing to be gained by an agent cooperating with agents that do not belong to the same connected component [101] of the coupling graph, i.e., those that are completely decoupled from $p$.

**Proposition 4.6 (Paths in $G^x$).** Suppose $\mathbf{U}_p^{\text{opt}}$ is a minimizer of (4.7) with cooperating set $\mathcal{N}_p \subset \mathcal{P}\backslash\{p\}$, where a path $\{p, \ldots, q\}$ exists in $G^x$ for all $q \in \mathcal{N}_p$. Then $\mathbf{U}_p^{\text{opt}}$ is also a minimizer of the optimization with cooperating set $\{\mathcal{N}_p, n\}$ if no path $\{p, \ldots, n\}$ exists in $G^x$.

**Proof.** The optimization for $p$ is over the set $\mathcal{U}_p^{N_p}(x; \mathbf{U}_p)$:

$$
\mathcal{U}_p^{N_p}(x; \mathbf{U}_p^{\text{opt}}) = \left\{ \mathbf{U}_p, \hat{\mathbf{U}}_{N_p} \right\},
$$

assuming the current, feasible plans are $\mathbf{U}_p$, $\forall p \in \mathcal{P}$. Consider the second constraint. If no path $\{p, \ldots, n\}$ exists in $G^x$, then no edge exists between the new subsystem $n$ and either $p$ or any $q \in \mathcal{N}_p$. By Proposition 4.5, it follows that $\mathbb{U}_{i,n} = \mathbb{U}_i \times \mathbb{U}_n$, $\forall i \in \{p, \mathcal{N}_p\}$, and, furthermore, $\mathbb{U}_{\{p, \mathcal{N}_p, n\}} = \mathbb{U}_{\{p, \mathcal{N}_p\}} \times \mathbb{U}_n$. Thus, the augmented optimization is over a decoupled feasible set

$$
\mathcal{U}_{p\{\mathcal{N}_p, n\}}(x; \mathbf{U}_p^{\text{opt}}) = \left\{ \mathbf{U}_p, \hat{\mathbf{U}}_{N_p}, \hat{\mathbf{U}}_n \right\},
$$
and the optimal values of $U_p$ are unaffected by the addition of $n$. □

This result leads to an analytical upper bound on size of the cooperation graph necessary for lowest local cost. Let $G^\text{pairs} = \{\mathcal{P}, \mathcal{P}^{(2)}\}$ denote the 'all-pairs' graph of nodes $\mathcal{P}$, where the set of edges $\mathcal{P}^{(2)}$ is the set of pairwise node permutations [101]. Also, supposing a graph $\mathcal{G} = \{\mathcal{P}, \mathcal{E}\}$ may be decomposed into $m$ independent, connected components, define $\Pi(\mathcal{E})$ as the set of edges of a graph that consists of the $m$ components, each component being an 'all-pairs' subgraph. That is, $\{p, q\} \in \Pi(\mathcal{E})$ if and only if $\{p, \ldots, q\}$ is a path in $\mathcal{G}$. Then, Proposition 4.6 implies that a sufficient choice of the cooperation graph $\mathcal{G}^N = \{\mathcal{P}, \mathcal{E}^N\}$ is given by the edges

$$\mathcal{E}^N \subseteq \Pi(\mathcal{E}^2) \begin{cases} = \mathcal{P}^{(2)} & \text{if } m = 1, \\ \subset \mathcal{P}^{(2)} & \text{if } m > 1. \end{cases}$$

An immediate consequence is that unnecessary computation and communication are avoided. In a sense, this result is trivial. However, it does not eliminate the possibility of cooperation with those beyond immediate neighbours, and the next section exploits this for a system-wide performance benefit. The DMPC algorithm is modified to include an adaptive choice of neighbourhood, based on currently-active coupling constraints.

### 4.4.2 Adaptive cooperation based on active couplings

Define the active coupling graph at a time $k$ as $G^\text{active}_k = \{\mathcal{P}, \mathcal{E}_k^{\text{active}}\} \subseteq G^\text{a}$. Motivated by the results of the previous section, at a time step $k$, it is proposed that the cooperation graph $\mathcal{G}^N$ is chosen according to the existence of paths in $G^\text{active}_k$ rather than $G^\text{a}$, as it is assumed that inactive constraints have little effect on the optimization results. The graph from the previous time step, $k - 1$, is used, as it is not known a priori which constraints will be active at the current step $k$. This compares with the method of Kuwata and How [43], where only current actively-coupled neighbours are involved in cooperation.
Figure 4.6 shows how the cooperating set decision may be made, based on the current active coupling graph. Also shown is an example coupling graph of six subsystems, one of which is disconnected, and the active and inactive constraints at an instant. A number of schemes are proposed for making the cooperating set decision, and these are defined in the table. The resulting cooperating set for subsystem 1 according to each scheme is then determined. The following relationships may be inferred:

- $E_3^N \subseteq E_2^N$;
- $E_4^N \subseteq E_3^N$;
- $E_5^N \subseteq E_3^N$;
- $E_6^N \subseteq E_5^N$;
- $E_6^N \subseteq E_4^N$;
- $E_4^N \subseteq E_7^N \subseteq E_3^N$.

Note that the ‘system’ scheme follows the ‘full’ cooperation approach of Venkat et al. [36], while ‘adjacent’ restricts cooperation to between directly-coupled subsystems. The ‘max’ scheme exploits the result of Proposition 4.6 to eliminate cooperation between agents of different connected components. Observe that it is the...
adaptive scheme ‘max active’—the main development of this section—that obtains the interesting result; in addition to the actively-coupled neighbour included by ‘adjacent active’, the scheme closest to those of Keviczky et al. [45] and Kuwata and How [43], the cooperating set also contains a non-directly coupled subsystem.

### 4.4.3 Implementation of active cooperation graph determination

Supposing the coupling constraints are polyhedral, so that

\[ g_c(z_{c1}, z_{c2}, \ldots, z_{cN_p}) \leq 0 \]

where \( z_{cp} \in \mathbb{R}^{N_z_c} \) and \( g_c : \mathbb{R}^{N_z_c} \rightarrow \mathbb{R}^{m_c} \), then with each constraint in the optimization (4.7) for \( p \) is associated a KKT multiplier \( \lambda_{cp} \in \mathbb{R}^{m_c} \). If \( U_p^{opt}(k) \) is a solution to the optimization at step \( k \), given plans \( U_q^*(k) \) for all \( q \neq p \), then

\[
\exists \lambda_{cp}^{opt}(k + j|k) \geq 0 : g_c(z_{cp}^{opt}(k + j|k), z_{c{-p}}^{opt}(k + j|k)) \leq 0,
\]

\[
\lambda_{cp}^{opt}(k + j|k) g_c(z_{cp}^{opt}(k + j|k), z_{c{-p}}^{opt}(k + j|k)) = 0,
\]

for all \( c \in C_p, j \in \{0, \ldots, N\} \), where \( z_{c{-p}}^{opt}(\cdot|k) \) denotes the collection

\[
\{z_{c1}^{*}(\cdot|k), \ldots, z_{c(p{-1})}^{*}(\cdot|k), z_{c(p+1)}^{*}(\cdot|k), \ldots, z_{cN_p}^{*}(\cdot|k)\}.
\]

Non-zero multipliers provide an indication of active constraints [86], and thus support the determination of the active coupling graph. To this end, we associate a weighting with each edge in the active coupling graph, based on the multiplier value. For the edge from \( p \) to \( q \), the weighting is defined as

\[
\omega_{pq}(k) = \begin{cases} 
1/\left(\max_{c \in C_p \cap C_q} \|\lambda_{cp}(k)\|\right) & \text{if } q \in Q_p \\
\infty & \text{if } q \notin Q_p,
\end{cases}
\]
where $\lambda_{cp}(k) = \{\lambda_{cp}(k|k), \ldots, \lambda_{cp}(k+N|k)\}$ is the collection, over the horizon, of multipliers for constraint $c$ from the optimization for $p$. The graph is now directed because, in general, the multiplier $\lambda_{cp}$ associated with constraint $c$ in $p$'s optimization is not necessarily equal to the corresponding multiplier in $q$'s optimization, where $q \in Q_p$. When constructing the active graph, multiplier values for an optimization are obviously not available a priori; therefore, candidate values associated with the default, feasible plan—based on values from the last time an agent updated—are available, and the decision on which constraints are currently active is based on these.

$$
\lambda_{cp}(k+j|k) = \begin{cases} 
\lambda_{cp}^*(k+j|k-1), & \forall j \in \{0, \ldots, N-1\}, \\
\lambda_{cp}^*(k+N-1|k-1), & j = N.
\end{cases}
$$

(4.12)

The use of weights allows conversion to a path problem; if a path can be found from $p$ to $q$ in the active coupling graph, with finite weight, then $q$ shall be included in the cooperating set for $p$. The DMPC algorithm is modified so that an optimizing subsystem, $p_k$, uses Dijkstra's shortest path algorithm [104] to determine a vector $D_p$ of shortest paths to each other agent in the active coupling graph.

To facilitate the path search, information requirements extra to those identified in the previous chapter apply to the multipliers for each subsystem. At a time step $k$, an updating agent $p_k$ must, prior to optimizing, make a decision on whom to cooperate with. The decision is based on the latest active coupling graph, $G_{k-1}^N$, as the current graph, $G_k^N$, is not known until optimizations are solved. A local agent can identify all necessary paths, therefore, if it has the most recent multipliers from each other agent it could cooperate with. That is, using the results of the previous section, if $p_k$ has received most-recent information

$$
\{\lambda_{cq}^*(\hat{k}_q|\hat{k}_q), \lambda_{cq}^*(\hat{k}_q + 1|\hat{k}_q), \ldots, \lambda_{cq}^*(\hat{k}_q + N|\hat{k}_q)\}_{c \in C_q}
$$

from all $q$ in the same connected component of the coupling graph, then the multipliers may be propagated successively to the current step by repeated application
Algorithm 4.1: Robust DMPC with adaptive cooperation

1. Design stabilizing controller $K_p$ and RPI set $R_p$;
2. Tighten sets $Y_p$, $Z_c$, $\forall c \in C_p$, via (2.16);
3. Wait for feasible solution $U_p^*(0)$, multipliers $\lambda_p^*(0)$, information $\ddot{Z}_p^*(0)$, terminal set $X_{F_p}$ and controller $\kappa_{F_p}$, and active coupling graph $G_0^{active}$ from central agent;

4. for $k = 1 : \infty$
   
   5. Sample current state $x_p(k)$;
   
   6. if $p_k = p$ then
      
      7. $D_p = \text{dijkstra}(G_{k-1}^{active}, p)$;
      
      8. $N_p \leftarrow \emptyset$;
      
      9. for $q \in P \setminus \{p\}$ do
         
         10. if $D_p[q]$ finite then $N_p \leftarrow N_p \cup \{q\}$
      
      end
      
      11. Obtain new plan $U_p^{opt}(k)$, $\lambda_p^{opt}(k)$ as solution to $P_{p}^{N_p}(x_p(k); \ddot{Z}_p^*(k))$
      
      12. Transmit new plan and multipliers to other agents;
   
   else
      
      13. Renew current plan and multipliers via (2.19) and (4.12);
   
   end
   
   14. Apply control (2.17): $u_p(k) = \bar{u}_p(k|k) + K_p(x_p(k) - \bar{x}_p(k|k))$
   
   15. Update $G_{k-1}^{active}$ to $G_k^{active}$;
   
   16. Wait one time step;

of (4.12), without any additional information required. The path search down-selects those agents to provide the set of agents that the optimizing agent will cooperate with. Following optimization, the agent transmits its new multipliers to all others in the same connected component.

Note that for the condition of line 10, more sophisticated conditions could be employed by the designer. For example, only paths shorter than some threshold length. In fact, a constraint’s multiplier value describes the change in optimization cost per unit movement of the constraint [86]. Thus, the shorter the path to another subsystem, the larger the multiplier value, and the greater the anticipated performance benefit of including that subsystem in the cooperating set.

148
Figure 4.7: ‘Ring-like’ coupling graph (left) for input constraints. The non-coupled subsystems are shaded according to update group; $P_{\text{odd}} = \{1, 3, 5\}, P_{\text{even}} = \{2, 4, 6\}$. Cooperation graph (right) for ‘system’ ($G^N_2$) or ‘max’ ($G^N_3$) scheme. Each edge is bidirectional. The cooperation graph for ‘adjacent’ coincides with $G^2$.

4.4.4 Numerical examples

The following example presents simulation results using the distributed NIPC cooperation schemes defined in Figure 4.6. The proposed ‘max active’ scheme is found to provide better performance than each of the immediate-neighbour cooperation schemes, rivalling the performance of the ‘max’ scheme, yet for less computation.

Example 4.3. The system to control comprises six point masses, each with the dynamics

$$x_p(k + 1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_p(k) + \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} u_p(k) + w_p(k).$$

The local velocity and control input constraints are $|x_{p,2}| \leq 5$ and $|u_p| \leq 1$ respectively. Coupling is in the form of maximum control input limits applied to each consecutive pair. That is,

$$|u_p| + |u_{p+1}| \leq 1.4, p \in \{1, \ldots, 5\},$$

$$|u_6| + |u_1| \leq 1.4,$$

so that the coupling graph is a ‘ring’, with each edge corresponding to a different coupling constraint. Figure 4.7 depicts this arrangement.
The stage and terminal costs are quadratic:

\[ l_p(x_p, u_p) = x_p^T Q x_p + u_p R u_p, \]
\[ F_p(x_p) = x_p^T P x_p, \]

where \( Q = I_2, R = 0.001, \) and

\[ P = \begin{bmatrix} 2.0007 & 0.5010 \\ 0.5010 & 1.2518 \end{bmatrix} \]

is that of the optimal, unconstrained, nominal LQR problem \( (A_p, B_p, Q, R) \), as is the tube controller \( K_p \). A outer-approximation to the minimal RPI set [79] is computed [84] for \( R_p \), given a disturbance set \( \mathcal{W}_p = \{ w_p : \| w_p \|_{\infty} \leq W_{\text{max}} \} \), where here \( W_{\text{max}} = 0.05 \). Constraint sets \( \mathcal{Y}_p, \mathcal{Z}_c \) are tightened accordingly. The terminal sets, \( \mathcal{X}_{F_p} \), are the maximal output-admissible invariant sets under \( \kappa_{F_p}(x_p) = K_p x_p \).

For initial states spaced evenly between \( x_1(0) = [12 \ 1]^T \) and \( x_6(0) = [8 \ 1]^T \), the masses were controlled by DMPC with the cooperation schemes proposed in Figure 4.6, in the previous section. Because the system consists of only one connected component, the ‘system’ and ‘max’ schemes are identical. The cooperating set agent weightings \( \alpha_{pq}, q \in \mathcal{N}_p, \) in the local optimizations are set to unity. For updating, the observation is made that odd- and even-numbered pairs are not directly coupled. Thus, as identified in Remark 2.6, we may replace the updating agent \( p_k \) with an updating set, \( \mathcal{P}_k \), of non-coupled subsystems. For this purpose, the system is divided into two groups of decoupled masses, correspondingly shaded in Figure 4.7, which then form the update sets at alternating time steps, i.e. \( \mathcal{P}_k = \mathcal{P}_{\text{odd}} = \{1, 3, 5\}, \mathcal{P}_{k+1} = \mathcal{P}_{\text{even}} = \{2, 4, 6\} \). To account for this parallel updating in the local cooperative optimizations, the initial constraint \( \hat{u}_q(k|k) = \hat{u}_q^*(k|k - 1), q \in \mathcal{N}_p, \) is omitted, allowing freedom of deviations in hypothetical plans from the initial step (see Remark 3.1). The horizon is 25 steps, and each resulting optimization problem is a quadratic program.
As each local objective places a heavier weighting on state than on control, a local agent seeks to steer a mass quickly to the origin using the control effort available. In each case, therefore, the system was initialized with a plan where control effort was minimized rather than state, so that all of the shared control effort available via the coupling constraints is not allocated by the initialization. At $k = 5$, each mass is subjected to a disturbance $w_p = -[W_{\text{max}} \ W_{\text{max}}]^T$ for one time step.

Figure 4.8 illustrates the position histories of the six masses, when controlled by three of the DMPC schemes; 'greedy', 'max active', and 'adjacent active'. Note that the latter two schemes lead to considerably different trajectories, proving that the extra cooperation in 'max active' is having some effect on the local optimizations. In fact, the 'max active' scheme result more closely resembles the 'greedy' result than the 'adjacent active' result. Figure 4.9 shows the maximal size of the cooperating set.
Figure 4.9: Maximal cooperating set sizes for six-mass example.

$N_p$ among updating agents, over the duration of the simulation. Only the adaptive schemes, identified in Figure 4.6, are time-varying. At times, the new 'max active' is solving the 'system' optimization, settling to an empty cooperating set when constraints are inactive.

A comparison between the schemes, in terms of closed-loop cost versus computation time, is shown in Figure 4.10. Where agents update simultaneously, the maximum computation time is taken. Means and standard deviations of times over a simulation are calculated, and averaged over 10 repetitions. All simulations were performed on a Pentium 4 with 2,048 MB RAM, using CPLEX 10.1 as the QP solver. As expected, the 'greedy' scheme results in shortest computation times and centralized the longest, with comparatively poor performance for the former. The 'system' and 'max' schemes, the graphs of which are shown in Figure 4.7, obtain the lowest cost of all the distributed schemes, closest to centralized, and for a comparable mean computation time. Surprisingly, 'adjacent', where cooperation is always
with immediate neighbours — and the cooperation graph coincides with the coupling graph — delivers the highest costs, indicating that this approach to cooperation is inadequate for the problem posed. Whereas it was shown earlier in the chapter that by adding any agent to the cooperating set, the steady-state limit the system converges is no worse, in terms of system-wide cost, here we see that such a result does not hold for transient performance. However, by modifying this static ‘adjacent’ scheme, adding cooperation with masses connected by active paths, (‘adjacent + max active’), performance is improved.

For the ‘adjacent active’ scheme, the performance degradation seen with ‘adjacent’ is avoided, and computation reduced, yet performance is still marginally worse than for ‘greedy’. The ‘max active’ scheme — the major development of this section — additionally includes masses coupled via active paths. Now a performance improvement over ‘greedy’ is seen, with the closed-loop cost closest to that of ‘system’, exploiting the benefits of cooperation, as identified in Section 4.4.1, but with-
out unnecessary computation. Furthermore, this confirms that the choice of *only* actively-coupled neighbours in the cooperating set is not sufficient to guarantee best DMPC performance. In conclusion, the new 'max active' scheme offers a middle-ground between the obvious choices of cooperation graph, *i.e.* the 'max' graph and the 'active adjacent' graph.

### 4.5 Summary

This chapter has examined in detail the use of inter-agent cooperation in distributed MPC. Firstly, by relating some game-theoretical concepts to the algorithm, it has been shown that the set of closed-loop state limit sets that the system, under a particular cooperation graph, can converge to is a subset of the corresponding set of limit sets for a more sparsely-connected graph. By example, it has been shown that, starting from non-cooperative DMPC, the controlled system can reach 'better' convergence state limits by using successively larger cooperating sets, though at the expense of computation.

In the second part of the chapter, the cooperation graph decision has been linked to the structure of the coupling graph, and a new, adaptive scheme for cooperation between agents proposed. By showing that cooperation between DMPC agents in different connected components of the coupling graph is redundant, when seeking to improve system-wide performance, an upper bound on the level of cooperation has been established. Conversely, cooperation with, in addition to directly-coupled neighbours, those connected by a path in the coupling graph may improve performance. The new adaptive approach, the 'active max' scheme, then proposes cooperation between subsystem agents connected via paths of active coupling constraints. Simulations have shown that, by adding cooperation with such connected agents, performance may improve over that where cooperation is between actively-coupled neighbours only, exploiting the benefits of cooperation, yet without unneces-
sary computation. Furthermore, this confirms that the set of immediately-adjacent neighbours is not necessarily the optimal cooperating set.
Chapter 5

Distributed MPC with Parallel Updates

An extension to the distributed MPC algorithm is proposed, allowing optimizations of control agents’ plans in parallel, while maintaining robust constraint satisfaction and feasibility.

5.1 Introduction

The robust DMPC algorithm proposed in Chapter 2 guarantees feasibility by permitting only one subsystem agent to optimize its plan at any time step, while all other agents adopt the feasible candidate plan. The new plan is subsequently transmitted to other, coupled subsystem agents. Such an approach is not without drawbacks; most significantly, all other subsystems are being controlled in an open-loop manner while waiting for their respective times in the sequence of updates. For example, for $N_p$ agents updating in a simple alternating sequence, any one agent spends an average of $N_p - 1$ time steps running open-loop.
Generally, the vast majority of DMPC methods in the literature propose parallel-update schemes; for example, Dunbar [35], Venkat et al. [36], Keviczky et al. [45], Alessio and Bemporad [47], Giovanini and Balderud [105] to name but a few. Though such schemes may suffer disadvantages, including convergence speed and quality of solution [106], benefits may accrue from the known advantages of higher update rates: increased bandwidth, shorter response times, and better disturbance rejection. The main challenge for parallel-update methods is of ensuring feasibility and stability of the closed-loop system, the approaches to which were reviewed in Chapter 1. On the contrary, feasibility and stability of the DMPC algorithm in this thesis are guaranteed, and the challenge is achieving good system-wide performance in the closed-loop. It is to be expected that performance of the proposed DMPC will degrade with increasing \(N_p\), even using inter-agent cooperation, and a number of numerical examples in previous chapters have confirmed this.

The situation may be ameliorated somewhat, as identified in Remark 2.6 in Chapter 2, because any pair of agents \(\{p, q\}\) may update simultaneously—without affecting feasibility and stability of the overall system—if \(p\) and \(q\) are not coupled, \(i.e., p \notin Q_q, q \notin Q_p\). Thus, the updating agent \(p_k\) is replaced by an updating set, \(\mathcal{P}_k\), of non-coupled subsystems. This was employed briefly in the previous chapter, in particular for Example 4.3, where odd- and even-numbered pairs optimized alternately. However, such a situation is conditional on the structure of the coupled constraints; if all agents share some constraint \(c\), so that the subsystems are ‘fully-connected’, then there does not exist any pair, or number, of agents whom may update simultaneously. The idea of simultaneously updating connected groups of subsystems is studied in Keviczky et al. [45], wherein it is acknowledged that achieving feasibility in the presence of coupling constraints is a considerable challenge.

In this chapter, a general extension of the DMPC algorithm is developed to permit simultaneous updates for any number of subsystem agents in the problem, even if those subsystems share coupling constraints. That is, the updating set \(\mathcal{P}_k\) is no longer restricted to contain only non-coupled subsystems. Robust feasibility is
guaranteed by agents being allowed a certain amount of ‘authority’ to update, and further tightening coupling constraints by some margin. A sufficient condition on the size of these margins is derived, and it is shown, for the case of polyhedral constraints, that the satisfaction of this condition is easily established or verified. Subsequently, the performance benefits associated with updates in parallel are promoted, though at the expense of increased computation, communication, and extra conservatism, but without resorting to iteration or bargaining to maintain feasibility.

The outline of the chapter is as follows. In Section 5.2, the DMPC method is extended to permit simultaneous updates by agents, by further tightening coupling constraints locally. A sufficient condition for robust feasibility and constraint satisfaction is subsequently presented. A numerical example is provided in Section 5.3. Finally, the chapter is summarized in Section 5.4.

5.2 Feasible parallel-update DMPC

The development of the parallel-update DMPC stems from the following observation; that distributed optimization over a product set may be completely decoupled, hence executed in parallel, without loss of feasibility. That is, if \( U = U_1 \times U_2 \), then parallel optimization over \( u_1 \in U_1 \) and \( u_2 \in U_2 \) always results in a feasible outcome \( \{u_1, u_2\} \in U \). The same is not true for a coupled feasible set \( U \subset U_1 \times U_2 \). However, what if a distributed optimization took place over a product set contained in the coupled set \( U \)? Then distributed agents could optimize simultaneously, albeit over a smaller decision space, while maintaining feasibility. Figure 5.1 illustrates this concept.

Note that, in general, infinitely many candidate product sets for parallel optimization might exist. However, two inferences may be made: firstly, if the contracted product set is entirely contained within the feasible set, then any solution generated in parallel is feasible; secondly, if the contracted product set contains the current
solution, then a feasible solution always exists. The results developed in the next section formalize these criteria, with application to the DMPC problem statement.

5.2.1 Margins for robust feasibility

The *centralized* optimization problem $F^C(x_1(k), \ldots, x_{N_p}(k))$, defined in Section 2.3, requires satisfaction of the coupled constraints (2.14g):

$$\sum_{p=1}^{N_p} \bar{z}_{cp}(k + j|k) \in \bar{Z}_c, \forall c \in C.$$ 

In Section 2.4, the distributed form of the tube MPC algorithm was developed, wherein the local optimization problem for an updating agent $p$ considers the equiv-
alent constraints (2.30g) with the outputs of all coupled subsystems fixed:

\[
\tilde{z}_{cp}(k + j|k) + \sum_{q \in P \setminus \{p\}} \tilde{z}_{cq}^*(k + j|k) \in \tilde{Z}_c, \forall c \in C_p,
\]

Now suppose some set \( P_k^{\text{opt}} \subseteq P \) of subsystem agents optimize in parallel. Previously, only non-coupled subsystems were contained in the update set \( P_k \) at time \( k \), but here this restriction is relaxed. Given that a constraint \( c \in C \) involves the set \( P_c \subseteq P \) of subsystems, the set \( P_k^{\text{opt}} \) contains some subset \( P_k^{\text{opt}} \cap P_c \) of subsystem agents that share that constraint, a total number \( P_c \triangleq n(P_k^{\text{opt}} \cap P_c) \). Then the following must hold to satisfy the coupling constraint \( c \),

\[
\sum_{p \in P_k^{\text{opt}} \cap P_c} \tilde{z}_{cp}(k + j|k) + \sum_{q \in P_c \setminus P_k^{\text{opt}}} \tilde{z}_{cq}^*(k + j|k) \in \tilde{Z}_c.
\]  

That is, the outputs of all the optimizing agents, together with those of non-optimizing agents, must remain compatible.

To proceed, motivated by the introduction to this section, it is proposed that by further tightening each \( \tilde{Z}_c \) in the local optimization, by some margin to be determined, satisfaction of the constraints is always achieved. The coupling constraints (2.30g) in the optimization for each \( p \) in the updating set \( P_k^{\text{opt}} \) are modified to include this extra tightening:

\[
\tilde{z}_{cp}(k + j|k) + \sum_{q \in P \setminus \{p\}} \tilde{z}_{cq}^*(k + j|k) \in \tilde{Z}_c \sim M_{cp}(j), \forall c \in C_p,
\]

where \( M_{cp}(j) \) is agent \( p \)'s extra margin for constraint \( c \) at prediction step \( j \). Generally, it is desirable to keep this extra margin small, to minimize the extra conservativeness brought about. The new, local optimization problem for a subsystem \( p \)
is $\mathbb{P}^{PD}_p(x_p(k); Z_p^*(k))$, defined by

$$V_p^{\text{opt}}(x_p(k); Z_p^*(k)) = \min_{U_p(k)} V_p(U_p(k)) \text{ subject to } (2.30a)-(2.30f) \text{ and } (5.2).$$

The following result represents a sufficient condition for the choice of margins to guarantee feasibility. The standing assumptions for robust feasibility apply; that is, Assumptions 2.1–2.3, pertaining to the existence of an RPI set $\mathcal{R}_p$, a terminal set $\chi_{F_p}$, and a terminal control law $\kappa_{F_p}(x_p)$.

**Proposition 5.1 (Margins for parallel updates).** Suppose the sequence of controls $U_p^*(k_0) = \{x_p^*(k_0|k_0), \bar{u}_p^*(k_0|k_0), \ldots, \bar{u}_p^*(k_0+N-1|k_0)\}, \forall p \in \mathcal{P}$, exists and is a feasible (but not necessarily optimal) solution to $\mathbb{P}^C(x_1(k_0), \ldots, x_{N_p}(k_0))$ at some time step $k_0$. Then, for all $x_p(k_0+1) \in A_p x_p(k_0) + B_p u_p(k_0) \oplus \mathcal{W}_p, \forall p \in \mathcal{P}$, where $u_p(k_0) = \bar{u}_p^*(k_0|k_0) + K_p(x_p(k_0) - \bar{x}_p^*(k_0|k_0))$, the collection of candidate sequences $U_p(k_0+1) = \begin{cases} U_p^{\text{feas}}(k_0 + 1), \text{ a sol'n of } \mathbb{P}^{PD}_p(x_p(k_0+1); Z_p^*(k_0 + 1)), \forall p \in \mathcal{P}\setminus k_{0+1}, \\ \tilde{U}_p(k_0 + 1), \text{ defined by } (2.19), \forall p \in \mathcal{P}\setminus k_{0+1} \end{cases}$

exists and is a feasible solution to $\mathbb{P}^C(x_1(k_0+1), \ldots, x_{N_p}(k_0+1))$ if the margins $M_{c,p}(j)$ for all $j \in \{0, \ldots, N-1\}$ are such that

$$\tilde{z}_{c,p}^*(k_0 + 1 + j|k_0) + \sum_{q \in \mathcal{P}\setminus\{p\}} \tilde{z}_{c,q}^*(k_0 + 1 + j|k_0) \in \tilde{z}_{c} \sim M_{c,p}(j), \forall c \in C_p, p \in \mathcal{P}_{k_{0+1}}^{opt},$$

(5.4a)

and,

$$\left[-(P_{(k_0+1),c} - 1) \sum_{i \in \mathcal{P}_c} \tilde{z}_{c,i}^*(k_0 + 1 + j|k_0) \right] \oplus \left( \bigoplus_{p \in \mathcal{P}_{(k_0+1),c}^{opt}} [\tilde{z}_{c} \sim M_{c,p}(j)] \right) \subseteq \tilde{z}_{c}, \forall c \in C,$$

(5.4b)
where $P_{(k_0+1),c}$ is the number of subsystem agents in $P^\text{opt}_{k_0+1}$ that share constraint $c$.

**Proof.** A feasible local solution $U_p^{\text{feas}}(k_0 + 1)$ to $P_p^{\text{PD}}(x_p(k_0 + 1); Z_p^*(k_0 + 1))$ satisfies the further-tightened coupled constraints (5.2):

$$
\bar{z}_{cp}^{\text{feas}}(k_0 + 1 + j|k_0 + 1) + \sum_{q \in P \setminus \{p\}} \bar{z}_{cq}^*(k_0 + 1 + j|k_0) \in \hat{\mathcal{Z}}_c \sim M_{cp}(j), \forall c \in C_p, \tag{5.5}
$$

for all $j \in \{0, \ldots, N - 1\}$, where the terms $\bar{z}_{cq}^* (\cdot|k_0)$ are the fixed, previously-published values of coupled subsystems $q \in Q_p$. For the solution (5.3) to be a feasible solution to $P^C(x_1(k_0 + 1), \ldots, x_{N_p}(k_0 + 1))$, the following must also hold

$$
\sum_{p \in P^\text{opt}_{k_0+1}} \bar{z}_{cp}^{\text{feas}}(k_0 + 1 + j|k_0 + 1) + \sum_{q \in P \setminus P^\text{opt}_{k_0+1}} \bar{z}_{cq}^*(k_0 + 1 + j|k_0) \in \hat{\mathcal{Z}}_c, \forall c \in C, \tag{5.6}
$$

for all $j \in \{0, \ldots, N - 1\}$, so that the sum of optimized outputs $\bar{z}_{cp}^{\text{feas}} (\cdot|k_0 + 1)$ over all $p$ in the set $P^\text{opt}_{k_0+1}$ must be compatible with the sum of fixed values $\bar{z}_{cq}^* (\cdot|k_0)$ over all $q$ not in the optimizing set.

Consider a single constraint $c \in C$. By summing (5.5), via Minkowski addition, over all $p \in P^\text{opt}_{k_0+1,c}$, it follows that

$$
\sum_{p \in P^\text{opt}_{(k_0+1),c}} \bar{z}_{cp}^{\text{feas}}(k_0 + 1 + j|k_0 + 1) + \left[ (P_{(k_0+1),c} - 1) \sum_{p \in P^\text{opt}_{(k_0+1),c}} \bar{z}_{cp}^*(k_0 + 1 + j|k_0) \right]
$$

$$
+ \left[ P_{(k_0+1),c} \sum_{q \in P_c \setminus P^\text{opt}_{(k_0+1),c}} \bar{z}_{cq}^*(k_0 + 1 + j|k_0) \right]
$$

$$
\in \bigoplus_{p \in P^\text{opt}_{(k_0+1),c}} \left[ \hat{\mathcal{Z}}_c \sim M_{cp}(j) \right].
$$
or, re-writing, using the fact that \( a + b \in C \) implies \( a \in -b \oplus C \),

\[
\sum_{p \in \mathcal{P}_{(k_0+1),c}^{\text{opt}}} \tilde{z}_{cp}^{\text{feas}}(k_0 + 1 + j|k_0 + 1) + \sum_{q \in \mathcal{P}_{c} \setminus \mathcal{P}_{(k_0+1),c}^{\text{opt}}} \tilde{z}_{cq}^{*}(k_0 + 1 + j|k_0)
\]

\[
\in \left[ -(P_{(k_0+1),c} - 1) \sum_{i \in \mathcal{P}_c} \tilde{z}_{ci}^{*}(k_0 + 1 + j|k_0) \right] \oplus \left( \bigoplus_{p \in \mathcal{P}_{(k_0+1),c}^{\text{opt}}} [\tilde{z}_c \sim \mathcal{M}_{cp}(j)] \right).
\]

Directly comparing with (5.6), the left-hand sides of both are the same, noting that \( \tilde{z}_{cr} = 0 \) for all \( r \notin \mathcal{P}_c \). It follows that the constraint \( c \) is always satisfied if

\[
\left[ -(P_{(k_0+1),c} - 1) \sum_{i \in \mathcal{P}_c} \tilde{z}_{ci}^{*}(k_0 + 1 + j|k_0) \right] \oplus \left( \bigoplus_{p \in \mathcal{P}_{(k_0+1),c}^{\text{opt}}} [\tilde{z}_c \sim \mathcal{M}_{cp}(j)] \right) \subseteq \tilde{z}_c.
\]

for all \( j \in \{0, \ldots, N - 1\} \). Thus, the solution (5.3), if it exists, is feasible for \( \mathbb{P}^{C}(x_1(k_0 + 1), \ldots, x_{N_{p}}(k_0 + 1)) \) if this condition is met. (Feasibility for the terminal step \( j = N \) is assured by construction). For existence, by Proposition 2.1, the candidate sequence \( \bar{U}_p(k_0 + 1), \forall p \), exists and is a feasible solution to \( \mathbb{P}^{C}(x_1(k_0 + 1), \ldots, x_{N_{p}}(k_0 + 1)) \). It also exists and is a feasible solution to each problem \( \mathbb{P}^{PD}_p(x_p(k_0 + 1); Z_p^*(k_0 + 1)) \), if the coupling constraints are satisfied despite the extra margins:

\[
\tilde{z}_{cp}^{*}(k_0 + 1 + j|k_0) + \sum_{q \in \mathcal{P}_c \setminus \{p\}} \tilde{z}_{cq}^{*}(k_0 + 1 + j|k_0) \in \tilde{z}_c \sim \mathcal{M}_{cp}(j), \forall c \in \mathcal{C}_p, p \in \mathcal{P}_{(k_0+1),c}^{\text{opt}}
\]

for all \( j \in \{0, \ldots, N - 1\} \). Therefore, the solution defined by (5.3) exists and is a feasible solution to \( \mathbb{P}^{C}(x_1(k_0 + 1), \ldots, x_{N_{p}}(k_0 + 1)) \) if the conditions (5.4) are satisfied.

Proposition 5.1 provides bounds on the 'size' of the margin sufficient to guarantee feasibility. Generally, a small margin is desirable to avoid excessive conservatism. Firstly, (5.4a) states that the margin must be sufficiently small so that the candi-
date solution remains available to any agent solving the modified optimization; an upper bound. Secondly, the sum of margins must be sufficiently large so that the parallel updates do not lead to constraint violation; a lower bound. The significance of the latter is that the minimum ‘size’ of the extra tightening margin of a coupled constraint $c$ depends on the margin left by the candidate solutions – that is, the previously-determined, feasible plans – of all subsystems involved in that constraint; effectively, the ‘slack’ remaining in the constraint is to be shared amongst the simultaneously-updating agents. How that slack is shared is open to question.

Note that if the optimizing set at a time $k$ contains only non-coupled agents, then $P_{k,c} = 1$ for all $c \in C$. The conditions (5.4) collapse to

$$
\tilde{z}_{cp}^*(k_0 + 1 + j|k_0) + \sum_{q \in \mathcal{P}_c \setminus \{p\}} \tilde{z}_{cq}^*(k_0 + 1 + j|k_0) \in \mathcal{Z}_c - \mathcal{M}_{cp}(j) \subseteq \mathcal{Z}_c,
$$

for all $j \in \{0, \ldots, N - 1\}, c \in C_p, p \in \mathcal{P}_k^{opt}$. Setting each margin $\mathcal{M}_{cp}(j)$ to be zero recovers the original form of coupling constraint in the DMPC optimization.

Now, for simplicity, we restrict the optimization objective function to the non-cooperative formulation of Chapter 2, i.e., $V_p(U_p) = J_p(U_p)$, so that

$$
V_p^{opt}(x_p(k); Z_p^*(k)) = \min_{U_p(k)} J_p(U_p(k)) \text{ subject to (2.30a)-(2.30f) and (5.2)},
$$

where

$$
J_p(U_p(k)) = F_p(\bar{x}_p(k + N|k)) + \sum_{j=0}^{N-1} l_p(\bar{x}_p(k + j|k), \bar{u}_p(k + j|k)).
$$

The next result shows that, under the assumption that the terminal cost $F_p$ is a local Lyapunov function in $\mathcal{X}_{F_p}$ (Assumption 2.4), the monotonicity of each local cost is maintained for updates in parallel.

**Proposition 5.2 (Monotonicity of the cost).** Suppose the sequence of controls $U_p^*(k_0) = \{x_p^*(k_0|k_0), u_p^*(k_0|k_0), \ldots, u_p^*(k_0 + N - 1|k_0)\}, \forall p \in \mathcal{P}$, exists and is a
feasible (but not necessarily optimal) solution to $\mathbb{P}^C(x_1(k_0), \ldots, x_{N_p}(k_0))$ at some
time step $k_0$. Then, for all $x_p(k_0 + 1) \in A_p x_p(k_0) + B_p u_p(k_0) \oplus W_p, \forall p \in \mathcal{P}$, where $u_p(k_0) = \bar{u}_p^*(k_0|k_0) + K_p (x_p(k_0) - \bar{x}_p^*(k_0|k_0))$, the upper bound on the local cost in
the problem $\mathbb{P}^D_p(x_p(k_0 + 1); Z_p^*(k_0 + 1))$ decreases monotonically:

$$V_p^*(x_p(k_0 + 1); Z_p^*(k_0 + 1)) \leq V_p^*(x_p(k_0); Z_p^*(k_0)) - l_p(\bar{x}_p^*(k_0|k_0), \bar{u}_p^*(k_0|k_0)),$$

for all $p \in \mathcal{P}$, where $V_p^*(x_p(k_0); Z_p^*(k_0)) = J_p(U_p^*(k_0)).$

Proof. Similar to Proposition 2.5. By Proposition 5.1, the candidate sequence $\bar{U}_p(k_0 + 1)$ exists and is a feasible solution to $\mathbb{P}^D_p(x_p(k_0 + 1); Z_p^*(k_0 + 1)), \forall p \in \mathcal{P}$; the collection of all such sequences is also a feasible solution to $\mathbb{P}^C(x_1(k_0 + 1), \ldots, x_{N_p}(k_0 + 1))$. Thus, for all $p$, the value of this local solution forms an upper
bound on the value of feasible solutions, and the result is established.  

Suffice it to say that asymptotic or exponential stability of parallel-update DMPC
follows, depending on further assumptions on the stage and terminal costs. Thus,
should the sufficient margins for feasibility be met, the parallel-update DMPC re-
tains all the robust feasibility and stability properties of the standard, single-update
DMPC of Chapter 2.

The next subsection shows that these sufficient conditions for robust feasibility of
parallel updates take on a simple, easily-verifiable form when the coupled constraints
are polyhedral. A control authority rule is proposed so that agents may, simulta-
neously, determine sufficient margins, and the resulting parallel-update algorithm is
presented.
5.2.2 Feasible parallel-update DMPC for polyhedral constraints

Suppose the tightened set $\tilde{Z}_c$ for a constraint $c$ is polyhedral, i.e., $\tilde{Z}_c = \{z : P_c z \leq \tilde{q}_c\}$. At a step $k$, a feasible solution satisfies,

$$P_c \left( \sum_{i \in \mathcal{P}_c} \tilde{z}_{ci}^*(k + j|k) \right) = \tilde{q}_c - P_c \delta \tilde{z}_c^*(k + j|k), \forall c \in \mathcal{C},$$

(5.7)

where $\delta \tilde{z}_c^*(k + j|k)$ is a slack variable for the constraint $c$. Likewise, constraint (5.1) becomes

$$P_c \left( \sum_{p \in \mathcal{P}_c^{opt}} \tilde{z}_{cp}(k + j|k) + \sum_{q \in \mathcal{P}_c \setminus \mathcal{P}_c^{opt}} \tilde{z}_{cq}^*(k + j|k) \right) \leq \tilde{q}_c.$$  

(5.8)

Now consider (5.2), the further-tightened form of the coupled constraints in the local optimization. Kerrigan [81, Corollary 3.3], states that if the polyhedra $\mathcal{A}$ and $\mathcal{B}$ are given by, respectively, $m$ and $n$ inequalities, then the Pontryagin difference $\mathcal{A} \sim \mathcal{B}$ is given by, at most, $m$ inequalities. This permits us to write the Pontryagin subtraction of margins $\mathcal{M}_c(j)$ from sets $\tilde{Z}_c$ as a simple subtraction of a vector of margins, $q_{cp}^{\text{mar}}(j)$, from the vector $\tilde{q}_c$, so that (5.2) becomes

$$P_c \left( \tilde{z}_{cp}(k + j|k) + \sum_{q \in \mathcal{P}_c \setminus \{p\}} \tilde{z}_{cq}^*(k + j|k) \right) \leq \tilde{q}_c - q_{cp}^{\text{mar}}(j), \forall p \in \mathcal{P}_c^{opt},$$

(5.9)

for all $j \in \{0, \ldots, N-1\}, c \in \mathcal{C}, p \in \mathcal{P}_c^{opt}$. The sufficient conditions (5.4) for feasibility are derived as follows. The upper bound (5.4a) follows immediately from (5.7) and (5.9) as

$$q_{cp}^{\text{mar}}(j) \leq P_c \delta \tilde{z}_c^*(k + j|k),$$

(5.10)

for all $j \in \{0, \ldots, N-1\}, c \in \mathcal{C}, p \in \mathcal{P}_c^{opt}$, while the lower bound (5.4b) is, using (5.7),

$$\sum_{p \in \mathcal{P}_c^{opt}} q_{cp}^{\text{mar}}(j) \geq (P_{k,c} - 1) P_c \delta \tilde{z}_c^*(k + j|k),$$

(5.11)

for all $j \in \{0, \ldots, N-1\}, c \in \mathcal{C}$. 

166
The question remains of how to determine suitable margins that satisfy these conditions; possible approaches include a priori rules or pre-allocated shares of the slack available, or convex optimization, given the current value of the slack variables $\delta z_c^*$. For example,

$$
\min_{Q^{\text{mar}}} \sum_{p=1}^{N_p} \sum_{c=1}^{N_c} \sum_{j=0}^{N-1} \|q_{cp}^{\text{mar}}(j)\| \text{ subject to (5.10) and (5.11)},
$$

where $Q^{\text{mar}}$ is the collection of all margins. This itself, of course, is an optimization problem with coupled constraints, and subject to the same issues as the original DMPC problem. Solutions might be sought by bargaining, or by some central, arbitrating agent, but such approaches renege on the principles on which the distributed algorithm was developed. The optimization admits a very simple solution if the agents ‘agree’ to share amongst themselves the total margin required. For example, an agent $p$ takes a proportion $\beta_p \in [0,1]$ of the total, with $\sum_{p \in P_c} \beta_p = 1$, the agent $p$’s margin for constraint $c$ at prediction step $j$ becomes

$$
q_{cp}^{\text{mar}}(j) = \beta_p (P_{k,c} - 1) P_c \delta z_c^*(k + j|k).
$$

The parameter $\beta_p$ is then an indication of the authority an agent has to update; for example, $\beta_p \to 0$ for some agent $p$ leads to the smallest, or zero, extra margin for that subsystem’s constraints, while other simultaneously-updating agents take on large margins, further constraining their problems.

Figure 5.2 illustrates this principle for a simple two-agent scenario where each seeks to minimize some function of a variable $z_p \geq 0, p \in \{1, 2\}$, subject to the coupled constraint $z_1 + z_2 \leq Z$. Including the tube MPC tightening for robustness, the feasible set is $\tilde{Z}$. For a given feasible solution $\{z_1^*, z_2^*\}$, also shown are three sets: the product set $\tilde{Z}_1 \times \tilde{Z}_2$ of the individual sets $\tilde{Z}_1(z_2^*)$ and $\tilde{Z}_2(z_1^*)$, including the infeasible region (shaded); and, for two different choices of margins (denoted $a$ and $b$ respectively), the product sets of the further-tightened individual sets. For choice
Figure 5.2: Coupled constraints further tightened by margins for parallel updates, with the feasible sets of \( \{\tilde{z}_1, \tilde{z}_2\} \) for the constraint \( \tilde{z}_1 + \tilde{z}_2 \leq Z \).

\( a \), the greater tightening (and smaller authority) is by agent 1; for choice \( b \), agent 2 adopts the larger margin. As these latter product sets are wholly contained within the set \( \tilde{Z} \), any simultaneous update—subject to the further-tightened constraints—shall be feasible, as was discussed at the outset of this section.

This predetermined authority approach is adopted in Algorithm 5.1 for robustly-feasible, parallel-update DMPC. The next example shows that such an approach, perhaps surprisingly, offers good closed-loop performance with a fast response.

### 5.3 Numerical example

The following example applies the parallel-update algorithm to a system of point masses. It is shown that, without the extra constraint tightening, constraint viola-
Algorithm 5.1: Robust feasible-parallel DMPC for a subsystem $p$

1. Design stabilizing controller $K_p$ and RPI set $R_p$;
2. Tighten sets $\mathcal{Y}_p, Z_c, \forall c \in C_p$, via (2.16);
3. Wait for feasible solution $U_p^*(0)$, information $Z^*_p(0)$, terminal set $X^*_F$, authority $\beta_p$ from central agent;
4. for $k = 1: \infty$ do
   5. Sample current state $x_p(k)$;
   6. if $p \in P^\text{opt}_k$ then
      7. Determine slackness $\delta Z^*_c(k + j|k), \forall c \in C_p, j \in \{0, \ldots, N - 1\}$ using candidate plans for $k$;
      8. Obtain margins $q^\text{mar}_c(j), \forall c \in C_p, j \in \{0, \ldots, N - 1\}$;
      9. Obtain new plan $U_p(k) = U^\text{opt}_p(k)$ as solution to $P^\text{PD}(x_p(k); Z^*_p(k))$;
     10. Transmit new plan to agents in $Q_p$;
   11. else
      12. Renew current plan via (2.19): $U_p(k) = \tilde{U}_p(k)$;
   13. end
   14. Apply control (2.17): $u_p(k) = \bar{u}_p(k|k) + K_p(x_p(k) - \bar{x}_p(k|k))$;
   15. Wait one time step;
5. end

Example 5.1 (Parallel-update DMPC of 1-D point masses). Consider four of the constrained point masses with the double-integrator dynamics of Example 2.1. Each is to remain within a maximum separation distance of $\Delta x = 2$ from the others, and each mass is subject to local constraints on velocity ($|x_p| \leq 5$) and control ($|u_p| \leq 1$). The local objectives are conflicting in that half of the masses favour minimizing state deviations from the origin, and the other half control:

$$l_p(x_p, u_p) = Q_p \|x_p\|_1 + R_p \|u_p\|_1,$$
where,

\[
Q_p = \begin{cases} 
0.001 & p \in \{1, 2\}, \\
1 & p \in \{3, 4\}, 
\end{cases} \quad R_p = \begin{cases} 
1 & p \in \{1, 2\}, \\
0.001 & p \in \{3, 4\}. 
\end{cases}
\]

Together with a zero terminal cost and the origin as the terminal set in the optimizations, this cost guarantees robust asymptotic stability of the set \( \mathcal{R}_p \). The disturbance set is again the hypercube \( \{ w_p \in \mathbb{R}^2 : \|w_p\|_\infty \leq W_{\text{max}} \} \), and the disturbances applied to the masses correspond to the different vertices of the set, designed to force apart the masses. For all \( k \in \mathbb{N} \):

\[
\begin{align*}
  w_1(k) &= W_{\text{max}} \begin{bmatrix} 1 & 1 \end{bmatrix}^T \\
  w_2(k) &= W_{\text{max}} \begin{bmatrix} -1 & -1 \end{bmatrix}^T \\
  w_3(k) &= W_{\text{max}} \begin{bmatrix} -1 & 1 \end{bmatrix}^T \\
  w_4(k) &= W_{\text{max}} \begin{bmatrix} 1 & -1 \end{bmatrix}^T
\end{align*}
\]

All other parameters are the same as in Example 2.1, bar the initial state and prediction horizon, which are \( x_p = [6 \ 0]^T \), \( \forall p \), and \( N = 10 \) respectively. Five different controllers are used:

1. ‘CMPC’: centralized MPC (Algorithm 2.1);

2. ‘SU-DMPC’: distributed MPC with a sole agent updating per time step (Algorithm 2.2), according to the simple alternating sequence \( \{1, 2, 3, 4, 1, 2, \ldots\} \);

3. ‘S-DMPC’: A modified distributed MPC (Algorithm 2.2) with agents updating in the sequence \( \{1, 2, 3, 4\} \) at each time step, as per the arrangement of [38];

4. ‘P-DMPC’: distributed MPC with all agents updating in parallel at each time step, but without the extra constraint margins described previously;

5. ‘FP-DMPC’: distributed MPC with all agents updating in parallel, with the extra constraint margins (Algorithm 5.1).
The coupling constraints were deliberately chosen to be pairwise, so that feasibility guarantees are lost if any number of agents update simultaneously without extra constraint tightening.

Firstly, Figure 5.3 shows the position histories of the four masses, together with the maximum separation constraints, when controlled by (a) P-DMPC and (b) FP-DMPC. For the former, a sustained constraint violation occurs between time steps 5 and 9, during which the local optimizations fail and the agents adopt the (infeasible) candidate solution. Though feasibility is eventually recovered, had the masses been open-loop unstable, for example, the system might well have gone unstable. This violation is successfully avoided by the FP-DMPC controllers, yet—despite the extra tightening—the full width of the constraints is used. Figure 5.4 offers an alternative depiction of this situation, via the maximum separation distance between any pair over time, and now including the remaining three controllers. All controllers bar P-DMPC satisfy the constraints; however, interestingly, the CMPC controller does not drive the system to the limit of the permitted separation. SU-DMPC, as predicted, appears to exhibit the slowest response of all the controllers, with a visible lag in the time taken to reach the constraint limits.
Figure 5.4: Maximum separations of four point masses controlled by feasible-parallel DMPC and other algorithms.

Table 5.1 shows the resulting closed-loop cost values for this 15-step simulation. Values are normalized as percentage increases over those of the benchmark centralized controller, which obtains lowest costs. Also shown are the number of data exchanges per step, and measures of computation times; the horizon was increased to 200 steps, to avoid an overhead-dominated computation time, and the time $t_c(k)$ is measured as the time elapsed from the beginning of the step $k$ until all optimizations taking place in that time interval have completed. That is,

$$
t_c(k) = \begin{cases} 
\sum_{p \in \mathcal{P}_{\text{opt}}} t^c_p(k) & \text{if sequential updates,} \\
\max_{p \in \mathcal{P}_{\text{opt}}} t^c_p(k) & \text{if parallel updates,}
\end{cases}
$$

where $t^c_p(k)$ is the time for agent $p$ to solve its optimization at step $k$. Delays are ignored, and it is assumed that any agent adopting the candidate plan takes zero time in doing so. This is a fair measure because, regardless of which algorithm is...
Table 5.1: Comparison of performance of parallel-update DMPC methods.

<table>
<thead>
<tr>
<th>Controller</th>
<th>CMPC</th>
<th>SU-DMPC</th>
<th>S-DMPC</th>
<th>P-DMPC</th>
<th>FP-DMPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{15} \sum_{k=1}^{15} N_{\text{data}}(k)$</td>
<td>6</td>
<td>3</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Cost Increase %</td>
<td>0</td>
<td>29.9</td>
<td>14.0</td>
<td>16.4</td>
<td>18.3</td>
</tr>
<tr>
<td>Mean $\bar{t}$ over 10 reps</td>
<td>0.7571</td>
<td>0.2867</td>
<td>1.1626</td>
<td>0.3460</td>
<td>0.3540</td>
</tr>
<tr>
<td>Mean $\sigma_t$ over 10 reps</td>
<td>0.1270</td>
<td>0.0262</td>
<td>0.0667</td>
<td>0.0279</td>
<td>0.0368</td>
</tr>
</tbody>
</table>

employed, it provides information on the duration of the decision making at each time step, from which a lower bound on the period, $\delta t$, of MPC updates might be obtained. Note that for the single-update DMPC, both measures are equal.

Subsequently, the mean and standard deviations of these computation times, $\bar{t}$ and $\sigma_t$ respectively, for each 15-step simulation were measured, and the averages of these values, over ten repetitions, are provided in the table. All simulations were performed on a Pentium 4 HT 3.2 GHz with 2,048 MB RAM, using CPLEX 10.1 as the LP solver.

The ranking of controllers by performance is as expected. SU-DMPC does worst; some lag is evident in the response, which is expected, given that the effective update frequency is quartered, yet communication and computation values are lowest. S-DMPC is closest to CMPC; at each time step, the $p^{th}$ in line to update has new information from the $p - 1$ preceding agents, thus, the rate at which the system is updated is highest, at the expense of the longest computation time and highest communication. Note that although computation times for local optimizations in this arrangement are known to scale favourably when compared to CMPC, [38], the drawback is that the whole sequence of updates, with communication following each update, must be completed within the length of the time step. Furthermore, all inter-agent data exchanges are assumed to be instantaneous, unlike in SU-DMPC where the entire remainder of the time step following optimization is available for communication.
For the parallel DMPC control scheme where extra tightening is not done (P-DMPC), although the closed-loop cost is relatively low, this is aided by a recovery of feasibility, which is by no means guaranteed. Finally, for FP-DMPC, the extra conservatism is evident by the increase in closed-loop cost over that of P-DMPC; however, feasibility is maintained throughout. Communication and computation are similar for the two parallel methods. The conclusion is that FP-DMPC maintains robust feasibility where it would otherwise be lost, and performance surpasses that of SU-DMPC, nearing that of S-DMPC, with extra communication but a computation time similar to that of the single-update method.

In conclusion, the key outcome of this example is that, while parallel updates are desirable for closed-loop performance, the primary advantage of the single-update DMPC algorithm is lost: namely, flexibility in communications. Recalling the motivation for using distributed control, some limitation on either communication or computation is usually present or inherent in the application, otherwise centralized MPC provides best performance. However, for the cost of this loss of communication flexibility, the proposed method has some attractive qualities, as demonstrated by the example. This is the first work to combine parallel updates with robust feasibility for subsystems coupled via the constraints.

5.4 Summary

The single-update DMPC algorithm has been extended to permit updates in parallel by control agents, while maintaining robust constraint satisfaction and feasibility. It was shown by simulation that performance of the parallel-update algorithm surpasses that of single-update DMPC, albeit when communication is not a limiting factor, and approaches that of a sequential implementation, yet requiring a fraction of the computation time needed for that approach.
Chapter 6

Multi-Vehicle Applications of Distributed MPC

The distributed algorithm is applied to two applications: (i) the problem of search, or coverage, of an area of unknown content by a team of vehicles, and (ii) the problem of controlling multiple dynamic sensors, and fusing gathered information, for the tracking and observation of uncertain, dynamic processes.

6.1 Introduction

In this chapter, two applications for the newly-developed DMPC algorithms are investigated. Both involve the use of a team of multiple vehicles, given some overall task or mission to complete. Though the DMPC algorithm is, of course, not restricted to path-planning for vehicles, such a case is of interest and exemplifies the problem statement considered in this thesis; vehicles have decoupled dynamics, local constraints, such as a flight envelope, and, as shown in previous chapters, the problem of collision avoidance may be cast as one of coupled constraint satisfaction. Furthermore, whereas predictive control has historically been the preserve of the
process control industry [6, 7], where dynamics are slow, continual advances in processor technology and optimization solvers mean that DMPC is becoming a viable option for real-time, on-line path-planning and control.

In Section 6.2, the DMPC algorithm is applied to the problem of multi-vehicle cooperative search, where a team of vehicles navigate through and search an unknown environment, while avoiding collision and duplication of efforts. Collision avoidance is the coupling constraint that must be robustly satisfied, while duplication of efforts is discouraged by the objective function. Key features that arise from applying the method to this problem are (i) path planning decisions are made autonomously and on-line, based on the dynamics model, by vehicles, (ii) collision avoidance is guaranteed at all times, and (iii) the control algorithm permits flexible communications between vehicles. In addition, it is shown that, by using the cooperative form of the DMPC, cooperation between efforts is encouraged and less duplication of efforts occurs, leading to better system-wide performance.

The problem of providing control for mobile, range-only sensing platforms to observe uncertain, dynamic targets is considered in Section 6.3. Using techniques from optimal estimation theory [107], a local subsystem objective is formulated that seeks to design local controls that minimize estimation errors over the prediction horizon, using a forward projection of the Kalman filter; at each step local agents exchange information plans, subsequently producing target state estimates based on locally-fused information. Combining with DMPC, a robustly-feasible, receding-horizon form of distributed data fusion ensues. When cooperative DMPC is employed, local agents additionally design hypothetical information plans for others. Simulation results show that this form of the DMPC again provides a benefit to system-wide performance.

Path-planning for vehicles in unknown environments is inherently uncertain with respect to convergence; the goal is usually to reach a target or complete a mission, but success is not guaranteed. The distributed MPC method originally proposed in Chapter 2 depended on three assumptions being met: Assumptions 2.2 and 2.3
are sufficient to guarantee robust feasibility and constraint satisfaction, while—in addition—stability or convergence may be guaranteed by the satisfaction of Assumption 2.4, i.e., the terminal cost being a Lyapunov function in the terminal set. For each of the two applications studied in this chapter, the development of the control problem leads to objective functions that do not satisfy the stronger Assumption 2.4. However, Assumptions 2.2 and 2.3 are met, so that robust feasibility and constraint satisfaction are guaranteed—in this case, collision avoidance and the availability of a plan to a vehicle at all times.
6.2 Multi-vehicle cooperative search

6.2.1 Introduction

Classical search theory is well established for static environments with a single agent. The problem of multi-vehicle search requires path-planning decisions to be made for each vehicle involved; ideally, the vehicles should search the unknown area thoroughly and to suit some performance objective, for example minimizing fuel use or time, yet coordinating to avoid duplication of efforts and collision with each other. One way of ensuring complete coverage of the area is to employ a pre-specified exhaustive search method, such as Zamboni search [108]; however, such a method makes no allowance for uncertainties in the environment, such as new threats or obstacles. Similarly, most results in the problem of coverage for robotics [109], for applications such as floor cleaning, harvesting, or mine hunting, do not consider search environments where path planning might need to be adaptive; moreover, no consideration is given to the dynamic or kinematic constraints of the vehicle. In an unknown environment, as vehicles progress, knowledge of the area increases and plans may be required to change, either in response to new information about the environment or changes in the intentions of others.

Research in this area includes a team of vehicles moving through the search area with a uniform longitudinal front, and only lateral relative motion between vehicles, with limited look-ahead [110], and swarms of vehicles arranging themselves into a formation with maximal sensing capability yet minimal inter-vehicle communication [111]. Some authors propose decomposition of the state space into searchable cells: in Sujit and Ghosh [112] and Yang et al. [113] the space is discretized so that vehicles have a finite number of heading choices, whereas in Baum and Passino [114] and Curtis and Murphey [115] waypoints are generated for the vehicle to follow using a low-level controller.

In this work, the vehicle dynamics model and kinematic constraints are included in the search problem, and the robustly-feasible distributed MPC is used as the con-
trol method. The vehicles plan finite-length paths through the search area, which is known in extent and decomposed into a cellular grid; thus, this method is comparable to a limited look-ahead approach, but with vehicle dynamics considered. Sensors on the vehicles search cells as the paths are followed, with the aim of collecting rewards associated with cells while avoiding collision and duplication of effort. The contribution of this section, therefore, mainly arises from the simplicity of the approach: the distributed receding-horizon algorithm leads to favourable computational complexity, when compared with solving a single centralized coverage problem, yet the dynamics model is included and collision avoidance guaranteed. Furthermore, by employing the cooperative form of the DMPC, performance may approach that of centralized.

The following section outlines the problem statement. Section 6.2.3 develops the main algorithm, by first designing a cost function that aims to maximize the cells searched by vehicles, and then combining with the DMPC algorithm. Section 6.2.4 presents simulation results.

6.2.2 Problem statement

The problem is to control a set of \(N_p\) vehicles with the aim of complete search of an area of known extent but unknown content. The vehicles are assumed to operate in the same plane, each vehicle being governed by the linear discrete-time state equation

\[
x_p(k + 1) = A_p x_p(k) + B_p u_p(k) + w_p(k), \forall k, p \in \mathcal{P},
\]

(6.1)

where \(x_p \triangleq [r_p^T \ v_p^T]^T\) is the state, and \(u_p \in \mathbb{R}^2\) is the control input applied. Each vehicle has an associated performance envelope, given by maximum velocity and force constraints:

\[
\|v_p(k)\|_2 \leq V_{p,\text{max}},
\]

\[
\|u_p(k)\|_2 \leq U_{p,\text{max}}.
\]

(6.2)
Furthermore, the vehicles must at all times avoid collision, i.e., at each time step \( k \) all pairs must maintain a minimum separation distance between their positions,

\[
\| r_p(k) - r_q(k) \|_2 \geq L, \forall k, p, q : p \neq q,
\]

which permits no other vehicle to enter a circular region, radius \( L \), around the position \( r_p(k) \) of vehicle \( p \) at time \( k \). As before, these 2-norm constraints may be written in a linear form by approximating as polyhedra [94]. Implementation of the collision avoidance constraints as coupling constraints implies that a constraint exists for each pair of vehicles, i.e., \( N_c = \binom{N_p}{2} \), and for each constraint \( c \) two matrices, \( E_{cp} \) and \( E_{cq} \), are non-zero.

The area to be searched is known in extent, and is decomposed into \( N_b \) non-overlapping cells, each with \( n_v \) vertices; the whole area has a total of \( N_v \) vertices, with \( n_v \leq N_v \). Each cell \( i \in \{1, \ldots, N_b\} \) is described by the coordinates of its \( n_v \) vertices, \( \{(r_{x,i}^1, r_{y,i}^1), \ldots, (r_{x,i}^{n_v}, r_{y,i}^{n_v})\} \). The team objective is to search the maximal number of cells over the duration of the mission. With each cell is associated a reward \( \phi_i \), which may differ from cell to cell, and may be collected only once; hence, the problem becomes to maximize the total reward obtained over the duration of the mission.

To search cells, each vehicle \( p \) is equipped with a sensor of range \( R_p \), assumed to be omni-directional. A cell \( i \) is deemed searched when it falls completely within a vehicle’s sensing range: that is, all vertices of cell \( i \) are within a distance \( R_p \) of the position of vehicle \( p \):

\[
\| r_i^j - r_p(k) \|_2 \leq R_p, \forall j \in \{1, \ldots, n_v\}.
\]

Again, this 2-norm expression is approximated by the polyhedral form

\[
P(r_i^j - r_p(k)) \leq R_p 1, \forall j \in \{1, \ldots, n_v\}.
\]
6.2.3 Multi-vehicle cooperative search using distributed MPC

This section extends the DMPC algorithm to the problem of multi-vehicle search by developing a local subsystem objective $J_p$ for the non-cooperative or cooperative optimizations that maximizes the reward obtained by vehicles searching cells. The MILP approach used, motivated by that of Richards and How [98], associates a binary variable with each cell and a state machine with each vehicle that flags which cells have been searched.

**Cell search objective function**

Suppose with each cell, $i \in \{1, \ldots, N_b\}$, is associated a binary variable $b_i(k) \in \{0, 1\}$. This variable acts as an indicator as to whether a particular cell has been searched; $b_i(k) = 1$ implies the cell $i$ has been searched by time step $k$, and $b_i(k) = 0$ implies the cell has not yet been searched. Assuming the reward associated with cell $i$ is $\phi_i$, the team objective may be stated as

$$\max_{i=1}^{N_b} \sum \phi_i b_i(T),$$

where $T$ is the mission time. However, in combining the problem of search with the DMPC algorithm, the vehicle dynamics are introduced; the planned trajectories of vehicles change over time, and so a record is needed of the history of cells searched so to avoid duplicated effort. Furthermore, the effect of other vehicles in the system must be taken into account: cells previously searched, and also cells planned to be searched, by those vehicles. Assume the system-wide objective function is separable, i.e., $J(U_1, \ldots, U_{N_p}) = \sum_{p=1}^{N_p} J_p(U_p)$. As usual, $U_p(k)$ is the collection of (nominal) initial state and the sequence of predicted controls:

$$U_p(k) = \{ \hat{x}_p(k|k), \hat{u}_p(k|k), \hat{u}_p(k + 1|k), \ldots, \hat{u}_p(k + N - 1|k) \}.$$
Then the following optimization problem forms the local subsystem objective $J_p(U_p(k))$ associated with vehicle $p \in P$.

$$J_p(U_p(k)) = \min_{b_p(k)} \tilde{J}_p(U_p(k), b_p(k))$$

$$= \min_{b_p(k)} F_p(\tilde{x}_p(k+N|k)) - \sum_{i=1}^{N_b} \phi_i y_{ip}(k+N|k)$$

subject to $\forall i \in \{0, \ldots, N_b\}, j \in \{0, \ldots, N - 1\}$:

$$y_{ip}(k+j+1|k) = y_{ip}(k+j|k) + w_i(j), \quad (6.6a)$$

$$y_{ip}(k|k) = y_{ip}^*(k|k-1), \quad (6.6b)$$

$$b_{ip}(k+j|k) + B_{ip}^*(k) + \sum_{q \in P \setminus \{p\}} b_{iq}^*(k+j|k) \geq w_i(j), \quad (6.6c)$$

$$b_{ip}(k+j|k) = 1 \implies P(r_i^* - [I\ 0] \tilde{x}_p(k+j|k)) \leq R_p, \forall v \in \{1, \ldots, n_v\}, \quad (6.6d)$$

$$y_{ip}(k+j|k) \in [0, 1], \quad (6.6e)$$

$$w_i(j) \in [0, 1], \quad (6.6f)$$

$$b_{ip}(k+j|k) \in \{0, 1\}, \quad (6.6g)$$

where,

- the decision variable is the collection of binary variables, across all cells and all steps of the prediction horizon,

$$b_{p}(k) \triangleq \left\{ \begin{array}{c} b_{1p}(k|k), \ldots, b_{N_{bp}}(k|k), \\ b_{1p}(k+1|k), \ldots, b_{N_{bp}}(k+1|k), \\ \vdots \\ b_{1p}(k+N|k), \ldots, b_{N_{bp}}(k+N|k) \end{array} \right\},$$

- the terms $b_{iq}^*(\cdot|k), q \neq p$, in constraint (6.6c) represent the binary variables associated with the previously-published plans for all other vehicles. For ex-
ample, for cell $i$ and vehicle $q$, the plan is

$$\{b_{iq}^*(k|k), b_{iq}^*(k+1|k), \ldots, b_{iq}^*(k+N|k)\}.$$ 

These terms are based on optimizations at previous time steps, and appear as fixed values (denoted by $^*$):

- the terms $B_{ip}^*(k), \forall i$, represent the values of binary variables for each cell prior to the current time step $k$. If a cell $i$ has been searched at some earlier time step $\hat{k}$, $B_{ip}^*(k) = 1, \forall k > \hat{k}$. Thus, the copy of $B_{ip}^*(k)$ that each vehicle $p$ holds provides information on which cells have already been searched by the team.

- $w_i(j) \in [0,1]$ is a secondary cell variable in the optimization, constrained to take a value between 0 and 1.

- $y_{ip} \in [0,1]$ is an auxiliary state for vehicle $p$ regarding cell $i$; the difference equation (6.6a) increments $y_{ip}(k+j|k)$ based on the value of $w_i(j)$. The initial condition for $y_{ip}(k|k)$ requires $y_{ip}^*(k|k-1)$, from the previous plan, to be known.

- $F_p(\hat{x}_p(k+N|k))$ is a terminal cost, based on the cost-to-go to the nearest unsearched cell.

Each of the fixed terms in the constraints, $b_{iq}^*, \forall q \neq p, B_{ip}^*$, and $y_{ip}^*(k|k-1)$, are required for the evaluation of this problem, and must therefore be available; how this information is obtained is detailed later.

In this optimization, searching of a cell is achieved by satisfaction of the sensing constraint (6.6d), i.e., so that all $n_v$ vertices of cell $i$ are within the sensing range of the vehicle. Note that this constraint may be re-written in the 'big-M' form

$$P(r_i^v - [I \ 0] \hat{x}_p(k+j|k)) \leq R_p(1 - M(1 - b_{ip}(k+j|k))), \forall v \in \{1, \ldots, n_v\},$$

where $M$ is a sufficiently large constant [87]. However, recent versions of the AMPL language [116], and the CPLEX solver [89], are able to process constraints in the
form of (6.6d) directly [117]. A value of $b_{ik}(k+j|k) = 1$ implies satisfaction of (6.6d) for cell $i$ at time step $j$ of the prediction. As previously stated, a reward may be collected only once, and only one search, by any vehicle, counts towards the mission objective. Hence, constraint (6.6c) is required, which may set the secondary cell variable $w_i(j)$ to unity in four possible ways:

1. cell $i$ being searched $j$ steps into the future by the updating vehicle $p$
   (i.e., $b_{ip}(k+j|k) = 1$):

2. cell $i$ being searched $j$ steps into the future by some other vehicle $q$
   ($b_{iq}^*(k+j|k) = 1$):

3. cell $i$ having already been searched by a member of the team ($B_{ip}^*(k) = 1$);
   and,

4. any combination of the previous.

Finally, an additional state variable $y_{ip}$ is introduced for vehicle $p$ and cell $i$; the difference equation (6.6a) increments $y_{ip}(k+j|k)$ based on the value of $w_i(j)$. The terminal value $y_{ip}(k+N|k)$ shall have a unity value should a cell $i$ be visited by the end of the prediction horizon, by any vehicle (predicted or otherwise). It is this value that appears in the cost function (6.5), which counts the rewards collected by the end of the horizon. Note that these additional cell variables $w_i, y_{ip}$, are not required to be binary variables, unlike the binary variable $b_{ip}^*$, but are each constrained to lie in the interval $[0,1]$. This keeps the number of binary variables minimal.

The optimization terminal cost represents the cost-to-go for vehicle $p$ beyond the end of the prediction horizon. Based on the approach of Kuwata et al. [49], the function $F_p$ is a measure of the distance from the terminal state to visible centres of unsearched cells. By being visible, the unsearched cell is reachable by following the shortest, straight-line path; computation of the cost-to-go for cells obscured from line of sight by, for example, obstacles is possible by following the cost map approach of [49], but this is beyond the scope of this example. The optimization chooses the
best unsearched cell \( m \in \mathcal{M} \) (where \( \mathcal{M} \subseteq \{1, \ldots, N_b\} \) is the set of all unsearched cells) such that \( F_p \) is minimized:

\[
F_p(x_p(k + N|k)) = \min_{m \in \mathcal{M}} \|\tilde{r}_p(k + N|k) - r_m\|_2,
\]

where \( \tilde{r}_p(k + N|k) \) is the (nominal) position of vehicle \( p \) at the end of the prediction horizon and \( r_m \) is the position of the centre of unsearched cell \( m \). Note that the list of unsearched cells at a step \( k \) is determinable from the collection of \( B_i^*(k) \) terms. Also, the cell centres, \( r_i, \forall i \in \{1, \ldots, N_b\} \) are determined as the centroids of the vertices for each cell. Lastly, this 2-norm constraint is implementable as a linear approximation [94].

**Combining with DMPC**

Combining this cell search cost function with the DMPC optimization, the local optimization for a vehicle \( p \) with state \( x_p(k) \) at time \( k \) becomes

**Non-cooperative search:**

\[
\min_{\{U_p(k), b_p(k)\}} \tilde{J}_p(U_p(k), b_p(k)) \tag{6.7}
\]

subject to (2.14), (6.6), and where

\[
\tilde{J}_p(U_p(k), b_p(k)) = F_p(x_p(k + N|k)) - \sum_{i=1}^{N_b} \phi_i y_{ip}(k + N|k).
\]

**Cooperative search:**

\[
\begin{aligned}
\min_{\{U_p(k), b_p(k), U_{Np}(k), b_{Np}(k)\}} & \tilde{J}_p(U_p(k), b_p(k)) + \sum_{q \in N_p(k)} \alpha_{pq} \tilde{J}_p(U_q(k), b_q(k)) \\
\end{aligned} \tag{6.8}
\]
subject to (2.14), (3.4), (6.6), and, $\forall q \in \mathcal{N}_{p_k}, i \in \{0, \ldots, N_b\}, j \in \{0, \ldots, N - 1\}$:

\begin{align}
\dot{y}_{iq}(k + j + 1|k) &= \dot{y}_{iq}(k + j|k) + \dot{w}_{iq}(j), \\
\dot{y}_{iq}(k|k) &= y_{iq}(k - 1),
\end{align}

\begin{align}
b_{ip}(k + j|k) + B^*_{ip}(k) + \sum_{n \in \mathcal{N}_{p}(k)} \hat{b}_{in}(k + j|k) \\
&+ \sum_{r \notin \{p, \mathcal{N}_{p}(k)\}} b_{ir}(k + j|k) \geq \hat{w}_{iq}(j),
\end{align}

\begin{align}
\hat{b}_{iq}(k + j|k) = 1 \implies P(r_i^* - [I \ 0] \dot{x}_q(k + j|k)) \leq R_q 1, \forall v \in \{1, \ldots, n_v\},
\end{align}

\begin{align}
\dot{y}_{iq}(k + j|k) \in [0, 1], \\
\dot{w}_{iq}(j) \in [0, 1], \\
\hat{b}_{iq}(k + j|k) \in \{0, 1\},
\end{align}

where again the cooperative cost function is designing hypothetical trajectories, and the associated cells that could be searched, by other vehicles, while maintaining compatibility with the previously-published plans of others and the plan for $p$ itself (constraint (6.9c)).

As before, one agent $p_k$ optimizes at a time step $k$, whilst all other agents $p \neq p_k$ renew their plans via (2.19), i.e.,

\begin{align}
\mathbf{U}_p(k + 1) &= \{x^*_p(k + 1|k), u^*_p(k + 1|k), \ldots, u^*_p(k + N - 1|k), \kappa_{p}(x^*_p(k + N|k)) \},
\end{align}

but now with the additional update equation for the cell binary variables and the auxiliary states. For all cells $i \in \{1, \ldots, N_b\}$ and all $p \neq p_k$:

\begin{align}
b_{ip}(k + j|k) &= b^*_{ip}(k + j|k - 1), \forall j \in \{0, \ldots, N - 2\}, \\
b_{ip}(k + N|k) &= b^*_{ip}(k + N - 1|k - 1), \\
y_{ip}(k + j|k) &= y^*_{ip}(k + j|j - 1), \forall j \in \{0, \ldots, N - 2\}, \\
y_{ip}(k + N|k) &= y^*_{ip}(k + N - 1|k - 1),
\end{align}
Algorithm 6.1: Multi-vehicle cooperative search using DMPC

1. Design stabilizing controller $K_p$ and RPI set $\mathcal{R}_p$;
2. Tighten sets $\mathcal{X}_p, Z_r, \forall r \in \mathcal{C}_p$ via (2.16);
3. Wait for feasible solution $U_p^*(0)$, information $\bar{Z}_p^*(0)$, and terminal set $\mathcal{X}_{F_p}$ and controller $\kappa_{F_p}$ from central agent;
4. for $k = 1 : \infty$ do
   5. Sample current state $x_p(k)$;
   6. if $p = p_k$ then
      7. Choose cooperating set $\mathcal{N}_p(k)$ and weightings $\alpha_{pq}, \forall q \in \mathcal{N}_p$;
      8. Obtain new plan $U_p^{opt}(k)$ and $b_p^{opt}(k)$ as solution to optimization;
      9. Transmit new plan to other agents;
   else
      10. Renew current plan via (2.19) and (6.10);
   end
   11. Apply control (2.17): $u_p(k) = \bar{u}_p(k|k) + K_p(x_p(k) - \bar{x}_p(k|k))$;
   12. Update $B_{ip}(k), \forall i \in \{1, \ldots, N_b\}$ using (6.11);
   13. Wait one time step;
14. end

where it is assumed that remaining in the terminal set contributes nothing further to the search (the terminal value of the auxiliary state $y_{ip}$ is non-increasing). Similarly, each vehicle $p$ in the system updates its cell variables $B_{ip}(k)$, based on the latest plans for each vehicle:

$$B_{ip}^*(k) = b_{i_1}^*(k|k) \lor \ldots \lor b_{ip}(k|k) \lor \ldots \lor b_{i_{N_b}}^*(k|k), \forall i \in \{1, \ldots, N_b\},$$ (6.11)

where ‘$\lor$’ denotes a logical-OR expression.

The distributed multi-vehicle search algorithm, executed by all agents in parallel, is given by Algorithm 6.1. In line 9 the updating vehicle is required to transmit its new plan data. Information and communication requirements were discussed thoroughly in Chapters 2 and 3. However, in addition to ensuring that the necessary data is available to evaluate the constraints (2.14g) and (3.4h), the latest plan data for the binary cell variables is required to evaluate constraints (6.6c), and, if using
the cooperative cost, constraints (6.9b) and (6.9c). It is sufficient for an agent \( p \) to transmit the plan \( b_p(k_p) \) to all other agents following optimization at step \( k_p \).

### 6.2.4 Numerical examples

This section presents results from simulations using the multi-vehicle search algorithm.

**Example 6.1.** The aim is to search a 70 m × 70 m area, which is divided into 49 square cells, each of 10 m length; with each cell a nominal unity reward is associated. There are two vehicles in the problem, assumed to have the linear dynamics

\[
\begin{align*}
\dot{r}_p &= v_p, \\
m_p\dot{v}_p &= f_p,
\end{align*}
\]

for all \( p \in \{1, 2\} \), which are discretized with a time step of 1 second. The dynamics in this example are deterministic, i.e., \( W_p = \{0\}, \forall p \), to aid clarity in the results obtained, and to enable a nominal comparison of the control methods to be made without considering the effects of disturbances.

Each vehicle has a maximum velocity of 10 m/s and a maximum acceleration of 3 m/s\(^2\), corresponding to a maximum turn rate of 0.3 rad/s. Each also has an omnidirectional sensor of range 10 m. The 2-norm velocity, force, and collision avoidance constraints, and the sensor description, are implemented as 20-sided polyhedra [94]. The minimum vehicle separation in the collision avoidance constraint is 5 m. For the optimization, each vehicle has a prediction horizon of 5 steps, and the terminal sets \( X_{F_p} \) used in the local optimizations are again safety sets, being any feasible point with zero velocity, so that all plans end with the vehicle stationary. As in previous safety set examples, constraints are extended to cover the terminal step \( j = N \), to ensure admissibility of the terminal set.

Simulations were run using four different algorithms:

1. Non-cooperative DMPC. Algorithm 6.1 with the non-cooperative cost (6.7).
2. Cooperative DMPC. Algorithm 6.1 with the cost (6.8), and with the other vehicle included in the cooperating set $N_p$.

3. Decoupled DMPC. A form of Algorithm 6.1 with non-cooperative cost (6.7) but no communication between vehicles. Collision avoidance is implemented by use of an emergency stop when other vehicle is sensed within range. Note that this form permits updates in parallel.

4. Centralized MPC.

In all cases, the vehicles were initialized with stationary plans at time zero; that is, plans that leave each vehicle stationary at its initial position. The control agents then proceed by optimizing in an alternating sequence. In the first instance, the two vehicles, $A$ and $B$, were started at positions $(-35, -35)$ and $(35, 35)$ respectively, i.e., in diagonally-opposite corners. Figure 6.1 shows the resulting cumulative sum of cells searched against time elapsed, for each of the four algorithms. Centralized MPC, as expected, offers the best performance, searching the whole area in the shortest time. This is followed by cooperative DMPC, non-cooperative DMPC, and, finally, decoupled DMPC, which is by far the slowest despite updates being made in parallel. Cooperative DMPC offers an improvement over non-cooperative DMPC, on both search rate and finishing time, its performance close to that of centralized MPC.

Secondly, the same problem was posed with 150 different, random starting positions, the same position used for all four algorithms. Figure 6.2 shows the range and mean of finishing times, i.e., the time to search all 49 cells, for each algorithm. The bars represent the range of finishing times for the 150 simulations, from minimum time to maximum. The mean finishing time for each algorithm is shown as a dot. "Good" performance is represented by a low mean time and a low maximum time to search. It can be seen that, again, centralized offers best performance, by having the smallest range of times around the lowest mean. The cooperative algorithm is next best, followed by non-cooperative and decoupled respectively. Although there
is only a small difference between the cooperative and non-cooperative means, the maximum time for cooperative is noticeably shorter, and the overall range is more comparable to that of centralized, indicating that the cooperative DMPC algorithm does offer improved system-wide performance when applied to the problem of multi-vehicle search.

A metric of interest in the literature on networks is the stretch, a non-dimensional value that permits comparisons between networks of different topologies. For the problem of object location, the stretch is defined, in general terms, as the ratio between the distance travelled by a query to an object and the minimal distance from the query origin to the object [118]. It is, therefore, the ratio of time taken to search for a target to the time taken by going straight to it. The optimal value is unity, with values close to unity being considered ‘good’. Where multiple objects are present, the mean- or max-stretch values are considered. In terms of the multi-
vehicle search problem, define the stretch metric $s_i$ for a cell $i$ as

$$s_i \triangleq \frac{\hat{k}_i}{\left(\frac{1}{0.5V_{\text{max}}}\right)\|r^c_i - r_{p_i}(0)\|_2},$$

where $\hat{k}_i$ is the time step at which cell $i$, which has centroid position $r^c_i$, was first searched by vehicle $p_i$. (Searches thereafter, by $p_i$ or any other, do not count). The stretch then relates the time taken to search a cell to the original proximity of the searching vehicle. The mean- and max-stretch metrics for a search problem of $N_b$ cells are then given by

$$s_{\text{mean}} = \frac{1}{N_b} \sum_{i=1}^{N_b} s_i,$$

$$s_{\text{max}} = \max_{i \in \{1, \ldots, N_b\}} s_i,$$

\textit{Figure 6.2}: Finishing times for 49-cell multi-vehicle search. Ranges and means for 150 random start positions, for each algorithm.
Table 6.1: Comparison of mean- and max-stretch values for search problem.

<table>
<thead>
<tr>
<th>Controller</th>
<th>Decoupled</th>
<th>Non-cooperative</th>
<th>Cooperative</th>
<th>Centralized</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_{\text{mean}}$</td>
<td>8.61</td>
<td>7.24</td>
<td>7.08</td>
<td>6.22</td>
</tr>
<tr>
<td>$s_{\text{max}}$</td>
<td>48.20</td>
<td>39.25</td>
<td>35.25</td>
<td>35.90</td>
</tr>
<tr>
<td>$\sigma_{s_{\text{mean}}}$</td>
<td>3.27</td>
<td>2.71</td>
<td>2.68</td>
<td>2.19</td>
</tr>
<tr>
<td>$\sigma_{s_{\text{max}}}$</td>
<td>35.10</td>
<td>30.80</td>
<td>23.75</td>
<td>33.20</td>
</tr>
</tbody>
</table>

Near unity values of $s_{\text{mean}}$ or $s_{\text{max}}$ would indicate that cells have been searched quickly by vehicles closest to them. Though this function is not minimized by the local optimizations, it may prove to be a useful means of comparison between the four algorithms. Thus, Table 6.1 shows mean and standard deviations of these two metrics, over the 150 simulations. The trends match those previously observed: centralized obtains the lowest stretch values, followed by, in ascending order of stretch values, cooperative DMPC, non-cooperative DMPC and decoupled DMPC.

It is useful to gather further insight into how decision-making in the cooperative algorithm compares with that for the non-cooperative algorithm. Figure 6.3 shows, at different time steps, the two vehicles searching cells when controlled by the cooperative DMPC, starting from the original initial positions $(-35, -35)$ and $(35, 35)$ respectively, while Figure 6.4 shows the same situation when non-cooperative DMPC is the control method. Initially, from diagonally-opposite starting positions, both vehicles head towards the bottom-right corner of the map. With cooperative DMPC, at $k = 8$ the vehicle heading right (‘A’) alters course, steering away from the corner. Subsequently, the other vehicle (‘B’) continues to the corner. In contrast, in Figure 6.4, vehicle A itself continues into the corner, and vehicle B subsequently deviates in response to this action. Some indication of which is the better outcome is provided by the number of cells that have been searched by $k = 15$, by which time the corner manoeuvres are complete. For cooperative DMPC, vehicle A has searched ten cells by this time, and vehicle B seven. By the same time, however, the vehicles have searched only seven cells each when controlled non-cooperatively.
Figure 6.3: Two-vehicle search using cooperative DMPC. Cooperation means vehicle $A$ avoids the lower-right corner to allow search of that by vehicle $B$. 

(a) Vehicles heading towards corner; vehicle $A$ changes course.

(b) Vehicle $B$ continues into corner.

(c) Upon reaching corner, vehicle $B$ plans new course.
Figure 6.4: Two-vehicle search using non-cooperative DMPC. Vehicle B is forced by vehicle A to change course. Subsequently, fewer cells are searched by time step $k = 15$. 

(a) Now vehicle A does not change course.

(b) Both vehicles approach corner.

(c) Vehicle B forced to change course.
6.3 Distributed control of multiple vehicles for dynamic target tracking

6.3.1 Introduction

Surveillance and reconnaissance missions are perhaps the *raisons d'etre* of unmanned vehicles, be they ground-, sea-, or air-based. A typical scenario might be for a team of sensor-equipped vehicles to locate and observe some target or targets, either stationary or moving, for the purpose of identification, tracking or destruction. Such a problem has many facets: local path-planning decisions must be made, subject to kinematic and dynamic constraints. Meanwhile coordination of the team must be ensured, so to provide best estimates of the target states, all while in the presence of sensor and target uncertainty.

The problem formulation in this section is for linear time-invariant mobile sensing platforms observing dynamic targets, subject to probabilistic uncertainty, arising from both sensing and process noise. The general objective is to control the sensors' movements, while sharing and fusing information, to minimize target state uncertainty. Approaches to this problem include, for range-bearing sensors, locally-optimal control laws generated via a gradient-descent method [119–121]; the authors show by simulation that the resulting sub-optimal approach is 'very nearly' optimal at steady-state, yet no consideration is given of constraints. A popular proposal bases control decisions on receding-horizon optimization gain of information, both for a single-step [122, 123] and multiple prediction steps [124, 125]. In the latter case, some prediction model must be assumed for non-stationary targets; Grocholsky [124] uses a Kalman filter to provide future estimation based on observations to date, while the method of Ryan et al. [125] explicitly minimizes the expected entropy over the horizon, using probability distributions.

The contribution of this section is a distributed receding-horizon control algorithm for cooperative tracking of dynamic targets. The formulation combines the robust DMPC method with an information-theoretic objective, based on the pre-
dictions of target state uncertainty. Using the information form of the Kalman filter [3], and a model and current estimate of target states, with each local agent’s planned trajectory is associated an information plan; that is, the information that is expected to be gained via observing and filtering over the horizon. By agents exchanging these plans at each time step, and locally-fusing the information received, an agent seeks to determine a trajectory that would maximize the expected total information gathered, similar to the approach of Grocholsky [124]. The algorithm differs from that of Grocholsky [124] because, while Gaussian process and sensing noises are present, local, bounded state disturbances act on vehicles, necessitating a robust control method. By a sole control agent updating at a time step, the local controllers guarantee robust constraint satisfaction for the vehicles.

The next section outlines some preliminaries, including the problem statement and a review of the information form of the Kalman filter. The algorithm is developed in the subsequent section. Finally, numerical simulations are provided.

6.3.2 Preliminaries

Problem Statement

The problem is concerned with controlling \( N_p \) sensing subsystems to estimate the states of \( N_i \) uncertain dynamic processes or targets. The state \( s_i \in \mathbb{R}^{N_{s,i}} \) of a target \( i \in I \triangleq \{1, \ldots, N_i\} \) evolves according to the linear state space equation

\[
s_i(k + 1) = F_i s_i(k) + d_i(k),
\]

where \( d_i \in \mathbb{R}^{N_{s,i}} \) is the process noise associated with the \( i^{\text{th}} \) target, assumed Gaussian, white, zero mean, with covariance \( \mathbb{E}[d_i d_i^T] = Q_i \in \mathbb{R}^{N_{s,i} \times N_{s,i}} \), and independent of all other process noise signals.

The characteristics of the sensing platforms are described by the earlier problem statement of Section 2.2, where the dynamics of each subsystem \( p \in \mathcal{P} \) is governed
by the linear state-space model

\[ x_p(k + 1) = A_p x_p(k) + B_p u_p(k) + w_p(k). \]

As previously, \( x_p, u_p, w_p \) are, respectively, the state, control input and additive disturbance for a subsystem \( p \); all realizations of the latter reside in a closed and bounded set \( \mathcal{W}_p \) that contains the origin. Each subsystem is subject to local and coupling constraints, described by (2.2) and (2.4).

With each subsystem is associated a sensor for observation of the dynamic processes. The observation \( \xi_{pi} \in \mathbb{R}^{N_i.p.i} \) a sensor \( p \in \mathcal{P} \) makes of a process \( i \in \mathcal{I} \) is assumed to be given by the linear measurement model

\[ \xi_{pi}(k) = H_{pi}(k)s_i(k) + \rho_{pi}(k) \]

where \( \rho_{pi}(k) \) is the sensing noise at step \( k \), also assumed independent, white, Gaussian, with covariance \( \mathbb{E}[\rho_{pi}\rho_{pi}^T] = R_{pi} \in \mathbb{R}^{N_i.p.i} \times \mathbb{R}^{N_i.p.i} \). The objective is to provide control inputs \( u_p \) to manipulate the states \( x_p \) to provide the best estimates of the states \( s_i \) of the \( N_i \) uncertain processes.

**Range-only sensing for vehicles**

Suppose the \( N_p \) sensing subsystems are vehicles, to be controlled by DMPC, and the \( N_i \) dynamic processes are moving targets. The mobile vehicles can sense the range of a target, i.e., an observation is given by the noisy range measurement function \( h_{pi} : \mathbb{R}^{N_x.p} \times \mathbb{R}^{N_y.i} \mapsto \mathbb{R}_0^+ \)

\[ h_{pi}(x_p, s_i) = \sqrt{(r_{p,x} - r_{i,x})^2 + (r_{p,y} - r_{i,y})^2} + \rho_{pi} \]

\[ = r_{pi} + \rho_{pi}, \]
The measurement noise variance $R_{pi} = \mathbb{E}[\rho_{pi}^2] = \sigma_{pi}^2$ is modelled by a range-dependent function

$$\sigma_{pi}^2 = f_r(r_{pi}) = a_2(r_{pi} - a_1)^2 + a_0, \quad (6.13)$$

as used in, for example, Chung et al. [120], so that the quality of sensing is highest at a ‘sweet spot’ a distance $a_1$ from the target. The linearization of this measurement function $h_{pi}(x_p, s_p)$ provides the observation matrix $H_{pi}(k)$:

$$H_{pi}(k) = H_{pi}(x_p(k), s_p(k)) = \begin{bmatrix} \sin(\theta_{pi}(k)) & \cos(\theta_{pi}(k)) & 0 & 0 \end{bmatrix},$$

given that $x_p = [r_{p,x} \ r_{p,y} \ v_{p,x} \ v_{p,y}]^T$ and $s_i = [r_{i,x} \ r_{i,y} \ v_{i,x} \ v_{i,y}]^T$, and where then $\theta_{pi} = \arctan \left( \frac{r_{i,x} - r_{p,x}}{r_{i,y} - r_{p,y}} \right)$ is the bearing from sensor $p$ to target $i$. Thus, it is evident that controlling the sensor state, given a target state, changes the observation obtained.

The use of range-only measurement models greatly increases the need for a cooperative approach to sensing; sensors have no idea of the bearing to a target; thus its whereabouts may only be estimated effectively with range measurements taken from a number of points and fused.

**Fusing information: the information Kalman filter**

Consider a linear, time-invariant dynamic process

$$s(k+1) = Fs(k) + d(k)$$

where $\mathbb{E}[dd^T] = Q$, and suppose at time $k$ an observation $\xi = Hs + \rho$, where $\mathbb{E}[\rho\rho^T] = R$, is made of the true state $s(k)$. Kalman filtering is a widely-applied technique for optimal estimation of such a process, given those observations; see, for example, Kamen and Sugie [107] for an overview. Briefly, an estimate $\hat{s}$ of the state $s$ is held, and with it an associated error covariance matrix $P$, a measure of the accuracy of the estimation. Through two phases, these variables are updated at each
time step: in the prediction phase, the process model acts on values \( \hat{s}(k-1|k-1) \) and \( P(k-1|k-1) \) from the previous time step \( k-1 \) to obtain a new predicted state and covariance, \( \hat{s}(k|k-1) \) and \( P(k|k-1) \) respectively. These are subsequently refined in the estimation phase, using the latest observation \( \xi(k) \), to produce \( \hat{s}(k|k) \) and \( P(k|k) \).

For example, the state estimate by applying an optimal Kalman gain, \( K^{\text{Kalman}}(k) \), to the error between observation and prediction (the innovation), i.e., \( \xi(k) - \hat{s}(k|k-1) \), and adding the resulting value to the prediction:

\[
\hat{s}(k|k) = \hat{s}(k|k-1) + K^{\text{Kalman}}(k) \left[ \xi(k) - H(k)\hat{s}(k|k-1) \right].
\]

The information form [3] of the Kalman filter replaces the estimate of the state, \( \hat{s} \), and associated error covariance matrix \( P \), with an information state \( \tilde{y} \) and information matrix \( \tilde{Y} \):

\[
\tilde{y} \triangleq P^{-1}\hat{s}, \\
\tilde{Y} \triangleq P^{-1}.
\]

An observation \( \xi(k) \) contributes an amount \( i(k) \) to the information state and an amount \( I(k) \) to the information matrix, where

\[
i = H^{T}R^{-1}\xi, \\
I = H^{T}R^{-1}H.
\]

The algorithm then proceeds as follows. Given observations up to a time \( k \), current estimates of information state \( \tilde{y}(k|k) \) and information matrix \( \tilde{Y}(k|k) \) are determined through two steps:
1. Prediction:

\[ \hat{y}(k|k-1) = (1 - \Omega(k))F^{-T}\tilde{y}(k-1|k-1), \quad (6.14a) \]
\[ Y(k|k-1) = M(k) - \Omega^T(k)\Sigma(k)\Omega(k), \quad (6.14b) \]

where,

\[ M(k) = F^{-T}P^{-1}(k-1|k-1)F^{-1}, \quad (6.14c) \]
\[ \Omega(k) = M(k)\Sigma^{-1}(k), \quad (6.14d) \]
\[ \Sigma(k) = M(k) + Q^{-1}. \quad (6.14e) \]

2. Estimation:

\[ \tilde{y}(k|k) = \hat{y}(k|k-1) + i(k), \quad (6.15a) \]
\[ Y(k|k) = Y(k|k-1) + I(k). \quad (6.15b) \]

The posterior estimate of the state is then obtained as \( \bar{s}(k|k) = Y^{-1}(k|k)\tilde{y}(k|k) \). The transformation of variables from \( \bar{s} \) and \( P \) to \( \tilde{y} \) and \( Y \), while resulting in increased complexity in the prediction stage of the filter, provides simpler update equations in the estimation stage. The importance of this becomes clear when multiple sensors are employed. The key property is that, whereas for the standard Kalman filter multiple observations may not be combined linearly,

\[ \bar{s}(k|k) \neq \bar{s}(k|k-1) + \sum_{p=1}^{N_p} K_{p,\text{Kalman}}(k) [\xi_p(k) - \mathbf{H}_p(k)\bar{s}(k|k-1)], \]

200
for the information form of the filter, they may be:

\[
\tilde{y}(k|k) = \tilde{y}(k|k - 1) + \sum_{p=1}^{N_p} \mathbf{y}_p(k),
\]

\[
Y(k|k) = Y(k|k - 1) + \sum_{p=1}^{N_p} \mathbf{I}_p(k).
\]

This is a particularly attractive feature for multi-sensor data fusion, meaning that the estimation of a process may be easily distributed. Agents make local observations and communicate them; subsequently, observations may be (locally or centrally) linearly-combined to produce estimates of the target states.

### 6.3.3 Distributed MPC for dynamic target tracking

In this subsection, the objective function for the team of vehicles is defined. The top-level aim is to control the sensing vehicles to minimize uncertainty, or maximize information, about the targets. The information Kalman filter forms the basis on which local decisions shall be made; a local agent shall design a trajectory, or a sequence of control inputs, that minimizes predictions of the target state estimation error.

**Maximizing expected information**

Based on the work of Grocholsky et al. [68, 122], we seek to maximize the expected entropic information by manipulating the observation matrix \( H_p(k) \) and covariance matrix \( R_p(k) \), each of which is dependent on the sensor state \( x_p(k) \), given a target state \( s_i(k) \). Generally, for each target \( i \in \mathcal{I} \), the objective is to maximize the
determinant of the information matrix$^1$:

$$\sum_{k=1}^{N_i} \prod_{i=1}^{N_i} |\mathbf{Y}_i(k|k)|,$$

(6.16)

where,

$$\mathbf{Y}_i(k|k) = \mathbf{Y}_i(k|k-1) + \sum_{p=1}^{N_p} \mathbf{I}_{pi}(k)$$

$$= \mathbf{Y}_i(k|k-1) + \sum_{p=1}^{N_p} \mathbf{H}_{pi}^T(x_p(k); s_i(k)) \mathbf{R}_{pi}^{-1}(x_p(k); s_i(k)) \mathbf{H}_{pi}^T(x_p(k); s_i(k)).$$

Despite the additive nature of the fusion process, whereby the local subsystem contributions are summed and added to the current information matrix, the objective function (6.16) is coupled and non-separable in these contributions, as, in general, $|A + B| \neq |A| + |B|$. This means that distribution of the objective is not straightforward. The approach used here is for each of the local control agents to attempt to maximize its own local representation $\mathbf{Y}_{pi}$ of the information matrix $\mathbf{Y}_i$, (noting that $\mathbf{Y}_i \neq \mathbf{Y}_{1i} + \ldots + \mathbf{Y}_{Npi}$). Thus, the target state estimation shall be local to each agent, facilitated by inter-agent communication of observations. Secondly, the information filter is projected forwards in time as part of the prediction model internal to the distributed MPC, so that predictions are made of $\mathbf{Y}_{pi}$, associated with state and control input predictions, and based on the resulting local information matrix predictions. Furthermore, in keeping with the cooperative algorithms developed so far, where hypothetical state trajectories are designed for vehicles in the cooperating set $\mathcal{N}_p$, included in the predictions for $\mathbf{Y}_{pi}$ are hypothetical information predictions for these others. The local objective function to minimize is then

$$\sum_{j=0}^{N_i} \sum_{i=1}^{N_i} -\log \left( |\mathbf{Y}_{pi}(k+j|k)| \right)$$

(6.17)

$^1$In Grocholsky et al. [68, 122], the determinant of a block-diagonal information matrix $\mathbf{Y} \triangleq \text{diag}(\mathbf{Y}_1, \ldots, \mathbf{Y}_{N_i})$ is maximized. However, it is well-known that $|\text{diag}(A, B)| = |A||B|$, so the formulation in this section is equivalent.
where $\forall j \in \{0, \ldots, N - 1\}, i \in I,$

\[
\begin{aligned}
\hat{Y}_{pi}(k + j + 1|k) &= \left[ F_i \hat{Y}_{pi}^{-1}(k + j|k) F_i^T + Q_i \right]^{-1} \\
&+ \hat{I}_{pi}(k + j|k) + \sum_{q \in \mathcal{N}_p(k)} \hat{I}_{qi}(k + j|k) + \sum_{r \notin \{p, \mathcal{N}_p(k)\}} \hat{I}_{ri}^*(k + j|k), \\
\end{aligned}
\tag{6.18a}
\]

\[
\hat{Y}_{pi}(k|k) = \hat{Y}_{pi}(k),
\tag{6.18b}
\]

\[
\begin{aligned}
\hat{I}_{pi}(k + j|k) &= \hat{H}_{pi}^T(k + j|k) \hat{R}_{pi}^{-1}(k + j|k) \hat{H}_{pi}(k + j|k), \\
\hat{H}_{pi}(k + j|k) &= \begin{bmatrix} \sin(\hat{\theta}_{pi}(k + j|k)) & \cos(\hat{\theta}_{pi}(k + j|k)) & 0 & 0 \end{bmatrix}, \\
\hat{R}_{pi}(k + j|k) &= a_2(\hat{\tau}_{pi}(k + j|k) - a_1)^2 + a_0, \\
\hat{\tau}_{pi}(k + j|k)^2 &= (\hat{r}_{pi,x}(k + j|k) - \hat{r}_{pi,x}(k + j|k))^2 \\
&+ (\hat{r}_{pi,y}(k + j|k) - \hat{r}_{pi,y}(k + j|k))^2, \\
\hat{\theta}_{pi}(k + j|k) &= \arctan \left( \frac{\hat{r}_{pi,x}(k + j|k) - \hat{r}_{pi,x}(k + j|k)}{\hat{r}_{pi,y}(k + j|k) - \hat{r}_{pi,y}(k + j|k)} \right),
\end{aligned}
\tag{6.18c}
\]

\[
\begin{aligned}
\hat{I}_{qi}(k + j|k) &= \hat{H}_{qi}^T(k + j|k) \hat{R}_{qi}^{-1}(k + j|k) \hat{H}_{qi}(k + j|k), \\
\hat{H}_{qi}(k + j|k) &= \begin{bmatrix} \sin(\hat{\theta}_{qi}(k + j|k)) & \cos(\hat{\theta}_{qi}(k + j|k)) & 0 & 0 \end{bmatrix}, \\
\hat{R}_{qi}(k + j|k) &= a_2(\hat{\tau}_{qi}(k + j|k) - a_1)^2 + a_0, \\
\hat{\tau}_{qi}(k + j|k)^2 &= (\hat{r}_{qi,x}(k + j|k) - \hat{r}_{qi,x}(k + j|k))^2 \\
&+ (\hat{r}_{qi,y}(k + j|k) - \hat{r}_{qi,y}(k + j|k))^2, \\
\hat{\theta}_{qi}(k + j|k) &= \arctan \left( \frac{\hat{r}_{qi,x}(k + j|k) - \hat{r}_{qi,x}(k + j|k)}{\hat{r}_{qi,y}(k + j|k) - \hat{r}_{qi,y}(k + j|k)} \right),
\end{aligned}
\tag{6.18g}
\]

and $\forall q \in \mathcal{N}_p(k),$

\[
\begin{aligned}
\hat{I}_{ri}(k + j|k) &= \hat{H}_{ri}^T(k + j|k) \hat{R}_{ri}^{-1}(k + j|k) \hat{H}_{ri}(k + j|k), \\
\hat{H}_{ri}(k + j|k) &= \begin{bmatrix} \sin(\hat{\theta}_{ri}(k + j|k)) & \cos(\hat{\theta}_{ri}(k + j|k)) & 0 & 0 \end{bmatrix}, \\
\hat{R}_{ri}(k + j|k) &= a_2(\hat{\tau}_{ri}(k + j|k) - a_1)^2 + a_0, \\
\hat{\tau}_{ri}(k + j|k)^2 &= (\hat{r}_{ri,x}(k + j|k) - \hat{r}_{ri,x}(k + j|k))^2 \\
&+ (\hat{r}_{ri,y}(k + j|k) - \hat{r}_{ri,y}(k + j|k))^2, \\
\hat{\theta}_{ri}(k + j|k) &= \arctan \left( \frac{\hat{r}_{ri,x}(k + j|k) - \hat{r}_{ri,x}(k + j|k)}{\hat{r}_{ri,y}(k + j|k) - \hat{r}_{ri,y}(k + j|k)} \right),
\end{aligned}
\tag{6.18h}
\]
and, finally,

\[ s_i(k + j + 1|k) = F_i s_i(k + j|k), \]  
\[ s_i(k|k) = s_{pi}(k), \]

where

- the overbar denotes, as it has in previous chapters, predictions generated via a nominal subsystem model, and the hat denotes predictions for those in the cooperating set \( N_p \);

- the dynamic equation (6.18a) for predictions \( \hat{Y}_pi(\cdot|k) \) includes both the prediction and estimation phases of the Kalman filter, (by judicious substitution of variables);

- the initial constraint (6.18b) sets the first predicted value \( \hat{Y}_pi(k|k) \) equal to the true value \( Y_{pi}(k) \), computed locally using latest observations;

- new information comprises a local prediction, \( \bar{I}_{pi}(k + j|k) \), predictions for those in the cooperating set, \( \bar{I}_{qi}(k + j|k), \forall q \in N_p(k) \), and a summation of previously-published, planned values from other sensors, \( \bar{I}_{ri}^*(k + j|k), \forall r \notin \{p,N_p\} \);

- the matrices \( \bar{H}_{pi} \) and \( \bar{R}_{pi} \) (and \( \bar{H}_{qi} \) and \( \bar{R}_{qi} \) for \( q \in N_p(k) \)) are time-varying, dependent on the predicted range and bearing from sensor to target;

- the constraints include a nominal dynamics model for the targets, with the current local estimate \( \bar{s}_{pi}(k) = Y_{pi}^{-1}(k)\tilde{y}_{pi}(k) \) as the initial state.

This objective couples the expected information predictions with the predicted states \( \bar{x}_p(\cdot|k) \) and \( \hat{x}_q(\cdot|k), \forall q \in N_p \), via, at each prediction time step, the range and bearing of each sensor's predicted trajectory from the targets' trajectories. By manipulating control inputs, state trajectories may be designed that maximize the prediction of information gathered over the horizon.
Note that uncertainty manifests itself via the target process model in each local optimization; open-loop predictions are made from a current local estimate of state, with no feedback mechanism present. Thus, errors may propagate—and be added to—over the horizon. Furthermore, the objective is highly nonlinear. Consequently, as discussed in the introduction to this chapter, no stability guarantees exist, but robust feasibility and constraint satisfaction are guaranteed.

**Implementation**

The information-theoretic objective, defined by (6.17) and (6.18), forms the local optimization objective function $V_p$. As in the previous section of this chapter, insertion into the general distributed MPC optimization $P^{D,N_p}(k)(x_p(k); \tilde{Z}_p^+(k))$, defined in Chapter 3, results in a cooperative distributed data fusion algorithm (Algorithm 6.2).

Note that, in common with the multi-vehicle search algorithm in the previous section, the control agents’ ‘plans’ are not limited to the sequence $U_p(k)$, but here also include a plan for information:

$$\{\hat{I}_{pi}(k|k), \hat{I}_{pi}(k+1|k), \ldots, \hat{I}_{pi}(k+N|k)\}_{i \in I_p},$$

where $I_p \subseteq I$ is the subset of targets allocated to be tracked by vehicle $p$. The algorithm follows the single-update form of previous algorithms, whereby, to guarantee coupling constraint satisfaction, only one agent optimizes at any particular instant. Therefore, just as non-updating agents renew their planned controls via the update equation (2.19), i.e., given

$$U_p^*(k) \triangleq \{\hat{x}_p^*(k|k), \hat{u}_p^*(k|k), \ldots, \hat{u}_p^*(k+N-1|k)\},$$

---

2 The problem of target allocation is not considered here, for simplicity of exposition; it is assumed that targets are allocated by the (centralized) initialization, and not changed thereafter. Target allocation for decentralized data fusion is an active area of research [126–128], and the combined problem of dynamic allocation and information gathering is of high complexity.
Algorithm 6.2: Robust DMPC for dynamic target tracking

1. Design stabilizing controller $K_p$ and RPI set $R_p$;
2. Tighten sets $\mathcal{Y}_p, \mathcal{Z}_c, \forall c \in C_p$, via (2.16);
3. Wait for feasible solution $U_p^*(0)$, coupling information $\tilde{Z}_c^*(0)$, terminal set $X_{F_p}$ and controller $\kappa_{F_p}$, set of targets $I_p$, and Kalman information $\tilde{y}_{pi}(0)$ and $Y_{pi}(0), \forall i \in I_p$, from central agent;
4. for $k = 1 : \infty$ do
   5. Sample current state $x_p(k)$;
   6. Make observations $\xi_{pi}(k)$ of targets $i \in I_p$;
   7. Calculate local information $\tilde{y}_{pi}(k)$ and $Y_{pi}(k), \forall i \in I_p$, via Kalman update equations (6.14) and (6.15);
   8. if $p_k = p$ then
      9. Choose cooperating set $N_p(k)$;
     10. Obtain new plan $U_p^{opt}(k)$ and \{ $I_{pi}^{opt}(k+j|k)$, $j \in \{0, \ldots, N\}, i \in I_p$, as solution to $F_{p}^{D,N_p}(x_p(k); \tilde{Z}_p^*(k))$, with optimization cost defined by (6.17) and (6.18);
     11. Transmit new plan to other agents;
   else
      12. Renew current plan via (2.19) and information via (6.19);
   end
   13. Apply control (2.17): $u_p(k) = \tilde{u}_p(k|k) + K_p(x_p(k) - \bar{x}_p(k|k))$;
   14. Wait one time step;
15. end

the renewed plan is,

$$\tilde{U}_p(k+1) \triangleq \{ \tilde{x}_p^*(k+1|k), \tilde{u}_p^*(k+1|k), \ldots, \tilde{u}_p^*(k+N-1|k), \kappa_{F_p}(\tilde{x}_p^*(k+N|k)) \},$$

the information plans are renewed in a similar fashion:

$$\{ \tilde{I}_{pi}^*(k+1|k), \tilde{I}_{pi}^*(k+2|k), \ldots, \tilde{I}_{pi}^*(k+N|k),$$

$$[ \tilde{H}_{pi}^T(k+N|k) \tilde{R}_{pi}^{-1}(k+N|k) \tilde{H}_{pi}(k+N|k) ] \}_{i \in I_p}. \quad (6.19)$$

The terminal information is that calculated for the new terminal state $A_p \bar{x}_p^*(k+N|k) + B_p \kappa_{F_p}(\tilde{x}_p^*(k+N|k))$. By inter-agent communication of these information
plans, each agent may construct a representation $Y_{pi}$ of information matrix $Y_i$ by fusion, in this case a simple summation of the information received.

An important practical consideration is the complexity of the information-theoretic objective; even where the local subsystem constraints for the DMPC optimization are polytopic or polyhedral, or even simply convex, the objective function is non-linear and non-convex, offering many local minima [119, 124]. Not to be deterred, the following example shows that sub-optimal solvers, namely the fmincon solver in MATLAB, may be employed to provide satisfactory solutions and closed-loop behaviour. Furthermore, cooperation makes a difference, resulting more information being gathered.

6.3.4 Numerical examples

The $N_p$ sensing vehicles assume an uncertain form of the linear dynamics employed previously in this chapter, i.e., those of Chapter 3, where

$$\dot{r}_p = v_p,$$

$$m_p \dot{v}_p = f_p + \Delta f_p,$$

so that the discrete-time state equation is

$$x_p(k+1) = \begin{bmatrix} r_p(k+1) \\ v_p(k+1) \end{bmatrix} = \begin{bmatrix} I_2 & \delta t I_2 \\ 0_2 & I_2 \end{bmatrix} \begin{bmatrix} r_p(k) \\ v_p(k) \end{bmatrix} + \begin{bmatrix} (\delta t)^2 I_2 \\ \delta t I_2 \end{bmatrix} (u_p(k) + w_p(k)),$$

where $m_p u_p(k) = f_p(k)$, and $m_p w_p(k) = \Delta f_p(k)$. The maximum speed and acceleration are again

$$\|v_p\|_2 \leq V_{\text{max}},$$

$$\|u_p\|_2 \leq U_{\text{max}},$$

207
where $V_{\text{max}} = 10$ m/s and $U_{\text{max}} = 3$ m/s$^2$. Polyhedral approximations to these circular constraints are assumed. The disturbance is limited to 20% of the acceleration:

$$w_p(k) \in W_p \triangleq \left\{ w_p \in \mathbb{R}^2 : \|w_p\|_\infty \leq 0.2U_{\text{max}} \right\}.$$  

As in the examples of Chapter 3, the feedback matrix for robustness, $K_p$, is the nilpotent controller for the nominal subsystem model, and the set $R_p$ is the corresponding mRPI set. Constraints are tightened accordingly. The terminal set $\mathcal{X}_{F_p}$ is the zero-velocity safety set

$$\mathcal{X}_{F_p} = \{ x_p \in \mathbb{R}^4 : Sx_p \leq 0 \},$$

where

$$S = \begin{bmatrix} 0_2 & I_2 \\ 0_2 & -I_2 \end{bmatrix}.$$  

Each vehicle is equipped with a range-only sensor, as described in Section 6.3.2, with the sensing uncertainty variance (6.13) defined by the parameter values $a_2 = 0.1568$, $a_1 = 15.6250$ and $a_0 = 0.0008$, as used by Chung et al. [119, 120].

The $N_i$ dynamic targets have similar, albeit autonomous, dynamics:

$$F_i = \begin{bmatrix} I_2 & \delta t I_2 \\ 0_2 & I_2 \end{bmatrix}, \forall i \in I,$$

with state uncertainty covariance matrices

$$Q_i = 0.1 \begin{bmatrix} (\delta t)^3 I_2 & (\delta t)^2 I_2 \\ (\delta t)^2 I_2 & \delta t I_2 \end{bmatrix}, \forall i \in I.$$  

**Example 6.2** (Two sensors observing a solitary target). Initially, a solitary, stationary target is to be observed by two sensing vehicles, each each with a prediction horizon of six time steps, where the discretization time $\delta t = 1$ second. In terms of
maximizing information, or minimizing error covariance, many locally-optimal configurations exist [119, 124]. However, all see the sensing platforms each at a range of \( \alpha_1 \) from the target, and separated by some constant angle; in the case of two sensors, observations are made from orthogonal viewpoints.

Figure 6.5 shows the resulting sensor trajectories and estimation errors for both non-cooperative and cooperative DMPC. The difference in closed-loop performance is pronounced. Under local control by non-cooperative DMPC, the vehicles do not settle to a clear steady-state, instead continually circling the target with varying degrees of eccentricity; consequently, the estimation errors are high. With cooperation present, however, the vehicles converge to orthogonal positions, with relatively
small estimation errors. It is known that cooperation improves the decentralized fusion process [124]; the key result here is that the outcome is that cooperative behaviour occurs even with the inclusion of the additional constraints required for robust feasibility.

Note that although the local control includes guaranteed robustness, no disturbances were actually applied either to the vehicles or to the target, so that the uncertainty present arises only from the sensing noise and the target modelling errors. The same sensing noise seed was used for all simulations. Parallels may be drawn with the results of Venkat et al. [70], where a non-cooperative approach is not sufficient to guarantee stability when the objective function is coupled. Figure 6.6 shows the associated histories of the local information matrix determinant, $|Y_{pi}(k)|$, for each control method. Cooperative DMPC produces a higher and relatively constant steady-state value, corresponding to lower uncertainty in target state estimation.

**Example 6.3 (Four sensors for two targets).** Extending the problem to four vehicles observing two targets, cooperative DMPC again shows a benefit (Figure 6.7). In the previous section, the problem of sensor-target assignment was mentioned. For this example, each sensor can ‘see’ every target, i.e., each sensor is assigned both targets. How the sensors then position themselves is the important problem, and includes, implicitly, a further degree of sensor-target assignment. For example, a ‘bad’ outcome is for all four sensors to move close to one target, neglecting the other. A ‘good’ arrangement would see the sensors deploying themselves evenly, sensing each target with a similar level of error; thus agents should cooperate when making decisions. Here, as shown in Figure 6.7, the non-cooperative algorithm results in an imbalanced assignment, leading to the more-uncertain observation of one of the targets. Without cooperation, the three sensors observing the bottom-right target have no incentive to change their plans and assist the sole agent observing the top-left target; system performance suffers as a consequence. The cooperative algorithm, however—the local cooperating set $N_p$ comprising only the next-in-line
vehicle to optimize—avoids such an outcome, the local control agents cooperating to evenly allocate sensors to targets, and subsequently reaching an outcome where information gathering is more accurate.

6.4 Summary

In this chapter, two applications for distributed MPC have been presented. The first considered the problem of search of an area of unknown content by a team of vehicles. The contribution of the method developed is the incorporation of the vehicle dynamics model, constraints and collision avoidance in the search. In the resulting algorithm, the local optimization is a mixed-integer program, where a local vehicle aims to maximize rewards collected over the horizon, while subject to the
Figure 6.7: Trajectories and estimation errors for four vehicles tracking two targets, controlled by (left) non-cooperative DMPC, and (right) cooperative DMPC.
plans of others. The problem calls for cooperation between control agents, to avoid duplication of efforts; examples showed that by using the cooperative form of DMPC, an improvement is seen over that of non-cooperative DMPC.

The second application presented was observation and tracking of uncertain, dynamic targets by sensing vehicles. Using techniques from optimal estimation theory, the information-theoretic objective maximizes information gathered over the horizon, based on observations to date and predictions from a Kalman filter. This function, though non-linear and coupled non-separably, forms the local optimization objective in the distributed MPC algorithm, leading to a robustly-feasible dynamic target tracking formulation. Examples again show the comparative performance benefits of using cooperative DMPC.
Chapter 7

Conclusions

7.1 Summary of contributions

The main contributions of this thesis are the development of a new and flexible formulation of distributed MPC, and analytical results on inter-agent cooperation in the presence of coupling constraints. The individual contributions of each chapter are summarized in what follows.

Distributed MPC for uncertain linear systems

In Chapter 2, a distributed MPC algorithm was developed for a group of linear time-invariant subsystems. The local subsystems, which are dynamically-decoupled, share coupling constraints, and each is subject to persistent and bounded state disturbances. Local agents in the formulation, which is based on the tube MPC method [27], achieve coupling constraint satisfaction by communication of planned ‘tubes’ following optimizations. Key features are that (i) only one subsystem agent updates its plan at each time step, (ii) robust stability is guaranteed for any choice of update sequence, and (iii) each agent communicates only after its update. An important contribution, then, is that this is the first work to combine robust DMPC with flexible communications.
An investigation of the trade between communication and performance was provided, and the new DMPC was compared with a related method, constraint-tightening DMPC [38]. Although the latter method subsumes the former, and offers better control performance, the unique initial constraint and sequence-independent constraint tightening of the proposed DMPC leads to a simple and flexible approach, with no reliance on instantaneous communications between agents.

**DMPC with cooperation**

Chapter 3 motivated the need for cooperation between agents, when otherwise objectives or coupled constraints would lead to conflict or greedy behaviour. A cooperative form of the DMPC algorithm was proposed, in which local agents are prepared to sacrifice local performance to permit a benefit to system-wide performance; in practice, a local agent designs, in addition to its own plan, hypothetical plans for others. The key contribution is that, while cooperative behaviour is promoted, robust constraint satisfaction, feasibility, and stability are maintained, and for any choices of cooperating sets of agents. Examples showed the elimination of ‘greedy’ decision-making, and the breaking of ‘deadlock’ situations, for multi-vehicle scenarios.

**An analysis of cooperation: convergence to state limit sets**

In the first part of Chapter 4, a formal analysis of the cooperative DMPC was presented. By posing the local optimizations as a game between subsystem agents, the Nash solution concept was employed to show that by increasing the ‘level’ of cooperation between agents, that is, by adding edges in a graph of cooperative agents, the set of Nash solutions to the game may grow no larger. Furthermore, the set always includes the (system-wide ‘optimal’) solution from a centralized optimization.

Under relaxed assumptions on the subsystem objectives, the controlled system is shown to converge to some state limit set. In that set, each agent is continually playing a Nash strategy. Thus, by implication, the set of such limit sets can also not
be enlarged by increasing cooperation. Though it is not possible, in a general sense, to prove strictness of this result, examples show that cases exist where, by using only partial cooperation, the system-wide optimal limit set is reached when otherwise it would not be. The key contribution, therefore, is an analytical confirmation of an intuitive concept: increasing cooperation does not harm the convergence outcome, even with the additional constraints to guarantee robust constraint satisfaction.

**Adaptive cooperation between agents**

The second part of Chapter 4 seeks to place an upper bound on the level of cooperation required for ‘good’ performance. From further analysis, relating choice of cooperating sets to the coupling structure, the primary findings are that (i) full cooperation is not always necessary to provide best distributed performance, yet (ii) the set of immediate, directly-coupled neighbours is not a sufficient choice. It is proposed that the ‘best’ set of agents to cooperate with is those connected, either directly or indirectly, in a graph of active coupling constraints. Subsequently, an adaptively-cooperative DMPC algorithm is introduced, where the cooperating set decision is made on-line, based on an agent’s currently-active constraints.

**Feasible updates in parallel**

The distributed MPC of earlier chapters assumed control agents optimized sequentially, one per time step, to maintain robust feasibility. In Chapter 5, the method was extended to permit local updates in parallel, despite the presence of coupling constraints and uncertainty. By increasing constraint margins, local agents may make decisions simultaneously without coupling constraint violation; sufficient conditions for robust feasibility are subsequently identified. Simulations show that this approach is not necessarily overly-conservative, and may improve performance, although at a cost of some loss of flexibility in communications.
Multi-vehicle applications

In Chapter 6, two vehicle-based application areas were explored. The first application considered the search or coverage of an area by a team of vehicles, the aim being to maximize rewards collected, or coverage achieved, while coordinating efforts to minimize duplication. Simultaneously, the vehicles must avoid collision. The cell-based search method proposed is simple, achieved by insertion of a search-theoretic, mixed-integer, linear cost in the DMPC optimizations. Furthermore, unlike other cell-based, look-ahead methods, the formulation includes a dynamics model and kinematic constraints for local path planning.

In the second part of the chapter, the problem considered is that of providing control to a team of sensing vehicles for observation and estimation of target uncertain, dynamic processes. Given a Gaussian model of process noise, together with Gaussian, range-dependent sensing errors, the distributed receding-horizon approach uses local, forward-projected Kalman filters to determine trajectories that would maximize information gathered over the horizon. Agents exchange and fuse information plans to determine local estimates. In practice, a non-linear, non-convex optimization results, yet simulations show that good performance, and a pronounced benefit through partial cooperation, can be obtained by using a commercially-available optimization solver.

7.2 Future research directions

Given the contributions made in this thesis, a number of future research areas may now be proposed.

Performance prediction for distributed MPC

The development of MPC is a consequence of the desire for optimal control, in terms of minimizing some performance index, in the presence of constraints. Yet, an analytical treatment of predicted performance is absent in the literature, with
the notable exception of Richards et al. [129], in which a performance prediction tool for MPC is proposed, based on identification and analysis of the regimes of operation for an MPC-controlled system. Otherwise, closed-loop performance is evaluated by numerical simulation, which—given the number of controller settings that may be available to the designer—requires extensive computation, and may be far from exhaustive. Simulating for a number of distributed controllers exacerbates the problem.

It would be therefore be of value to produce a performance prediction tool for distributed MPC, complementary to, or based on, the tool proposed by Richards et al. [129]. Such a tool would necessarily allow for the different sequencing of local optimizations. One possible approach is to rewrite the set of distributed, local problems as a centralized problem, with concatenated states and inputs, and block-diagonal system and constraint matrices. Subsequently, only part of the combined decision variable, corresponding to the optimizing agent’s plan, would be optimized at a time step, the remainder constrained to follow candidate plans. The method of Richards et al. [129] could be applied to the resulting constrained centralized problem. Further development would include inter-agent cooperation.

**Bounding sub-optimality**

Supposing that the solutions that arise from centralized MPC are ‘optimal’, at least in terms of system-wide performance, it would be of significant value to derive analytical bounds on the sub-optimality that arises from applying distributed MPC. Simulations suggest the key factors are, in no particular order, the number of agents, the update sequence, the coupling structure, the local disturbances, and the cooperation graph. It is most likely that any meaningful result will depend on the coupling constraints sets having ‘volume’; equality-constrained couplings do not permit agents to deviate from the provided initial feasible solution.
Optimal update sequencing

The distributed algorithms proposed in this thesis introduced update sequencing with greater flexibility than comparable robust methods [38]. The investigations of Chapter 2 illustrated how performance varies with sequence, and in Chapter 4 it was seen that even the steady-state limits of the system may depend on the order in which subsystems optimize. Many heuristics are possible for choosing the order of updates apart from cyclical sequencing. For example, the next agent to optimize might be decided by, inter alia, auctioning, the authority of agents to update, the relative magnitudes of previous disturbances, or comparison of the 'constrainedness' of local problems. Does choosing at each time step the agent who has most to gain, i.e., the greatest decease in open-loop cost, necessarily lead to best closed-loop performance?

Linking with the development of a performance prediction tool for DMPC, it would be interesting to determine explicitly the relationship between predicted performance and update sequence. Of particular value would be an outcome where a tractable optimization problem, parametric in disturbance sequence, could be solved that minimized expected closed-loop performance by manipulating the update sequence.

Continuous-time systems and asynchronous optimization

This thesis has focused exclusively on discrete-time systems, and has assumed control update times are synchronous. True synchronous distributed optimization algorithms, though often sufficient for guaranteed feasibility and convergence, present significant practical difficulties in real-world applications. Furthermore, an implicit assumption throughout this thesis is that optimization times are shorter than control update times. An asynchronous formulation would permit replanning by any agent at any time, though with some time limit imposed on optimizations, updating controls as and when required. A continuous-time system model might aid identification of when replanning is required, by comparing the state evolution with predictions.
Imperfect or constrained communication

All inter-agent communication in this thesis has been assumed timely and error-free, and the communication graph has been assumed sufficient to meet any requirements. Were the communication graph insufficient, perhaps owing to select transmission links being broken, it would be interesting to identify under what conditions the distributed MPC is robust to absent or corrupt communications. Multi-hop transmissions may introduce significant delay—how does performance degrade with increasing delay? Another practical concern is bandwidth constraints. How are the update sequence and the inter-agent cooperation arrangements best chosen given an imposed upper limit on communication?

Coupled dynamics

With dynamically-decoupled subsystems coupled through the constraints, a local agent includes a nominal representation of other subsystems’ plans, with uncertainty accounted for by tightening constraint limits. Many application systems, such as those in the process control industries, may not be decomposed in this way, subsystem dynamics being coupled. Generalization to this case would introduce considerable difficulties to the distributed MPC design problem. The additional local uncertainty arising from uncertainty around interacting subsystems’ plans may be bounded, and constraints tightened further, yet local problems may soon become excessively conservative, and sets of admissible controls prohibitively small, for strong couplings.

Drawing these separate topics together, the desire is to generalize the methods developed in this thesis, broadening the applicable problem class. Furthermore, the creation of performance prediction tools that account for update sequence and cooperation would enable a designer to, a priori, answer the question of how cooperation and computation should be used to provide best performance.
Bibliography


226


227


