Robust Constrained Model Predictive Control

by

Arthur George Richards

Master of Science
Massachusetts Institute of Technology, 2002
Master of Engineering
University of Cambridge, 2000

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2005

Author .....
Department of Aeronautics and Astronautics
November 22, 2004

Accepted by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Robust Constrained Model Predictive Control

by

Arthur George Richards

Accepted by .....
Jonathan P. How
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Eric M. Feron
Associate Professor of Aeronautics and Astronautics

Accepted by .....
John J. Deyst Jr.
Associate Professor of Aeronautics and Astronautics

Accepted by .....
Dr. Jorge Tierno
ALPHATECH Inc.

Robust Constrained Model Predictive Control

by

Arthur George Richards

Submitted to the Department of Aeronautics and Astronautics
on November 22, 2004, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis extends Model Predictive Control (MPC) for constrained linear systems subject to uncertainty, including persistent disturbances, estimation error and the effects of delay. Previous work has shown that feasibility and constraint satisfaction can be guaranteed by tightening the constraints in a suitable, monotonic sequence. This thesis extends that work in several ways, including more flexible constraint tightening, applied within the prediction horizon, and more general terminal constraints, applied to ensure feasible evolution beyond the horizon. These modifications reduce the conservatism associated with the constraint tightening approach.

Modifications to account for estimation error, enabling output feedback control, are presented, and we show that the effects of time delay can be handled in a similar manner. A further extension combines robust MPC with a novel uncertainty estimation algorithm, providing an adaptive MPC that adjusts the optimization constraints to suit the level of uncertainty detected. This adaptive control replaces the need for accurate *a priori* knowledge of uncertainty bounds. An approximate algorithm is developed for the prediction of the closed-loop performance using the new robust MPC formulation, enabling rapid trade studies on the effect of controller parameters.

The constraint tightening concept is applied to develop a novel algorithm for Decentralized MPC (DMPC) for teams of cooperating subsystems with coupled constraints. The centralized MPC optimization is divided into smaller subproblems, each solving for the future actions of a single subsystem. Each subproblem is solved only once per time step, without iteration, and is guaranteed to be feasible. Simulation examples involving multiple Uninhabited Aerial Vehicles (UAVs) demonstrate that the new DMPC algorithm offers significant computational improvement compared to its centralized counterpart.

The controllers developed in this thesis are demonstrated throughout in simulated examples related to vehicle control. Also, some of the controllers have been implemented on vehicle testbeds to verify their operation. The tools developed in this thesis improve the applicability of MPC to problems involving uncertainty and high complexity, for example, the control of a team of cooperating UAVs.

Acknowledgements

I'd like to thank my advisor, Prof. Jonathan How, for all his guidance, support and for letting me loose on some interesting problems. I also thank my committee members, Prof. Eric Feron, Prof. John Deyst and Dr. Jorge Tierno for their input and oversight, and Prof. Steve Hall for participating in the defense.

Many fellow students have shaped my MIT experience, at work and beyond, including the members of the Aerospace Controls Lab, the Space Systems Lab and the various students of the Laboratory for Random Graduate Study in Room 409. Special thanks to the other founding members of Jon How's MIT group for making settling in such a pleasure and for their continuing friendship. Yoshiaki Kuwata and Nick Pohlman were good enough to read this hefty document and help its preparation with their feedback. Simon Nolet generously gave much time and effort to help me with the SPHERES experiments.

I thank my mother for her constant support throughout this endeavour and all that came before. Finally, I dedicate this thesis to two other family members who both had a hand in this somehow.

For AER and GTB

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis makes extensions to the theory of Model Predictive Control (MPC) to suit high-performance problems combining uncertainty, constraints and multiple agents. In particular, the thesis considers the challenges of *robustness*, guaranteeing constraint satisfaction, and *scalability* of computation with the number of agents.

This introduction begins with a motivation in Section 1.1, defining the class of problems to be considered, presenting examples of such problems from the field of aerospace, and outlining the challenges to be addressed. Section 1.2 gives more background information on the core technologies employed in the thesis: robust MPC and decentralized control. Finally, Section 1.3 presents an outline of the thesis layout, including the contributions of each chapter.

1.1 Motivation

Fig. 1-1 shows two applications from the aerospace sector for the types of controller developed in this thesis. Fig. 1-1(a) shows a Predator Uninhabited Aerial Vehicle (UAV) [1]. Future USAF concepts involve teams of UAVs acting cooperatively to achieve mission goals in high risk environments. Fig. 1-1(b) shows an artist's impression of a formation-flying interferometer, part of NASA's Terrestrial Planet Finder (TPF) mission [2]. By combining light collected from different spacecraft, detailed, high-resolution information on distant objects can be determined, with the

(a) Predator UAV (b) Terrestrial Planet Finder

Figure 1-1: Applications

aim of idenitified habitable planets around other stars. This requires the relative positions of the spacecraft to be controlled accurately.

With these applications in mind, this thesis considers the problem of controlling linear systems with the following four features, which define a broad class of problems including the examples in Fig. 1-1:

- uncertainty;
- constraints;
- coupled subsystems;
- high performance objectives.

Table 1.1 illustrates these features by example, identifying where they arise in the examples shown in Fig. 1-1. The core of our approach is to apply Model Predictive Control (MPC) [3, 4, 5] to these problems. MPC is a feedback scheme using online solutions to optimizations, and therefore is a natural choice for problems involving both hard constraints and a performance metric to be somehow optimized. However, the uncertainty and coupled subsystems pose challenges to the resulting MPC controller: robustness and scalability. This thesis addresses these two issues, each of which is described in more detail in the following subsections.

Table 1.1: Features of Problems of Interest

Feature	UAV	Spacecraft Formation
Uncertainty	Wind disturbance GPS position error	Ranging measurement error State estimation error Solar or atmospheric drag Unmodeled gravitational effects
Constraints	Speed limit Turn rate limit Collision avoidance	Thruster force limit Relative position requirements Plume impingement avoidance
Coupled subsystems	UAVs in a team	Spacecraft in a formation
Performance objective	Minimize time Minimize risk	Minimize fuel use Minimize control energy

1.1.1 Robust feasibility and constraint satisfaction

The combination of uncertainty and constraints make robustness a key concern. As its name implies, Model Predictive Control uses predictions of future behavior to make control decisions. If those predictions are uncertain, care must be taken to ensure that the control actions do not lead to constraint violations. Also, since the controller depends on solving a constrained numerical optimization, it is necessary to ensure that a solution to the optimization always exists. For example, consider an avoidance problem where an aircraft is required to remain outside a specified region. Nominal MPC (*i.e.* not considering uncertainty), based on optimization, would generate a plan that just touches the edge of that region on its way past, and only the slightest disturbance is needed to push the aircraft into the avoidance region.

1.1.2 Decentralization/Scalability

Centralized control of a team of vehicles leads to a single computation which often scales poorly with the size of the team. Also, it is not readily compatible with exploiting the attractive features of a team approach, such as the ability to add members and reconfigure to accommodate failures. Cooperative team operation requires a dis-

tributed, or *decentralized*, controller in which decision making is shared among the team. The challenge then is to ensure that decisions made by the different planning agents are consistent with the rest of the team's (coupled) goals and constraints. This challenge is increased by the presence of uncertainty, making predictions of the actions of other team members uncertain.

1.2 Background

This section describes existing work in the relevant areas and identifies the open issues in the application of MPC to the class of problems of interest.

1.2.1 Model Predictive Control

Model Predictive Control (MPC) [3, 4, 5, 6] has been widely adopted in the field of process control, and continual improvement in computer technology mow makes it a promising technology for many aerospace applications [7, 8, 9, 10]. It uses the online solution of a numerical optimization problem, generating plans for future actions, and can readily accommodate hard constraints, such as relative position tolerances (or "error boxes") in spacecraft formation flight [8] and collision avoidance in UAV guidance [10]. Because MPC explicitly considers the operating constraints, it can operate closer to hard constraint boundaries than traditional control schemes. Stability and results for constrained MPC are well-addressed by existing work (see, for example, Refs. [15, 16] and the comprehensive survey in Ref. [11]).

Robustness has also received much attention and remains an active research area. The recent survey in Ref. [18] gives a more thorough list of work, and this section highlights some of the different approaches. Some approaches consider robustness *analysis* of nominal MPC formulations [19, 20], while others propose various schemes for the *synthesis* of explicitly robust schemes. Uncertainty is considered as either plant variability [21, 22, 23, 24, 25], affine disturbances [26, 27, 20, 17], state knowledge uncertainty [41, 42, 43, 44, 45], or an effect of delay [40]. The resulting optimization problems typically involve Linear Matrix Inequalities [26, 23, 25],

dynamic programming [24], or linear programming (LP), and/or quadratic programming (QP) [21, 22, 27, 12]. The QP or LP forms can involve min-max optimizations [12] or be embedded in other parameter searches [27].

A key challenge of designing a robust MPC algorithm is to achieve the desired robustness guarantee without either being too conservative or making the computation significantly more complex. Nominal MPC typically involves no more than QP or LP optimization. The approach of constraint-tightening (sometimes called "constraint restriction") was first suggested by Gossner *et al.* [13] and generalized by Chisci *et al.* [14]. The key idea is to achieve robustness by tightening the constraints in a monotonic sequence, retaining margin for future feedback action that becomes available to the MPC optimization as time progresses. This approach has the significant benefit that the complexity of the optimization is unchanged from the nominal problem by the robustness modifications. Only nominal predictions are required, avoiding both the large growth in problem size associated with incorporating multivariable uncertainty in the prediction model and the conservatism associated with worst case cost predictions, which is a common alternative, *e.g.* [12]. Some authors suggest that this approach leads to conservatism in terms of feasibility, *i.e.* that the tightened constraints unduly restrict the set of feasible initial conditions [4]. This thesis includes an analysis of this issue and shows that constraint tightening need not be conservative.

1.2.2 Decentralized Control

Decentralized control (not necessarily MPC) has been investigated for applications in complex and distributed systems, *e.g.* [62], investigating robust stability of the interconnections of systems with local controllers. Attention has recently focused on decentralized MPC [9], with various approaches, including robustness to the actions of others [51, 52], penalty functions [53, 54], and partial grouping of computations [55]. The open question addressed by this thesis is how to guarantee robust constraint satisfaction.

1.3 Outline and Summary of Contributions

- Chapter 2 provides two new formulations for robust MPC of constrained linear systems, guaranteeing feasibility and constraint satisfaction under the assumption of an unknown but bounded, persistent disturbance. We extend the work of Chisci *et al.* [14], in the area of robustness via constraint tightening, by generalizing the candidate control policies used to determine uncertainty margin. This extension leads to a less constrained optimization and hence a less conservative controller. Furthermore, we do not place any restrictions on the cost function, allowing the designer to pick a form that is consistent with the overall system objectives. Numerical examples are presented, comparing the feasible set for the new MPC controllers with the maximal feasible set for *any* robust controller. These results show that the new formulations alleviate the problems of conservatism reported in Ref. [4] for the constraint tightening approach.

  The second part of Chapter 2 considers transient problems, such as maneuvering problems, where the objective is to enter a given target set in finite time. However, these target sets are rarely invariant, so the approach of the first part cannot be adapted to suit these problems. We combine the constraint tightening ideas from the first part with the variable horizon approach of Scokaert and Mayne [12], providing a control that guarantees finite-time entry of an arbitrary target set, as well as feasibility and constraint satisfaction. Simulation examples are provided showing the variable horizon MPC controlling UAVs for collision avoidance.

- Chapter 3 also has two parts. The first extends the fixed-horizon MPC from Chapter 2 to account for inaccurate state information by applying the method of Bertsekas and Rhodes [39] to convert the problem to an equivalent case with perfect information. A suitable estimator is proposed for an output-feedback MPC using this approach. Analysis of the method for accommodating time delay in [40] is also presented to show that it can be cast as a problem with estimation error and thus handled in the same framework.

In the second part, an algorithm is presented to estimate uncertainty bounds, required by the robust controllers, from online measurements. This algorithm is combined with robust MPC to provide an adaptive MPC in which the uncertainty model, and hence also the controller, are updated as more information is acquired.

- Chapter 4 presents a novel algorithm for decentralized MPC, using the constraint tightening ideas from Chapter 2. The algorithm guarantees feasibility and satisfaction of coupled constraints for a team of subsystems, each subject to disturbances. The controllers for each subsystem exchange information via communication, but each plans only for its own actions, giving a more scalable control scheme than a centralized planning process. The method is demonstrated in simulation for control of multiple UAVs subject to collision avoidance constraints. The new algorithm is shown to offer significant computational benefits over centralized MPC with very little degradation of performance, here in terms of UAV flight time. Extensions of the approach are presented to account for delay in computation and communication.

- Chapter 5 develops an approximate algorithm for predicting the performance of the robust MPC algorithm presented in Chapter 2, in terms of the RMS of an output chosen by the designer. This enables rapid trade studies on the effects of controller settings, such as constraint limits, horizon length and expected disturbance level, and of environmental parameters such as actual disturbance level and sensing noise.

- Chapter 6 presents results of hardware experiments, employing the controllers from the previous chapters to control vehicle testbeds. The robust MPC from Chapter 2, with some of the extensions for delay and estimation error from Chapter 3, is employed for station-keeping of a spacecraft simulator on an air table and for the rendezvous of two rovers. The decentralized MPC from Chapter 4 is demonstrated for the control of multiple rovers subject to collision avoidance constraints.

Chapter 2

Robust MPC

The *constraint-tightening* approach to robust Model Predictive Control of constrained linear systems is extended. Previous work has shown that by monotonically tightening the constraints in the optimization, robust feasibility can be guaranteed for a constrained linear system subject to persistent, unknown but bounded disturbances. This robust feasibility follows from the form of the constraints: it does not depend on the choice of cost function or on finding the optimal solution to each problem, and the modification to account for robustness does not increase the complexity of the optimization.

This chapter presents several extensions to the constraint-tightening approach, including more flexible constraint tightening, applied within the prediction horizon, and more general terminal constaints, applied to represent the future beyond the horizon. These modifications reduce the conservatism associated with the constraint tightening approach.

2.1 Introduction

This chapter extends previous work on robust Model Predictive Control (MPC) in which robustness is achieved by only tightening the constraints in the optimization [13, 14]. MPC is a feedback scheme in which an optimal control problem is solved at each time step [3]. The first control input of the optimal sequence is applied and the

optimization is repeated at each subsequent time step. Since optimization techniques naturally handle hard constraints, this enables a controller to be explicitly aware of the constraints and operate close to, but within, their boundaries. However, when uncertainty is present, the state evolution will not match the prediction, so it is important to ensure that the optimization at each time step will be feasible. The work in this chapter extends the approach of constraint tightening [13, 14], modifying only the constraints of the optimization to achieve robustness. The extensions presented modify the work in Ref. [14] to enlarge the set of feasible initial conditions by allowing more flexible terminal constraints and more flexibility in constraint tightening.

Results concerning the nominal stability of MPC are well-established [15, 16, 11], but robustness remains the subject of considerable attention, as discussed in the recent survey [18]. Some approaches consider robustness *analysis* of nominal MPC formulations [19, 20], while others propose various schemes for the *synthesis* of explicitly robust schemes. Uncertainty is considered as either plant variability [21, 22, 23, 24, 25] or affine disturbances [26, 27, 20, 17]. The resulting optimization problems typically involve Linear Matrix Inequalities [26, 23, 25], dynamic programming [24], or linear and/or quadratic programming [21, 22, 27, 12]. The latter can involve min-max optimizations [12] or be embedded in outer parameter searches [27].

Gossner *et al.* [13] introduced a synthesis method for SISO systems with input limits in which robustness was achieved by tightening the constraints in a monotonic sequence. This was extended to the multivariable case by Chisci *et al.* [14]. The key idea is to to retain a "margin" for future feedback action, which becomes available to the MPC optimization as time progresses. Since robustness follows only from the constraint modifications, only nominal predictions are required, avoiding both the large growth in problem size associated with incorporating multivariable uncertainty in the prediction model and the conservatism associated with worst case cost predictions, which is a common alternative, *e.g.* [12]. Constraint tightening is based on a choice of candidate feedback policy used to determine the amount of margin to be retained. Gossner [13] required a nilpotent policy for this purpose and Chisci *et al.* [14] use a stabilizing LTI state feedback.

This chapter presents two formulations that extend the work of Chisci *et al.* [14]. The first formulation generalizes the candidate control policy used to perform constraint tightening, leading to a less constrained optimization and hence a less conservative controller. The generalization uses a time-varying control law, in place of a constant law, and derives its terminal constraint set from the *maximal robust control invariant set* [28], which can be found using some recently-developed tools [31, 28]. Furthermore, we do not place any restrictions on the cost function, allowing the designer to pick a form that is consistent with the overall system objectives.

The second MPC formulation removes the requirement for the terminal constraint set to be invariant. This controller uses the variable horizon approach of Scokaert and Mayne [12], providing a control that guarantees finite-time entry of an arbitrary target set, as well as feasibility and constraint satisfaction. The variable horizon formulation is particularly applicable to transient control problems, such as vehicle maneuvering. An example application of this controller would be a spacecraft rendezvous problem. The controller is required to maneuver the spacecraft, using limited thrusters and obeying sensor visibility constraints, into a region in which docking can occur. This target set could also involve relative velocity requirements for docking. Once the target set is reached, the work of the controller is done: it is not necessary for the controller to make the target set invariant.

Section 2.2 presents a detailed problem statement. Section 2.3 presents the fixed horizon MPC formulation and proves its robust feasibility. Section 2.4 provides a robust covergence result for the fixed-horizon form. Section 2.5 presents some numerical examples of the fixed-horizon robust MPC in simulation and a verification of robustness for particular problems using set visualization. Section 2.6 presents the variable-horizon form of the controller and proves its properties. Section 2.7 shows aircraft and spacecraft control examples using variable-horizon MPC, including some with non-convex constraints.

2.2 Problem Statement

The aim is to control a linear system with discretized dynamics

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{w}(k) \tag{2.1}$$

where $\mathbf{x}(k) \in \Re^{N_x}$ is the state vector, $\mathbf{u}(k) \in \Re^{N_u}$ is the input, $\mathbf{w}(k) \in \Re^{N_x}$ is the disturbance vector. Assume the system $(\mathbf{A}, \mathbf{B})$ is controllable and the complete state $\mathbf{x}$ is accessible. The disturbance lies in a bounded set but is otherwise unknown

$$\forall k \ \mathbf{w}(k) \in \mathcal{W} \subset \Re^{N_x} \tag{2.2}$$

The assumptions that the complete state $\mathbf{x}$ and the bound $\mathcal{W}$ are known are relaxed in Chapter 3, where the problem is extended to an output feedback case and to derive the uncertainty bound from online measurements.

The control is required to keep an output $\mathbf{y}(k) \in \Re^{N_y}$ within a bounded set for all disturbances. The form of the output constraints

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \tag{2.3}$$

$$\mathbf{y}(k) \in \mathcal{Y} \subset \Re^{N_y}, \ \forall k \tag{2.4}$$

can capture both input and state constraints, or mixtures thereof, such as limited control magnitude or error box limits [28]. The matrices $\mathbf{C}$ and $\mathbf{D}$ and the set $\mathcal{Y}$ are all chosen by the designer. The objective is to minimize the cost function

$$J = \sum_{k=0}^{\infty} \ell\left(\mathbf{u}(k), \mathbf{x}(k)\right) \tag{2.5}$$

where $\ell(\cdot)$ is a stage cost function. Typically, this would be a quadratic function, resulting in a quadratic program solution, or a convex piecewise linear function (*e.g.* $|\mathbf{u}| + |\mathbf{x}|$) that can be implemented with slack variables in a linear program [29]. The robust feasibility results require no assumptions concerning the nature of this

cost, but assumptions are added in Section 2.4 to enable the robust convergence and completion results.

Definition (Robust Feasibility) Assume that the MPC optimization problem can be solved from the initial state of the system. Then the system is *robustly-feasible* if, for all future disturbances $\mathbf{w}(k) \in \mathcal{W}\ \forall k \geq 0$, the optimization problem can be solved at every subsequent step.

Remark 2.1. (Robust Feasibility and Constraint Satisfaction) Robust feasibility is a stronger condition than robust constraint satisfaction, *i.e.* the satisfaction of the constraints at all times for all disturbances within the chosen bound. Feasibility requires constraint satisfaction as the constraints are applied to the first step of the plan, hence robust feasibility implies robust constraint satisfaction. However, in some cases there may be states which satisfy the constraints but from which no feasible plan can be found, *e.g.* if a vehicle is going so fast it will violate its position constraint at the following time step. The formulations in this thesis all provide robust feasibility and therefore robust constraint satisfaction as well.

2.3 Robustly-Feasible MPC

The online optimization approximates the complete problem in Section 2.2 by solving it over a finite horizon of N steps. A control-invariant terminal set constraint is applied to ensure stability [11]. Predictions are made using the nominal system model, *i.e.* (2.1) and (2.3) without the disturbance term. The output constraints are tightened using the method presented in [14], retaining a margin based on a particular candidate policy.

Optimization Problem Define the MPC optimization problem $\mathsf{P}(\mathbf{x}(k), \mathcal{Y}, \mathcal{W})$

$$J^*(\mathbf{x}(k)) = \min_{\mathbf{u},\mathbf{x},\mathbf{y}} \sum_{j=0}^{N} \ell\left(\mathbf{u}(k+j|k), \mathbf{x}(k+j|k)\right) \tag{2.6}$$

subject to $\forall j \in \{0 \dots N\}$

$$\begin{aligned}
\mathbf{x}(k+j+1|k) &= \mathbf{A}\mathbf{x}(k+j|k) + \mathbf{B}\mathbf{u}(k+j|k) && \text{(2.7a)}\\
\mathbf{y}(k+j|k) &= \mathbf{C}\mathbf{x}(k+j|k) + \mathbf{D}\mathbf{u}(k+j|k) && \text{(2.7b)}\\
\mathbf{x}(k|k) &= \mathbf{x}(k) && \text{(2.7c)}\\
\mathbf{x}(k+N+1|k) &\in \mathcal{X}_F && \text{(2.7d)}\\
\mathbf{y}(k+j|k) &\in \mathcal{Y}(j) && \text{(2.7e)}
\end{aligned}$$

where the double indices $(k+j|k)$ denote the prediction made at time k of a value at time $k+j$. The constraint sets are defined by the following recursion, which is the extension of the constraint form in Ref. [14] to include a time-varying $\mathbf{K}$

$$\begin{aligned}
\mathcal{Y}(0) &= \mathcal{Y} && \text{(2.8a)}\\
\mathcal{Y}(j+1) &= \mathcal{Y}(j) \sim (\mathbf{C} + \mathbf{D}\mathbf{K}(j))\,\mathbf{L}(j)\mathcal{W}, \quad \forall j \in \{0 \dots N-1\} && \text{(2.8b)}
\end{aligned}$$

where $\mathbf{L}(j)$ is defined as the state transition matrix for the closed-loop system under a candidate control law $\mathbf{u}(j) = \mathbf{K}(j)\mathbf{x}(j)$ $j \in \{0 \dots N-1\}$

$$\begin{aligned}
\mathbf{L}(0) &= \mathbf{I} && \text{(2.9a)}\\
\mathbf{L}(j+1) &= (\mathbf{A} + \mathbf{B}\mathbf{K}(j))\,\mathbf{L}(j), \quad \forall j \in \{0 \dots N-1\} && \text{(2.9b)}
\end{aligned}$$

Remark 2.6 describes a method to compare different choices of $\mathbf{K}(j)$, and Remark 2.8 discusses the particular significance of choosing $\mathbf{K}(j)$ to render the system nilpotent. The operator '$\sim$' denotes the Pontryagin difference [28]

$$\mathcal{A} \sim \mathcal{B} \stackrel{\triangle}{=} \{\mathbf{a} \mid \mathbf{a} + \mathbf{b} \in \mathcal{A},\ \forall \mathbf{b} \in \mathcal{B}\} \tag{2.10}$$

This definition leads to the following important property, which will be used in the proof of robust feasibility

$$\mathbf{a} \in (\mathcal{A} \sim \mathcal{B}), \mathbf{b} \in \mathcal{B} \Rightarrow (\mathbf{a} + \mathbf{b}) \in \mathcal{A} \tag{2.11}$$

The matrix mapping of a set is defined such that

$$\mathbf{A}\mathcal{X} \triangleq \{\mathbf{z} \mid \exists \mathbf{x} \in \mathcal{X} : \mathbf{z} = \mathbf{A}\mathbf{x}\} \tag{2.12}$$

A Matlab$^{\text{TM}}$ toolbox for performing the necessary operations, in particular the calculation of the Pontryagin difference, for polyhedral sets is available [30]. The calculation of the sets in (2.8) can be done offline.

There is some flexibility in the choice of terminal constraint set $\mathcal{X}_F$. It is found by the following Pontryagin difference

$$\mathcal{X}_F = \mathcal{R} \sim \mathbf{L}(N)\mathcal{W} \tag{2.13}$$

where $\mathcal{R}$ is a robust control invariant admissible set [31] *i.e.* there exists a control law $\kappa(\mathbf{x})$ satisfying the following

$$\forall \mathbf{x} \in \mathcal{R} \quad \mathbf{A}\mathbf{x} + \mathbf{B}\kappa(\mathbf{x}) + \mathbf{L}(N)\mathbf{w} \in \mathcal{R}, \ \forall \mathbf{w} \in \mathcal{W} \tag{2.14a}$$

$$\mathbf{C}\mathbf{x} + \mathbf{D}\kappa(\mathbf{x}) \in \mathcal{Y}(N) \tag{2.14b}$$

Remark 2.2. (Choice of Terminal Constraint Set) A variety of methods can be employed to calculate a suitable terminal set $\mathcal{X}_F$, based on identifying different sets $\mathcal{R}$ satisfying (2.14). For the greatest feasible set, use the Maximal Robust Control Invariant set [31], the largest set for which conditions (2.14) can be satisfied by any nonlinear feedback κ. If it is desired to converge to a particular target control, use the Maximal Robust Output Admissible set [28], the largest set satisfying (2.14) for a particular choice of controller $\kappa(\mathbf{x}) = \mathbf{K}\mathbf{x}$. Remark 2.8 discusses additional possibilities that arise if $\mathbf{K}(j)$ is chosen to render the system nilpotent.

Remark 2.3. (Comparison with Ref. [14]) In Ref. [14], the terminal constraint set is required to be invariant under the same constant, stabilizing LTI control law used to perform the constraint tightening. This is equivalent to restricting $\kappa(\mathbf{x}) = \mathbf{K}\mathbf{x}$ and $\mathbf{K}(j) = \mathbf{K}$ for some stabilizing $\mathbf{K}$.

This completes the description of the optimization problem and its additional design parameters. The following algorithm summarizes its implementation.

Algorithm 2.1. (Robustly Feasible MPC)

1. Solve problem $\mathsf{P}(\mathbf{x}(k), \mathcal{Y}, \mathcal{W})$
2. Apply control $\mathbf{u}(k) = \mathbf{u}^*(k|k)$ from the optimal sequence
3. Increment k. Go to Step 1

Theorem 2.1. ***(Robust Feasibility)*** *If* $\mathsf{P}(\mathbf{x}(0), \mathcal{Y}, \mathcal{W})$ *has a feasible solution then the system (2.1), subjected to disturbances obeying (2.2) and controlled using Algorithm 2.1, is robustly-feasible.*

Proof: It is sufficient to prove that if at time k_0 the problem $\mathsf{P}(\mathbf{x}(k_0), \mathcal{Y}, \mathcal{W})$ is feasible for some state $\mathbf{x}(k_0)$ and control $\mathbf{u}^*(k_0|k_0)$ is applied then the next problem $\mathsf{P}(\mathbf{x}(k_0+1), \mathcal{Y}, \mathcal{W})$ is feasible for all disturbances $\mathbf{w}(k_0) \in \mathcal{W}$. The theorem then follows by recursion: if feasibility at time k_0 implies feasibility at time k_0+1 then feasibility at time 0 implies feasibility at all subsequent steps. A feasible solution is assumed for time k_0 and used to construct a candidate solution for time k_0+1, which is then shown to satisfy the constraints for time k_0+1.

Assume $\mathsf{P}(\mathbf{x}(k_0), \mathcal{Y}, \mathcal{W})$ is feasible. Then it has a feasible (not necessarily optimal) solution, denoted by *, with states $\mathbf{x}^*(k_0+j|k_0),\ j \in \{0, \ldots, N+1\}$, inputs $\mathbf{u}^*(k_0+j|k_0),\ j \in \{0, \ldots, N\}$ and outputs $\mathbf{y}^*(k_0+j|k_0),\ j \in \{0, \ldots, N\}$ satisfying all of the constraints (2.7). Now, to prove feasibility of the subsequent optimization, a candidate solution is constructed and then shown to satisfy the constraints. Consider the following candidate solution, denoted by $\hat{\cdot}$, for problem $\mathsf{P}(\mathbf{x}(k_0+1), \mathcal{Y}, \mathcal{W})$

$$\begin{aligned} \hat{\mathbf{u}}(k_0+j+1|k_0+1) &= \mathbf{u}^*(k_0+j+1|k_0) \\ &+ \mathbf{K}(j)\mathbf{L}(j)\mathbf{w}(k_0), \quad \forall j \in \{0 \ldots N-1\} \end{aligned} \tag{2.15a}$$

$$\hat{\mathbf{u}}(k_0+N+1|k_0+1) = \kappa\left(\hat{\mathbf{x}}(k_0+N+1|k_0+1)\right) \tag{2.15b}$$

$$\hat{\mathbf{x}}(k_0+j+1|k_0+1) = \mathbf{x}^*(k_0+j+1|k_0) \tag{2.15c}$$

$$
\begin{aligned}
& + \mathbf{L}(j)\mathbf{w}(k_0), \quad \forall j \in \{0 \dots N\} \\
\hat{\mathbf{x}}(k_0+N+2|k_0+1) &= \mathbf{A}\hat{\mathbf{x}}(k_0+N+1|k_0+1) \qquad (2.15\text{d}) \\
& + \mathbf{B}\kappa(\hat{\mathbf{x}}(k_0+N+1|k_0+1)) \\
\hat{\mathbf{y}}(k_0+j+1|k_0+1) &= \mathbf{C}\hat{\mathbf{x}}(k_0+j+1|k_0+1) \qquad (2.15\text{e}) \\
& + \mathbf{D}\hat{\mathbf{u}}(k_0+j+1|k_0+1), \quad \forall j \in \{0 \dots N\}
\end{aligned}
$$

This solution is formed by shifting the previous solution by one step, *i.e.* removing the first step, adding one step of the invariant control law κ at the end, and adding perturbations representing the rejection of the disturbance by the candidate controller $\mathbf{K}$ and the associated state transition matrices $\mathbf{L}$. The construction of this candidate solution also illustrates the generalization compared to Ref. [14]: the controller $\mathbf{K}(j)$ may be time varying and the terminal law $\kappa(\mathbf{x})$ is a general nonlinear law, known to exist, according to (2.14) but not necessarily known explicitly.

To prove that the candidate solution in (2.15) is feasible, it is necessary to show that it satisfies all of the constraints (2.7a)–(2.7e) at timestep $k = k_0 + 1$ for any disturbance $\mathbf{w}(k_0) \in \mathcal{W}$.

Dynamics constraints (2.7a): Since the previous plan satisfied (2.7a), we know

$$
\mathbf{x}^*(k_0+j+2|k_0) = \mathbf{A}\mathbf{x}^*(k_0+j+1|k_0) + \mathbf{B}\mathbf{u}^*(k_0+j+1|k_0), \ \forall j \in \{0 \dots N-1\} \quad (2.16)
$$

Substituting on the both sides for $\mathbf{x}^*$ and $\mathbf{u}^*$ from the definitions of the candidate solution (2.15a) and (2.15c)

$$
\begin{aligned}
[\hat{\mathbf{x}}(k_0+j+2|k_0+1) - \mathbf{L}(j+1)\mathbf{w}(k_0)] &= \mathbf{A}[\hat{\mathbf{x}}(k_0+j+1|k_0+1) - \mathbf{L}(j)\mathbf{w}(k_0)] \\
& + \mathbf{B}[\hat{\mathbf{u}}(k_0+j+1|k_0+1) - \mathbf{K}(j)\mathbf{L}(j)\mathbf{w}(k_0)] \\
&= \mathbf{A}\hat{\mathbf{x}}(k_0+j+1|k_0+1) \qquad (2.17) \\
& + \mathbf{B}\hat{\mathbf{u}}(k_0+j+1|k_0+1) \\
& - (\mathbf{A}+\mathbf{B}\mathbf{K}(j))\mathbf{L}(j)\mathbf{w}(k_0)
\end{aligned}
$$

Then using (2.9b), the definition of the state transition matrices $\mathbf{L}$, the last term on each side cancels leaving

$$\hat{\mathbf{x}}(k_0+j+2|k_0+1) = \mathbf{A}\hat{\mathbf{x}}(k_0+j+1|k_0+1)+\mathbf{B}\hat{\mathbf{u}}(k_0+j+1|k_0+1),\ \forall j \in \{0\dots N-1\} \tag{2.18}$$

identical to the dynamics constraint (2.7a) for steps $j \in \{0\dots N-1\}$. The final step of the candidate plan (2.15b) and (2.15d) satisfy the dynamics model by construction, hence (2.7a) is satisfied for steps $j \in \{0\dots N\}$.

System constraints (2.7b): the candidate outputs (2.15e) are constructed using the output constraints (2.7b) and therefore satisfy them by construction.

Initial constraint (2.7c): The true state at time k_0+1 is found by applying control $\mathbf{u}^*(k_0|k_0)$ and disturbance $\mathbf{w}(k_0)$ to the dynamics (2.1)

$$\mathbf{x}(k_0+1) = \mathbf{A}\mathbf{x}(k_0) + \mathbf{B}\mathbf{u}^*(k_0|k_0) + \mathbf{w}(k_0) \tag{2.19}$$

Compare this equation with the constraints (2.7a) for step $j=0$ at time $k=k_0$

$$\begin{aligned}\mathbf{x}^*(k_0+1|k_0) &= \mathbf{A}\mathbf{x}(k_0|k_0) + \mathbf{B}\mathbf{u}^*(k_0|k_0)\\ &= \mathbf{A}\mathbf{x}(k_0) + \mathbf{B}\mathbf{u}^*(k_0|k_0)\end{aligned} \tag{2.20}$$

Then subtracting (2.20) from (2.19) shows that the new state can be expressed as a perturbation from the planned next state

$$\mathbf{x}(k_0+1) = \mathbf{x}^*(k_0+1|k_0) + \mathbf{w}(k_0) \tag{2.21}$$

Substituting $\mathbf{L}(0) = \mathbf{I}$ from (2.9a) into (2.15c) with $j=0$ gives

$$\hat{\mathbf{x}}(k_0+1|k_0+1) = \mathbf{x}^*(k_0+1|k_0) + \mathbf{w}(k_0) \tag{2.22}$$

From (2.21) and (2.22) we see that $\hat{\mathbf{x}}(k_0+1|k_0+1) = \mathbf{x}(k_0+1)$, satisfying (2.7c).

Terminal constraint (2.7d): substituting into (2.15c) for $j = N$ gives

$$\hat{\mathbf{x}}(k_0 + N + 1|k_0 + 1) = \mathbf{x}^*(k_0 + N + 1|k_0) + \mathbf{L}(N)\mathbf{w}(k)$$

Feasibility at time k_0 requires $\mathbf{x}^*(k_0+N+1|k_0) \in \mathcal{X}_F$ according to (2.7d) and therefore, using the property 2.11 of the Pontryagin difference (2.13) defining $\mathcal{X}_F$, we have

$$\hat{\mathbf{x}}(k_0 + N + 1|k_0 + 1) \in \mathcal{R} \tag{2.23}$$

The invariance condition (2.14a) ensures

$$\mathbf{A}\hat{\mathbf{x}}(k_0 + N + 1|k_0 + 1) + \mathbf{B}\kappa(\hat{\mathbf{x}}(k_0 + N + 1|k_0 + 1)) + \mathbf{L}(N)\mathbf{w} \in \mathcal{R},\ \forall \mathbf{w} \in \mathcal{W} \tag{2.24}$$

which from (2.15d) implies

$$\hat{\mathbf{x}}(k_0 + N + 2|k_0 + 1) + \mathbf{L}(N)\mathbf{w} \in \mathcal{R},\ \forall \mathbf{w} \in \mathcal{W} \tag{2.25}$$

Using the definition of the Pontryagin difference 2.10 and the terminal set (2.13) this shows

$$\hat{\mathbf{x}}(k_0 + N + 2|k_0 + 1) \in \mathcal{X}_F \tag{2.26}$$

which satisfies the terminal constraint (2.7d) for time $k = k_0 + 1$.

Output constraints (2.7e): Begin by testing steps $j = 0 \dots N-1$ of the candidate solution. Substituting the state and control perturbations (2.15c) and (2.15a) into the output definition (2.15e) gives

$$\hat{\mathbf{y}}((k_0+1)+j|k_0+1) = \mathbf{y}^*(k_0+(j+1)|k_0)+(\mathbf{C}+\mathbf{D}\mathbf{K}(j))\mathbf{L}(j)\mathbf{w}(k_0),\ \ \forall j \in \{0 \dots N-1\} \tag{2.27}$$

Also, given feasibility at time k_0, we know

$$\mathbf{y}^*(k_0 + (j + 1)|k_0) \in \mathcal{Y}(j + 1),\ \forall j \in \{0 \dots N - 1\} \tag{2.28}$$

Recall the definition of the constraint sets (2.8b)

$$\mathcal{Y}(j+1) = \mathcal{Y}(j) \sim (\mathbf{C} + \mathbf{DK}(j))\,\mathbf{L}(j)\mathcal{W}, \quad \forall j \in \{0 \dots N-1\}$$

Now the property of the Pontryagin difference (2.11) can be applied to (2.27) and (2.28)

$$\begin{gathered}\mathbf{y}^*(k_0+j+1|k_0) \in \mathcal{Y}(j+1) \Rightarrow \\ \hat{\mathbf{y}}(k_0+j+1|k_0+1) = \mathbf{y}^*(k_0+j+1|k_0) + (\mathbf{C}+\mathbf{DK}(j))\mathbf{L}(j)\mathbf{w}(k_0) \in \mathcal{Y}(j), \\ \forall \mathbf{w}(k_0) \in \mathcal{W},\ \forall j \in \{0 \dots N-1\}\end{gathered}$$

proving that the first N steps of the candidate outputs (2.15e) satisfy the constraints (2.7e) at time $k = k_0 + 1$. Also, it follows from (2.23) and the admissibility requirement (2.14b) that the final control step using $\kappa(\hat{\mathbf{x}}(k+N+1|k+1))$ is admissible for set $\mathcal{Y}(N)$ according to (2.14b).

Having shown that the candidate solution (2.15) satisfies the constraints at time k_0+1, given that a solution for time k_0 exists, then feasibility at time k_0 must imply feasibility at time $k_0 + 1$, and hence at all future times, by recursion. □

Fig. 2-1 illustrates for a simple example how the constraint tightening in (2.8b) allows margin for feedback compensation in response to the disturbance. Here, an output y is constrained to be less than or equal to $Y_{\max}$ at all times. A two-step nilpotent control policy has been chosen as $\mathbf{K}(j)$ and the plan constraints are tightened after the first and second steps accordingly. At time k, the planned output signal $y(\cdot|k)$ must remain beneath the constraint line marked for time k. Suppose the plan at time k is right against this boundary. The first step is executed and the same optimization is solved at time $k+1$, but now subject to the constraint boundary marked for time $k+1$. The upward arrows show the margin returned to the optimization at step $k+1$: the nilpotent control policy can be added to the previous plan (the line of constraint for time k) resulting in a new plan that would satisfy the new constraints *i.e.* below the $k+1$ constraint line. The downward arrows show the margin retained for later steps.

The decision variables for the robust problem are a single sequence of controls,

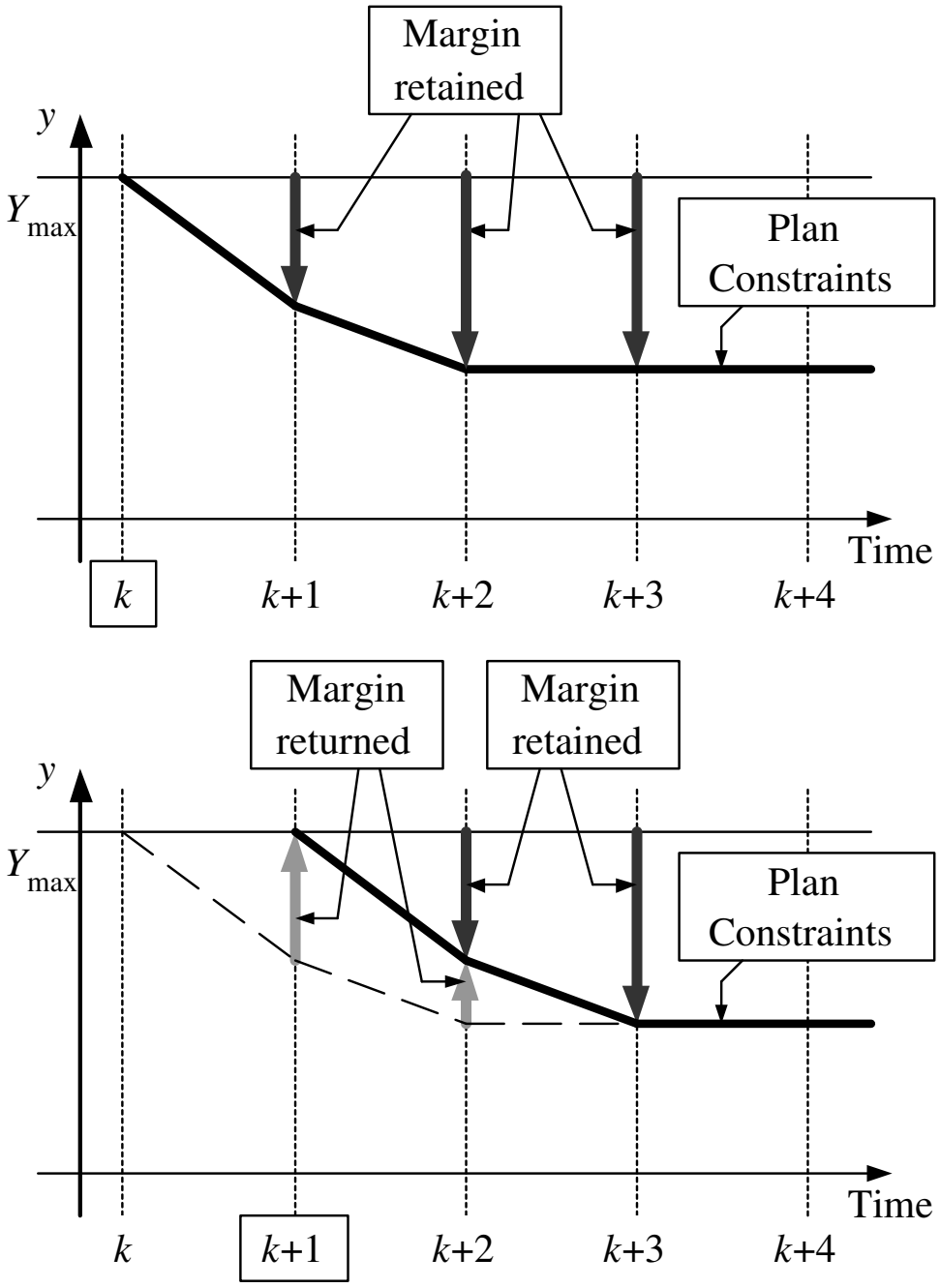


Fig. 2-1: Illustration of Constraint Tightening for Robustness

states and outputs extending over the horizon. The number of variables grows linearly with horizon length. If the original constraint set $\mathcal{Y}$ and the disturbance set $\mathcal{W}$ are polyhedra then the modified constraint sets $\mathcal{Y}(j)$ are also polyhedra [28, 31] and the number of constraints required to specify each set $\mathcal{Y}(j)$ is the same as that required to specify the nominal set $\mathcal{Y}$ [31]. Therefore the type and size of the optimization is unchanged by the constraint modifications. For example, if the nominal optimal control problem, ignoring uncertainty, is a linear program (LP), the robust MPC optimization is also an LP of the same size.

Remark 2.4. (Anytime Computation) The result in Theorem 2.1 depends only on finding *feasible* solutions to each optimization, not necessarily *optimal* solutions. Furthermore, the robust feasibility result follows from the ability to construct a feasible solution to each problem before starting the optimization. This brings two advantages, first that this solution may be used to initialize a search procedure, and second that the optimization can be terminated at any time knowing that a feasible solution exists and can be used for control. Then, for example, an interior point method [33] can be employed, initialized with the feasible candidate solution.

Remark 2.5. (Reparameterizing the Problem) Some optimal control approaches [32] simplify the computation by reparameterizing the control in terms of basis functions, reducing the number of degrees of freedom involved. It is possible to combine this basis function approach with constrait tightening for robustness as developed in this Chapter. Robust feasibility of the reparameterized optimization can be assured by expressing the control sequence as a perturbation from the candidate solution. This is equivalent to substituting the following expression for the control sequence

$$\mathbf{u}(k+j|k) = \hat{\mathbf{u}}(k+j|k) + \sum_{n=1}^{N_b} \alpha_n b_n(j)$$

where $\hat{\mathbf{u}}(k+j|k)$ is the candidate solution from (2.15), $b_n(j)$ denotes a set of N_b basis functions, defined over the horizon $j = 0 \ldots N$ and chosen by the designer, and α_n are the new decision variables in the problem. Since the candidate solution is known to be feasible, $\alpha_n = 0 \ \forall n$ is a feasible solution to the revised problem.

Remark 2.6. (Comparing Candidate Policies) The choice of the candidate control policy $\mathbf{K}$ is an important parameter in the robustness scheme. One way of comparing possible choices of $\mathbf{K}$ is to evaluate the largest disturbance that can be accommodated by the scheme using that candidate policy. Assuming the disturbance set (2.2) is a mapped unit hypercube $\mathcal{G} = \{\mathbf{v} \mid \|\mathbf{v}\|_\infty \leq 1\}$ with a variable scaling δ

$$\mathcal{W} = \delta \mathbf{E}\mathcal{G} \tag{2.29}$$

then it is possible to calculate the largest value of δ for which the set of feasible initial states is non-empty by finding the largest δ for which the smallest constraint set $\mathcal{Y}(N)$ is non-empty. Given the property (2.11) of the Pontryagin difference, this can be found by solving the following optimization

$$\begin{aligned} \delta_{\max} &= \max_{\delta,\mathbf{p}} \delta \\ \text{s.t.} \mathbf{p} + \delta\mathbf{q} &\in \mathcal{Y} \qquad \forall \mathbf{q} \in (\mathbf{C} + \mathbf{DK}(0))\mathbf{L}(0)\mathbf{E}\mathcal{G} \oplus \\ &\qquad\qquad (\mathbf{C} + \mathbf{DK}(1))\mathbf{L}(1)\mathbf{E}\mathcal{G} \oplus \ldots \oplus \\ &\qquad\qquad (\mathbf{C} + \mathbf{DK}(N-1))\mathbf{L}(N-1)\mathbf{E}\mathcal{G} \end{aligned} \tag{2.30}$$

where $\oplus$ denotes the Minkowski or vector sum of two sets. Since the vertices of $\mathcal{G}$ are known, the vertices of the Minkowski sum can be easily found as the sum of all cominbations of vertices of the individual sets [31]. The number of constraints can be large but this calculation is performed offline for controller analysis, so solution time is not critical. This calculation is demonstrated for the examples in Section 2.5 and shown to enable comparison of the "size" of the sets of feasible initial conditions when using different candidate controllers.

Remark 2.7. (Approximate Constraint Sets) In some cases it may be preferable to use approximations of the constraint sets (2.8) rather than calculating the Pontryagin difference exactly. The proof of robust feasibility in Theorem 1 depends on the property (2.11) of the Pontryagin difference. This property is a weaker condition than the definition of the Pontryagin difference (2.10), which is the *largest* set obeying (2.11). Therefore, the constraint set recursion in (2.8) can be replaced by

any sequence of sets for which the following property holds

$$\mathbf{y} + (\mathbf{C} + \mathbf{DK}(j))\,\mathbf{L}(j)\mathbf{w} \in \mathcal{Y}(j),\ \forall \mathbf{y} \in \mathcal{Y}(j+1),\ \forall \mathbf{w} \in \mathcal{W},\ \forall j \in \{0 \dots N-1\}$$

This test can be accomplished using norm bounds, for example.

Remark 2.8. (Significance of Nilpotency) If the candidate control is restricted to be nilpotent, then the final state transition matrix $\mathbf{L}(N) = \mathbf{0}$ according to (2.9). Referring back to the requirements of the terminal set (2.14a), (2.14b) and (2.13), this means that the set $\mathcal{R}$ can be a *nominally* control invariant set and $\mathcal{X}_F = \mathcal{R}$. Nominal control invariance is a weaker condition than robust control invariance, and in some cases, a nominally-invariant set can be found when no robustly-invariant set exists. For example, a nominal control invariant set can have no volume, but this is impossible for a robust control invariant set. In the spacecraft control example in Section 2.5.2, a coasting ellipse or "passive aperture" is a particularly attractive terminal set and is shown to offer good performance. However, a passive aperture is only a nominally invariant set, not robustly invariant, so the candidate control must be nilpotent if the passive aperture terminal constraint is used.

Remark 2.9. (Nilpotent Controllers) Constant nilpotent controllers can be synthesized by using pole-placement techniques to put the closed-loop poles at the origin. For a controllable system of order M, this generates a constant controller $\mathbf{K}(j) = \mathbf{K}\ \forall j$ that guarantees convergence of the state to the origin in at most N_x steps, *i.e.* $\mathbf{L}(N_x) = \mathbf{0}$. This satisfies the requirement that $\mathbf{L}(N) = \mathbf{0}$ provided $N \geq N_x$. Greater flexibility in the choice of $\mathbf{K}(j)$ can be achieved by using a time-varying controller. A finite-horizon (M-step) LQR policy with an infinite terminal cost is suitable, found by solving

$$\begin{aligned}
\mathbf{P}(M) &= \infty\mathbf{I} \\
\forall j \in \{1 \dots M\}\ \mathbf{P}(j-1) &= \mathbf{Q} + \mathbf{A}^T\mathbf{P}(j)\mathbf{A} - \mathbf{A}^T\mathbf{P}(j)\mathbf{B}(\mathbf{R} + \mathbf{B}^T\mathbf{P}(j)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}(j)\mathbf{A} \\
\mathbf{K}_L(j-1) &= -(\mathbf{R} + \mathbf{B}^T\mathbf{P}(j)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{P}(j)\mathbf{A}
\end{aligned}$$

Most generally, a general M-step nilpotent policy, applying control at some steps

$\mathcal{J} \subset \{0 \dots M-1\}$ with $N_x \leq M \leq N$, can be found by solving the state transition equation for the inputs $\mathbf{u}(j)$ $j \in \mathcal{J}$

$$\mathbf{0} = \mathbf{A}^M \mathbf{x}(0) + \sum_{j \in \mathcal{J}} \mathbf{A}^{(M-j-1)} \mathbf{B} \mathbf{u}(j) \tag{2.31}$$

If the system is controllable, this equation can be solved for a set of controls as a function of the initial state $\mathbf{u}(j) = \mathbf{H}(j)\mathbf{x}(0)$. This can be re-arranged into the desired form $\mathbf{u}(j) = \mathbf{K}(j)\mathbf{x}(j)$.

Proposition 2.1. ***(Repeating Trajectories as Terminal Constraints)*** *For any system, an admissible trajectory that repeats after some period, defined by a set*

$$\mathcal{X}_F = \{\mathbf{x} \mid \mathbf{A}^{N_R}\mathbf{x} = \mathbf{x},\ \mathbf{C}\mathbf{A}^j\mathbf{x} \in \mathcal{Y}(N),\ \forall j = 0 \dots (N_R - 1)\} \tag{2.32}$$

where N_R is the chosen period of repetition, is nominally control invariant, i.e. *satisfies (2.14b) and (2.14a), under the policy $\kappa(\mathbf{x}) = \mathbf{0}$ $\forall \mathbf{x}$*

Proof: see Section 2.A.1.

2.4 Robust Convergence

For some control problems, robust feasibility and constraint satisfaction are sufficient to meet the objectives. This section provides an additional, stronger result, showing convergence to a smaller bounded set within the constraints. A typical application would be dual-mode MPC [16], in which MPC is used to drive the state into a certain set, obeying constraints during this transient period, before switching to another, steady-state stabilizing controller.

The result in this section proves that the state is guaranteed to enter a set $\mathcal{X}_C$. This set is not the same as the terminal constraint set $\mathcal{X}_F$. Nor does the result guarantee that the set $\mathcal{X}_C$ in invariant, *i.e.* the state may leave this set, but it cannot remain outside it for all time. Some other forms of robust MPC *e.g.* [26, 12] guarantee invariance of their terminal constraint sets, but they have stronger restrictions on the

choice of terminal constraints. The variable-horizon MPC formulation in Section 2.6 also offers a robust set arrival result. In that case, the target set $\mathcal{Q}$ is chosen freely by the user, and not restricted to the form of $\mathcal{X}_C$ dictated below, but the variable horizon optimization is more complicated to implement.

First, an assumption on the form of the cost function is necessary. This is not restrictive in practice because the assumed form is a common standard.

Assumption (Stage Cost) The stage cost function $\ell(\cdot)$ in (2.6) is of the form

$$\ell(\mathbf{u}, \mathbf{x}) = \|\mathbf{u}\|_{\mathbf{R}} + \|\mathbf{x}\|_{\mathbf{Q}} \tag{2.33}$$

where $\|\cdot\|_{\mathbf{R}}$ and $\|\cdot\|_{\mathbf{Q}}$ denote weighted norms. Typical choices would be quadratic forms $\mathbf{u}^T\mathbf{R}\mathbf{u}$ or weighted one-norms $|\mathbf{R}\mathbf{u}|$.

Theorem 2.2. ***(Robust Convergence)*** *If* $\mathsf{P}(\mathbf{x}(0), \mathcal{Y}, \mathcal{W})$ *has a feasible solution, the optimal solution is found at each time step, and the objective function is of the form (2.33), then the state is guaranteed to enter the following set*

$$\mathcal{X}_C = \{\mathbf{x} \in \Re^{N_x} \mid \|\mathbf{x}\|_{\mathbf{Q}} \leq \alpha + \beta\} \tag{2.34}$$

where α *and* β *are given by*

$$\alpha = \max_{\mathbf{w} \in \mathcal{W}} \sum_{i=0}^{N} (\|\mathbf{K}(i)\mathbf{L}(i)\mathbf{w}\|_{\mathbf{R}} + \|\mathbf{L}(i)\mathbf{w}\|_{\mathbf{Q}}) \tag{2.35}$$

$$\beta = \max_{\mathbf{x} \in \mathcal{X}_F} (\|\mathbf{x}\|_{\mathbf{Q}} + \|\kappa(\mathbf{x})\|_{\mathbf{R}}) \tag{2.36}$$

These represent, respectively, the maximum cost of the correction policy for any disturbance $\mathbf{w} \in \mathcal{W}$ and the maximum cost of remaining in the terminal set for one step from any state $\mathbf{x} \in \mathcal{X}_F$. Note that the common, albeit restrictive, choice of the origin as the terminal set $\mathcal{X}_F = \{\mathbf{0}\}$ yields $\beta = 0$.

Proof: Theorem 2.1 showed that if a solution exists for problem $\mathsf{P}(\mathbf{x}(k_0), \mathcal{Y}, \mathcal{W})$ then a particular candidate solution (2.15) is feasible for the subsequent problem $\mathsf{P}(\mathbf{x}(k_0 + 1), \mathcal{Y}, \mathcal{W})$. The cost of this candidate solution $\hat{J}(\mathbf{x}(k_0 + 1))$ can be used to bound the

optimal cost at a particular time step $J^*(\mathbf{x}(k_0+1))$ relative to the optimal cost at the preceding time step $J^*(\mathbf{x}(k_0))$. A Lyapunov-like argument based on this function will then be used to prove entry to the target set $\mathcal{X}_C$. Since the theorem requires optimal solutions to be found at each step, redefine the superscript * to denote the optimal solution, giving the following cost for the optimal solution at time k_0

$$J^*(\mathbf{x}(k_0)) = \sum_{i=0}^{N} \{\|\mathbf{u}^*(k_0+j|k_0)\|_{\mathbf{R}} + \|\mathbf{x}^*(k_0+j|k_0)\|_{\mathbf{Q}}\} \tag{2.37}$$

and the cost of the candidate solution in (2.15)

$$\begin{aligned} \hat{J}(\mathbf{x}(k_0+1)) &= \sum_{j=0}^{N} \{\|\hat{\mathbf{u}}(k_0+j+1|k_0+1)\|_{\mathbf{R}} + \|\hat{\mathbf{x}}(k_0+j+1|k_0+1)\|_{\mathbf{Q}}\} && (2.38) \\ &= \sum_{j=0}^{N-1} \{\|\mathbf{u}^*(k_0+j+1|k_0) + \mathbf{K}(j)\mathbf{L}(j)\mathbf{w}(k_0)\|_{\mathbf{R}} + \\ &\qquad \|\mathbf{x}^*(k_0+j+1|k_0) + \mathbf{L}(j)\mathbf{w}(k_0)\|_{\mathbf{Q}}\} + && (2.39) \\ &\qquad \|\mathbf{x}^*(k_0+N+1|k_0)\|_{\mathbf{Q}} + \|\kappa(\mathbf{x}^*(k_0+N+1|k_0))\|_{\mathbf{R}} \end{aligned}$$

The cost of the candidate solution can be bounded from above using the triangle inequality, applied to each term in the summation

$$\begin{aligned} \hat{J}(\mathbf{x}(k_0+1)) &\leq \sum_{j=0}^{N-1} \{\|\mathbf{u}^*(k_0+j+1|k_0)\|_{\mathbf{R}} + \|\mathbf{K}(j)\mathbf{L}(j)\mathbf{w}(k_0)\|_{\mathbf{R}} + \\ &\qquad \|\mathbf{x}^*(k_0+j+1|k_0)\|_{\mathbf{Q}} + \|\mathbf{L}(j)\mathbf{w}(k_0)\|_{\mathbf{Q}}\} + && (2.40) \\ &\qquad \|\mathbf{x}(k_0+N+1|k_0)\|_{\mathbf{Q}} + \|\kappa(\mathbf{x}(k_0+N+1|k_0))\|_{\mathbf{R}} \end{aligned}$$

This bound can now be expressed in terms of the previous optimal cost in (2.37)

$$\begin{aligned} \hat{J}(\mathbf{x}(k_0+1)) &\leq J^*(\mathbf{x}(k_0)) - \|\mathbf{u}(k_0)\|_{\mathbf{R}} - \|\mathbf{x}(k_0)\|_{\mathbf{Q}} + \\ &\qquad \sum_{i=0}^{N-1} \{\|\mathbf{K}(i)\mathbf{L}(i)\mathbf{w}(k_0)\|_{\mathbf{R}} + \|\mathbf{L}(i)\mathbf{w}(k_0)\|_{\mathbf{Q}}\} + && (2.41) \\ &\qquad \|\mathbf{x}^*(k_0+N+1|k_0)\|_{\mathbf{Q}} + \|\kappa(\mathbf{x}^*(k_0+N+1|k_0))\|_{\mathbf{R}} \end{aligned}$$

The summation term is clearly bounded by the quantity α from (2.35). Also, since the constraints require $\mathbf{x}^*(k_0+N+1|k_0) \in \mathcal{X}_F$, the final two terms are bounded by the maximum (2.36). Finally, using the non-negativity of $\|\mathbf{u}(k_0)\|_{\mathbf{R}}$, (2.41) can be rewritten as

$$\hat{J}(\mathbf{x}(k_0+1)) \leq J^*(\mathbf{x}(k_0)) - \|\mathbf{x}(k_0)\|_{\mathbf{Q}} + \alpha + \beta \tag{2.42}$$

and since the cost of the candidate solution forms an upper bound on the optimal cost $J^*(\mathbf{x}(k_0+1)) \leq \hat{J}(\mathbf{x}(k_0+1))$, this implies that

$$J^*(\mathbf{x}(k_0+1)) - J^*(\mathbf{x}(k_0)) \leq -\|\mathbf{x}(k_0)\|_{\mathbf{Q}} + \alpha + \beta \tag{2.43}$$

Recall from (2.34) the definition of the convergence set $\mathcal{X}_C = \{\mathbf{x} \in \Re^{N_x} \mid \|\mathbf{x}\|_{\mathbf{Q}} \leq \alpha + \beta\}$. Therefore if the state lies outside this set, $\mathbf{x}(k_0) \notin \mathcal{X}_C$, the right-hand side of (2.43) is negative and the optimal cost is strictly decreasing. However, the optimal cost $J^*(\cdot)$ is bounded below, by construction, so the state $\mathbf{x}(k_0)$ cannot remain outside $\mathcal{X}_C$ forever and must enter $\mathcal{X}_C$ at some time. □

The smallest region of convergence $\mathcal{X}_C$ for a given system is attained by setting the target set $\mathcal{X}_F$ to be the origin, yielding $\beta = 0$, and weighting the state much higher than the control. This would be desirable for a scenario to get close to a particular target state. The set $\mathcal{X}_C$ becomes larger as the control weighting is increased or as the terminal set is enlarged. An example in the following section demonstrates the use of the convergence property for transient control subject to constraints using a dual-mode approach. A final property of the convergence set, that it can be no smaller than the uncertainty set, is captured in the following proposition.

Proposition 2.2. *The convergence set contains the disturbance set, $\mathcal{X}_C \supseteq \mathcal{W}$.*

Proof: See Section 2.A.2

2.5 Numerical Examples

The following subsections demonstrate the properties of the new formulation using numerical examples. The first uses a simple system in simulation to illustrate the

effectiveness of the robust controller in comparison to nominal MPC. The second example considers steady-state spacecraft control, showing in particular how the choice of terminal constraint sets can change performance. The third set of examples use set visualization to verify robust feasibility. The final example shows the exploitation of the convergence result to enlarge the region of attraction of a stabilizing control scheme, subject to constraints. In all examples, set calculations were performed using the Invariant Set Toolbox for Matlab$^{\text{TM}}$ [30].

2.5.1 Robust Feasibility Demonstration

This section shows a very simple example comparing nominal MPC with the robustly-feasible formulation of Section 2.3 in simulation, including random but bounded disturbances. The system is a point mass moving in one dimension with force actuation, of up to one unit in magnitude, and a disturbance force of up to 30% of the control mgnitude. The control is required to keep both the position and the velocity within ± 1 while minimizing the control energy. In the notation of Section 2.2

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \quad \mathcal{W} = \{\mathbf{w} = \mathbf{B}z \mid |z| \leq 0.3\} \\ \mathbf{C} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathcal{Y} = \{\mathbf{y} \in \Re^3 \mid \|\mathbf{y}\|_\infty \leq 1\} \end{aligned}$$

The horizon was set to $N = 10$ steps. The Matlab implementation of Ackermann's formula was used to design a nilpotent controller by placing both closed-loop poles at $s = 0$. The resulting candidate controller was

$$\mathbf{K} = [-1 \ -1.5]$$

Using this controller in the expression (2.8) for constraint tightening gave the following constraint sets

$$\begin{aligned}
\mathcal{Y}(0) &= \{\mathbf{y} \in \Re^3 \mid |y_1| \leq 1 \quad |y_2| \leq 1 \quad |y_3| \leq 1 \quad \} \\
\mathcal{Y}(1) &= \{\mathbf{y} \in \Re^3 \mid |y_1| \leq 0.85 \quad |y_2| \leq 0.7 \quad |y_3| \leq 0.4 \quad \} \\
\mathcal{Y}(j) &= \{\mathbf{y} \in \Re^3 \mid |y_1| \leq 0.7 \quad |y_2| \leq 0.4 \quad |y_3| \leq 0.1 \quad \}, \ \forall j = \{2 \dots 10\}
\end{aligned}$$

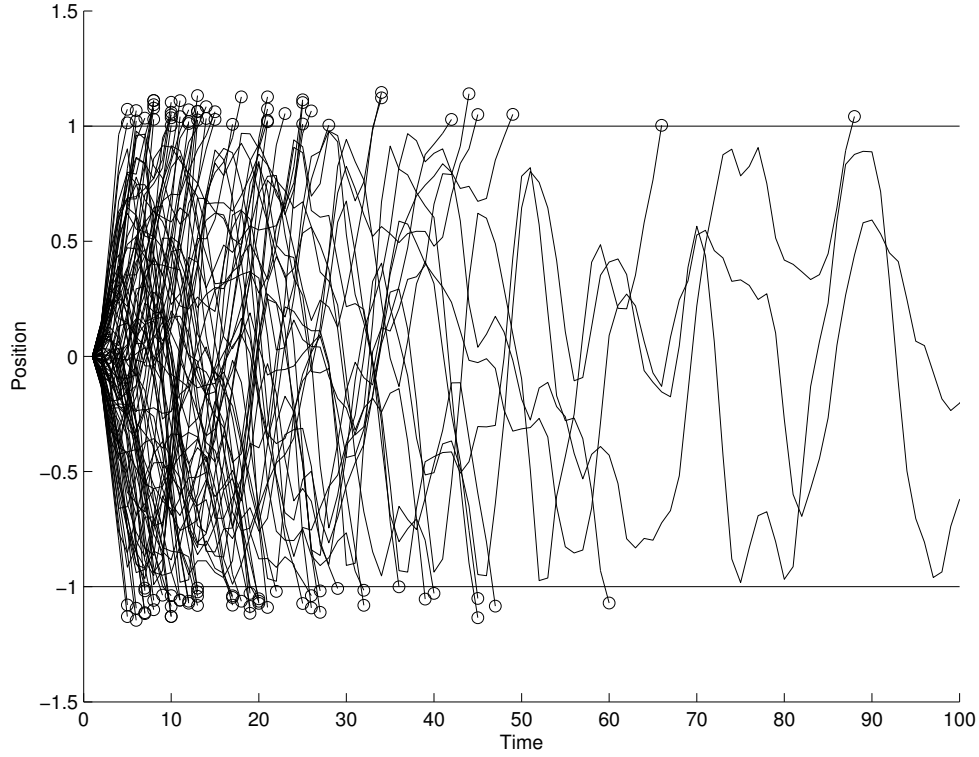
Note that since the controller is chosen to be nilpotent, the sets remain constant after a finite number of steps. The terminal constraint set was chosen to be the origin, $\mathcal{X}_F = \{\mathbf{0}\}$. The stage cost was quadratic, with the weighting heaviliy biased to the control

$$\ell(\mathbf{x}, \mathbf{u}) = \frac{1}{1000}\mathbf{x}^T\mathbf{x} + 100u^2$$

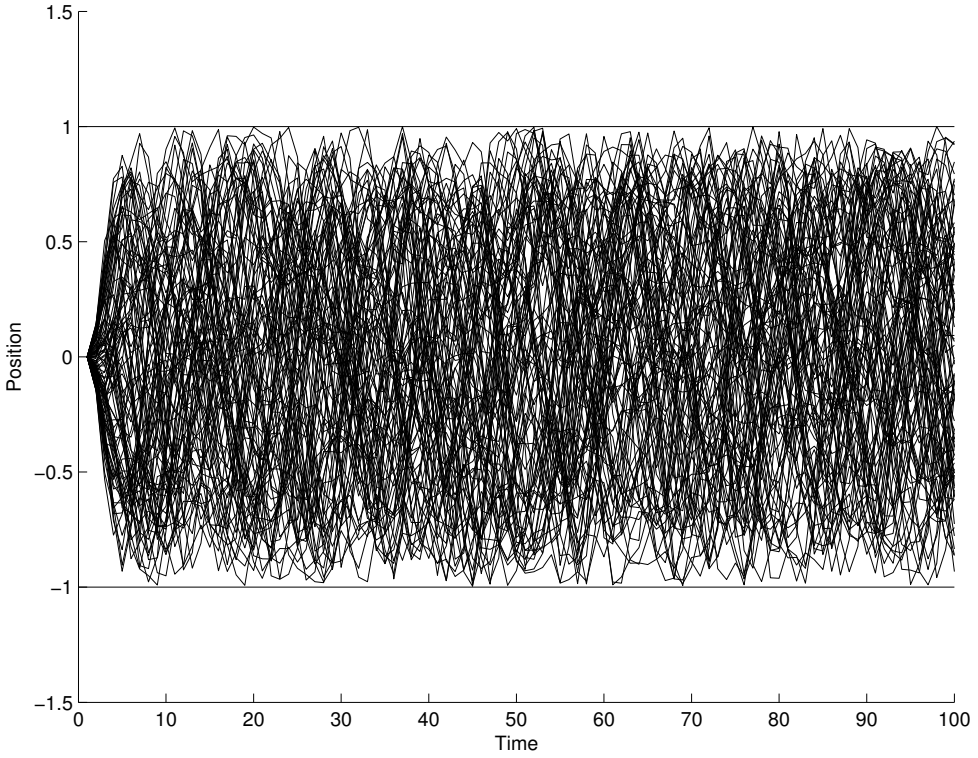
Fig. 2-2(a) shows the position time histories from 100 simulations, each with randomly-generated disturbances, using nominal MPC, *i.e.* without constraint tightening. The circles mark where a problem becomes infeasible. Of the 100 total simulations, 98 become infeasible before completion. Fig. 2-2(b) shows position time histories for a further 100 simulations using robustly-feasible MPC as in Section 2.3. Now all 100 simulations complete without infeasibility. Note that the mass goes all the way out to the position limits of ± 1, but never exceeds them. This is to be expected from a control-minimizing feedback law and shows that the controller is robust but still using all of the constraint space available, thus retaining the primary advantage of MPC.

2.5.2 Spacecraft Control

This section shows the application of robust MPC to precision spacecraft control. The scenario requires that a spacecraft remain within a 20m-sided cubic "error box" relative to a reference orbit using as little fuel as possible. In particular, this example is concerned with the choice of terminal constraint set for this scenario. Recall that the MPC optimization approximates the infinite horizon control problem by solving over a finite horizon and applying a terminal constraint to account for behavior beyond



(a) Nominal MPC



(b) Robust MPC

Figure 2-2: State Time Histories from 100 Simulations comparing Nominal MPC. 'o' denotes point at which problem becomes infeasible.

the end of the plan. Ideally, we expect the spacecraft to drift around inside the box on a trajecory close to a *passive aperture*, an elliptical path that repeats itself without any control action [34]. In this section, we demonstrate the use of a passive aperture as a terminal constraint for robust MPC. This is compared with a common alternative constraint, requiring nominal trajectory to end at the error box center. The latter is clearly a worse approximation as in practice the spacecraft will not go to the box center and remain there. The passive aperture constraint is shown to offer a significant performance benefit over the box center constraint.

As discussed in Remark 2.8, the robust MPC formulation in Section 2.3 requires only a *nominally* control invariant terminal constraint set, as opposed to *robust* control invariance, if the candidate controller is chosen to be nilpotent. The example in this section makes use of this property, as a passive aperture is a nominally invariant set but not robustly invariant, since it has no "volume".

The spacecraft MPC problem has been extensively studied for application to spacecraft formation flight [8] and can be posed in linear optimization using Hills equations [35] to model relative spacecraft motion. (Note that the numbers used in this example are not intended to accurately represent any particular spacecraft scenario, and better performance could probably be obtained by adjusting the horizon and time step lengths. However, the performance comparison between the two terminal constraints and the applicability of the method to spacecraft control are still demonstrated.) The continuous time equations of motion in state space form, under no external force other than gravity, are

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3\omega^2 & 0 & 0 & 0 & 2\omega & 0 \\ 0 & 0 & 0 & -2\omega & 0 & 0 \\ 0 & 0 & -\omega^2 & 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t) = \mathbf{A}_C \mathbf{x}(t) \tag{2.44}$$

where the state is defined in a rotating reference frame, relative to a point in a

circular reference orbit of angular rate ω. In this example, the angular rate was set to correspond to a 90 minute low-Earth orbit. The state elements are defined as

x_1	Radial displacement	x_4	Radial velocity
x_2	In $-$ track displacement	x_5	In $-$ track velocity
x_3	Out $-$ of $-$ plane displacement	x_6	Out $-$ of $-$ plane velocity

With some abuse of notation, define the discrete-time state $\mathbf{x}(k)$ as the continuous time state sampled at time interval of Δt *i.e.* $\mathbf{x}(k\Delta t)$. This example uses a time step of $\Delta t = 1$ minute and a planning horizon of 15 minutes or 1/6 of an orbit. The continuous time dynamics (2.44) can be discretized to give the form

$$\mathbf{x}(k+1) = \underbrace{e^{\mathbf{A}_C \Delta t}}_{\mathbf{A}} \mathbf{x}(k) + \underbrace{e^{\mathbf{A}_C \Delta t} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}}_{\mathbf{B}} \mathbf{u}(k) \tag{2.45}$$

where $\mathbf{u}(k)$ is the control input, defined to be an impulsive velocity change. The constraints limit the force magnitude and displacement (the "error box"), written in output form as

$$\mathbf{C} = \frac{1}{D_{\max}} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \qquad \mathbf{D} = \frac{1}{U_{\max}} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \qquad \mathcal{Y} = \{\mathbf{y} \in \Re^3 \mid \|\mathbf{y}\|_\infty \leq 1\}$$

where $D_{\max} = 10$ is the maximum displacement and $U_{\max} = 1$ is the maximum control. Constraint tightening was performed using a three step bang-off-bang policy as the candidate $\mathbf{K}$. The terminal constraint is that the spacecraft be on a passive aperture that lies entirely within the error box. This is written as two constraints

$$\mathbf{C}\mathbf{A}^j \mathbf{x}(k+N+1|k) \in \mathcal{Y}(N), \ \forall j \in \{1 \dots N_R\} \tag{2.46a}$$

$$\mathbf{A}^{N_R} \mathbf{x}(k+N+1|k) = \mathbf{x}(k+N+1|k) \tag{2.46b}$$

where $N_R = 90$ is the number of time steps corresponding to a complete orbit. The first constraint (2.46a) captures the requirement that the ellipse remains inside

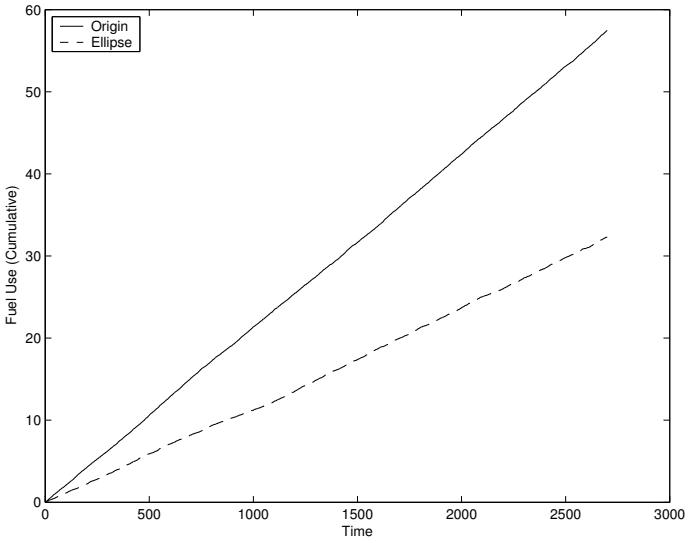
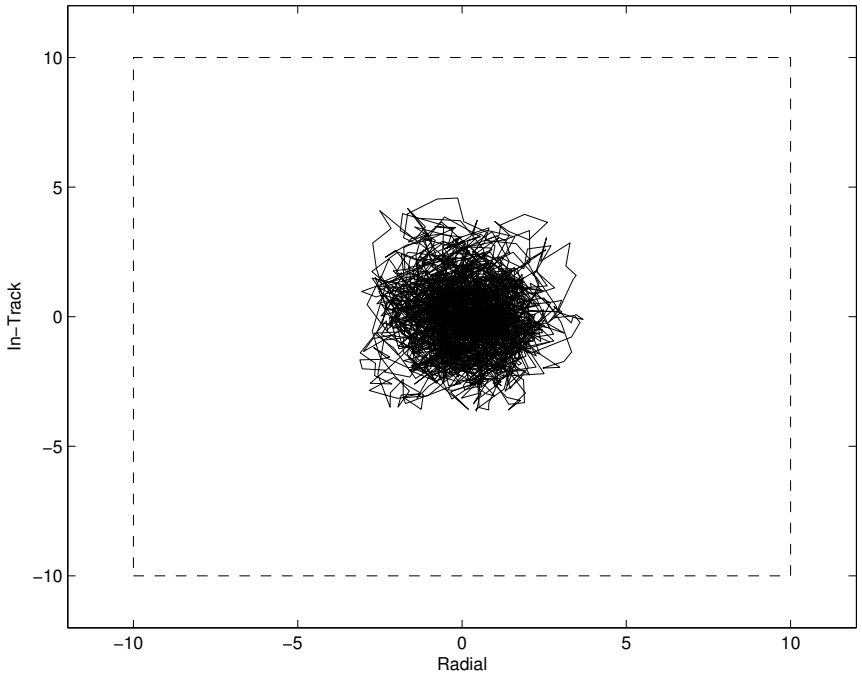


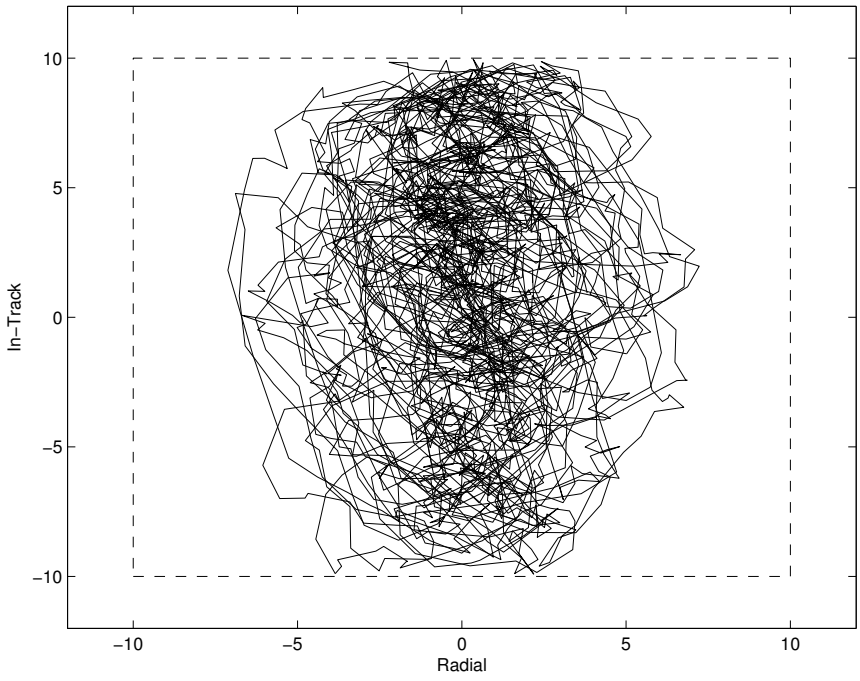
Figure 2-3: Fuel Use Comparison for Spacecraft MPC with Origin and Ellipse Terminal Constraints

the error box and the second constraint (2.46b) requires that the trajectory repeat after N_R steps without any control. This set is a suitable terminal set, *i.e.* it is control invariant, as shown by Proposition 2.1.

Fig. 2-4(a) shows, for comparison, the trajectory taken by the spacecraft, in simulation using the terminal constraint $\mathbf{x}(k+N+1|k) = \mathbf{0}$. The simulation included uniformly-distributed random disturbances of 1.5% of the control ΔV. The error box is shown dashed, and it is clear that the problem is artificially constrained close to its center by the terminal constraint. Fig. 2-4(b) shows the trajectory when the terminal constraints are the ellipse form (2.46). Fig. 2-3 shows the cumulative fuel use of the two controllers. The new terminal constraints yield a fuel use reduction of approximately 40%. Comparing Fig. 2-4(b) with Fig. 2-4(a), it is clear that the ellipse terminal conditions are much less constraining. Also, observe that the spacecraft motion in Fig. 2-4(b) is roughly within a 2-by-1 envelope, which is the proportion of the free-flying passive aperture. This illustrates that the spacecraft actual motion is similar to the approximation made beyond the horizon, *i.e.* that the vehicle would drift on an ellipse. It is logical that a better terminal approximation leads to better performance.



(a) Origin Terminal Constraint



(b) Ellipse Terminal Constraint

Figure 2-4: Trajectories of Spacecraft under MPC, comparing Origin and Ellipse Terminal Constraints

2.5.3 Graphical Proof of Robustness

This section demonstrates robust feasibility by calculating and comparing two sets. The first is the set of feasible initial conditions for the problem $\mathsf{P}(\mathbf{x}, \mathcal{Y}, \mathcal{W})$, denoted by $\mathcal{X}_{feas}$. The second is the *robust closed-loop reach set* $\mathcal{X}_{next}$, that is the set of states $\mathbf{x}(1) = \mathbf{A}\mathbf{x}(0) + \mathbf{B}\mathbf{u}(0) + \mathbf{w}(0)$ that can be reached in one step from all initial states in $\mathbf{x}(0) \in \mathcal{X}_{feas}$ under all possible disturbances in $\mathbf{w}(0) \in \mathcal{W}$ and all feasible controls $\mathbf{u}(0)$ generated by solving problem $\mathsf{P}(\mathbf{x}(0), \mathcal{Y}, \mathcal{W})$. Formally,

$$\mathcal{X}_{feas} = \{\mathbf{x}(k) \mid \exists(\mathbf{u}(k|k \dots k+N), \mathbf{x}(k|k \dots k+N+1), \mathbf{y}(k|k \dots k+N)) \\ \text{satisfying (2.7a)} - \text{(2.7e)}\}$$

$$\mathcal{X}_{next} = \{\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k|k) + \mathbf{w}(k) \mid \\ \mathbf{x}(k) \in \mathcal{X}_{feas}, \mathbf{u}(k|k) \text{ from a feasible soln. to } \mathsf{P}(\mathbf{x}(\mathrm{k}), \mathcal{Y}, \mathcal{W}), \mathbf{w}(k) \in \mathcal{W}\}$$

Then robust feasibility follows if $\mathcal{X}_{next} \subseteq \mathcal{X}_{feas}$ [31, 4]. This section will consider 2-D systems and graphically check this condition to demonstrate the robust feasibility of control by Algorithm 2.1, already proven analytically in Theorem 2.1.

If the constraint set $\mathcal{Y}$ and the terminal set $\mathcal{X}_F$ are polytopes, then the set $\mathcal{X}_{feas}$ can be found directly from the constraints of $\mathsf{P}(\mathbf{x}(0), \mathcal{Y}, \mathcal{W})$ by inequality projection. To calculate $\mathcal{X}_{next}$, it is first necessary to use a similar projection to find the set of all feasible control-state pairs

$$\mathcal{K}_{feas} = \{(\mathbf{x}(k)^T \mathbf{u}(k|k)^T)^T \mid \exists(\mathbf{u}(k|k+1 \dots k+N), \mathbf{x}(k|k \dots k+N+1), \mathbf{y}(k|k \dots k+N)) \\ \text{satisfying (2.7a)} - \text{(2.7e)}\}$$

Then the reach set can be found by mapping the set of control-state pairs and adding the effect of disturbance

$$\mathcal{X}_{next} = [\mathbf{A}\ \mathbf{B}]\mathcal{K}_{feas} \oplus \mathcal{W}$$

Each example uses one of the following system $\mathbf{A}$ matrices and a common input

$\mathbf{B}$ matrix

$$\mathbf{A}_1 = \begin{bmatrix} 0.9 & 0.2 \\ 0.0 & 0.8 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} 1.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \quad \mathbf{A}_3 = \begin{bmatrix} 1.1 & 1.0 \\ 0.0 & 1.3 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$

representing stable, neutrally-stable and unstable systems, respectively. The following limits define the constraints on the state and input

$$|x_1(k)| \leq 10 \quad |x_2(k)| \leq 10 \quad |u(k)| \leq 1$$

where $x_i(k)$ denotes element i of the state at time k. These are converted into output constraint form using the following system matrices and constraint set

$$\mathbf{C} = \begin{bmatrix} \frac{1}{10} & 0 \\ 0 & \frac{1}{10} \\ 0 & 0 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathcal{Y} = \{\mathbf{y} \in \Re^3 \mid \|\mathbf{y}\|_\infty \leq 1\}$$

The disturbance set is a simple square

$$\mathcal{W} = \{\mathbf{w} \mid \|\mathbf{w}\|_\infty \leq 0.1\}$$

In all examples, constraint tightening is performed using a constant candidate policy $\mathbf{K}(j) = \mathbf{K}_{LQR}(\mathbf{A}, \mathbf{B})\ \forall j$ (although the formulation permits $\mathbf{K}(j)$ to be time-varying) where $\mathbf{K}_{LQR}(\mathbf{A}, \mathbf{B})$ is the steady-state LQR controller for the relevant system with weightings $\mathbf{Q} = \mathbf{I}$ on the state and $\mathbf{R} = 5$ on the control. The horizon length was $N = 4$ steps.

For comparison, two forms of terminal constraints are employed, distinguished by the choice of the invariant set $\mathcal{R}$ in (2.14)

I. $\mathcal{R}$ is the maximal robust control invariant set admissible in $\mathcal{Y}(N)$ under disturbance $\mathbf{L}(N)\mathcal{W}$

II. $\mathcal{R}$ is the maximal robust output admissible set for the closed loop system $[\mathbf{A} + \mathbf{B}\mathbf{K}_{LQR}(\mathbf{A}, \mathbf{B})]$ for output set $\mathcal{Y}(N)$ (equivalent to the method of Chisci *et al.* [14])

Both of these sets can be calculated using the Invariant Set Toolbox for Matlab [30] and are consistent with the requirements for robust feasibility laid out in (2.14).

In Figs. 2-5–2-10, the lightly-shaded area is the set of feasible initial states $\mathcal{X}_{feas}$ and the darker shaded region is the reach set $\mathcal{X}_{next}$. It is clear that $\mathcal{X}_{next} \subseteq \mathcal{X}_{feas}$ in every case. The dash-dot line marks the terminal set $\mathcal{X}_F$, shown for illustration. The solid line, which frequently coincides with the edge of the shaded region, marks the boundary of the *maximal robust control invariant set* $\tilde{\mathcal{C}}_\infty$ [31] for the given constraints. This is the largest set within the constraints that can be made invariant, despite the action of the disturbance, by *any* feedback control policy. The feasible region of any robustly-feasible MPC scheme must be a subset of $\tilde{\mathcal{C}}_\infty$, which is therefore used here as a benchmark for feasibility of the new MPC formulation.

Comparing Fig. 2-5 with 2-6, Fig. 2-7 with 2-8 and Fig. 2-9 with 2-10, it is clear that the terminal constraints from Method A, based on the maximal robust invariant set, lead to larger terminal constraint sets and hence larger feasible region. This clearly illustrates the main contribution of this work: the larger feasible sets indicate a less constrained optimization and hence a less conservative control. Also, as the feasible sets in Figs. 2-5 and 2-7 fill the maximal robust invariant sets, indicating that the controller is no more conservative than robustness demands, and the feasible set in Fig. 2-9 covers nearly all of the maximal set, indicating that there is little room for improvement. These results suggest that constraint tightening need not be inherently conservative.

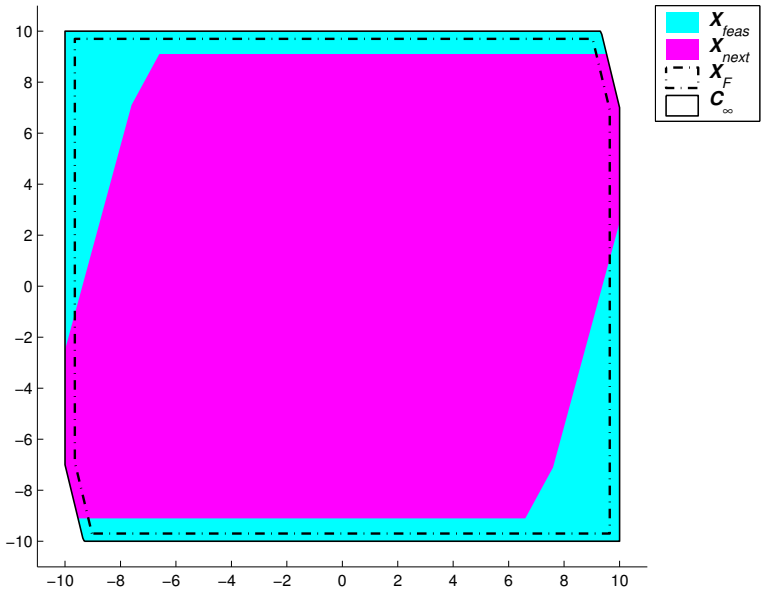


Figure 2-5: Comparison of Sets for Stable Example System $\mathbf{A}_1$ using Terminal Constraints I. Observe that $\mathcal{X}_{next} \subseteq \mathcal{X}_{feas}$, implying robust feasibility, and that $\mathcal{X}_{feas}$ fills $\tilde{\mathcal{C}}_\infty$ showing as good a region of feasibility as can be achieved by any controller.

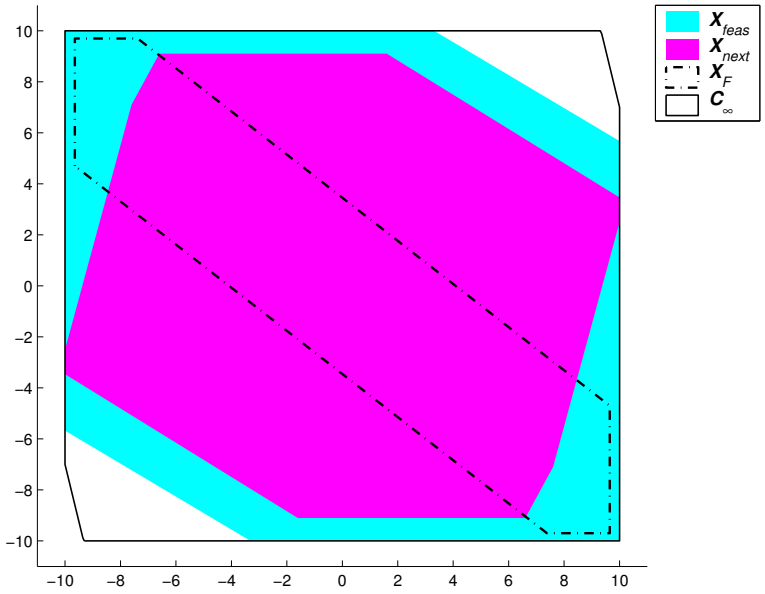


Figure 2-6: Comparison of Sets for Stable Example System $\mathbf{A}_1$ using Terminal Constraints II. Compare with Fig. 2-5: both the terminal constraint set and the feasible set are smaller with constraints II.

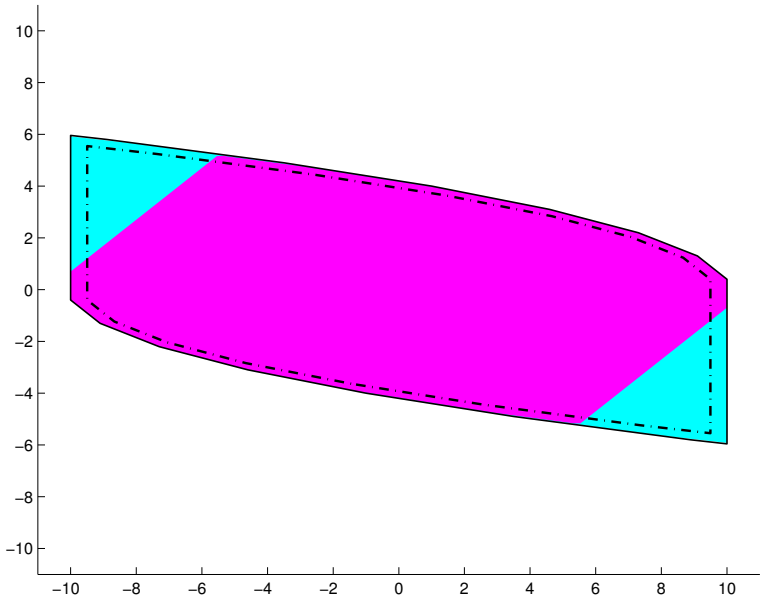


Figure 2-7: Comparison of Sets for Neutrally-Stable Example System $\mathbf{A}_2$ using Terminal Constraints I. Observe that $\mathcal{X}_{next} \subseteq \mathcal{X}_{feas}$, implying robust feasibility, and that $\mathcal{X}_{feas}$ fills $\tilde{\mathcal{C}}_\infty$ showing as good a region of feasibility as can be achieved by any controller.

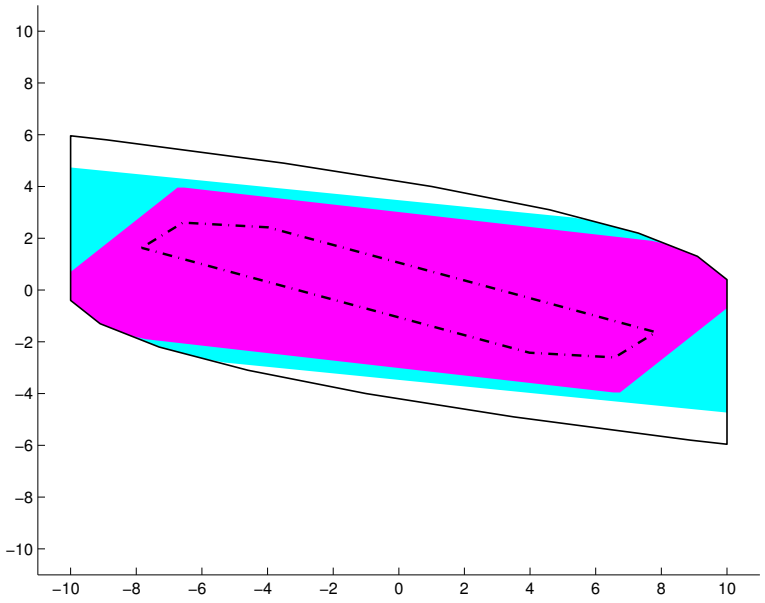


Figure 2-8: Comparison of Sets for Neutrally-Stable Example System $\mathbf{A}_2$ using Terminal Constraints II. Compare with Fig. 2-7: both the terminal constraint set and the feasible set are smaller with constraints II.

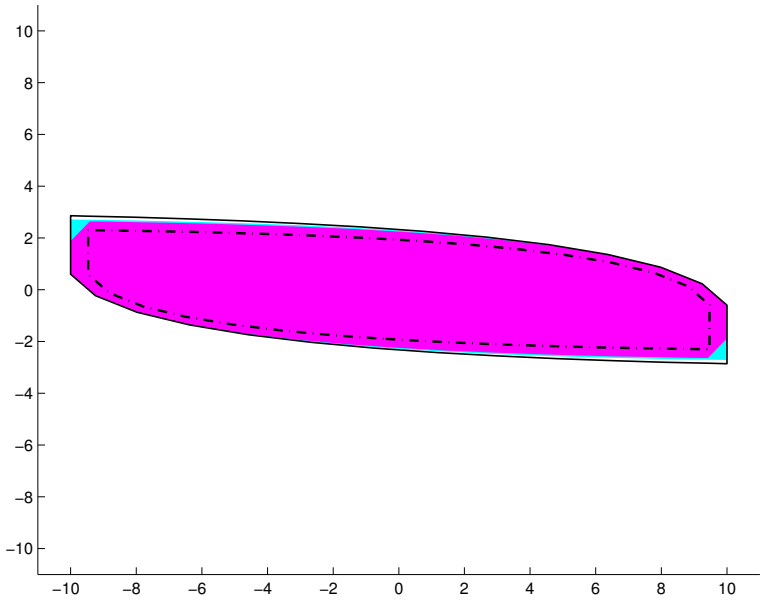


Figure 2-9: Comparison of Sets for Unstable Example System $\mathbf{A}_3$ using Terminal Constraints I

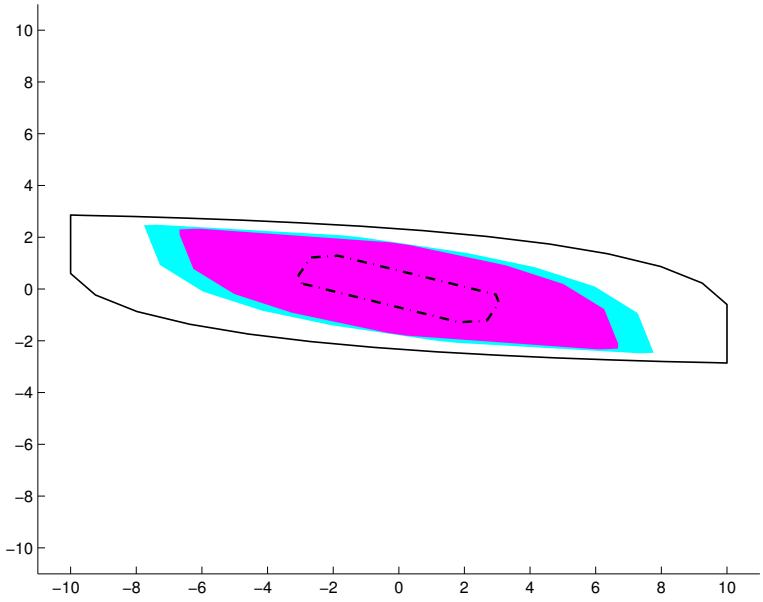


Figure 2-10: Comparison of Sets for Unstable Example System $\mathbf{A}_3$ using Terminal Constraints II. (Compare with Fig. 2-9)

2.5.4 Robust Convergence

This example demonstrates the robust convergence property for a double integrator system. The disturbance set and system dynamics are the same as used in the previous section, using the matrix $\mathbf{A}_2$. The constraints are

$$\begin{aligned} |x_1(k)| &\leq 10 \\ |x_2(k)| &\leq 1.2 \\ |u(k)| &\leq 1 \end{aligned}$$

A state feedback controller $\mathbf{K}_L$ has been designed to place both closed-loop poles at $s = -0.1$. It is desired to use this controller in steady state to stabilize the system about the origin. Fig. 2-11 shows the constraint set $\mathcal{X}_{cons}$ and the maximal robust positively-invariant set [31, 30] $\mathcal{X}_L$ within $\mathcal{X}_{cons}$ for the closed-loop system $\mathbf{A} + \mathbf{B}\mathbf{K}_L$. This is the largest set within the constraints such that, if the state starts within that set, it will remain inside for all future steps. Therefore, once the state is inside $\mathcal{X}_L$, the controller $\mathbf{K}_L$ can be used and the constraints will be satisfied. If the state is initially outside $\mathcal{X}_L$, use of the controller $\mathbf{K}_L$ does not guarantee satisfaction of the constraints. Therefore, another controller is required to steer the state into $\mathcal{X}_L$ while satisfying the constraints. Robust MPC is used for this purpose. The terminal set is the origin $\mathcal{X}_F = \{\mathbf{0}\}$ and the stage cost is

$$\ell(\mathbf{u}, \mathbf{x}) = |\mathbf{x}| + \frac{1}{100}|u|$$

Then the convergence target region $\mathcal{X}_C$ for the MPC controller, calculated using (2.34), is the diamond region shown by the dashed outline in Fig. 2-11. If the initial MPC optimization is feasible, the state is guaranteed to enter $\mathcal{X}_C \subseteq \mathcal{X}_L$. Example state trajectories obtained by simulation are shown in the figure. The overall scheme is a dual-mode controller [16]: if the state is inside $\mathcal{X}_L$, use controller $\mathbf{K}_L$, otherwise use MPC.

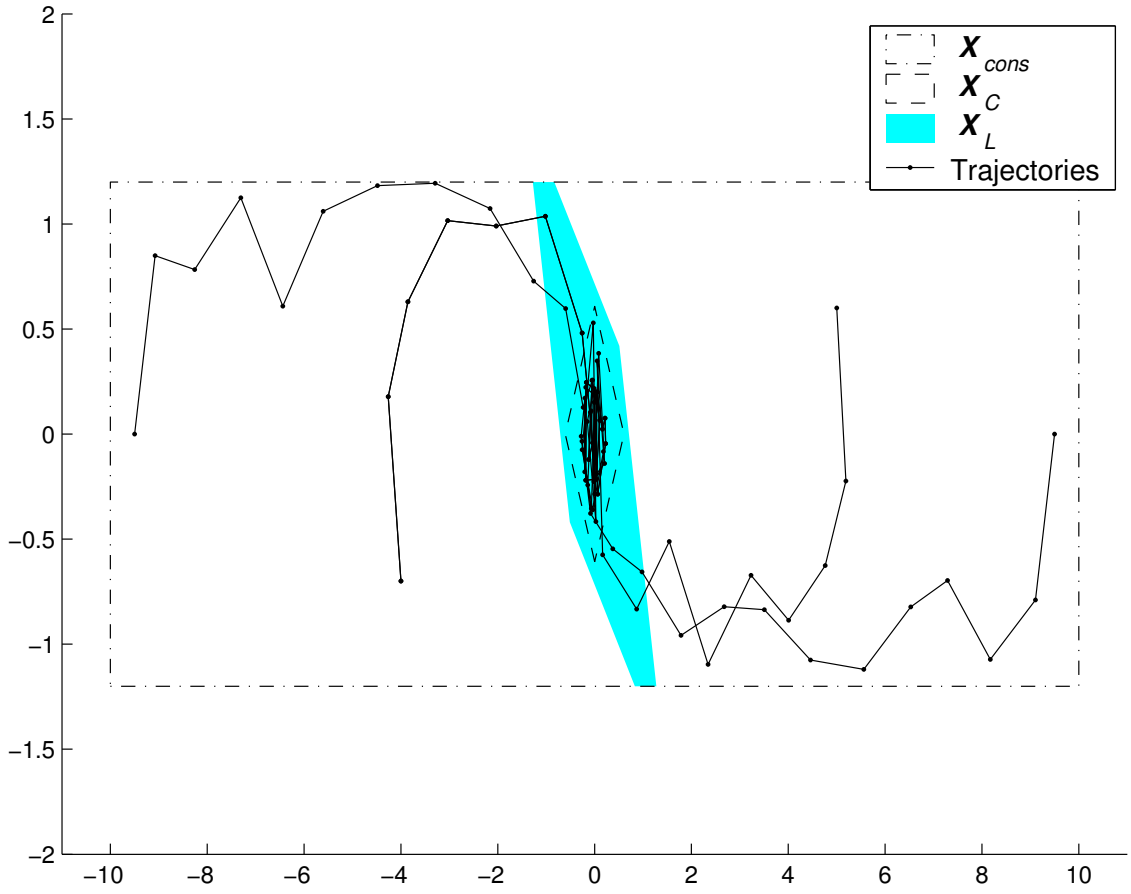


Figure 2-11: Phase-Plane Demonstration of Convergence for Example System, showing state constraint set $\mathcal{X}_{cons}$, MPC convergence set $\mathcal{X}_C$, region of operation for LTI controller $\mathcal{X}_L$ and four simulated state trajectories under MPC for different initial conditions. Observe that all trajectories remain in $\mathcal{X}_{cons}$ and reach $\mathcal{X}_C \subseteq \mathcal{X}_L$ so the LTI control can then be employed.

2.6 Variable Horizon MPC

This section combines the new robustness formulation from Section 2.3 with a variable horizon length [12] to provide a controller suitable transient control problems, such as finite-time maneuvering, with a predetermined target set. The signifcance of this result is that the target set need not be invariant, so the method of Section 2.3 cannot be applied directly. The variable-horizon MPC controller guarantees robust feasibility and finite-time entry of the target set. The distinction between this result and the convergence result in Section 2.4 is that variable-horizon MPC can use an arbitrary terminal set, whereas the convergence result for fixed-horizon MPC involved a norm-bounded set around the origin. The greater flexibility of the variable-horizon form comes at the expense of a more complicated computation.

An example application of this controller would be a spacecraft rendezvous problem. The controller is required to maneuver the spacecraft, using limited thrusters and obeying sensor visibility constraints, into a region in which docking can occur. This target set could also involve relative velocity requirements for docking. Once the target set is reached, the work of the controller is done: it is not necessary for the controller to make the target set invariant.

2.6.1 Problem Statement

The dynamics and constraints are the same as in Section 2.2. The designer specifies a predetermined terminal set $\mathcal{Q}$ and the objective is to minimize the arrival time N_a, with a weighting on control, to reach that set

$$\min J = \sum_{k=0}^{N_a} (1 + \gamma \|\mathbf{u}(k)\|) \tag{2.47}$$

subject to

$$\mathbf{x}(N_a) \in \mathcal{Q} \tag{2.48}$$

There are no restrictions on the nature of the terminal set $\mathcal{Q}$.

2.6.2 Variable Horizon Controller

Define the MPC optimization problem $\mathsf{P}_{VH}(\mathbf{x}(k), \mathcal{Y}, \mathcal{Q}, \mathcal{W})$ in which the horizon is now a decision variable $N(k)$

$$J^*_{VH}(\mathbf{x}(k)) = \min_{\mathbf{u},\mathbf{x},\mathbf{y},N(k)} \sum_{j=0}^{N(k)} (1 + \gamma \|\mathbf{u}(k+j|k)\|) \tag{2.49}$$

subject to $\forall j \in \{0 \dots N(k)\}$

$$\begin{aligned}
\mathbf{x}(k+j+1|k) &= \mathbf{A}\mathbf{x}(k+j|k) + \mathbf{B}\mathbf{u}(k+j|k) && \text{(2.50a)}\\
\mathbf{y}(k+j|k) &= \mathbf{C}\mathbf{x}(k+j|k) + \mathbf{D}\mathbf{u}(k+j|k) && \text{(2.50b)}\\
\mathbf{x}(k|k) &= \mathbf{x}(k) && \text{(2.50c)}\\
\mathbf{y}(k+j|k) &\in \mathcal{Y}(j) && \text{(2.50d)}\\
\mathbf{x}(k+N(k)+1|k) &\in \mathcal{Q}(N(k)+1) && \text{(2.50e)}
\end{aligned}$$

where $N(k)$ is the predicted time to entry of $\mathcal{Q}$ at step k. The constraint sets $\mathcal{Y}(j)$ are tightened according to (2.8) as before. The terminal constraints are tightened in a similar manner

$$\begin{aligned}
\mathcal{Q}(0) &= \mathcal{Q} && \text{(2.51a)}\\
\mathcal{Q}(j+1) &= \mathcal{Q}(j) \sim \mathbf{L}(j)\mathcal{W} && \text{(2.51b)}
\end{aligned}$$

where the state transition matrix $\mathbf{L}$ is defined as in (2.9).

Algorithm 2.2. (Robust Variable-Horizon MPC)

1. If $\mathbf{x}(k) \in \mathcal{Q}$, stop (target reached).
2. Solve optimization $\mathsf{P}_{VH}(\mathbf{x}(k), \mathcal{Y}, \mathcal{Q}, \mathcal{W})$
3. Apply control $\mathbf{u}(k) = \mathbf{u}^*(k|k)$ from the optimal sequence
4. Increment k. Go to Step 1.

Theorem 2.3. ***(Robust Feasibility)*** *If* $\mathsf{P}_{VH}(\mathbf{x}(0), \mathcal{Y}, \mathcal{Q}, \mathcal{W})$ *has a feasible solution then the system (2.1), subjected to disturbances obeying (2.2) and controlled using Algorithm 2.2, is robustly-feasible.*

Proof: The proof if this result is very similar to that of Theorem 2.1, showing that feasibility of problem $\mathsf{P}_{VH}(\mathbf{x}(k_0), \mathcal{Y}, \mathcal{Q}, \mathcal{W})$ at some time k_0 implies feasibility of the subsequent problem $\mathsf{P}_{VH}(\mathbf{x}(k_0+1), \mathcal{Y}, \mathcal{Q}, \mathcal{W})$ for all disturbances $\mathbf{w}(k_0)$ obeying (2.2).

Assume $\mathsf{P}_{VH}(\mathbf{x}(k_0), \mathcal{Y}, \mathcal{Q}, \mathcal{W})$ is feasible with a horizon of $N(k_0)$. Then it has a feasible (not necessarily optimal) solution, denoted by *, with states $\mathbf{x}^*(k_0+j|k_0),\ j \in \{0, \ldots, N(k_0)+1\}$, inputs $\mathbf{u}^*(k_0+j|k_0),\ j \in \{0, \ldots, N(k_0)\}$ and outputs $\mathbf{y}^*(k_0+j|k_0),\ j \in \{0, \ldots, N(k_0)\}$ satisfying all of the constraints (2.7). Now, to prove feasibility of the subsequent optimization, a candidate solution is constructed and then shown to satisfy the constraints. Consider the following candidate solution, denoted by $\hat{\cdot}$, for problem $\mathsf{P}(\mathbf{x}(k_0+1), \mathcal{Y}, \mathcal{Q}, \mathcal{W})$

$$\hat{N}(k_0+1) = N(k_0) - 1 \tag{2.52a}$$

$$\begin{aligned} \hat{\mathbf{u}}(k_0+j+1|k_0+1) &= \mathbf{u}^*(k_0+j+1|k_0) \\ &+ \mathbf{K}(j)\mathbf{L}(j)\mathbf{w}(k_0),\ \forall i \in \{0 \ldots N(k_0)-1\} \end{aligned} \tag{2.52b}$$

$$\begin{aligned} \hat{\mathbf{x}}(k_0+j+1|k_0+1) &= \mathbf{x}^*(k_0+j+1|k_0) \\ &+ \mathbf{L}(j)\mathbf{w}(k_0),\ \forall i \in \{0 \ldots N(k_0)\} \end{aligned} \tag{2.52c}$$

$$\begin{aligned} \hat{\mathbf{y}}(k_0+j+1|k_0+1) &= \mathbf{C}\hat{\mathbf{x}}(k_0+j+1|k_0+1) \\ &+ \mathbf{D}\hat{\mathbf{u}}(k_0+j+1|k_0+1),\ \forall i \in \{0 \ldots N(k_0)-1\} \end{aligned} \tag{2.52d}$$

where $\mathbf{K}(j)$ denotes a stabilizing feedback law and the corresponding state transition matrices $\mathbf{L}(j)$ are defined as in (2.9). Unlike the candidate solution constructed in Theorem 2.1, there is no additional step added at the end, hence the horizon length $\hat{N}(k_0+1)$ for the candidate solution is one step shorter than the previous solution.

Satisfaction of the dynamics (2.50a) and (2.50b), initial condition (2.50c) and output constraints (2.50d) follows from arguments identical to those in Theorem 2.1. It remains to show that the solution (2.52) satisfies the terminal constraint (2.50e).

Feasibility of $\mathsf{P}_{VH}(\mathbf{x}(k_0), \mathcal{Y}, \mathcal{Q}, \mathcal{W})$ implies $\mathbf{x}^*(k + N(k_0) + 1|k) \in \mathcal{Q}(N(k_0) + 1)$ from (2.50e). The candidate solution state for this step is

$$\hat{\mathbf{x}}(k_0 + N(k_0) + 1|k_0 + 1) = \mathbf{x}^*(k_0 + N(k_0) + 1|k_0) + \mathbf{L}(N(k_0))\mathbf{w}(k_0) \tag{2.53}$$

Substsituting $j = N(k_0)$ in the expression for the terminal constraints (2.51b) gives the Pontryagin difference

$$\mathcal{Q}(N(k_0) + 1) = \mathcal{Q}(N(k_0)) \sim \mathbf{L}(N(k_0))\mathcal{W} \tag{2.54}$$

Combining (2.53) and (2.54) with the property of the Pontryagin difference (2.11) gives the result

$$\begin{aligned} \mathbf{x}^*(k_0 + N(k_0) + 1|k_0) &\in \mathcal{Q}(N(k_0) + 1) \\ \Rightarrow \hat{\mathbf{x}}(k_0 + N(k_0) + 1|k_0 + 1) &= \mathbf{x}^*(k_0 + N(k_0) + 1|k_0) + \mathbf{L}(N(k_0))\mathbf{w}(k_0) \\ &\in \mathcal{Q}(N(k_0)),\ \forall \mathbf{w}(k_0) \in \mathcal{W} \end{aligned}$$

Rewriting and substituting the new horizon $\hat{N}(k_0) = N(k_0) - 1$

$$\hat{\mathbf{x}}((k_0 + 1) + \hat{N}(k_0) + 1|(k_0 + 1)) \in \mathcal{Q}(\hat{N}(k_0) + 1),\ \forall \mathbf{w}(k_0) \in \mathcal{W} \tag{2.55}$$

which satisfies the terminal constraint (2.50e) for time $k_0 + 1$.

Having shown that the candidate solution (2.52) satisfies the constraints (2.50) for all $\mathbf{w}(k_0)$ obeying (2.2), it follows that feasibility at time k_0 implies feasibility at time $k_0 + 1$ and therefore feasibility at time 0 implies feasibility at all future times $k > 0$. □

Theorem 2.4. ***(Robust Completion)*** *If* $\mathsf{P}_{VH}\left(\mathbf{x}(0), \mathcal{Y}, \mathcal{Q}, \mathcal{W}\right)$ *has a feasible solution with cost* $J^*\left(\mathbf{x}(0)\right)$ *and*

$$\lambda = 1 - \gamma \max_{\mathbf{w}\in\mathcal{W}} \sum_{i=0}^{\infty} \|\mathbf{K}(i)\mathbf{L}(i)\mathbf{w}\| > 0 \tag{2.56}$$

then the state $\mathbf{x}(k)$ *of the system (2.1), subject to disturbances obeying (2.2) and controlled using Algorithm 2.2, will enter target set* $\mathcal{Q}$ *in* $N_{\max}$ *steps or fewer, where* $N_{\max}$ *is the integer part of* $\frac{J^*(\mathbf{x}(0))}{\lambda}$

Proof The proof of this result is very similar to that of Theorem 2.2, deriving a bound on the optimal cost and using a Lyapunov-like argument. Suppose, at some time-step k_0, the state is not within the target, *i.e.* $\mathbf{x}(k_0) \notin \mathcal{Q}$, then using the triangle inequality, the cost of the candidate solution (2.52) for the subsequent time-step k_0+1 can be bounded by

$$\hat{J}_{VH}(\mathbf{x}(k_0+1)) \leq J^*_{VH}(\mathbf{x}(k_0)) - \lambda \tag{2.57}$$

where the λ reduction arises from the horizon reduction $\hat{N}(k_0+1) = N(k_0)-1$ and the addition of the perturbation terms in (2.52). This gives a bound on the subsequent optimal cost

$$J^*_{VH}(\mathbf{x}(k_0+1)) \leq J^*_{VH}(\mathbf{x}(k_0)) - \lambda \tag{2.58}$$

Therefore, if the state remains outside the target set for P steps, the cost is

$$J^*_{VH}(\mathbf{x}(P)) \leq J^*_{VH}(\mathbf{x}(0)) - P\lambda \tag{2.59}$$

and the left hand side would be negative if $P > N_{\max}$. This is a contradiction, as the cost must be positive, by construction. Hence the target set $\mathcal{Q}$ must be reached in $N_{\max}$ steps or fewer. □

2.6.3 MILP Implementation

The horizon variation in the optimization P_{VH} is implemented using MILP optimization to perform the integer choice of horizon length. Assume the terminal constraint

sets are polyhedral

$$\mathcal{Q}(j) = \{\mathbf{x} \mid \mathbf{p}_i^T \mathbf{x} \leq q_i(j),\ \forall i \in 1 \dots N_Q\}$$

Then the terminal constraint (2.50e) can be written as

$$\mathbf{p}_i^T \mathbf{x}(k+j|k) \leq q_i(j) + Mb(j),\ \forall i \in 1 \dots N_Q$$

where M is a large positive scalar, greater than any state value encountered in the problem, and $b(j)$ is a binary decision variable defined such that $b(j) = 1$ if $N(k) = j$ and $b(j) = 0$ otherwise. The cost function (2.49) is written as a summation over a fixed, maximal horizon $\bar{N}$ in terms of b

$$J^*(\mathbf{x}(k)) = \min_{\mathbf{u},\mathbf{x},\mathbf{y},\mathbf{b}} \sum_{j=0}^{\bar{N}} (jb(j) + \gamma \|\mathbf{u}(k+j|k)\|)$$

subject to the additional constraint

$$\sum_{j=0}^{\bar{N}} b(j) = 1$$

to ensure that the optimization selects a finishing time step. Finally, since the optimization is converted to a fixed-horizon problem of horizon $\bar{N}$, it is necessary to relax the constraints on those steps after the selected finishing time. Assuming the constraints, like the target set, are polyhedral

$$\mathcal{Y}(j) = \{\mathbf{y} \mid \mathbf{r}_i^T \mathbf{y} \leq s_i(j),\ \forall i \in 1 \dots N_c\}$$

then a similar "big-M" formulation can be employed

$$\mathbf{r}_i^T \mathbf{y}(k+j|k) \leq s_i(j) + M \sum_{n=1}^{j} b(n),\ \forall i \in 1 \dots N_c$$

in which the constraints on step j of the plan are relaxed if the horizon is shorter than j.

Remark 2.10. (Non-convex Constraint Sets) MILP optimization also enables non-convex constraints to be applied, such as collision avoidance [36, 37]. It is necessary to consider how to tighten these constraints in accordance with (2.8b). The Pontryagin difference for a union of polytopes $\mathcal{A}$, which may be a non-convex set, is described by Kerrigan [31]

$$\mathcal{X} \sim \mathcal{Y} = [\mathcal{X}^c \oplus (-\mathcal{Y})]^c \tag{2.60}$$

where $\cdot^c$ denotes the complement of a set. Here, that definition is specialized for the case of a set exclusion, such as an avoidance constraint. Define a set subtraction operation

$$\mathcal{A} \backslash \mathcal{B} \triangleq \{\mathbf{z} \in \mathcal{A} \mid \mathbf{z} \notin \mathcal{B}\} \tag{2.61}$$

Proposition 2.3. ***(Pontryagin Difference for a Set Exclusion)*** *The Pontryagin difference for a set exclusion as defined by (2.61) is given by*

$$(\mathcal{A} \backslash \mathcal{B}) \sim \mathcal{C} = (\mathcal{A} \sim \mathcal{C}) \backslash (\mathcal{B} \oplus (-\mathcal{C})) \tag{2.62}$$

Proof: See Section 2.A.3.

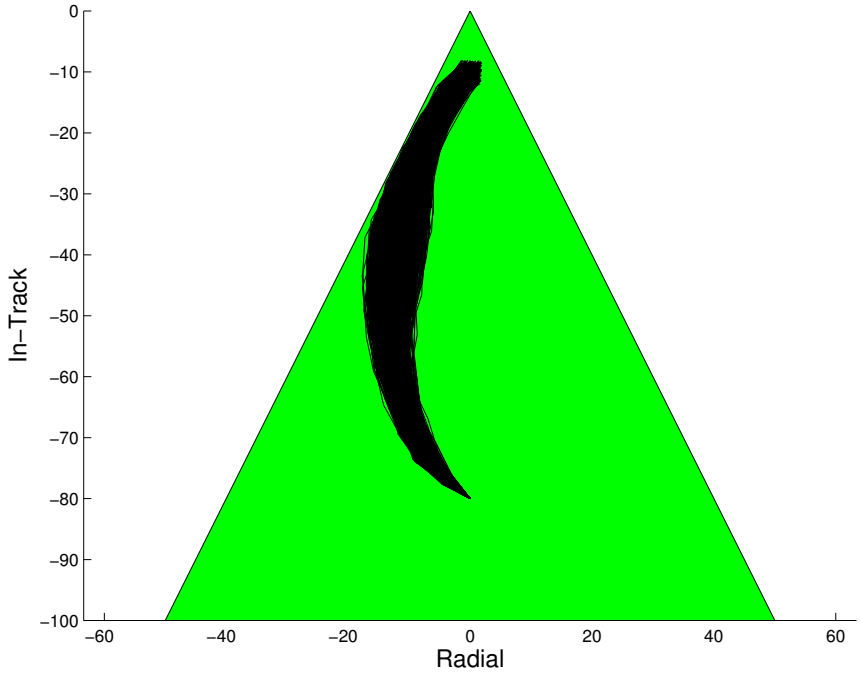
For obstacle avoidance problems, $\mathcal{B}$ would represent the obstacle set and $(-\mathcal{C})$ some mapping of the uncertainty, so the Minkowski summation leads to an effective enlargement of the obstacles at future steps of each plan, using the set tightening defined in (2.8b). Similarly, for avoidance of collisions between vehicles, the avoidance region around each vehicle is enlarged. Since the obstacles and avoidance regions are defined as polyhedra, as are all other constraints, the Pontryagin difference in (2.62) can be easily performed. The constraint set approximations described in Remark 2.7 are usefully employed here to avoid complicated corner effects arising from the Minkowski sum in (2.62). The number of constraints is then unchanged by the constraint tightening.

2.7 Variable Horizon Examples

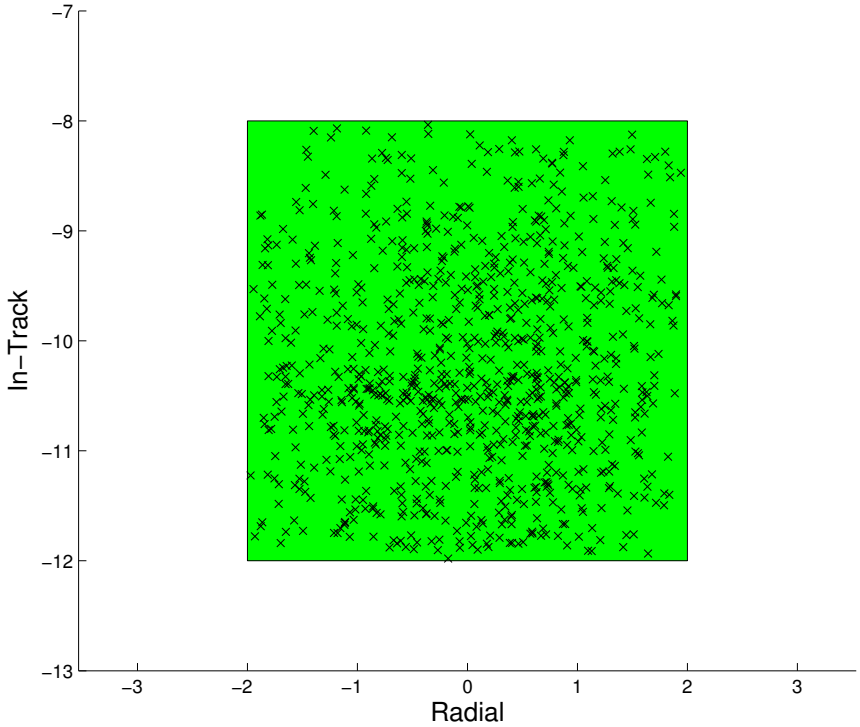
2.7.1 Space Station Rendezvous

Fig. 2-12(a) shows the results of approximately 900 simulations of a spacecraft rendezvous controlled by variable-horizon MPC. In each case, the spacecraft starts with an 80m in-track separation, shown at the bottom of the figure. It is required to enter a 4m by 4m box centered 10m behind the reference using minimum fuel. For sensor visibility, it should remain in a cone with vertex at the origin and half-angle $\tan^{-1}(0.5)$, shown shaded in the figure, at all times. The dynamics model uses Hills equations [35], as in the earlier example in Section 2.5.2, and a disturbance force of up to 10% of the control thrust acts on the spacecraft.

In Fig. 2-12(a), observe that many of the trajectories pass close to the edge of the visibility cone, taking a curved trajectory to reduce fuel use by coasting under gravitational force, and some brush up against the edge of the region, but all remain inside the constraint. In Fig. 2-12(b), the shaded region marks the target set $\mathcal{Q}$ and the crosses mark the end points of all the simulated trajectories. Every trajectory ended inside the required region. Finally, Fig. 2-13 shows the cost reduction at each step for all the simulations. The horizontal line at the top marks the theoretical decrease λ defined in (2.56). At every step and for all simulations, the cost decreased by more than λ, as predicted by (2.57).



(a) Paths. The vehicle starts at the bottom.



(b) Final states. The target set is shaded.

Figure 2-12: Results of 900 Rendezvous Simulations

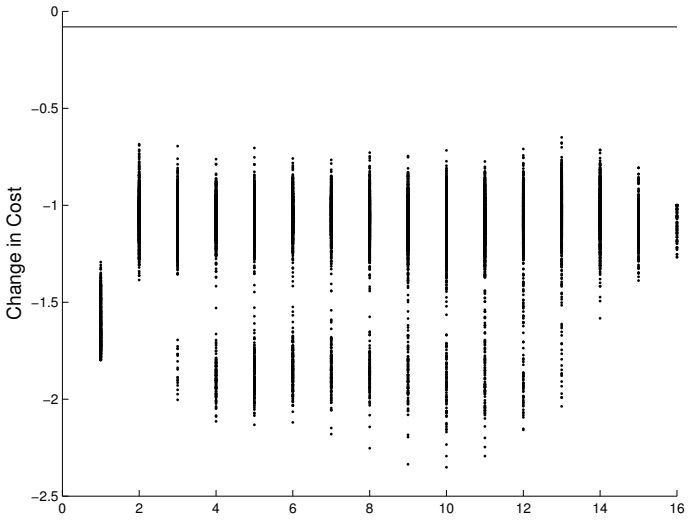


Figure 2-13: Cost Change at Each Plan Step using Robust MPC. The dashed line is the predicted upper bound.

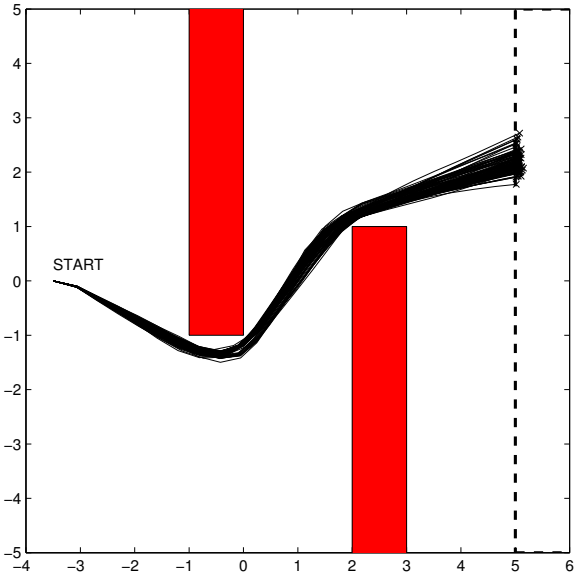


Figure 2-14: 100 Simulations using Robustly-Feasible Controller.

2.7.2 UAV Obstacle Avoidance

Fig. 2-14 shows the trajectories from 100 simulations of a UAV obstacle avoidance scenario, using the linearized constrained UAV dynamics model from Ref. [38]. The simulations had different randomly-generated disturbances of up to 10% of the control force magnitude. The starting point, common to all simulations, is marked. The objective was to reach the target box, shown dashed at the right, in minimum time while avoiding the two obstacles. Each trajectory is slightly different, visible amid the thick bunch of lines, but all remain outside the obstacles and terminate in the target set, as marked by a cross. Observe how the paths get very close to the obstacles but never encroach.

Fig. 2-15 shows further simulation results involving randomly-generated obstacle fields. Six representative scenarios are shown and for each one, ten trajectories are plotted, each from a different simulation with random disturbance forces. It is interesting to observe that in four of the scenarios, different disturbances caused the UAV to pass on different sides of obstacles. This corresponds to a change in integer decisions within the MILP.

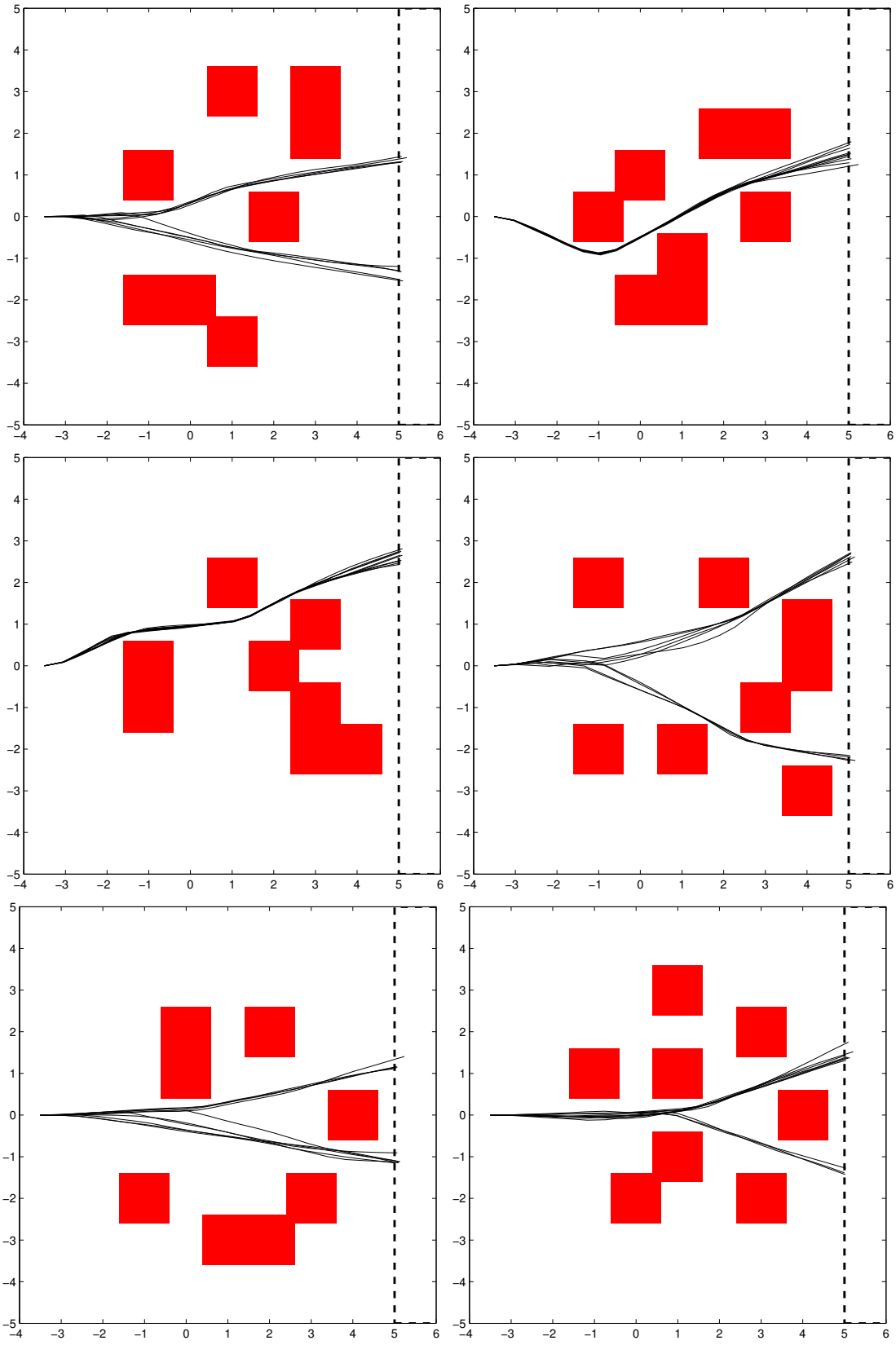


Figure 2-15: 10 UAV Simulations of Six Scenarios using Robustly-Feasible Controller.

2.8 Summary

Two forms of robust Model Predictive Control (MPC) have been presented, both using constraint tightening to guarantee robust constraint satisfaction and feasibility. The first new controller extends previous work [14] by generalizing the candidate control policy used for constraint tightening. This has been shown to give larger feasible sets than previous work, indicating a less conservative controller. The controller has been demonstrated in several examples including station-keeping control of a spacecraft.

The second new controller relaxes the requirement of an invariant terminal constraint set. By employing a variable horizon [12], finite-time entry of an arbitrary target set is guaranteed. This controller is applicable to transient or maneuvering problems, in which the target set is specified as part of the problem. Simulation examples have been presented concerning rendezvous control for spacecraft and UAV flight control subject to obstacle avoidance.

2.A Proof of Propositions

2.A.1 Proof of Proposition 2.1

Proposition 2.1 (Repeating Trajectories as Terminal Constraints) *For any system, an admissible trajectory that repeats after some period without control is control invariant admissible set, satisfying (2.14b) and (2.14a). Define the set as follows*

$$\mathcal{X}_F = \{\mathbf{x} \mid \mathbf{A}^{N_R}\mathbf{x} = \mathbf{x},\ \mathbf{C}\mathbf{A}^j\mathbf{x} \in \mathcal{Y}(N),\ \forall j = 0 \dots (N_R - 1)\} \tag{2.63}$$

where N_R is the chosen period of repetition. Also define the invariance control law $\kappa(\mathbf{x}) = \mathbf{0}\ \forall \mathbf{x}$.

Proof Choose a point $\mathbf{x}_0$ in $\mathcal{X}_F$. Then by the set definition (2.32) we have

$$\mathbf{A}^{N_R}\mathbf{x}_0 = \mathbf{x}_0 \tag{2.64a}$$

$$\mathbf{C}\mathbf{A}^j\mathbf{x}_0 \in \mathcal{Y}(N),\ \forall j = 0 \dots (N_R - 1) \tag{2.64b}$$

The admissibility condition (2.14b) requires

$$\mathbf{C}\mathbf{x}_0 + \mathbf{D}\kappa(\mathbf{x}_0) \in \mathcal{Y}(N)$$

and since $\kappa(\cdot) = \mathbf{0}$, this condition is ensured by the constraint (2.64b) with $j = 0$. The inviariance condition requires

$$\mathbf{A}\mathbf{x}_0 + \mathbf{B}\kappa(\mathbf{x}_0) \in \mathcal{X}_F$$

Again, since $\kappa(\cdot) = \mathbf{0}$, this is equivalent to the requirement

$$\mathbf{x}_1 = \mathbf{A}\mathbf{x}_0 \in \mathcal{X}_F$$

Begin by testing the equality constraint

$$\begin{aligned}\mathbf{A}^{N_R}\mathbf{x}_1 &= \mathbf{A}^{N_R}\mathbf{A}\mathbf{x}_0 \\ &= \mathbf{A}\mathbf{A}^{N_R}\mathbf{x}_0\end{aligned}$$

and substitute for $\mathbf{A}^{N_R}\mathbf{x}_0$ from (2.64a)

$$\begin{aligned}\mathbf{A}^{N_R}\mathbf{x}_1 &= \mathbf{A}\mathbf{x}_0 \\ &= \mathbf{x}_1\end{aligned}$$

satisfying the equality constraint. Now test the set inclusion constraint, beginning by rewriting (2.64b) in terms of $\mathbf{x}_1$

$$\begin{aligned}\mathbf{C}\mathbf{A}^{j}\mathbf{x}_0 &\in \mathcal{Y}(N),\ \forall j = 0 \ldots (N_R - 1) \\ \Rightarrow \mathbf{C}\mathbf{A}^{(j-1)}\mathbf{A}\mathbf{x}_0 &\in \mathcal{Y}(N),\ \forall j = 1 \ldots (N_R - 1) \\ \text{and } \mathbf{C}\mathbf{x}_0 &\in \mathcal{Y}(N)\end{aligned}$$

Substitute for $\mathbf{x}_0$ in the last line using (2.64a) and factorize again giving

$$\begin{aligned}\mathbf{C}\mathbf{A}^{(j-1)}\mathbf{A}\mathbf{x}_0 &\in \mathcal{Y}(N),\ \forall j = 1 \ldots (N_R - 1) \\ \text{and } \mathbf{C}\mathbf{A}^{N_R}\mathbf{x}_0 &\in \mathcal{Y}(N) \\ \Rightarrow \mathbf{C}\mathbf{A}^{N_R-1}\mathbf{A}\mathbf{x}_0 &\in \mathcal{Y}(N)\end{aligned}$$

Now combine the first and last lines

$$\mathbf{C}\mathbf{A}^{(j-1)}\mathbf{A}\mathbf{x}_0 \in \mathcal{Y}(N),\ \forall j = 1 \ldots N_R$$

Finally substitute $\mathbf{x}_1 = \mathbf{A}\mathbf{x}_0$ and $i = j - 1$ to get

$$\mathbf{C}\mathbf{A}^{i}\mathbf{x}_1 \in \mathcal{Y}(N),\ \forall i = 0 \ldots (N_R - 1)$$

which shows that $\mathbf{x}_1$ satisfies the constraints of $\mathcal{X}_F$, hence for any $\mathbf{x}_0 \in \mathcal{X}_F$, it follows that $\mathbf{x}_1 = \mathbf{A}\mathbf{x}_0 \in \mathcal{X}_F$ and so $\mathcal{X}_F$ is a control-invariant admissible set. □

2.A.2 Proof of Proposition 2.2

Proposition 2.2 *The convergence set can be no smaller than the disturbance set,* $\mathcal{X}_C \supseteq \mathcal{W}$.

Proof First, define a quantity γ as the norm bound of the disturbance

$$\gamma = \max_{\mathbf{w}\in\mathcal{W}} \|\mathbf{w}\|_Q$$

Also define the norm-bounded set

$$\mathcal{B}_Q(\lambda) = \{\mathbf{v} \in \Re^{N_x} \mid \|\mathbf{v}\|_Q \leq \lambda\}$$

Therefore it follows that $\mathcal{W} \subseteq \mathcal{B}_Q(\gamma)$. Now, considering the term with $i = 0$ for the summation in (2.35) for the quantity α and recalling $\mathbf{L}(0) = \mathbf{I}$, it follows that $\alpha \geq \gamma$. Therefore the set inclusion above can be extended

$$\mathcal{W} \subseteq \mathcal{B}_Q(\gamma) \subseteq \mathcal{B}_Q(\alpha) \subseteq \mathcal{B}_Q(\alpha + \beta) = \mathcal{X}_C$$

□

2.A.3 Proof of Proposition 2.3

Proposition 2.3 (Pontryagin Difference for a Set Exclusion) *The Pontryagin difference for a set exclusion as defined by (2.61) is given by*

$$(\mathcal{A}\backslash\mathcal{B}) \sim \mathcal{C} = (\mathcal{A} \sim \mathcal{C})\backslash(\mathcal{B} \oplus (-\mathcal{C})) \tag{2.65}$$

Proof Apply the result (2.60) with $\mathcal{X} = \mathcal{A}\backslash\mathcal{B}$ and $\mathcal{X} = \mathcal{C}$. Begin by writing the set exclusion as an intersection

$$\mathcal{X} = \mathcal{A}\backslash\mathcal{B} = \mathcal{A} \cap \mathcal{B}^c$$

Taking the complement

$$\mathcal{X}^c = \mathcal{A}^c \cup \mathcal{B}$$

then performing the Minkowski sum

$$\mathcal{X}^c \oplus (-\mathcal{Y}) = (\mathcal{A}^c \oplus (-\mathcal{C})) \cup (\mathcal{B} \oplus (-\mathcal{C}))$$

and finally taking the complement again to get the result

$$\mathcal{X} \sim \mathcal{Y} = [\mathcal{X}^c \oplus (-\mathcal{Y})]^c = (\mathcal{A}^c \oplus (-\mathcal{C}))^c \cap (\mathcal{B} \oplus (-\mathcal{C}))^c$$

Substituting for $(\mathcal{A}^c \oplus (-\mathcal{C}))^c$ using (2.60) and rewriting as a set subtraction again

$$\begin{aligned} (\mathcal{A} \backslash \mathcal{B}) \sim \mathcal{C} &= \mathcal{X} \sim \mathcal{Y} \\ &= (\mathcal{A} \sim \mathcal{C}) \backslash (\mathcal{B} \oplus (-\mathcal{C})) \end{aligned}$$

□

Chapter 3

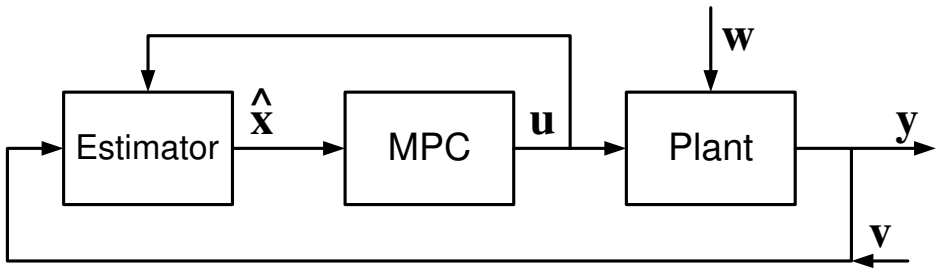
MPC with Imperfect Information

This chapter presents two extensions to robust Model Predictive Control (MPC) involving imperfect information. In the first extension, the controller from Chapter 2 is modified to account for an unknown but bounded state estimation error. As an example, a simple estimator is proposed and analyzed to provide the necessary error bounds. Furthermore, it is shown that delayed state information can be handled in the same framework. These analyses depend on knowledge of bounds on the measurement and disturbance uncertainties. The second extension provides a method of estimating these bounds using available data, providing an alternative "adaptive" form of the controller for cases where the error levels are poorly known *a priori*.

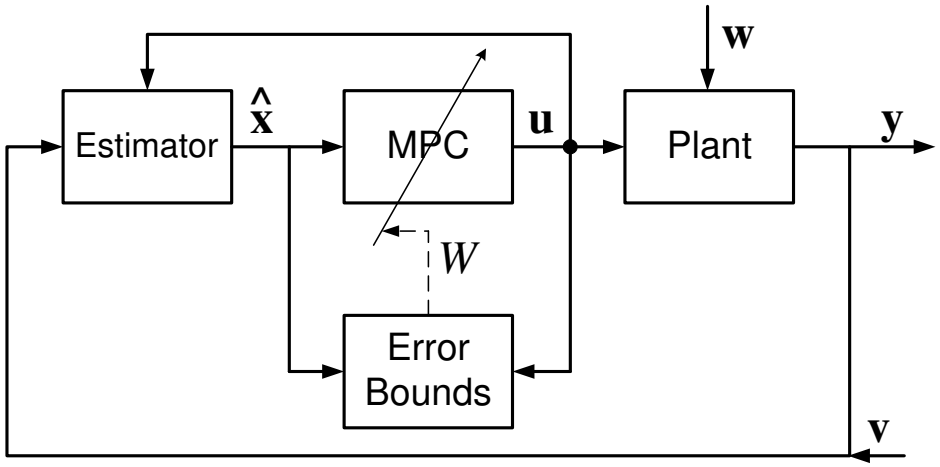
3.1 Introduction

This chapter presents two extensions to robust Model Predictive Control (MPC). The first combines robust full-state feedback MPC from Chapter 2 with an estimator, as shown in Fig. 3-1(a), to provide output feedback control, accounting for inaccuracy and time delay in state information. The second extension, outlined in Fig. 3-1(b), is applicable when the levels of disturbance and measurement noise are not well known. It is a form of adaptive control, using past data to estimate the uncertainty set and modifying the MPC controller to suit.

The vast majority of research on MPC assumes full state information. Recently,



(a) MPC with Output Feedback via Estimator



(b) MPC with Adaptive Error Bounding

Figure 3-1: System Block Diagrams. In Fig. 3-1(a), a constant error set $\mathcal{W}$ is embedded in MPC to account for uncertainties $\mathbf{w}$ and $\mathbf{v}$. In Fig. 3-1(b), the set $\mathcal{W}$ is estimated from online measurements.

attention has turned to formal analysis of the output feedback case. We adopt the approach of Bertsekas and Rhodes [39] of transforming the problem to an equivalent form with perfect information. A simple finite-memory estimator is shown to provide state information with suitably-bounded errors. Further, we provide a formal analysis of the method proposed in Ref. [40] for handling time delay, showing that it can be considered in the same framework as estimation error. Closest to this work is that of Lee and Kouvaritakis [41], who derive invariant sets for the estimation error and then employ their own robust MPC method with input saturation limits. Other approaches involve forms of separation principle for nonlinear MPC [42, 43],

including the application of receding horizon estimation [44] and set-membership estimation [63]. In an exception to the separation principle approach, Lee *et al.* [45] use receding horizon optimization to design output-feedback regulators of a certain class.

Analytical consideration of sensing noise and delay requires knowledge of bounds on the process and measurement uncertainty. While judgement and experience can often fulfill this need, exact levels of uncertainty are rarely known in practice. The second development in this chapter provides an alternative approach, deriving error bounds online from measurements of the system performance, resulting in an adaptive scheme. Since the adaptation is based on estimating the level of noise, the method is most closely related to adaptive tuning for Kalman filtering [46, 47], using the innovation as measurement data to update estimated levels of process and sensing noise. The adaptive MPC scheme proposed in this chapter is outlined in Fig. 3-1(b). The prediction error can be directly measured and used to update estimates of the bounding sets that determine the constraint tightening. Adaptive filtering methods usually seek the variance of the signal [46], but the MPC robustness depends on bounds on the prediction error. Therefore, a novel algorithm is developed to predict bounds on future signals from past measurements. Since the bounds are estimated from imperfect information, there is a probability that they might be lower than the true bound and that the problem could become infeasible. The bounding algorithm is set up to allow the designer to choose the probability of this occurrence as a parameter in the adaptation scheme.

Section 3.2 shows how problems with imperfect information, including estimation error in Section 3.2.1 and time delay in Section 3.2.5, can be accommodated by the controller from Section 2.3. Section 3.3 develops the adaptive method for estimating error bounds online.

3.2 Handling Imperfect State Information

3.2.1 MPC with Estimation Error

This section shows how the MPC formulation of Chapter 2 is modifed to account for a bounded state estimation error, using the arrangement in Fig. 3-1(a). The core of the method is the transformation of the problem to consider the dynamics of the estimate [39], which is perfectly known, by definition. Let the state estimate $\hat{\mathbf{x}}$ be the sum of the true state $\mathbf{x}$ and an error $\mathbf{e}$

$$\hat{\mathbf{x}}(k) = \mathbf{x}(k) + \mathbf{e}(k) \tag{3.1}$$

$$\hat{\mathbf{x}}(k+1) = \mathbf{x}(k+1) + \mathbf{e}(k+1) \tag{3.2}$$

where the error is unknown but bounded

$$\mathbf{e}(k) \in \mathcal{E} \ \forall k \tag{3.3}$$

Section 3.2.2 describes a simple estimator that is suitable for this purpose. Substituting (3.1) and (3.2) into the true dynamics (2.1) gives the dynamics of the estimate

$$\hat{\mathbf{x}}(k+1) = \mathbf{A}\hat{\mathbf{x}}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{w}(k) + \mathbf{e}(k+1) - \mathbf{A}\mathbf{e}(k) \tag{3.4}$$

This is the same form as the dynamics of the true state (2.1), driven by an "effective process noise"

$$\hat{\mathbf{w}}(k) = \mathbf{w}(k) + \mathbf{e}(k+1) - \mathbf{A}\mathbf{e}(k) \tag{3.5}$$

Define the set $\hat{\mathcal{W}}$ such that

$$\hat{\mathbf{w}}(k) \in \hat{\mathcal{W}} \ \forall k \tag{3.6}$$

A suitable set can be found using the disturbance (2.2) and estimate (3.3) error sets in a vector (Minkowski) summation

$$\hat{\mathcal{W}} = \mathcal{W} \oplus \mathcal{E} \oplus (-\mathbf{A})\mathcal{E} \tag{3.7}$$

This is a suitable bound for (3.6) and will suffice for robust feasibility, but it ignores possible correlations between the estimation error and disturbance signals. This will be examined further in Section 3.2.3.

Since the constraints are now applied to the estimate of the state and not the true state, the constraint set must be adjusted to account for possible discrepancies. This operation uses a Pontryagin difference (2.10)

$$\hat{\mathcal{Y}} = \mathcal{Y} \sim (-\mathbf{C})\mathcal{E} \tag{3.8}$$

The importance of this change is demonstrated in Theorem 3.1 below. The output feedback control algorithm is similar to the perfect information case in Algorithm 2.1, changing only the uncertainty and constraint sets in the optimization and including the estimation step.

Algorithm 3.1. (Robustly Feasible Output Feedback MPC)

1. Take measurements and form estimate $\hat{\mathbf{x}}(k)$
2. Solve problem $\mathsf{P}(\hat{\mathbf{x}}(k), \hat{\mathcal{Y}}, \hat{\mathcal{W}})$
3. Apply control $\mathbf{u}(k) = \mathbf{u}^*(k|k)$ from the optimal sequence
4. Increment k. Go to Step 1

Theorem 3.1. ***(Robust Feasibility)*** *If* $\mathsf{P}(\hat{\mathbf{x}}(0), \hat{\mathcal{Y}}, \hat{\mathcal{W}})$ *has a feasible solution then the system (2.1), with disturbances and estimate errors (3.5) obeying (3.6) and controlled using Algorithm 3.1, is robustly-feasible and satisfies the constraints (2.4).*

Proof The result of Theorem 2.1 applies for the system (3.4), with the estimate $\hat{\mathbf{x}}$ in place of the true state $\mathbf{x}$. Perfect information is known for this system and it is acted upon by an effective process noise (3.5) bounded by (3.6). Therefore Theorem 2.1 guarantees robust feasibility and satisfaction of the following constraints

$$\hat{\mathbf{y}}(k) = \mathbf{C}\hat{\mathbf{x}}(k) + \mathbf{D}\mathbf{u}(k) \in \hat{\mathcal{Y}} \tag{3.9}$$

It remains to show that $\hat{\mathbf{y}}(k) \in \hat{\mathcal{Y}}$ implies $\mathbf{y}(k) \in \mathcal{Y}$. The true output is given by

$$\begin{aligned} \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \\ &= \mathbf{C}(\hat{\mathbf{x}}(k) - \mathbf{e}(k)) + \mathbf{D}\mathbf{u}(k) \\ &= \hat{\mathbf{y}}(k) + (-\mathbf{C})\mathbf{e}(k) \end{aligned} \tag{3.10}$$

So, combining (3.9) and (3.10) by the definition of the modified constraint set (3.8) the property (2.11) of the Pontryagin difference means

$$\hat{\mathbf{y}}(k) \in \hat{\mathcal{Y}} \Rightarrow \mathbf{y}(k) \in \mathcal{Y} \; \forall \mathbf{e}(k) \in \mathcal{E} \tag{3.11}$$

satisfying the true constraints (2.7). □

3.2.2 Finite Memory Estimator

This section describes the implementation of a finite memory estimator that is suitable for employment within the MPC scheme described in Section 3.2.1. Let the measurement be

$$\mathbf{z}(k) = \mathbf{F}\mathbf{x}(k) + \mathbf{v}(k) \tag{3.12}$$

where $\mathbf{v}(k)$ is a random noise vector belonging to a bounded set $\mathcal{V}$ for all k. Define the following notation for stacked vectors

$$\mathbf{r}(j;k) = \begin{pmatrix} \mathbf{r}(j) \\ \vdots \\ \mathbf{r}(k) \end{pmatrix}$$

for a general discrete time vector signal $\mathbf{r}(\cdot)$. Then the combined equations for N_m past measurements can be written as

$$\begin{aligned} \mathbf{z}(k - N_m + 1; k) &= \mathbf{R}\mathbf{x}(k - N_m + 1) + \mathbf{S}\mathbf{u}(k - N_m + 1; k - 1) \\ &+ \mathbf{T}\mathbf{w}(k - N_m + 1; k - 1) + \mathbf{v}(k - N_m + 1; k) \end{aligned} \tag{3.13}$$

where

$$\mathbf{R} = \begin{bmatrix} \mathbf{F} \\ \mathbf{FA} \\ \vdots \\ \mathbf{FA}^{(N_m-1)} \end{bmatrix}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{FB} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{FAB} & \mathbf{FB} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{FA}^{(N_m-2)}\mathbf{B} & \mathbf{FA}^{(N_m-3)}\mathbf{B} & \cdots & \mathbf{FB} \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{F} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{FA} & \mathbf{F} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{FA}^{(N_m-2)} & \mathbf{FA}^{(N_m-3)} & \cdots & \mathbf{F} \end{bmatrix}$$

If the system $(\mathbf{A}, \mathbf{F})$ is observable, then the matrix $\mathbf{R}$ has full rank and its pseudo-inverse, denoted by $\mathbf{R}^\dagger$, can be found such that $\mathbf{R}^\dagger\mathbf{R} = \mathbf{I}$. This inverse can be used to get an estimate of $\mathbf{x}(k - N_m + 1)$ using only the known quantities at time k, the past measurements and controls

$$\hat{\mathbf{x}}(k - N_m + 1|k) = \mathbf{R}^\dagger\mathbf{z}(k - N_m + 1; k) - \mathbf{R}^\dagger\mathbf{S}\mathbf{u}(k - N_m + 1; k - 1) \tag{3.14}$$

which can then be propagated forward, again using only the known controls, to get an estimate of $\mathbf{x}(k)$ using measurements $\mathbf{z}(k - N_m + 1; k)$

$$\begin{aligned} \hat{\mathbf{x}}(k) &= \mathbf{A}^{(N_m-1)}\hat{\mathbf{x}}(k - N_m + 1|k) + \mathbf{P}\mathbf{u}(k - N_m + 1; k - 1) && (3.15) \\ &= \mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger\mathbf{z}(k - N_m + 1; k) && (3.16) \\ &\quad + \left(\mathbf{P} - \mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger\mathbf{S}\right)\mathbf{u}(k - N_m + 1; k - 1) \end{aligned}$$

where

$$\mathbf{P} = \left[\mathbf{A}^{(N_m-2)}\mathbf{B} \cdots \mathbf{AB}\ \mathbf{B}\right]$$

The true state propagation can be written in similar form

$$\mathbf{x}(k) = \mathbf{A}^{(N_m-1)}\mathbf{x}(k-N_m+1) + \mathbf{P}\mathbf{u}(k-N_m+1; k-1) + \mathbf{Q}\mathbf{w}(k-N_m+1; k-1) \tag{3.17}$$

where

$$\mathbf{Q} = \left[\mathbf{A}^{(N_m-2)} \cdots \mathbf{A}\ \mathbf{I}\right]$$

Substituting (3.13) and $\mathbf{R}^\dagger\mathbf{R} = \mathbf{I}$ into (3.16) and subtracting (3.17) gives the error in the estimate

$$\begin{aligned} \mathbf{e}(k) &= \hat{\mathbf{x}}(k) - \mathbf{x}(k) \\ &= \mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger\mathbf{v}(k-N_m+1;k) \\ &+ \left(\mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger\mathbf{T} - \mathbf{Q}\right)\mathbf{w}(k-N_m+1;k-1) \end{aligned} \tag{3.18}$$

If the measurement noise $\mathbf{v}$ and process noise $\mathbf{w}$ lie in bounded sets $\mathcal{V}$ and $\mathcal{W}$, respectively, vector summation can be used to find the set of all possible estimation errors

$$\begin{aligned} \mathcal{E} &= \mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger(\mathcal{V}\times\mathcal{V}\times\cdots\times\mathcal{V}) \\ &\oplus \left(\mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger\mathbf{T} - \mathbf{Q}\right)(\mathcal{W}\times\mathcal{W}\times\cdots\times\mathcal{W}) \end{aligned} \tag{3.19}$$

such that $\mathbf{e}(k) \in \mathcal{E}$ for all k.

3.2.3 Combining Finite Memory Estimator with MPC

The expression (3.19) for the estimation error can be used in (3.7) to find the set of all possible prediction errors. However, the expression (3.7) assumes that the quantities $\mathbf{e}(k)$, $\mathbf{e}(k+1)$ and $\mathbf{w}(k)$ are independently distributed within their given sets. Inspection of the estimation error expression (3.18) shows that successive errors $\mathbf{e}(k)$ and $\mathbf{e}(k+1)$ are determined by common measurement noise instances $\mathbf{v}(k-N_m+2;k)$ and process noise instances $\mathbf{w}(k-N_m+2;k-1)$. Therefore the terms $\mathbf{e}(k)$, $\mathbf{e}(k+1)$ and $\mathbf{w}(k)$ in the summation in (3.5) are not independent, and treating them as such gives a conservative approximation of the prediction error set $\hat{\mathcal{W}}$.

The expression for the estimation error (3.18) can be substituted into the predic-

tion error (3.5) to give an expression for $\hat{\mathbf{w}}(k)$ in terms of independent quantities

$$\hat{\mathbf{w}}(k) = \tilde{\mathbf{P}}\mathbf{v}(k - N_m + 1; k + 1) + \tilde{\mathbf{Q}}\mathbf{w}(k - N_m; k)$$

where

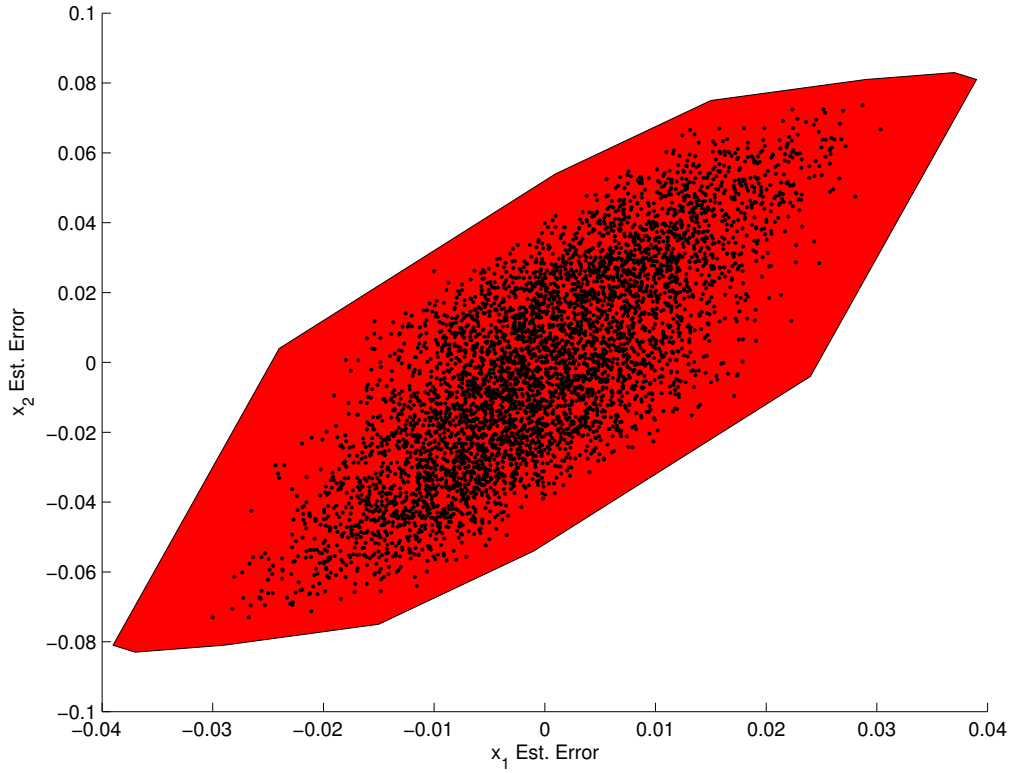
$$\begin{aligned}
\tilde{\mathbf{P}} &= \left[-\mathbf{A}^{N_m}\mathbf{R}^\dagger \mid \mathbf{0}\right] + \left[\mathbf{0} \mid \mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger\right] \\
\tilde{\mathbf{Q}} &= \left[-\mathbf{A}\left(\mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger\mathbf{T} - \mathbf{Q}\right) \mid \mathbf{0}\right] + \\
&\quad \left[\mathbf{0} \mid \left(\mathbf{A}^{(N_m-1)}\mathbf{R}^\dagger\mathbf{T} - \mathbf{Q}\right)\right] + \left[\mathbf{0} \cdots \mathbf{0} \mid \mathbf{I}\right]
\end{aligned}$$

Then an improved set bound for the prediction error (3.6), based on independent quantities, is given by

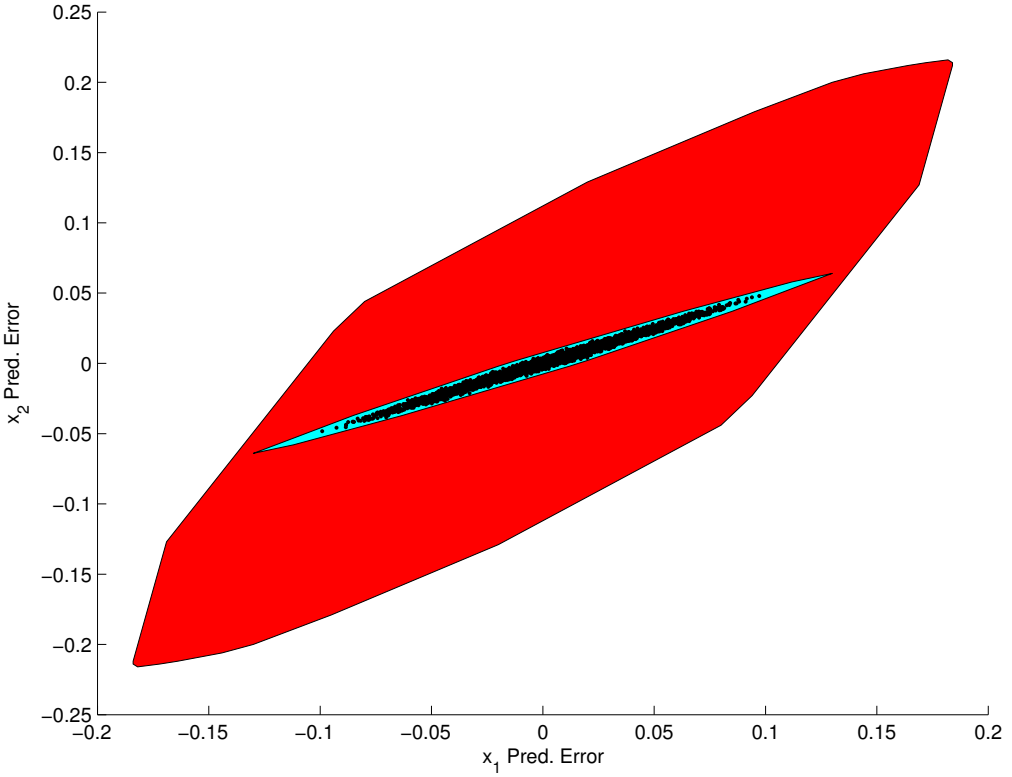
$$\hat{\mathcal{W}} = \tilde{\mathbf{P}}(\mathcal{V} \times \mathcal{V} \times \cdots \times \mathcal{V}) \oplus \tilde{\mathbf{Q}}(\mathcal{W} \times \mathcal{W} \times \cdots \times \mathcal{W}) \tag{3.20}$$

3.2.4 Example

This section repeats the simple double integrator control example of Section 2.5.1, replacing perfect full-state feedback with imperfect output feedback. Position measurements were provided to the controller, corrupted by additive errors of up to ±1cm, *i.e.* 1% of the ±1m position tolerance. The system was controlled according to Algorithm 3.1 with the estimator from Section 3.2.2 using four previous measurements to estimate the current state. Fig. 3-2(a) shows the estimation error $\mathbf{e}(k)$ at each step plotted in the state space. Observe that the errors $\mathbf{e}(k)$ lie within the set of all possible estimation errors $\mathcal{E}$ obtained from (3.19), shown shaded. Fig. 3-2(b) shows the state prediction error, again plotted as points in state space. The points all lie within the inner set, obtained using (3.20), the expression for the prediction error set that incorporates the correlation between estimation errors and process noise. The larger set, shown for comparison, is the set found using (3.7), assuming that estimation errors and process noise are all independent. This is a very large superset of the actual error set, showing that the assumption of independence leads to a very conservative controller, and that is is very important to analyze the estimator for prediction error.



(a) State Estimation Errors $\mathbf{e}(k)$. The shaded region is the error set predicted by (3.19).



(b) Prediction Errors $\hat{\mathbf{w}}(k)$. The outer shaded region is the prediction error bound from (3.7). The inner set is the much tighter bound from (3.20).

Figure 3-2: Error Sets and Simulation Results.

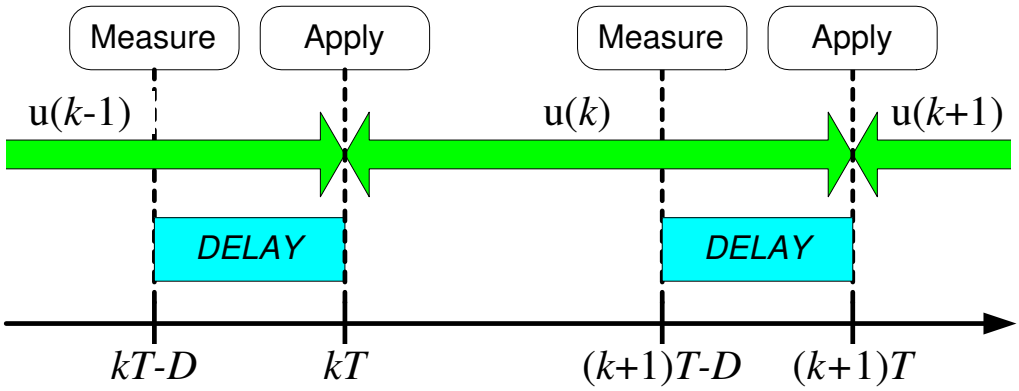


Figure 3-3: Timing Diagram for Problem with Delay

3.2.5 Time Delay

This section extends the formulation to account for a time delay in the loop. Delay can arise from several effects, including measurement, computation and communication latencies. The timing arrangement is shown graphically in Fig. 3-3. We assume that the delay is of a known length D and less than the time step *i.e.* $D < T$. Adopting the approach of [40], the state measurement is propagated forward by the length of the delay to obtain the initial condition for each MPC optimization. This may be regarded as an estimate of the state at the time the plan begins. The analysis proceeds by converting the delayed-state case to the uncertain-state case of Section 3.2.1.

Consider an LTI system, in continuous time, with state $\mathbf{x}(t)$ at time t (the use of a time argument t, T, τ rather than a discrete argument k, j, i denotes the difference between sampled and continuous states) and the following dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{A}_c\mathbf{x}(t) + \mathbf{B}_c\mathbf{u}(t) + \mathbf{E}_c\mathbf{w}_c(t) \tag{3.21}$$

The control is applied using a zero-order hold

$$\mathbf{u}(t) = \mathbf{u}(k) \quad kT \leq t < (k+1)T \tag{3.22}$$

hence the dynamics of discrete-time state can be written in the form of (2.1) as

$$\overbrace{\mathbf{x}((k+1)T)}^{\mathbf{x}(k+1)} = \overbrace{e^{\mathbf{A}_c T}}^{\mathbf{A}} \overbrace{\mathbf{x}(kT)}^{\mathbf{x}(k)} + \overbrace{\int_0^T e^{\mathbf{A}_c(T-\tau)} d\tau \mathbf{B}_c}^{\mathbf{B}} \mathbf{u}(k) + \overbrace{\int_0^T e^{\mathbf{A}_c(T-\tau)} \mathbf{E}_c \mathbf{w}_c(kT+\tau) d\tau}^{\mathbf{w}(k)} \tag{3.23}$$

Remark 3.1. (Discretization Method) The analysis shown here assumes zero-order hold control application. Altervative methods, *e.g.* instantaneous application, common for spacecraft thruster firing, could be readily applied using the same approach.

The discrete-time state estimates are based on measurements delayed by time D,

$$\hat{\mathbf{x}}(k) = \hat{\mathbf{x}}(kT|kT-D) \tag{3.24}$$

At time $t = kT - D$, a measurement is taken and used to form a state estimate. This estimate is corrupted by an error $\mathbf{n}(k)$ giving

$$\hat{\mathbf{x}}(kT-D|kT-D) = \mathbf{x}(kT-D) + \mathbf{n}(k) \tag{3.25}$$

This estimate is then used to form the estimate $\hat{\mathbf{x}}(k)$ of the state at time kT. This operation can be written as a state transition operating on the uncertain estimate

$$\begin{aligned} \hat{\mathbf{x}}(kT|kT-D) &= e^{\mathbf{A}_c D} \hat{\mathbf{x}}(kT-D|kT-D) + \int_0^D e^{\mathbf{A}_c(D-\tau)} d\tau \mathbf{B}_c \mathbf{u}(k-1) \\ &= e^{\mathbf{A}_c D} \mathbf{x}(kT-D) + e^{\mathbf{A}_c D} \mathbf{n}(k) + \int_0^D e^{\mathbf{A}_c(D-\tau)} d\tau \mathbf{B}_c \mathbf{u}(k-1) \end{aligned} \tag{3.26}$$

A similar expression, without the estimation error but including the disturbance, gives the true state at time kT in terms of the true state at $kT - D$

$$\begin{aligned} \mathbf{x}(kT) &= e^{\mathbf{A}_c D} \mathbf{x}(kT-D) + \int_0^D e^{\mathbf{A}_c(D-\tau)} d\tau \mathbf{B}_c \mathbf{u}(k-1) \\ &+ \int_0^D e^{\mathbf{A}_c(D-\tau)} \mathbf{E}_c \mathbf{w}_c(kT-D+\tau) d\tau \end{aligned} \tag{3.27}$$

Subtracting (3.27) from (3.26) gives an equation for the effective estimation error at time kT in the same form as the discrete-time estimation error expression in (3.1)

$$\overbrace{\hat{\mathbf{x}}(kT|kT-D)}^{\hat{\mathbf{x}}(k)} - \overbrace{\mathbf{x}(kT)}^{\mathbf{x}(k)} = \overbrace{e^{\mathbf{A}_c D}\mathbf{n}(k) - \int_0^D e^{\mathbf{A}_c(D-\tau)}\mathbf{E}_c\mathbf{w}_c(kT-D+\tau)d\tau}^{\mathbf{e}(k)} \quad (3.28)$$

The system with delay has now been cast in the form of the problem in Section 3.2.1. The dynamics of the delayed estimate (3.23) are, similar to (2.1), driven by an affine disturbance. The state information (3.28) is, similar to (3.1), corrupted by an additive error term. If bounding sets are known for disturbance $\mathbf{w}_c(t)$ and estimation error $\mathbf{n}(k)$, then a bound $\hat{\mathcal{W}}$ satisfying (3.6) can be obtained and the formulation of Section 3.2.1 is applicable. The only change to the algorithm is the prediction in the estimate, Step 2.

Algorithm 3.2. (Robust MPC with Time Delay)

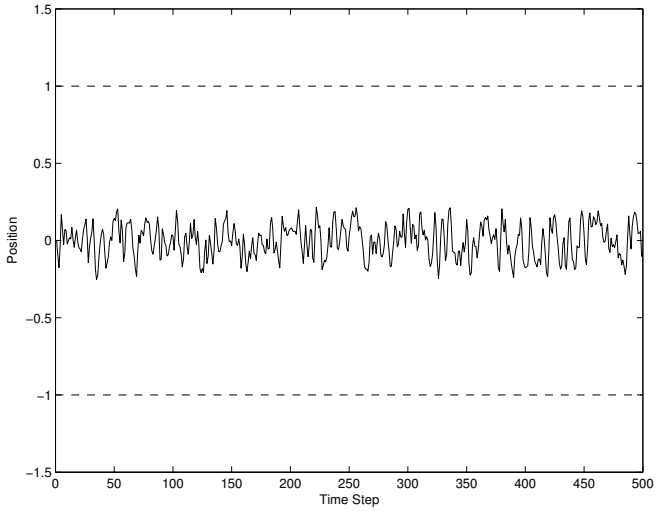
1. (Time $t = kT - D$): Form state estimate $\hat{\mathbf{x}}(kT - D)$
2. Predict state D into the future $\hat{\mathbf{x}}(kT|kT - D)$ using (3.26)
3. Solve MPC optimization starting from predicted state $\mathsf{P}(\hat{\mathbf{x}}(kT|kT - D), \hat{\mathcal{Y}}, \hat{\mathcal{W}})$
4. (Time $t = kT$): Apply first control input $\mathbf{u}(k) = \mathbf{u}^*(k|k)$ as in (3.22)
5. Increment k. Go to Step 1

3.2.6 Time Delay Example

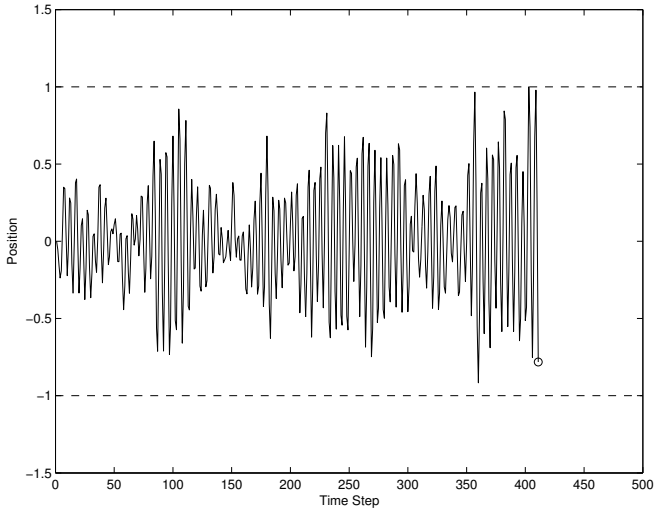
This section adds a time delay to the example from Section 2.5.1. The example parameters are the same as in Section 2.5.1 except for a change in the cost function, now chosen to primarily penalize the position error

$$\ell(\mathbf{x}, \mathbf{u}) = 5x_1^2 + x_2^2 + u^2$$

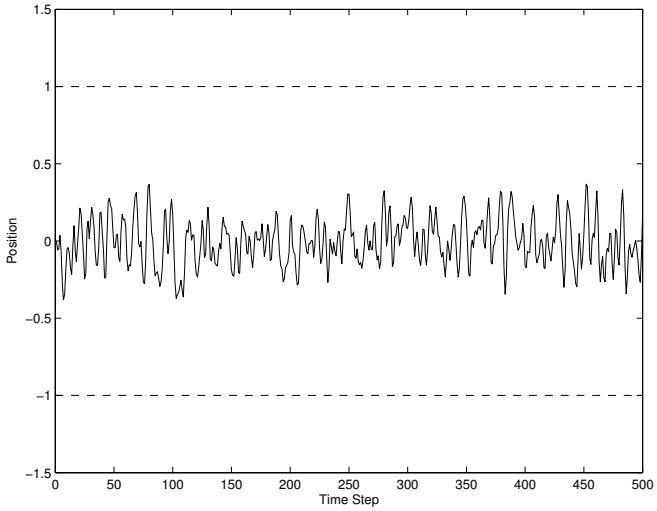
For comparison, Fig. 3-4(a) shows a simulation result for the double integrator example system with perfect, immediate state information and using the controller from



(a) No delay, control by Alg. 2.1



(b) 40% delay, control by Alg. 2.1 (ignoring delay)



(c) 40% delay, control by Alg. 3.2 (accounts for delay)

Figure 3-4: Position Histories showing Effect of Time Delay and New Controller

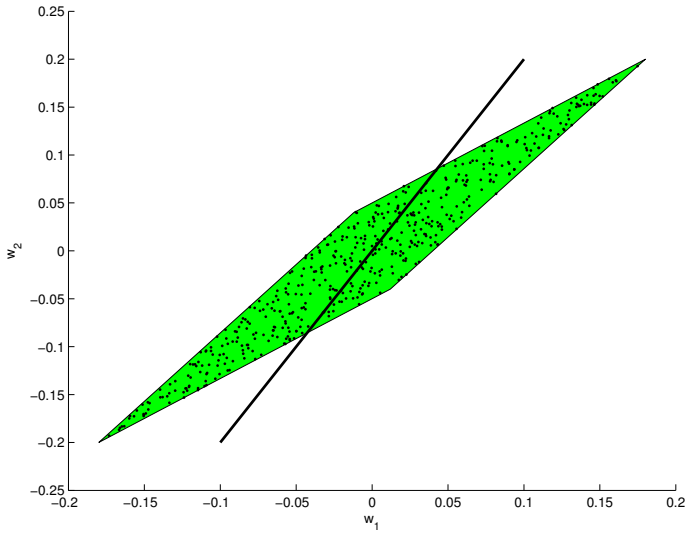


Figure 3-5: Prediction Errors in Presence of Delay. The shaded region is the set of prediction errors expected accounting for the delay. Without delay, all prediction errors would lie on the heavy line.

Section 2.5.1. Fig. 3-4(b) shows the position time history for the same system with the addition of a time delay in the state information of 0.4s (40% of the time step length) but using the same controller, robust to the disturbance but ignoring the delay. There are large oscillations in position and the optimization becomes infeasible after about 400 steps. Finally, Fig. 3-4(c) shows the result when the controller is modified to account for the delay using Algorithm 3.2, propagating the estimate forward and including extra constraint tightening to suit. The oscillation is reduced and feasibility is maintained throughout, although the position deviation is greater than in Fig. 3-4(a), the case without delay, due to the extra uncertainty introduced by the state prediction, as expected according to (3.28).

Fig. 3-5 shows the prediction errors, plotted in the state space, from the simulation in Fig. 3-4(c). The shaded region is the set of prediction errors found using the equivalent state uncertainty expression (3.28) substituted into the prediction error (3.5). The heavy line is the set of prediction errors expected due to the disturbance alone, ignoring the effect of the delay. This shows that the delay introduces additional uncertainty into the problem.

3.3 Adaptive Prediction Error Bounds

This section describes a method for deriving estimated bounds on the prediction error from online measurements. The method develops, online, an estimated set $\tilde{\mathcal{W}}(k)$ such that, if the MPC constraints are tightened to accommodate disturbances in $\tilde{\mathcal{W}}(k)$, then the problem will remain feasible with some probability λ' for a specified number of future time steps.

3.3.1 Bounding Set Estimation for Vector Signals

Section 3.A describes the calculation of a bound function $G_{NM}(x, \lambda)$ such that if $X(k)$ is a random Gaussian scalar signal with unknown variance and its measured maximum over a measured N samples is x, then with probability λ, a further M samples will not exceed $G_{NM}(x, \lambda)$. This function is employed in this section to identify bounds on multivariable signals. Assume a vector random signal $\mathbf{x}(k)$ is white and Gaussian (although cross-correlation between components of the vector is permitted). Then for any vector $\mathbf{e}$ of suitable size, $X(k) = \mathbf{e}^T\mathbf{x}(k)$ is a scalar, white Gaussian random signal and the analysis in the Appendix can be used to derive an estimated bound on $\mathbf{e}^T\mathbf{x}(k)$. Consider a set of L such vectors $\{\mathbf{e}_i\}$ such that the polytope defined by

$$\mathbf{E}\mathbf{x} \leq \mathbf{q}, \qquad \mathbf{E} = [\mathbf{e}_1 \cdots \mathbf{e}_L]^T \tag{3.29}$$

is bounded for all finite $\mathbf{q}$. Then choose the threshold values as follows

$$q_i = G_{NM}\left(\max_{k\in\{1\ldots N\}} \mathbf{e}_i^T\mathbf{x}(k), \lambda\right) \tag{3.30}$$

Then the probability that a further M samples of the signal $\mathbf{y}$ lie inside the polytope (3.29) is given by

$$\begin{aligned} P\left(\mathbf{E}\mathbf{y} \leq \mathbf{q}\right) &= 1 - P\left((\mathbf{e}_1^T\mathbf{y} > q_1) \cup \ldots \cup (\mathbf{e}_L^T\mathbf{y} > q_L)\right) \\ &\geq 1 - \sum_{i=1}^{L} P\left(\mathbf{e}_i^T\mathbf{y} > q_i\right) = 1 - L(1-\lambda) \end{aligned}$$

The probability is underbounded, with the "$\geq$" arising in (3.3.1) from the extension of the fundamental probability relation

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \leq P(A) + P(B)$$

So if the desired probability associated with the polytope is λ', the individual probability parameters are found by

$$\lambda = 1 - \frac{(1 - \lambda')}{L} \tag{3.31}$$

i.e. if the desired probability of remaining inside the polytope (3.29) is λ', the individual bounds should be set according to (3.30) with the parameter λ set according to (3.31).

3.3.2 Adaptive MPC

This section describes the combination of the bound estimation from Section 3.3.1 and the robustly-feasible MPC from Section 2.3 to form an adaptive controller, using the arrangement shown in Fig. 3-1(b). The disturbance model $\mathcal{W}$ enters the controller optimization (2.7) via the constraint tightening recursion (2.8). Hence online updates of the disturbance model can be used to update the constraint sets $\mathcal{Y}(j)$ used to compute the control.

Subtracting the first state prediction $\mathbf{x}(k+1|k) = \mathbf{A}\hat{\mathbf{x}}(k) + \mathbf{B}\mathbf{u}(k)$, constrained by (2.7a) with $j = 0$, from the true dynamics (2.1) gives an expression for the prediction error

$$\hat{\mathbf{w}}(k) = \hat{\mathbf{x}}(k+1) - \mathbf{x}(k+1|k) \tag{3.32}$$

Both quantities on the righthand side are known at time $k+1$. The method does not distinguish between the effects of disturbance, sensing noise and delay. These measurements of $\hat{\mathbf{w}}$ are used to form an estimated error set $\tilde{\mathcal{W}}(k)$ using the method in Section 3.3.1, such that with probability λ', all future prediction errors $\hat{\mathbf{w}}(k+j)$ over some window $j = \{1 \dots M\}$ lie within set $\tilde{\mathcal{W}}(k)$, and the optimization constraints are

tightened accordingly. The probability parameter λ' is chosen by the designer to set a level of "risk" for the adaptation. It was shown in Section 3.3.1 that the probability of all errors lying within the estimated set (3.29) is greater than λ'. Furthermore, while feasibility is guaranteed only if all errors lie in the chosen error bound, the optimization need not become infeasible if a single error exceeds the bound. Therefore, the probability of remaining feasible using the adaptive scheme is at least λ'.

A further complication when employing the bound estimation within adaptive MPC is that the error bound must always be shrinking, *i.e.* it is required that $\tilde{\mathcal{W}}(k+1) \subseteq \tilde{\mathcal{W}}(k)$. If the error set were to grow, the feasible region of the optimization would shrink, and feasibility might be lost. The guarantee of robust feasibility only holds if the feasible region remains the same or grows. Therefore, restrict the adaptation algorithm so that the error bound estimates are all monotonically decreasing, using the estimate equation (3.30) only if the bound is decreasing

$$q_i(k+1) = \min \left\{ q_i(k), G_{NM} \left(\max_{j \in \{0 \ldots k\}} \mathbf{e}_i^T \hat{\mathbf{w}}(j), \lambda \right) \right\} \tag{3.33}$$

where $q_i(k)$ is the estimated bound in direction i at time k.

As discussed in Remark 3.2, the assumption of independent maximum samples requires relatively long observation periods. Also, reforming the constraints for a new error set estimate requires calculating Pontryagin differences and can be computationally intensive. Therefore, the error set estimate update should be done relatively infrequently, compared to the dynamics timescales of the system.

Algorithm 3.3. (Adaptive MPC)

1. (Initialization): Choose weighting matrix $\mathbf{E}$ and vector $\mathbf{q}(0)$ forming set $\tilde{\mathcal{W}}(0)$
2. Form current estimate $\hat{\mathbf{x}}(k)$ (including propagation for delay if necessary as in Section 3.2.5)
3. Measure and store prediction error $\hat{\mathbf{w}}(k)$ using (3.32)
4. **If** k is an error-set update step, **then** update error bounds using (3.33) and reform constraint sets $\mathcal{Y}(j)$, **else** $\tilde{\mathcal{W}}(k) = \tilde{\mathcal{W}}(k-1)$ and constraints are unchanged.

5. Solve MPC optimization starting from estimate $\mathsf{P}(\hat{\mathbf{x}}(k), \hat{\mathcal{Y}}, \tilde{\mathcal{W}}(k))$

6. Apply first control input $\mathbf{u}(k) = \mathbf{u}^*(k|k)$

7. Increment k. Go to Step 2.

The method discussed in Remark 2.6, for calculating the largest possible disturbance leading to a nonempty feasible set, can be used to initialize the algorithm with a conservative estimate of the disturbance set.

3.3.3 Adaptive MPC Examples

Fig. 3-6 shows the results of a simulation using adaptive MPC from Algorithm 3.3 for the double integrator example, previously seen in Section 3.2.4, in which the aim is to keep the position within limits of ± 1 with minimum control energy. Error set updates were performed every 1000 steps, using all the past data to predict the maximum over the remainder of the 6000-step simulation *i.e.* $N = k$ and $M = 6000 - k$. The top plot shows the position time history, with the dashed lines marking the constraints on the second plan step $\mathcal{Y}(1)$. The set $\mathcal{Y}(1)$ is formed by tightening the original constraint set $\hat{\mathcal{Y}}$ to suit the disturbance set $\tilde{\mathcal{W}}$ according to the recursion (2.8), hence the greater the anticipated disturbance, the tighter the constraint. In Fig. 3-6, the constraints can be seen to be relaxed at each update, most significantly at the first update, and the controller makes use of the relaxation, allowing greater state deviations and reducing the control energy use *i.e.* the slope of the bottom plot. The middle plot shows the time history of the prediction error and the estimated bounds. The bounds can be seen to be tightening but always overbounding the error. In this case, the first 1000 samples give a good estimate of the error set. The significant relaxation of the constraints at the first update occurs because the initial disturbance set estimate was chosen to be very conservative, for the sake of safety.

Fig. 3-7 shows the successive estimated error sets for a similar simulation to Fig. 3-6 but with a multivariable disturbance. The true disturbance in the simulation was

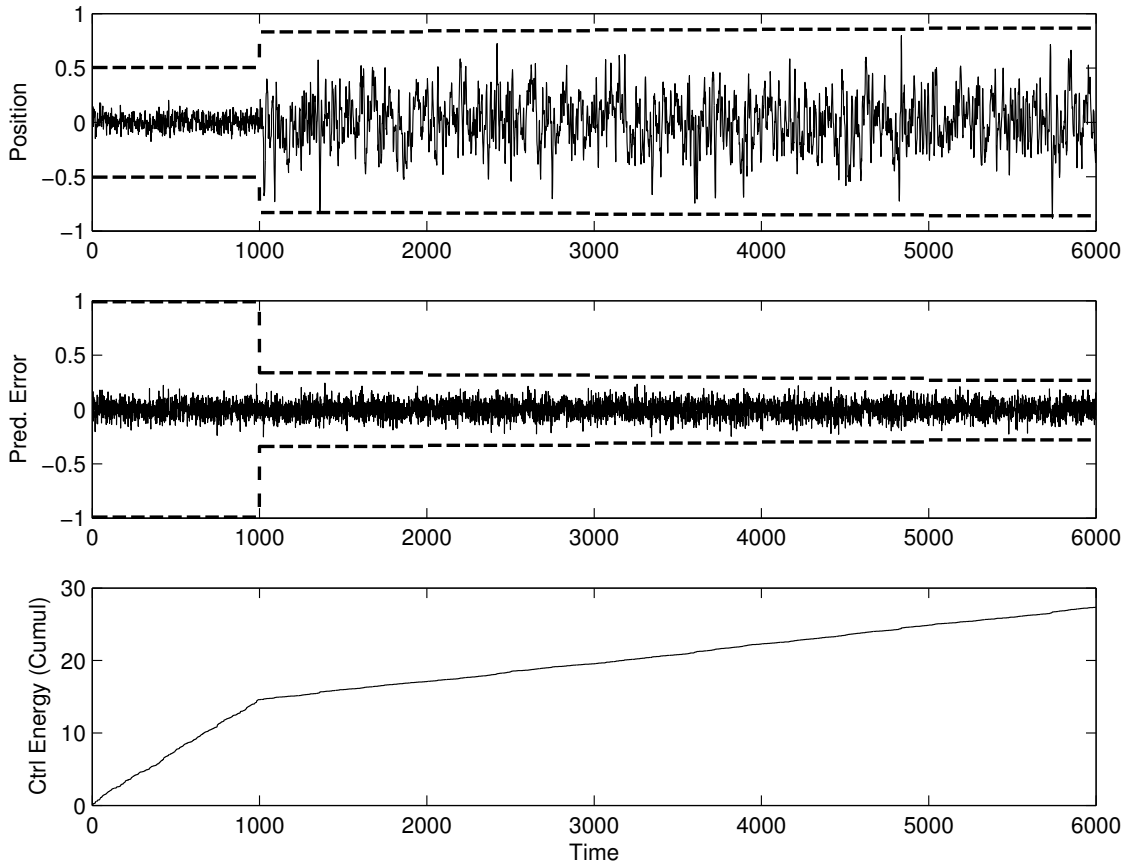


Figure 3-6: Time Histories from Adaptive MPC

a Gaussian, white vector signal with auto-correlation matrix

$$E[\mathbf{w}(k)\mathbf{w}(k)^T] = \begin{bmatrix} 0.0156 & -0.0115 \\ -0.0115 & 0.0289 \end{bmatrix}$$

This signal was generated by taking a vector of two independent signals, with standard deviations of 1 and 0.5 respectively, and rotating it by 30^o. Therefore the set of errors is expected to look like a 2-by-1 ellipse inclined at 30^o to the axes.

Eight vectors $\mathbf{e}_1 \ldots \mathbf{e}_8$, arranged at 45^o intervals, form the directions for the bounding set estimation, as in (3.29). Remark 2.6 shows how to calculate the maximum disturbance norm a particular controller could accommodate. For this example, it was found that disturbances obeying $\|\mathbf{w}(k)\|_\infty \leq 1$ could be tolerated, so the unit cube

was chosen as the initial error set estimate. The corresponding settings parameters $\mathbf{E}$ and $\mathbf{q}(0)$ are

$$\mathbf{E} = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \quad \mathbf{q}(0) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

Fig. 3-7 shows the sets shrinking at each adaptation step, closing in on the errors. Observe that one error sample, at the bottom right, falls outside the innermost estimated set. This is consistent with the analysis as the errors lie within the estimated set only with probability λ'. However, exceeding the error set does not necessarily lead to infeasibility. In this and 1000 similar tests, with the probability parameter λ' set to 0.99, all simulations remained feasible throughout. This is consistent with expectations, as the probability of infeasibility is expected to be smaller than $1-\lambda'$, as discussed above.

3.4 Summary

Output feedback Model Predictive Control has been developed, guaranteeing robust constraint satisfaction and feasibility despite the presence of unknown but bounded measurement and process uncertainty. Robust MPC has been modified to accommodate state information with unknown but bounded errors and a suitable estimator has been provided. The same controller can accommodate time delay in the loop by including prediction in the estimator. Furthermore, for cases where the uncertainty bounds are not well known *a priori*, an adaptive form has been developed using online measurements to estimate the set of possible prediction errors.

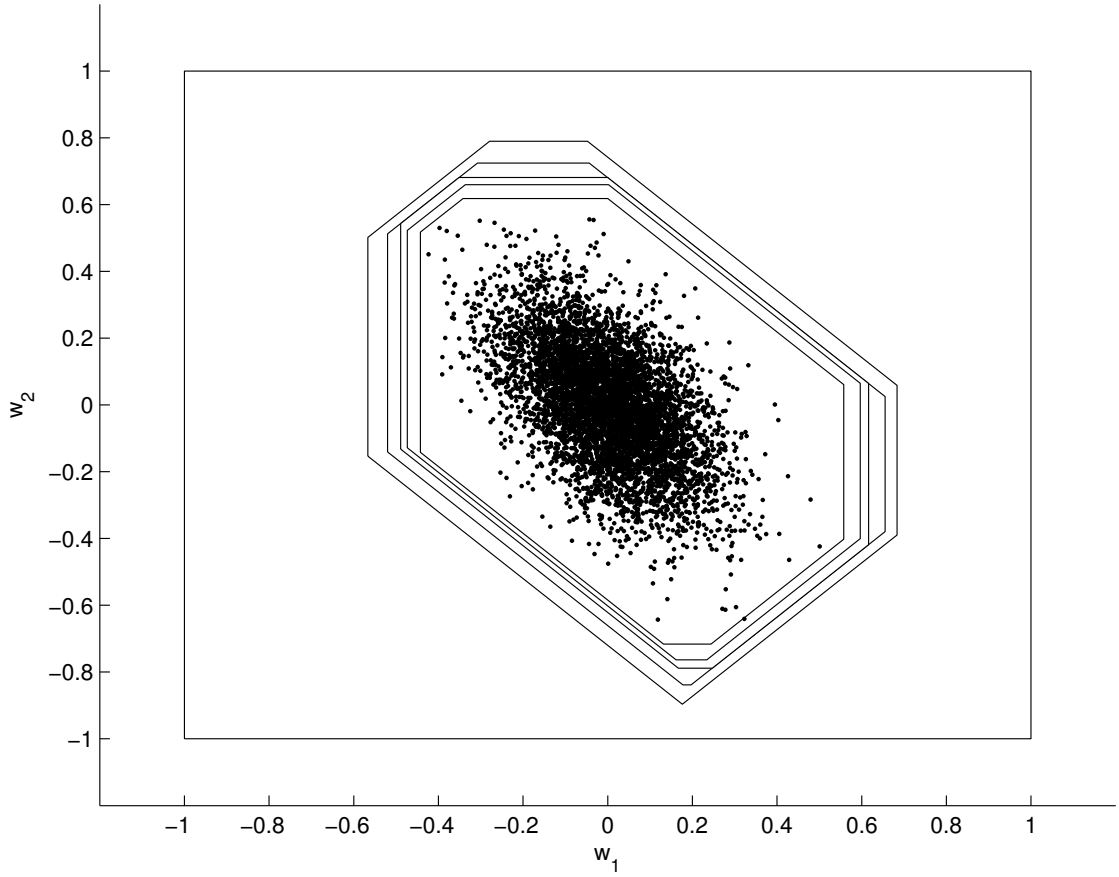


Figure 3-7: Successive Sets from Prediction Error Bounding. The initial set $\tilde{\mathcal{W}}(0)$ is the square $\|\mathbf{w}\|_\infty \leq 1$. Five further updates are performed, each new set lying inside its predecessor.

3.A Bound Estimation for Scalar Signals

This section describes a method for estimating a bound on future values of a random scalar signal given past measurements. The problem statement is to choose, given a set of N samples, a threshhold such that with probability λ, the signal will not exceed that threshhold in the next M samples. The approach is to find the conditional cumulative probability density function of the maximum of a series of M samples given the maximum value over a (different) series of N samples of the same distribution.

Assume the discrete-time scalar random signal $X(k)$ is white and Gaussian (see Remark below) with zero mean and Σ standard deviation, where Σ is a random

variable. Assume the distribution of Σ is uniform in $[\sigma_{\min}, \sigma_{\max}]$. Define the random variable Z as the maximum value of the first N samples of $X(k)$

$$Z = \max_{k \in \{1..N\}} X(k)$$

Given a value of Σ, the cumulative distribution of Z can be found

$$\begin{aligned} F_{Z|\Sigma}(z,\sigma) &= P(Z \leq z|\Sigma) \\ &= \prod_{k=1}^{N} P(X(k) \leq z|\Sigma) \\ &= \left[F_{X|\Sigma}(z,\sigma)\right]^N \end{aligned} \tag{3.34}$$

where $F_{X|\Sigma}(z,\sigma)$ is the cumulative distribution function of a normally-distributed random variable with standard deviation Σ. Differentiating gives the probability density function (PDF) for Z

$$\begin{aligned} f_{Z|\Sigma}(z,\sigma) &= \frac{\partial F_{Z|\Sigma}(z,\sigma)}{\partial z} \\ &= N\left[F_{X|\Sigma}(z,\sigma)\right]^{(N-1)} f_{X|\Sigma}(z,\sigma) \end{aligned} \tag{3.35}$$

Similarly, define Y as the maximum over a further M samples of the same signal

$$Y = \max_{k \in \{(N+1)..(N+M)\}} X(k)$$

and the PDF of Y given the standard deviation, $f_{Y|\Sigma}(y,\sigma)$, has the same form as (3.35). The aim of this analysis is to find the cumulative distribution $F_{Y|Z}(\cdot)$ of Y, the future maximum, given Z, the past maximum. Under the assumption that the signal $X(k)$ is white, then, given Σ, the elements are independent of eachother and therefore the maximum values over non-overlapping intervals are also independent, giving the following relation for the joint distribution.

$$f_{YZ|\Sigma}(y,z,\sigma) = f_{Y|\Sigma}(y,\sigma) f_{Z|\Sigma}(z,\sigma) \tag{3.36}$$

Bayes' rule gives the following two relations for the joint distributions of X, Y and Σ

$$f_{YZ\Sigma}(y,z,\sigma) = f_{YZ|\Sigma}(y,z,\sigma)f_{\Sigma}(\sigma) \tag{3.37}$$

$$f_{Z\Sigma}(z,\sigma) = f_{Z|\Sigma}(z,\sigma)f_{\Sigma}(\sigma) \tag{3.38}$$

where $f_{\Sigma}(\sigma)$ is the PDF of Σ, given by

$$f_{\Sigma}(\sigma) = \begin{cases} \frac{1}{(\sigma_{\max}-\sigma_{\min})} & \sigma_{\min} \leq \sigma \leq \sigma_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{3.39}$$

The marginal distribution of Z can be found by integrating (3.38) and substituting (3.39) giving

$$\begin{aligned} f_Z(z) &= \int_{-\infty}^{+\infty} f_{Z\Sigma}(z,\sigma)d\sigma \\ &= \int_{-\infty}^{+\infty} f_{Z|\Sigma}(z,\sigma)f_{\Sigma}(\sigma)d\sigma \\ &= \frac{1}{(\sigma_{\max}-\sigma_{\min})}\int_{\sigma_{\min}}^{\sigma_{\max}} f_{Z|\Sigma}(z,\sigma)d\sigma \end{aligned} \tag{3.40}$$

This expression cannot be extended analytically due to the presence of the integral of the Gaussian in $f_{Z|\Sigma}$ (3.34). However, it can be evaluated numerically. Similarly, the joint distribution of Y and Z can be found

$$f_{YZ}(y,z) = \frac{1}{(\sigma_{\max}-\sigma_{\min})}\int_{\sigma_{\min}}^{\sigma_{\max}} f_{YZ|\Sigma}(y,z,\sigma)d\sigma \tag{3.41}$$

Using Bayes rule, combine (3.40) and (3.41) to get the conditional PDF of Y given Z

$$\begin{aligned} f_{Y|Z}(y,z) &= \frac{f_{YZ}(y,z)}{f_Z(z)} \\ &= \frac{\int_{\sigma_{\min}}^{\sigma_{\max}} f_{YZ|\Sigma}(y,z,\sigma)d\sigma}{\int_{\sigma_{\min}}^{\sigma_{\max}} f_{Z|\Sigma}(z,\sigma)d\sigma} \end{aligned} \tag{3.42}$$

Finally, (3.42) can be integrated, again numerically, to find the cumulative distribu-

tion of Y given Z

$$F_{Y|Z}(y,z) = \int_{-\infty}^{y} f_{Y|Z}(u,z)du \tag{3.43}$$

Therefore, if the maximum value observed over the first N samples is z, then the probability that all of the next M samples will lie below y is $F_{Y|Z}(y,z)$. As the function $F_{Y|Z}(y,z)$ is a cumulative probability density function, it is monotonic in y and therefore invertible. Define the inverse G by

$$G_{NM}(z,\lambda) = y \Leftrightarrow F_{Y|Z}(y,z) = \lambda \tag{3.44}$$

The function G can be evaluted using a bisection search, for example. This is the basis of a bound-estimation algorithm. Given the measured maximum of N previous samples z, then with probability λ, the following M samples will not exceed $G(z,\lambda)$.

Fig. 3-8 shows the function $G_{NM}(z,\lambda)$ for probability values $\lambda = \{0.66, 0.95, 0.99\}$ with the other settings $\sigma_{\min} = 1$, $\sigma_{\max} = 3$, $N = M = 100$. Observe that higher values of the probability λ lead to higher bound predictions, as the algorithm becomes more "cautious" to achieve a more likely bound. Also, the functions level off at very low and very high values of the measured, past maximum. For instance, the $\lambda = 0.95$ line tends to $3\sigma_{\max}$ at the high end and $3\sigma_{\min}$ at the low end.

Remark 3.2. (Assumption of Independence) The formulation above assumed that the signal $X(k)$ is white and Gaussian. The assumption of a Gaussian distribution is common and justifiable via the central limit theorem. Independence (whiteness) is less realistic, as there will often be physical effects leading to correlation and filtering of the noise sources. However, the formulation used the independence assumption to ensure that the maximum over the first N samples Z is independent of the maximum of the next M samples Y. If the sampling windows M and N are chosen to be much longer than any correlation times present in the signal X, then it is reasonable that Y and Z should be *almost* independent.

Figs. 3-9 and 3-10 show some numerical results to verify intermediate steps in this analysis. Fig. 3-9 compares the expression (3.40) for the probability density of the

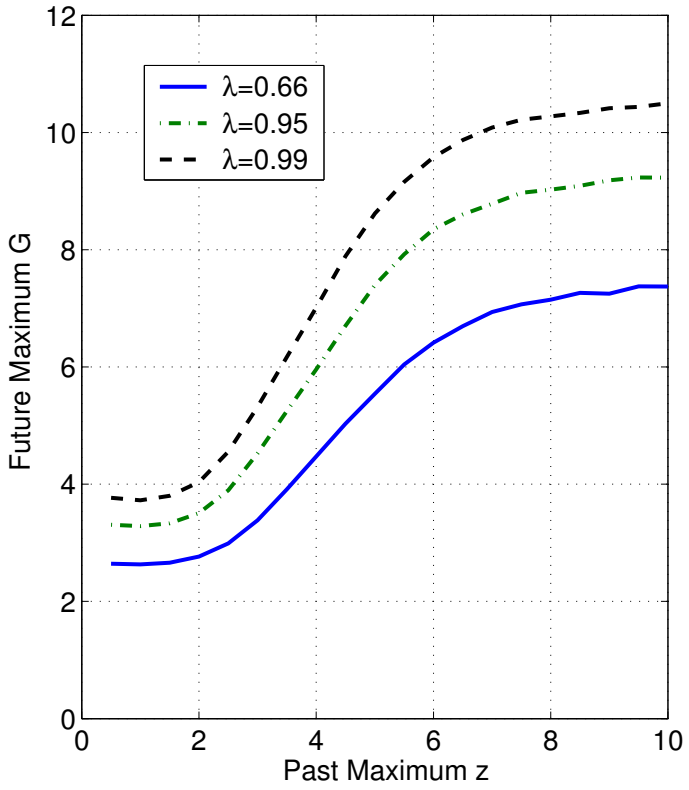


Figure 3-8: Bound Prediction Function $G_{NM}(z, \lambda)$ for Three Values of λ

maximum of a signal, Z, with a histogram of the maximum values of 1000 simulated instances. The two are a very good match. Fig. 3-10 verifies the threshhold prediction function from (3.44). The figure shows a plot of the parameter λ, on the X-axis, against the proportion of instances in 1000 random simulations in which the maximum value over the second 1000 time steps exceeded $G_{NM}(z, \lambda)$, with $N = M = 1000$ and z being the maximum value over the first 1000 steps. The analysis predicts that this proportion should also be λ, and indeed the graph is very close to the line $y = x$, confirming the prediction.

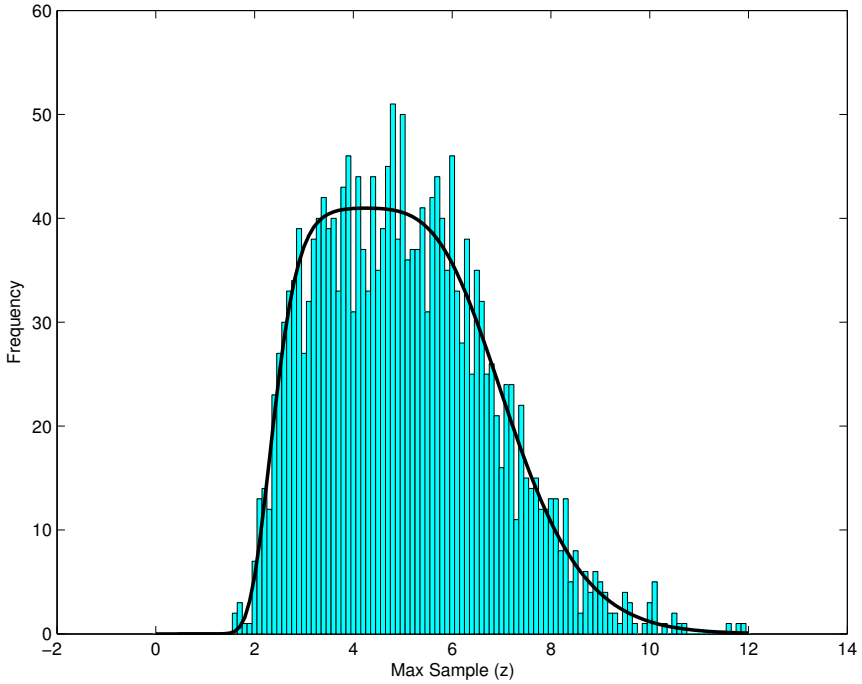


Figure 3-9: Verification of f_Z Probability Density Function

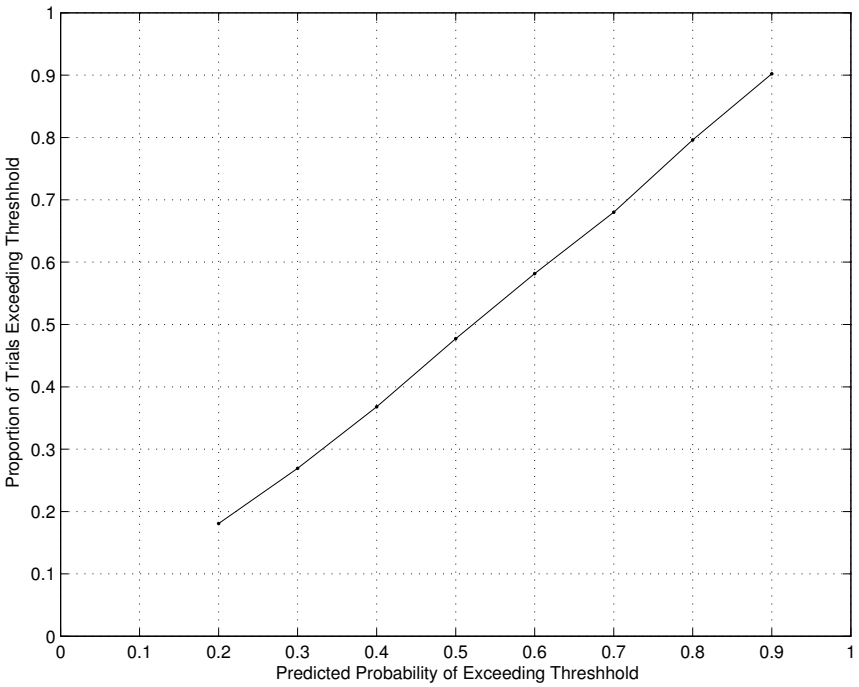


Figure 3-10: Verification of Probability Evaluation

Chapter 4

Decentralized MPC

This chapter presents a formulation for Decentralized Model Predictive Control (DMPC) of systems with coupled constraints. The single large planning optimization described in Chapter 2 is divided into smaller subproblems, each planning only for the states and controls of a particular subsystem. Relevant plan data is exchanged between susbsystems to ensure that all decisions satisfy the coupled constraints. The key property of the algorithm in this chapter is that if an initial feasible plan can be found, then all subsequent optimizations are guaranteed to be feasible, and hence the constraints will be satisfied, despite the action of unknown but bounded disturbances on each subsystem. This property is demonstrated in simulated examples, also showing the associated benefit in computation time.

The Decentralized Model Predictive Control (DMPC) is implemented for a team of cooperating Uninhabited Aerial Vehicles (UAVs). Collision avoidance is used as an example of coupled constraints and the chapter shows how the speed, turn rate and avoidance distance limits in the optimization should be modified in order to guarantee robust constraint satisfaction. Integer programming is used to solve the non-convex problem of path-planning subject to avoidance constraints. Numerical simulations are presented to compare computation time and target arrival time under decentralized and centralized control and investigate the impact of decentralization on team performance. The results show that the computation required for DMPC is significantly lower than for its centralized counterpart and scales better with the size

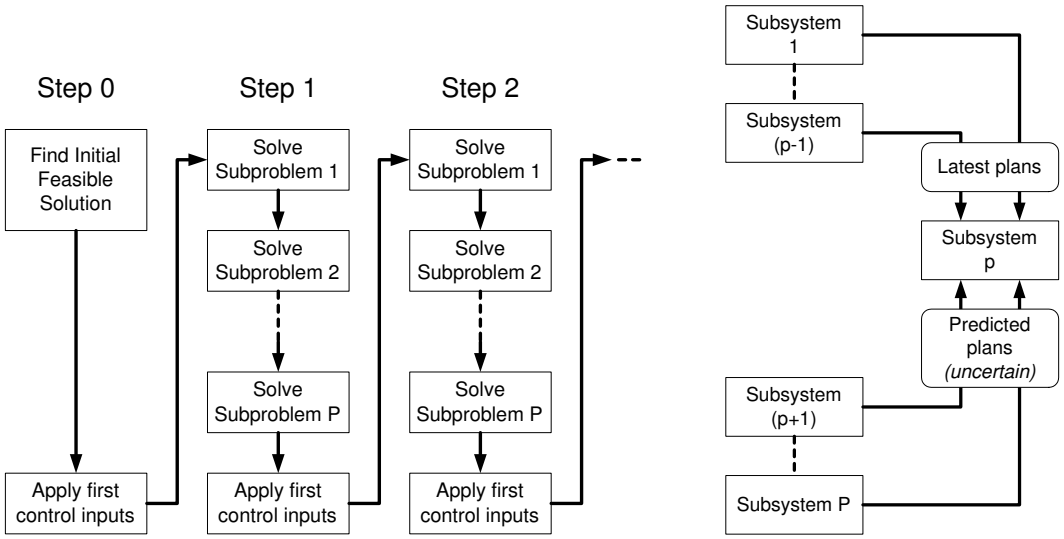
of the team, at the expense of only a small increase in UAV flight times.

4.1 Introduction

This chapter presents a *decentralized* form of Model Predictive Control (DMPC) for systems comprised of multiple subsystems, each with independent dynamics and disturbances but with coupled constraints. The principle of constraint tightening, used in Chapter 2 to accommodate uncertainty in future behavior, is applied in this chapter to account for uncertainty in the actions of other subsystems. The algorithm guarantees robust feasibility of all optimizations involved and satisfaction of the coupled constraints. Therefore, by embedding team performance goals in the constraints, cooperative behavior can be achieved without using a centralized controller.

Centralized MPC has been widely developed for constrained systems [3], with many results concerning stability [11, 48] and robustness [12, 20, 10], and has been applied to the co-operative control of multiple vehicles [10, 49, 50, 7]. However, solving a single optimization problem for the entire team typically requires significant computation, which scales poorly with the size of the system (*e.g.* the number of vehicles in the team). To address this computational issue, attention has recently focused on decentralized MPC [9], with various approaches, including robustness to the actions of others [51, 52], penalty functions [53, 54], and partial grouping of computations [55]. The key point is that, when decisions are made in a decentralized fashion, the actions of each subsystem must be consistent with those of the other subsystems, so that decisions taken independently do not lead to a violation of the coupled constraints. The decentralization of the control is further complicated when disturbances act on the subsystems, making the prediction of future behavior uncertain.

This chapter presents a new approach to DMPC that addresses both of these difficulties. The key features of this algorithm are that each subsystem (*e.g.* a vehicle) only solves a subproblem for its own plan, and each of these subproblems is solved only once per time step, without iteration. Under the assumption of bounded disturbances, each of these subproblems is guaranteed to be feasible, thus ensuring robust



(a) Procedure Flow (b) Information Flow

Figure 4-1: Overview of Decentralized Algorithm

constraint satisfaction across the group. The method employs at each time step a sequential solution procedure, outlined in Fig. 4-1(a), in which the subsystems solve their planning problems one after the other. The plan data relevant to the coupled constraints is then communicated to the other subsystems. Fig. 4-1(b) shows the information requirements for subproblem p. Each subproblem accommodates (a) the latest plans of those subsystems earlier in the sequence and (b) predicted plans of those later in the sequence. The disturbance is accommodated by including "margin" in the constraints, tightening them in a monotonic sequence, using an extension of ideas from Chapter 2. At initialization, it is necessary to find a feasible solution to the centralized problem, although this need not be optimal.

The latter part of this chapter devotes particular attention to DMPC for teams of cooperating Uninhabited Aerial Vehicles (UAVs). Mixed-Integer Linear Programming (MILP) path-planning [56, 38] is combined with DMPC to provide a decentralized algorithm for cooperative guidance of UAVs. The performance of this algorithm is investigated by simulating its behavior for many different multi-UAV collision avoidance problems. The avoidance constraints couple the behavior of the UAVs and

therefore exercise the ability of the controller to make cooperative decisions. The DMPC algorithm was designed primarily to ensure satisfaction of constraints, and critical performance goals can be embedded in these constraints. However, secondary goals remain in the cost function, *i.e.* quantities which should be small but have no explicit limit. This chapter investigates the effect of decentralization on these secondary goals. The metrics of interest in the UAV example are the flight time, which appears in the cost function of the MPC optimizations, and computation time. In particular, we investigate how the algorithm scales with the number of UAVs involved and compare with the behavior of the corresponding centralized MPC.

Section 4.2 presents the problem statement for decentralized MPC in a general form. Section 4.3 presents centralized MPC for the decentralized problem, used for comparison and initialization later in the chapter. Section 4.4 develops the DMPC algorithm and provides the robust feasibility result, with details of the proof in Section 4.A. Section 4.5 presents illustrative examples of DMPC in simulation. Section 4.6 develops the MPC algorithm for the special case of UAV collision avoidance. Simulation results for the UAV scenario are presented in Section 4.7. Finally, Section 4.8 shows how the problem should be modified to explicitly account for computation and communication delays.

4.2 Problem Statement

Consider the problem of controlling N_p subsystems. Each subsystem, denoted by subscript $p \in \{1, \ldots, N_p\}$, has linear, time-invariant, discretized dynamics

$$\mathbf{x}_p(k+1) = \mathbf{A}_p\mathbf{x}_p(k) + \mathbf{B}_p\mathbf{u}_p(k) + \mathbf{w}_p(k) \tag{4.1}$$

where $\mathbf{x}_p(k) \in \Re^{N_p^x}$ is the state vector of subsystem p at time k, $\mathbf{u}_p(k) \in \Re^{N_p^u}$ is the control input to subsystem p and $\mathbf{w}_p(k) \in \Re^{N_p^x}$ is the disturbance acting upon subsystem p. Assume all subsystems $(\mathbf{A}_p, \mathbf{B}_p)$ are controllable and the complete states $\mathbf{x}_p$ are available.

The disturbances are unknown *a priori* but lie in independent bounded sets

$$\forall k, p \quad \mathbf{w}_p(k) \in \mathcal{W}_p \subset \Re^{N_p^x} \tag{4.2}$$

The whole system is subjected to the following constraints upon summed outputs $\mathbf{y}_p(k) \in \Re^{N_y}$, which may include both decoupled constraints, such as individual control magnitude limits and state constraints, and coupled constraints, such as collision avoidance

$$\mathbf{y}_p(k) = \mathbf{C}_p\mathbf{x}_p(k) + \mathbf{D}_p\mathbf{u}_p(k) \tag{4.3}$$

$$\sum_{p=1}^{N_p} \mathbf{y}_p(k) \in \mathcal{Y} \subset \Re^{N_y} \ \forall k \tag{4.4}$$

where the matrices $\mathbf{C}_p$ and $\mathbf{D}_p$ for each p and the set $\mathcal{Y}$ are all chosen by the designer. The objective function is assumed to be decoupled between subsystems and therefore written as the following summation

$$J = \sum_{p=1}^{N_p} \sum_{k} \ell_p(\mathbf{x}_p(k), \mathbf{u}_p(k)) \tag{4.5}$$

A decoupled objective is typical of many problems of interest, such as fuel minimization for multiple spacecraft. This assumption is not strictly necessary for the robust feasibility result, which depends only on the constraints. However, decoupling the objectives is consistent with the approach taken here of using constraints to capture coupled behavior between subsystems. The requirement is to control the subsystems (4.1) such that the constraints (4.3) and (4.4) are satisfied for all disturbance sequences satisfying (4.2).

4.3 Centralized Robust MPC Problem

This section presents the centralized form of the robust MPC problem for the problem in Section 4.2. It is identical to the controller in Section 2.3, treating the group of subsystems as a single system. The centralized formulation is relevant here for

three reasons: first; because a feasible solution to the centralized problem is required to intialize the decentralized algorithm; second, because demonstrating satisfaction of the centralized constraints is an intermediate step in proving robustness of the decentralized algorithm; and third, because centralized MPC is used in examples later in this chapter for comparison with the new decentralized controller.

The online optimization approximates the complete problem in Section 4.2 by solving it over a finite horizon of N steps. A control-invariant terminal set constraint is applied to ensure stability [11]. Predictions are made using the nominal system model, *i.e.* (4.1) and (4.3) without the disturbance term. The output constraints are tightened in a monotonic sequence [10] to ensure robust feasibility, retaining a margin based on a particular candidate policy.

Define the centralized optimization problem

$$\mathsf{P}_C\left(\mathbf{x}_1(k),\ldots,\mathbf{x}_{N_p}(k)\right) \;:\; J^* = \min_{\left\{\begin{array}{c}\mathbf{u}_1\ldots\mathbf{u}_{N_p},\\ \mathbf{x}_1\ldots\mathbf{x}_{N_p},\\ \mathbf{y}_1\ldots\mathbf{y}_{N_p}\end{array}\right\}} \sum_{p=1}^{N_p}\sum_{j=0}^{N} \ell_p(\mathbf{x}_p(k+j|k),\mathbf{u}_p(k+j|k))$$

subject to $\forall j \in \{0\ldots N\}\ \forall p \in \{1\ldots N_p\}$

$$\begin{aligned}
\mathbf{x}_p(k+j+1|k) &= \mathbf{A}_p\mathbf{x}_p(k+j|k)+\mathbf{B}_p\mathbf{u}_p(k+j|k) && (4.6a)\\
\mathbf{y}_p(k+j|k) &= \mathbf{C}_p\mathbf{x}_p(k+j|k)+\mathbf{D}_p\mathbf{u}_p(k+j|k) && (4.6b)\\
\mathbf{x}_p(k|k) &= \mathbf{x}_p(k) && (4.6c)\\
\mathbf{x}_p(k+N+1|k) &\in \mathcal{Q}_p && (4.6d)\\
\sum_{p=1}^{N_p}\mathbf{y}_p(k+j|k) &\in \mathcal{Y}_{N_p}(j) && (4.6e)
\end{aligned}$$

where the constraint sets are found using the following recursions

$$\begin{aligned}
\mathcal{Y}_{N_p}(0) &= \mathcal{Y} && (4.7a)\\
\mathcal{Y}_{(p-1)}(j) &= \mathcal{Y}_p(j) \sim \left(\mathbf{C}_p+\mathbf{D}_p\mathbf{K}_p(j)\right)\mathbf{L}_p(j)\mathcal{W}_p\ \forall j\in\{0\ldots N\}\forall p\in\{2\ldots N_p\} && (4.7b)\\
\mathcal{Y}_{N_p}(j+1) &= \mathcal{Y}_1(j) \sim \left(\mathbf{C}_1+\mathbf{D}_1\mathbf{K}_1(j)\right)\mathbf{L}_1(j)\mathcal{W}_1\ \forall j\in\{0\ldots N-1\} && (4.7c)
\end{aligned}$$

where the operator "$\sim$" represents the Pontryagin difference (2.10) and $\mathbf{K}_p(j)$ $j \in \{0 \dots N-1\}$ are candidate control laws for each subsystem, with associated state transition matrices $\mathbf{L}_p(j)$ obeying

$$\begin{aligned} \mathbf{L}_p(0) &= \mathbf{I} && \text{(4.8a)} \\ \mathbf{L}_p(j+1) &= (\mathbf{A}_p + \mathbf{B}_p\mathbf{K}_p(j))\,\mathbf{L}_p(j) \;\; \forall j \in \{0 \dots N-1\} && \text{(4.8b)} \end{aligned}$$

The recursions (4.7) tighten the constraints at future plan steps, ensuring the existence of a margin to allow for future feedback action in response to disturbance. The tightening procedure is divided into separate steps for each subsystem. The reason for this will become apparent when the decentralized algorithm is presented in the next section. The centralized problem uses only the constraint sets for the last subsystem N_p. The choice of the candidate policies $\mathbf{K}_p(j)$ is left to the designer. The constraint tightening can be performed offline and, for polyhedral sets, the Pontryagin difference is implemented in the Matlab Invariant Set Toolbox [30].

The terminal constraint sets $\mathcal{Q}_p$ are given by

$$\mathcal{Q}_p = \mathcal{R}_p \sim \mathbf{L}_p(N)\mathcal{W}_p \tag{4.9}$$

where $\mathcal{R}_p$ are robust control invariant sets, with associated control laws κ_p satisfying

$$\begin{aligned} \mathbf{A}_p\mathbf{x}_p + \mathbf{B}_p\kappa_p(\mathbf{x}_p) + \mathbf{L}_p(N)\mathbf{w}_p &\in \mathcal{R}_p, && \text{(4.10a)} \\ &\qquad \forall \mathbf{x}_p \in \mathcal{R}_p,\ \forall \mathbf{w}_p \in \mathcal{W}_p,\ \forall p \in \{1 \dots N_p\} \\ \sum_{p=1}^{N_p} \{\mathbf{C}_p\mathbf{x}_p + \mathbf{D}_p\kappa_p(\mathbf{x}_p)\} &\in \mathcal{Y}_{N_p}(N), && \text{(4.10b)} \\ &\qquad \forall (\mathbf{x}_1^T \dots \mathbf{x}_{N_p}^T)^T \in \{\mathcal{Q}_1 \times \dots \times \mathcal{Q}_{N_p}\} \end{aligned}$$

Remark 2.2 discusses how to choose suitable terminal sets. In particular, Remark 2.8 discusses the significance of making $\mathbf{K}_p$ a nilpotent controller upon this choice.

Solving this optimization problem in a centralied MPC scheme using Algorithm 2.1 guarantees robust feasibility, according to Theorem 2.1.

4.4 Decentralized MPC Algorithm

This section describes a decentralized algorithm for solving the problem in Section 4.2. The centralized optimization P_C from Section 4.3 is divided into N_p subproblems, each involving the trajectory of only one subsystem. Fig. 4-1 shows an outline of the resulting algorithm. The subproblems are solved one after the other, but the overall scheme is not iterative: each subproblem is solved once per time step and is guaranteed to be feasible. The proof of robust feasibility is given at the end of this section.

Remark 4.1. (Time Delays) Latencies introduced by computation and communication are neglected in the initial development. Section 4.8 then shows how the same formulation can be modified to accommodate such delays. It will be shown that each subproblem must account only for the delay of its own computation, not for the whole solution sequence.

Define the p^{th} subproblem, with parameters $\tilde{\mathbf{y}}_p(k,\ldots,k+N|k)$, representing the intentions of the other subsystems, and $\mathbf{x}_p(k)$, the current state of subsystem p

$$\mathsf{P}_p\left(\mathbf{x}_p(k), \tilde{\mathbf{y}}_p(k\ldots k+N|k)\right) \;:\; J_p^* = \min_{\mathbf{u}_p,\mathbf{x}_p,\mathbf{y}_p} \sum_{j=0}^{N} \ell_p(\mathbf{x}_p(k+j|k), \mathbf{u}_p(k+j|k))$$

subject to $\forall j \in \{0\ldots N\}$

$$\begin{aligned}
\mathbf{x}_p(k+j+1|k) &= \mathbf{A}_p\mathbf{x}_p(k+j|k) + \mathbf{B}_p\mathbf{u}_p(k+j|k) && \text{(4.11a)}\\
\mathbf{y}_p(k+j|k) &= \mathbf{C}_p\mathbf{x}_p(k+j|k) + \mathbf{D}_p\mathbf{u}_p(k+j|k) && \text{(4.11b)}\\
\mathbf{x}_p(k|k) &= \mathbf{x}_p(k) && \text{(4.11c)}\\
\mathbf{x}_p(k+N+1|k) &\in \mathcal{Q}_p && \text{(4.11d)}\\
\mathbf{y}_p(k+j|k) + \tilde{\mathbf{y}}_p(k+j|k) &\in \mathcal{Y}_p(j) && \text{(4.11e)}
\end{aligned}$$

The output sequence $\tilde{\mathbf{y}}_p(k,\ldots,k+N|k)$ has two components: (a) the most recent plans of those subsystems earlier than p in the planning sequence and (b) predicted plans for subsystems later in the sequence. The predictions are constructed from the

remainder of the plan from the previous time-step and a single step of the control law κ_q, defined in (4.10), to keep the subsystem state in its terminal set

$$\begin{aligned}\tilde{\mathbf{y}}_p(k+j|k) &= \left[\sum_{q\in\{1\dots N_p|q<p\}} \mathbf{y}_q^*(k+j|k)\right] && \text{(4.12a)}\\ &+ \left[\sum_{q\in\{1\dots N_p|q>p\}} \mathbf{y}_q^*(k+j|k-1)\right] \ \forall j\in\{0\dots N-1\}\\ \tilde{\mathbf{y}}_p(k+N|k) &= \left[\sum_{q\in\{1\dots N_p|q<p\}} \mathbf{y}_q^*(k+N|k)\right] && \text{(4.12b)}\\ &+ \left[\sum_{q\in\{1\dots N_p|q>p\}} \mathbf{C}_q\mathbf{x}_q^*(k+N|k-1)+\mathbf{D}_q\kappa_q(\mathbf{x}_q^*(k+N|k-1))\right]\end{aligned}$$

where $\mathbf{y}_q^*(k+j|k)$ denotes the outputs from the solution to subproblem q at time k.

The output constraints in (4.11e) use the intermediate sets $\mathcal{Y}_p$ from the recursions in (4.7), whereas the centralized problem only used those for the final subproblem $\mathcal{Y}_{N_p}$. These recursions involve constraint tightening from one subproblem to the next and one time step to the next, consistent with the need to retain margin for both future disturbances and the intentions of those subsystems yet to plan. Tightening from one subproblem to the next is necessary because of the inclusion of predicted intentions $\mathbf{y}_q^*(k+j|k-1)$ for those subsystems later in the sequence $q>p$. These are nominal predictions only and the actual plans of those subsystems will differ due to the disturbances, hence margin must be retained for those modifications.

The subproblems P_p are employed in the following algorithm, also shown outlined in Fig. 4-1.

Algorithm 4.1. (Decentralized MPC)

1. Find a solution to the initial centralized problem $\mathsf{P}_C(\mathbf{x}_1(0),\dots,\mathbf{x}_{N_p}(0))$. If a solution cannot be found, stop (problem is infeasible).
2. Set $k=0$
3. Apply control $\mathbf{u}_p^*(k|k)$ to each subsystem p

4. Increment k
5. For each subsystem p in order $1, \ldots, N_p$:

   (a) Gather, by communication, the plan data $\tilde{\mathbf{y}}_p(k, \ldots, k+N|k)$ from other subsystems, defined by (4.12)

   (b) Solve subproblem $\mathsf{P}_p\left(\mathbf{x}_p(k), \tilde{\mathbf{y}}_p(k, \ldots, k+N|k)\right)$
6. Go to 3

Note that Step 1 does not require finding the optimal solution of the centralized problem, although that is one way of finding the initial plan. Remark 4.3 discusses the implications of the ordering required in Step 5 and how it can be changed during operation if required.

Theorem 4.1. ***(Robust Feasibility)*** *If a feasible solution to the initial centralized problem $\mathsf{P}_C(\mathbf{x}_1(0), \ldots, \mathbf{x}_{N_p}(0))$ can be found, then the systems (4.1), subjected to disturbances obeying (4.2) and controlled using Algorithm 4.1, are robustly feasible and satisfy constraints (4.3) and (4.4).*

Proof An outline of the proof is presented here, with the details in Section 4.A.

It is necessary to consider only one time step and show that feasibility at time k_0 implies feasibility at time $k_0 + 1$ for all disturbances $\mathbf{w}_p(k_0)$ obeying (4.2). Then, by recursion, feasibility at time 0 implies feasibility at all future steps. The recursion is broken into multiple stages:

1. Assume a set of solutions is known for all subsystems at time k_0 satisfying the centralized problem $\mathsf{P}_C(k_0)$ (where the shortened index notation $\mathsf{P}(k_0)$ denotes the problem for the states at that time $\mathsf{P}(\mathbf{x}_1(k_0), \cdots)$);

2. Show that, given the assumption in Step 1, a feasible solution can be found to the first subproblem $\mathsf{P}_1(k_0+1)$ for all disturbances $\mathbf{w}_1(k_0)$ by showing feasibility of a candidate solution. The candidate is constructed by shifting the previous plan for subsystem 1, assumed known in Step 1, by one time step and adding a perturbation sequence using the predetermined controller $\mathbf{K}_1$;

3. Show that, under the assumption in Step 1, given any solution to the subproblem $\mathsf{P}_{p_0}(k_0+1)$ for $p_0 \in \{1,\ldots,N_p-1\}$ then the next subproblem $\mathsf{P}_{(p_0+1)}(k_0+1)$ is feasible, again by showing feasibility of a candidate sequence. The candidate is constructed by shifting the previous plan for subsystem p_0+1, assumed known in Step 1, by one time step and adding a perturbation sequence using the predetermined controller $\mathbf{K}_{(p_0+1)}$;

4. Show that if a feasible solution to the final subproblem $\mathsf{P}_{N_p}(k_0+1)$ can be found, then the set of solutions to all subproblems $\mathsf{P}_p(k_0+1)$ for all $p \in \{1,\ldots,N_p\}$ satisfies the constraints of the centralized problem $\mathsf{P}_C(k_0+1)$

So the recursion sequence is:

solutions at k_0 satisfy $\mathsf{P}_C(k_0)$ (Step 1 of Proof)

$\Downarrow$

subproblem $\mathsf{P}_1(k_0+1)$ is feasible (by Step 2)

$\Downarrow$

subproblem $\mathsf{P}_2(k_0+1)$ is feasible (by Step 3)

$\Downarrow$

$\vdots$

$\Downarrow$

subproblem $\mathsf{P}_{N_p}(k_0+1)$ is feasible (by Step 3)

$\Downarrow$

solutions at k_0+1 satisfy $\mathsf{P}_C(k_0+1)$ (by Step 4) $\square$

This recursion sequence is analogous to the procedure flow in Fig. 4-1(a): the feasibility of each optimization problem follows from the feasibility of the one immediately before it.

Remark 4.2. (Communication Requirements) The formulation presented involves a general form of team-wide output constraints, with all constraints, coupling and individual, represented in the same output vector. However, each subsystem is

typically only affected by a subset of these constraints, and this structure can be exploited to reduce the communication load. Two circumstances in particular offer significant simplification: (a) if a constraint does not affect a particular subsystem, it can be ignored in that subproblem, and (b) if a constraint applies to only one subsystem, it can be ignored by all others. These cases can be expressed formally in terms of the output matrices. For (a), if all entries in row i of matrices $\mathbf{C}_p$ and $\mathbf{D}_p$ are zero, then the corresponding output and constraints can be omitted from subproblem P_p and subsystem p does not need to receive information on that output from other subsystems. For (b), if all entries in row i of matrices $\mathbf{C}_q$ and $\mathbf{D}_q$ for all *other* subsystems $q \neq p$ are zero, then output i and the corresponding constraints affect *only* subsystem p. Then subproblem P_p must still include output i and enforce constraints upon it, but no information on that output need be exchanged with other subsystems.

Remark 4.3. (Planning Order) Algorithm 4.1 involves solving subproblems in a particular order, 1 to N_p. The assignment of positions in this order to physical subsystems, *e.g.* vehicles, is arbitrary and determined by the designer. However, if required, the ordering can be changed between successive time steps, *i.e.* immediately before Step 5 of Algorithm 4.1, provided the constraints (4.7) are reformed accordingly. For example, in a problem with three subsystems using the initial planning order "ABC", the order may be changed to any other combination after subsystem "C" completes its plan at some time-step. The ability to change ordering follows from the fact that the combined solutions to all subproblems must satisfy the constraints of the centralized problem, which are independent of subsystem ordering. Thus the subsystems can be reordered without violating the assumption in Step 1 of the Proof of Theorem 4.1. No clear pattern has been found governing the effect of planning order on system performance. Simulation results in Section 4.7 address this issue.

Remark 4.4. (Set Approximation) In some cases, the Pontryagin differences in the set recursions (4.7) can lead to an increase in the total number of constraints in the problem [31]. In these circumstances, it may be desirable to use inner approximations

of the constraint sets, leading to a simpler but conservative controller. Therefore, in place of the sets defined in (4.7), the designer may substitute any sets obeying the following conditions

$$\begin{aligned}
\mathcal{Y}_{N_p}(0) &\subseteq \mathcal{Y} \\
\mathcal{Y}_{(p-1)}(j) &\subseteq \mathcal{Y}_p(j) \sim (\mathbf{C}_p + \mathbf{D}_p\mathbf{K}_p(j))\,\mathbf{L}_p(j)\mathcal{W}_p \;\forall j \in \{0 \dots N\}\; \forall p \in \{2 \dots N_p\} \\
\mathcal{Y}_{N_p}(j+1) &\subseteq \mathcal{Y}_1(j) \sim (\mathbf{C}_1 + \mathbf{D}_1\mathbf{K}_1(j))\,\mathbf{L}_1(j)\mathcal{W}_1 \;\forall j \in \{0 \dots N-1\}
\end{aligned}$$

This is the extension of the set approximation from Remark 2.7 to the decentralized problem. It is often possible to use techniques such as norm bounding and induced norms to find suitable sets. Constraint approximation is applied in the robust UAV example in Section 4.6.2 later in this chapter, using simple obstacle enlargement in place of a more complicated Pontryagin difference operation.

Remark 4.5. ("Anytime" Computation) The result in Theorem 4.1 depends only on finding *feasible* solutions to each optimization, not necessarily *optimal* solutions. This is true for both the centralized initialization problem in Step 1 of Algorithm 4.1 and the decentralized subproblems in Step 5b. Furthermore, the robust feasibility result follows from the ability to construct a feasible solution to each problem before starting the optimization. This brings two advantages, first that this solution may be used to initialize a search procedure, and second that the optimization can be terminated at any time knowing that a feasible solution exists and can be employed for control.

4.5 DMPC Examples

The first example involves a simple problem in which the positions of two point masses moving in 1-D are to remain within a specified distance of each other. The dynamics of each subsystem are identical to those of the example system in Section 2.5.1. The constraints limit the positions of each to within ± 1 of the origin, the actuation force to less than 1 in magnitude, and the distance between the two masses to be less

than 0.3. The constraints in the form of (4.3) and (4.4) are

$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{0.3} & 0 \end{bmatrix} \quad \mathbf{D}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{C}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ -\frac{1}{0.3} & 0 \end{bmatrix} \quad \mathbf{D}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathcal{Y} = \{\mathbf{y} \in \Re^5 \mid \|\mathbf{y}\|_\infty \leq 1\}$$

As in Section 2.5.1, the cost function is the control energy. A random disturbance force of up to 0.2 in magnitude acts on each mass. Referring to Remark 4.2 about communication requirements, note that the outputs y_1 and y_2 depend only on subsystem 1, hence their constraints can be ignored in subproblem 2, and *vice versa* for outputs y_3 and y_4. Only output y_5 involves both subsystems, so only plan data for this output is exchanged between the subsystems in the DMPC algorithm.

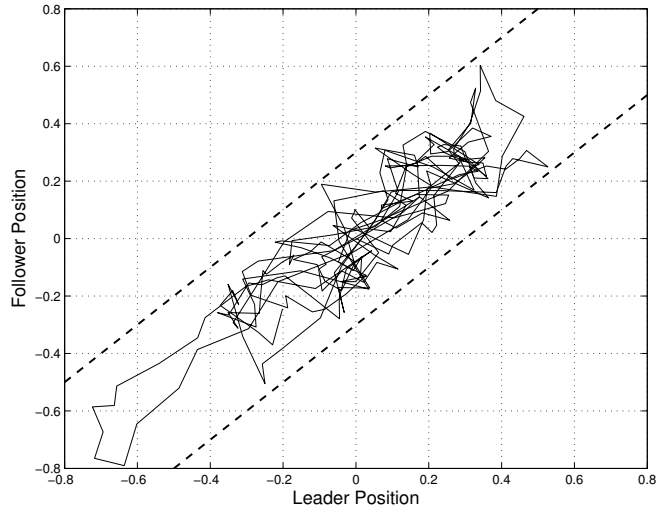
For comparison, the problem is solved by three different methods, centralized MPC, decentralized MPC and a decoupled approach, in which each position is individually constrained to remain within 0.15 m of the origin, ensuring satisfaction of the coupled constraints without solving a coupled problem. Fig. 4-2(a) is a graph of the position states of the follower plotted against those of the leader, both under the control of centralized MPC. The coupling constraint requires that the states remain between the dashed lines. The figure shows that the constraints are satisfied at all times. Fig. 4-2(b) shows the results from the decentralized algorithm, also obeying the constraints at all times and giving a similar result to centralized MPC. Fig. 4-2(c) shows the results from the decoupled method. The dashed box shows the decoupled constraints used to enforce satisfaction of the coupling constraints. The results satisfy the coupling constraints but it is clear that the problem is far more constrained than either of the other two methods. Note that the results from the decentralized method in Fig. 4-2(b) do not remain within any possible decoupling constraints. Table 4.1 compares the total control energy used by the three algorithms, averaged over four

Table 4.1: Performance Comparison of Algorithms: total control energy cost averaged over four randomly-disturbed simulations.

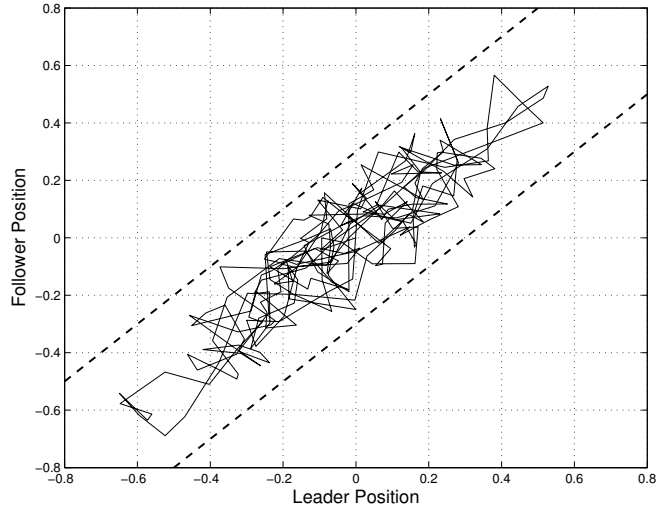
Algorithm	Control Cost
Centralized	1.94
Decentralized	2.51
Decoupled	3.71

different simulations, each with randomly-chosen disturbances. The centralized algorithm has the lowest cost, as it finds the system-wide optimal solution to the coupled problem. The decoupled method has the highest cost, since it involves artificially tight constraints. As might be expected, the cost of the decentralized method lies in between that of the other two methods.

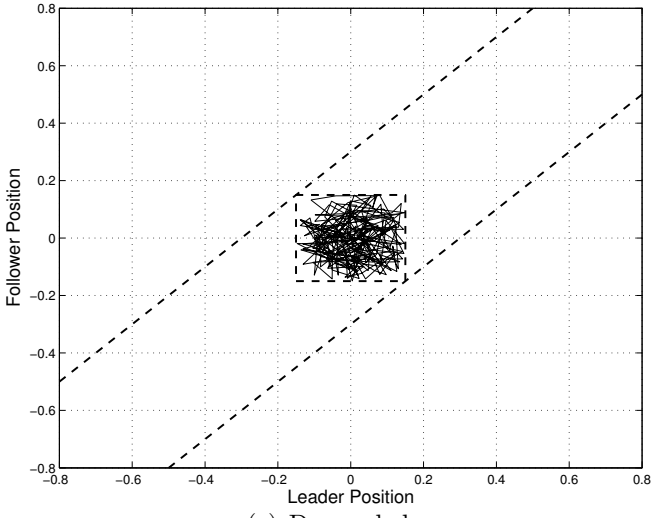
Fig. 4-3 shows the position time histories under decentralized MPC for a similar problem with six subsystems, each the same double-integrator as before and each pair constrained to remain within 0.3 m. The lower plot in Fig. 4-3 shows that the maximum separation is always less than 0.3 m, hence the coupling constaints are satisfied. Note that in the upper plot, the positions move beyond the ± 0.15 m range of the decoupled solution, showing that the decentralized method is again making use of its greater constraint space. Fig. 4-4 compares averaged solution times using the centralized and decentralized MPC schemes. The decentralized MPC subproblem times are shown stacked as they are solved sequentially. These results show that the decentralized MPC has faster computation and is also more scalable. Fig. 4-5 compares the control RMS results for the different controllers for different numbers of subsystems. The MPC objective function was to minimize control energy. As expected, DMPC uses slightly more control effort in every case, as it solves the same problem as centralized MPC but in a more constrained manner. However, the increase is small. Also note that there is no clear advantage to planning first or last in the sequence: the control effort is distributed roughyl equally between the subsystems.



(a) Centralized



(b) Decentralized



(c) Decoupled

Figure 4-2: Follower Positions vs. Leader Position using Three Forms of MPC. The dashed lines mark the constraints.

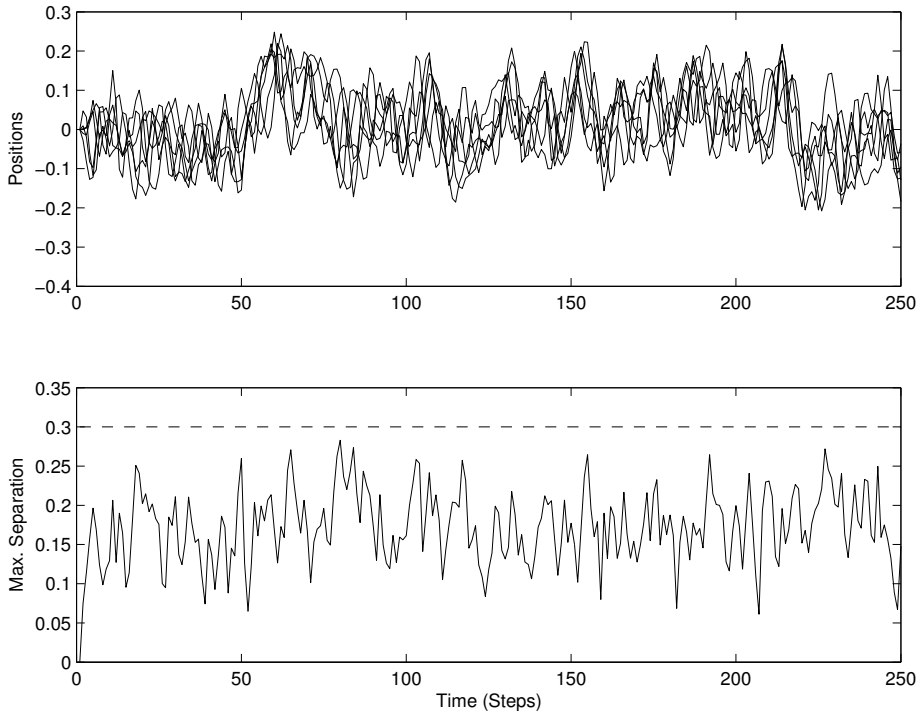


Figure 4-3: Position Time Histories (top) and Maximum Separation History (bottom) under DMPC.

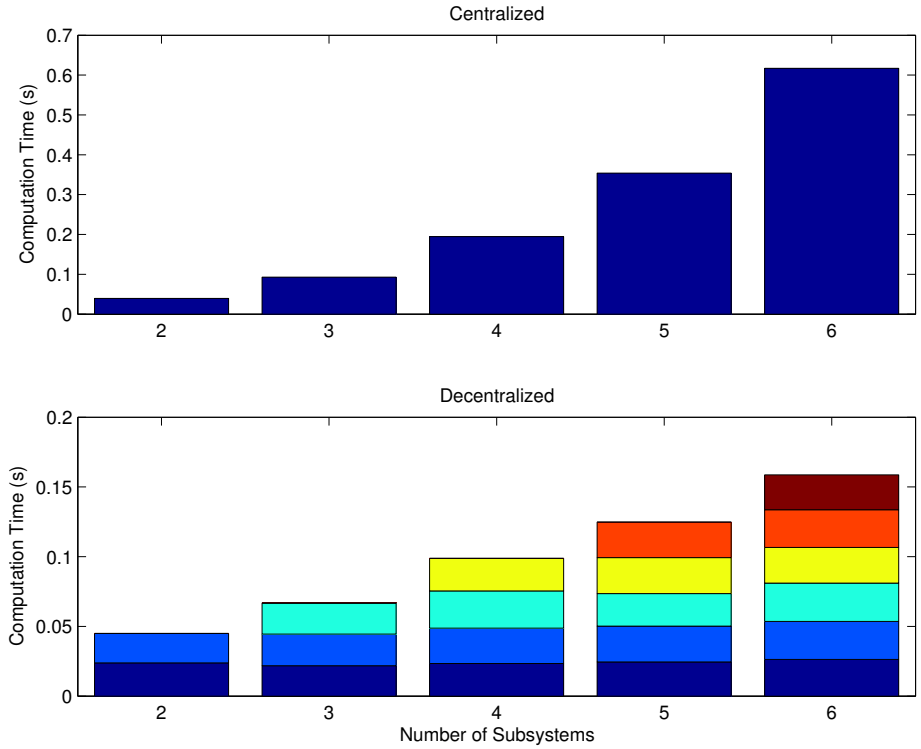


Figure 4-4: Solution time comparisons of MPC and DMPC for different numbers of subsystems. Note different vertical scales.

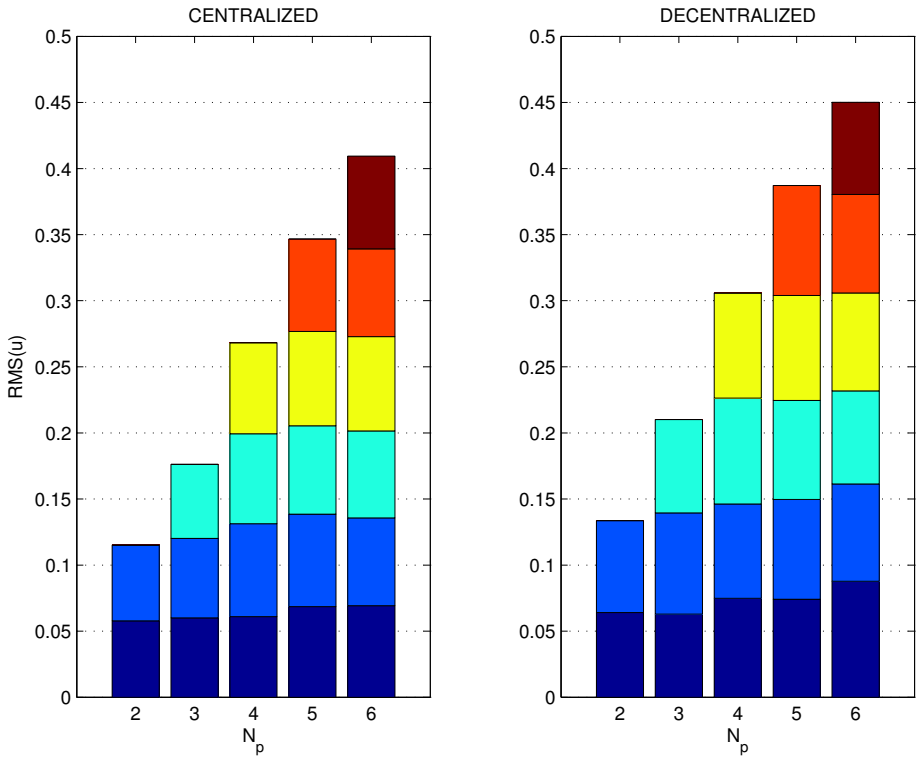


Figure 4-5: Control RMS comparisons of MPC and DMPC for different numbers of subsystems, broken down by individual subsystem

4.6 DMPC for UAVs

This section applies the decentralized MPC formulation to UAV collision avoidance. The section begins with the multi-UAV problem statement in Section 4.6.1. Then Section 4.6.2 lays out the control formulation for this problem. Section 4.7 gives the results of simulations using this controller for many different scenarios and team sizes.

4.6.1 UAV Problem Statement

The problem is to control multiple UAVs, each assumed to fly at constant altitude and with restricted speed and rate of turn. Each UAV has a single pre-assigned goal point. The objective is for all UAVs to reach their goals in minimum time without colliding, *i.e.* maintaining a minimum separation between every pair of UAVs at all times. This problem statement exercises the ability of the decentralized method to handle non-convex constraints and coupling between the actions of the UAVs.

Since the decentralized MPC method requires linear dynamics, we adopt the linear approximation of aircraft dynamics from Ref. [38]. Each vehicle is modeled as a unit point mass moving in two dimensions subject to speed and force limits. A disturbance force is also included in the model to account for uncertainty. Let UAV p have position $\mathbf{r}_p(t) \in \Re^2$ and velocity $\mathbf{v}_p(t) \in \Re^2$ and be acted on by a control force $\mathbf{f}_p(t) \in \Re^2$ and a disturbance force $\tilde{\mathbf{f}}_p(t) \in \Re^2$ giving dynamics

$$\dot{\mathbf{r}}_p(t) = \mathbf{v}_p(t) \quad \dot{\mathbf{v}}_p(t) = \mathbf{f}_p(t) + \tilde{\mathbf{f}}_p(t)$$

and constraints

$$\|\mathbf{v}_p(t)\|_2 \leq V_p \quad \|\mathbf{f}_p(t)\|_2 \leq F_p$$

(This model becomes equivalent to Dubins' car model [57] with the addition of a minimum speed constraint. This can be implemented using MILP, but was not done in these experiments for simplicity.) The disturbance force is assumed to be unknown but bounded

$$\|\tilde{\mathbf{f}}_p(t)\|_\infty \leq \tilde{F}_p$$

The collision avoidance is expressed in terms of a square exclusion region of side $2L$ around each UAV which no other UAV may enter

$$\|\mathbf{r}_p(t) - \mathbf{r}_q(t)\|_\infty \geq L \ \forall t, p, q : q \neq p$$

Finally, let the goal of UAV p be $\mathbf{g}_p \in \Re^2$.

4.6.2 DMPC for UAVs

This section presents the robust MPC optimization problems for the UAV problem described in Section 4.2 and the algorithm in which they are employed. Both centralized and decentralized optimizations are presented. The centralized form is relevant because it is used to initialize the decentralized algorithm and also used in comparison tests in Section 4.7.

The MPC algorithm requires a linear discrete-time state-space dynamics model and linear, although not necessarily convex, constraints. It is straightforward to convert the dynamics to the standard state space form of (4.1) with state $\mathbf{x}_p(k) = [\mathbf{r}_p(kT)^T \ \mathbf{v}_p(kT)^T]^T$, control input as $\mathbf{u}_p(k) = \mathbf{f}_p(kT)$, disturbance $\mathbf{w}_p(k) = \tilde{\mathbf{f}}_p(kT)$ and time step T. The speed and force limits are approximated by polyhedra and applied at each time step, along with the collision avoidance constraints

$$[\mathbf{0} \ \mathbf{G}]\mathbf{x}_p(k) \leq \mathbf{1}V_p \tag{4.13a}$$

$$\mathbf{G}\mathbf{u}_p(k) \leq \mathbf{1}F_p \tag{4.13b}$$

$$\|[\mathbf{I} \ \mathbf{0}]\left(\mathbf{x}_p(k) - \mathbf{x}_q(k)\right)\|_\infty \geq L' \ \forall p, q : q \neq p \tag{4.13c}$$

where the rows of $\mathbf{G}$ are unit vectors and $L' > L$ includes some additional margin to account for the the discrete-time application of the constraints. Ref. [38] showed the use of binary logical variables within MILP to encode non-convex avoidance constraints such as (4.13c).

Centralized Optimization

$$\mathsf{P}_C\left(\mathbf{x}_1(k),\ldots,\mathbf{x}_P(k)\right) \;:\; J_C^* = \min_{\left\{\begin{smallmatrix}\mathbf{u}_1, & \mathbf{x}_1, & T_1\\ \vdots & \vdots & \vdots\\ \mathbf{u}_{N_p}, & \mathbf{x}_{N_p}, & T_{N_p}\end{smallmatrix}\right\}} \sum_{p=1}^{P} T_p$$

subject to $\forall p \in \{1\ldots P\}\ \forall j \in \{0\ldots(T_p-1)\ \forall q > p\}$

$$\begin{aligned}
\mathbf{x}_p(k+j+1|k) &= \mathbf{A}\mathbf{x}_p(k+j|k) + \mathbf{B}\mathbf{u}_p(k+j|k)\\
\mathbf{x}_p(k|k) &= \mathbf{x}_p(k)\\
[\mathbf{0}\ \mathbf{G}]\,\mathbf{x}_p(k+j|k) &\le \mathbf{1}(\bar{V}_p - c_p(j))\\
\mathbf{G}\mathbf{u}_p(k+j|k) &\le \mathbf{1}(\bar{f}_p - d_p(j))\\
\|[\mathbf{I}\ \mathbf{0}]\mathbf{x}_p(k+T_p|k) - \mathbf{g}_p\|_\infty &\le a_p(T_p)\\
\|[\mathbf{I}\ \mathbf{0}]\mathbf{x}_p(k+j|k) - \mathbf{r}_q(k+j|k)\|_\infty &\ge L' + b_{pq}(j)
\end{aligned}$$

where the quantities a, b, c, d are modifications to the constraints to provide margin for robustness

$$\begin{aligned}
a_p(0) &= 0\\
a_p(1) &= \|[1\ 0\ 0\ 0]\mathbf{B}\|_2\,\tilde{F}_p\\
a_p(j) &= a_p(1) + \|[1\ 0\ 0\ 0]\mathbf{L}\mathbf{B}\|_2\tilde{F}_p \qquad \forall j \ge 2\\
b_{pq}(j) &= a_p(j) + a_q(j) \qquad \forall j\\
c_p(0) &= 0\\
c_p(1) &= \|[0\ 0\ 1\ 0]\mathbf{B}\|_2\tilde{F}_p\sqrt{2}\\
c_p(j) &= c_p(1) + \|[0\ 0\ 1\ 0]\mathbf{L}\mathbf{B}\|_2\tilde{F}_p\sqrt{2} \quad \forall j \ge 2\\
d_p(0) &= 0\\
d_p(1) &= \|[1\ 0]\mathbf{K}\mathbf{B}\|_2\tilde{F}_p\sqrt{2}\\
d_p(j) &= d_p(1) + \|[1\ 0]\mathbf{K}\mathbf{L}\mathbf{B}\|_2\tilde{F}_p\sqrt{2} \qquad \forall j \ge 2
\end{aligned}$$

where $\mathbf{K}$ is chosen to be the nilpotent controller for the system $(\mathbf{A}, \mathbf{B})$ and $\mathbf{L} = (\mathbf{A} + \mathbf{B}\mathbf{K})$. A suitable controller $\mathbf{K}$ can be found since the system is a combination of two independent double-integrators. Since $\mathbf{K}$ is chosen to make the system nilpotent, only two steps of constraint tightening are needed and the sets are constant thereafter.

Decentralized Optimization (for UAV p)

$$\mathsf{P}_p\left(\mathbf{x}_p(k), \tilde{\mathbf{r}}_{pq}(k \ldots k+T|k)\right) \;:\; J_p^* = \min_{\mathbf{u}_p, \mathbf{x}_p, T_p} T_p$$

subject to $\forall j \in \{0 \ldots T_p - 1\}\ \forall q \neq p$

$$\begin{aligned}
\mathbf{x}_p(k+j+1|k) &= \mathbf{A}_p\mathbf{x}_p(k+j|k) + \mathbf{B}_p\mathbf{u}_p(k+j|k) \\
\mathbf{x}_p(k|k) &= \mathbf{x}_p(k) \\
[\mathbf{0}\ \mathbf{G}]\,\mathbf{x}_p(k+j|k) &\leq \mathbf{1}(\bar{V}_p - c_p(j)) \\
\mathbf{G}\mathbf{u}_p(k+j|k) &\leq \mathbf{1}(\bar{f}_p - d_p(j)) \\
\|[\mathbf{I}\ \mathbf{0}]\mathbf{x}_p(k+T_p|k) - \mathbf{g}_p\|_\infty &\leq a_p(T_p) \\
\|[\mathbf{I}\ \mathbf{0}]\mathbf{x}_p(k+j|k) - \tilde{\mathbf{r}}_{pq}(k+j|k)\|_\infty &\geq L' + \hat{b}_{pq}(j)
\end{aligned}$$

where $\tilde{\mathbf{r}}_{pq}(k \ldots k+T|k)$ represents the latest known intentions of other vehicles, equivalent to the outputs $\tilde{\mathbf{y}}_p$ in (4.12), rationalized according to Remark 4.2 to communicate only the data that couples the subsystems.

$$\tilde{\mathbf{r}}_{pq}(k \ldots k+j|k) = \begin{cases} [\mathbf{I}\ \mathbf{0}]\mathbf{x}_q(k+j|k) & q < p \\ [\mathbf{I}\ \mathbf{0}]\mathbf{x}_q(k+j|k-1) & q > p \end{cases}$$

This comprises the latest plans for those UAVs before p in the planning sequence and projected plans, using the continuation of the previous plans, for those after p. This data is exchanged by communication between UAVs and appears as a constrant parameter in sub-problem p.

Only the avoidance margins $\hat{b}$ are modified for the decentralized problem, as these are the only constraints that couple the UAVs.

$$\begin{aligned}
\hat{b}_{pq}(0) &= 0 && q < p \\
\hat{b}_{pq}(0) &= a_q(0) && q > p \\
\hat{b}_{pq}(1) &= a_p(0) + a_q(0) && q < p \\
\hat{b}_{pq}(1) &= a_p(0) + a_q(1) && q > p \\
\hat{b}_{pq}(j) &= a_p(1) + a_q(1) && q \neq p\ \forall j \geq 2
\end{aligned}$$

This accounts for the additional uncertainty in the projected plans of later vehicles, as the actual plans of those UAVs will differ from the predictions due to the action of the disturbances. The other constraint margins a, c, d are the same as used in the centralized problem.

4.7 UAV Simulation Results

This section presents results of numerical simulations using the optimizations P_C and P_p from Section 4.6.2 within DMPC (Algorithm 4.1). In particular, we investigate its performance in comparison to the centralized counterpart (CMPC). One hundred random instances of problems for each team size between two and seven were generated and simulated using DMPC. In every case, the UAVs began on the line between $(-5, 5)$ and $(-5, -5)$, heading in the $+X$ direction. The goals were chosen randomly on the line between $(5, 5)$ and $(5, -5)$. Fig. 4-6 shows a representative example of a problem of each size. The simulation model included a randomly-generated disturbance force of up to 10% of the control force. The simulations were run in Matlab on a desktop PC with Pentium 2.2 GHz processor and 512MB RAM using CPLEX 7.0 for MILP optimization. The same computer was used to solve all UAV sub-problems, consistent with the sequential nature of the algorithm. For comparison, the same fifty instances for team sizes up to five were also simulated, on the same computer, using the CMPC. The same initial solution was used for both algorithms, with CPLEX configured to search primarily for feasible, rather than optimal, solutions and taking the first one found by its search procedure. For the on-line optimizations, CPLEX was configured to seek optimal solutions and subjected to a ten minute computation time-limit, after which the best feasible solution available was employed. The larger problems, with six and seven UAVs, were not attempted using the centralized method as the computation times became prohibitively long.

In all the scenarios, all UAVs reached their respective goals without incurring infeasibility and maintaining the required separations. The examples in Fig. 4-6 show how the UAVs divert from their direct routes to their goals to avoid collisions.

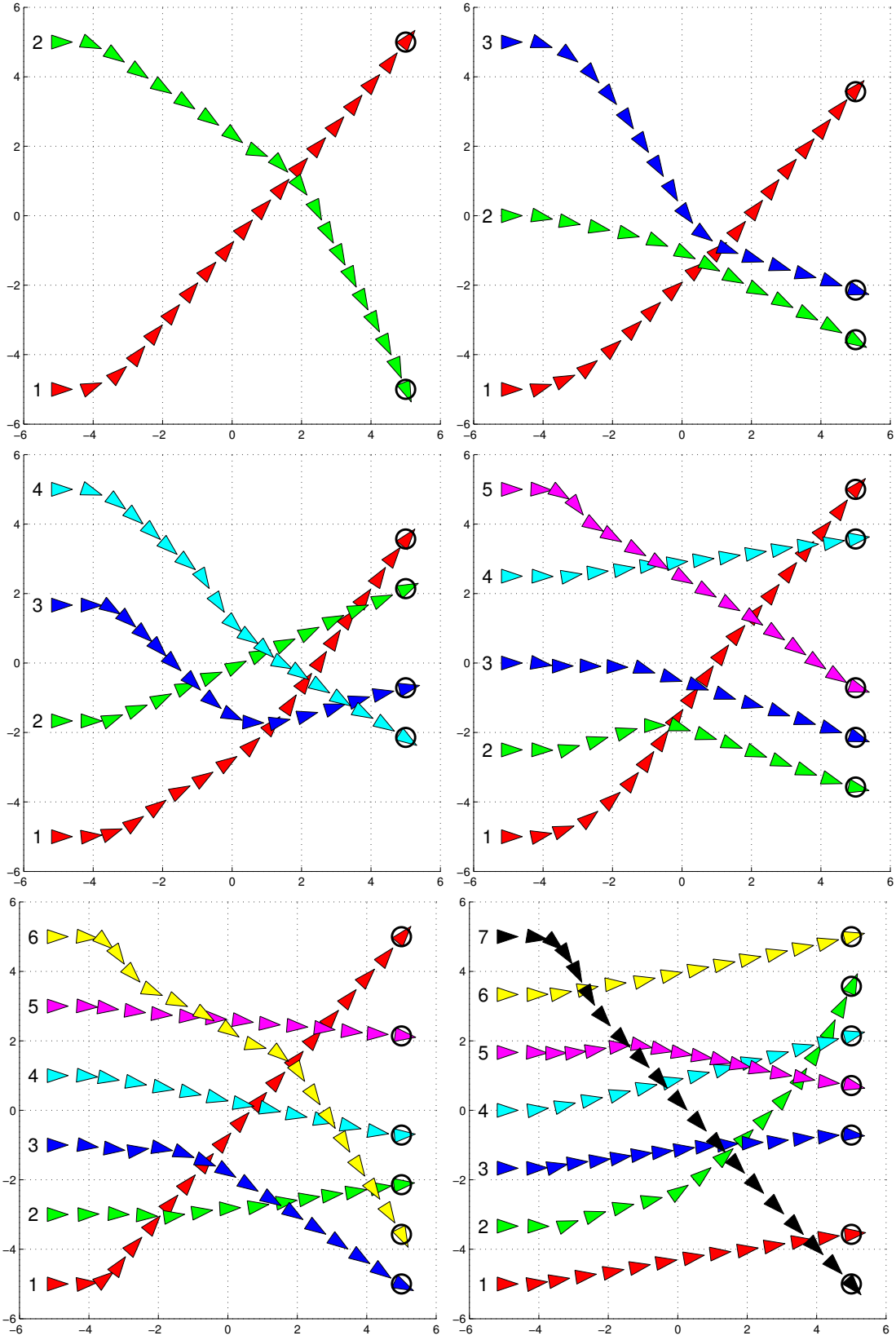


Figure 4-6: Example Trajectories for Simulations of Teams of 2-7 UAVs using Decentralized MPC. The numbers denote the planning order and circles mark the goals.

The order of planning, outlined in Fig. 4-1(a), is marked in the figures. The results indicate that the significance of planning order is unclear and highly problem specific. In the two-, three- and six-vehicle cases, UAV 1 goes straight to its goal, but in the four- and five-vehicle cases, UAV 1 diverts to avoid others. Thus there is no apparent consistent advantage to planning first or last.

4.7.1 Computation Time Comparison

Fig. 4-7 and Table 4.2 present data comparing the total computation times for the centralized and decentralized algorithms. Only computation times for the second time step are considered, as the problem complexity varies as UAVs proceed towards their goals. A computation time limit of ten minutes was applied to the optimization in the centralized case, leading to the sharp cut-off at 600 seconds. The results for the decentralized algorithm are the summed times for all the sub-problems, since they are solved sequentially as shown in Fig. 4-1(a). Looking at the mean times (black dot) and the standard devation envelopes (dark gray bar), it is clear that the decentralized method offers a considerable computational advantage, averaging over twenty times faster than centralized for problems of five UAVs. The ranges (light gray bar) show that there are, however, some rare cases (only one instance in these experiments) in which the DMPC controller can take over ten minutes to solve. The DMPC computation load still grows nonlinearly with team size, to be expected as the inherent complexity of the problem grows as the UAV "density" increases. However, it is clear that the rate of increase is much slower than for CMPC. Fig. 4-8 shows the average solution times for the decentralized MPC experiments broken down by sub-problem. This shows that there is no clear pattern to the distribution of computation between subproblems.

4.7.2 Flight Time Comparison

The results in Section 4.7.1 showed that the DMPC algorithm offers a considerable improvement over its centralized counterpart in terms of computation time. This

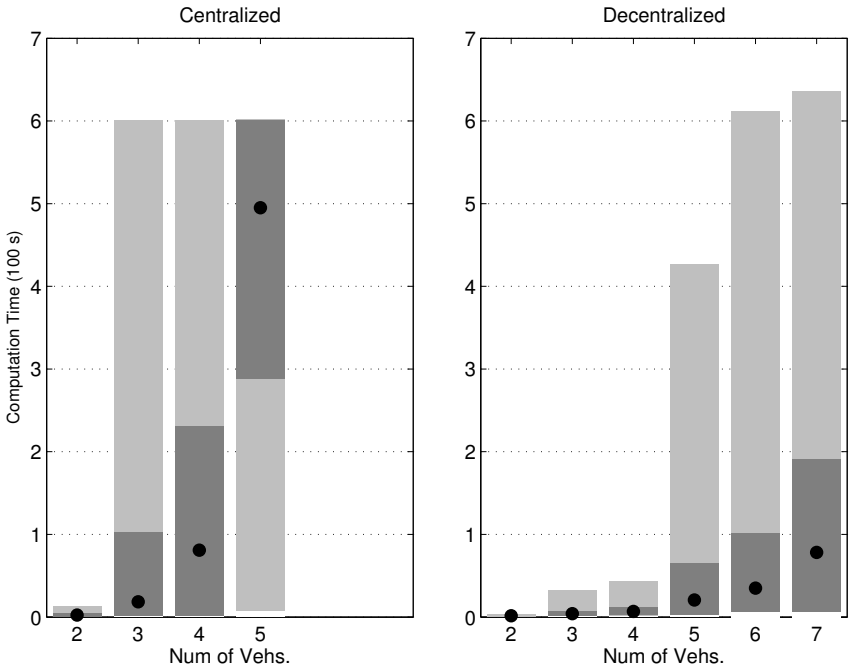


Figure 4-7: Comparison of Computation Times for Randomly-Chosen Instances. The plot shows the mean computation times (·), range of times (union of light and dark gray) and standard deviation about mean (dark gray).

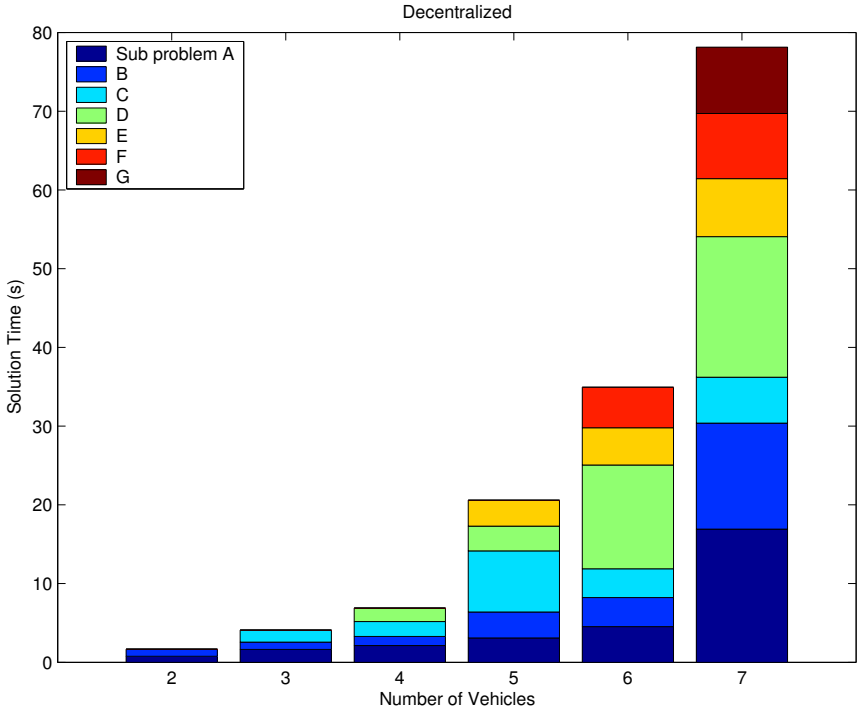
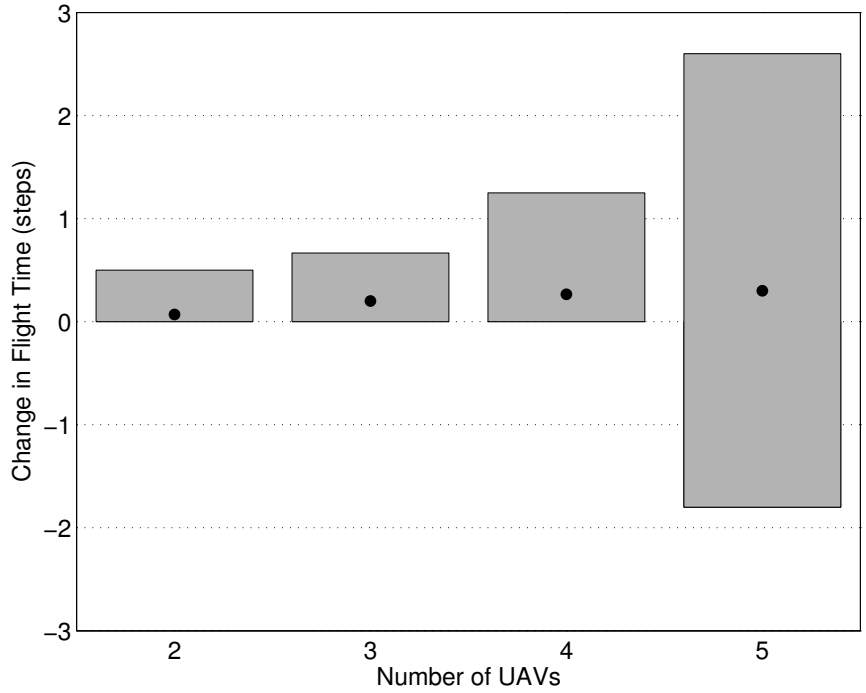


Figure 4-8: Mean Solution Times for DMPC with up to Seven UAVs, showing Breakdown among Sub-Problems.

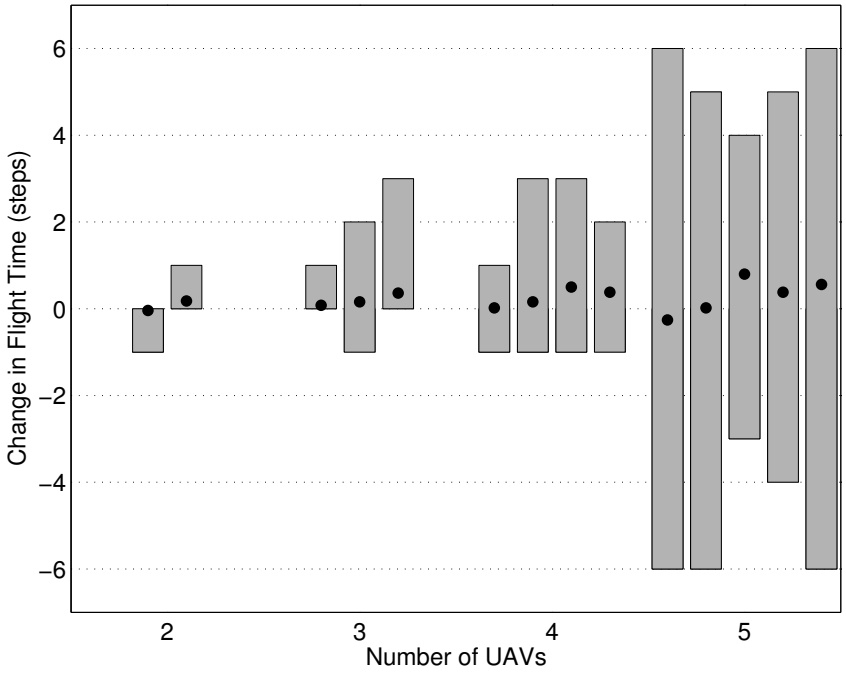
Table 4.2: Computation Times

Number of Vehicles	Computation Times (s)			
	Min	Mean	Max	St.Dev
	Centralized			
2	0.73	2.51	13.14	2.64
3	1.77	19.45	600.50	87.07
4	1.67	71.91	600.59	134.14
5	8.00	495.12	601.25	206.21
	Decentralized			
2	0.92	1.65	3.55	0.61
3	1.53	4.07	32.49	3.51
4	2.11	6.88	42.86	5.50
5	3.25	20.59	426.14	45.19
6	6.41	34.94	611.70	66.69
7	6.38	78.14	635.61	112.75

section compares the performance of the two algorithms *i.e.* the optimization cost function, in this case the UAV flight time from start to goal. Fig. 4-9 shows the differences in flight time between DMPC and CMPC for the same fifty random simultions described in Section 4.7.1. Fig. 4-9(a) shows the mean and range of the average difference, *i.e.* averaging across all UAVs in the team for each simulation. For up to four UAVs, total flight-times using DMPC are no shorter than for CMPC, but only longer by about one time step per UAV. It is intuitive that DMPC should be no better than CMPC as both solve the same problem, but DMPC in a more constrained manner. However, for the five-vehicle cases, some DMPC results are better than CMPC, occurring when the CMPC optimization is terminated by its computation time-limit with an inferior solution. Fig. 4-9(b) shows the flight time comparison broken down by individual UAV. The trends in the mean flight time differences suggest a slight advantage to being early in the planning order, but there is considerable variation and no firm conclusion can be drawn.



(a) Team Average Flight Time Difference

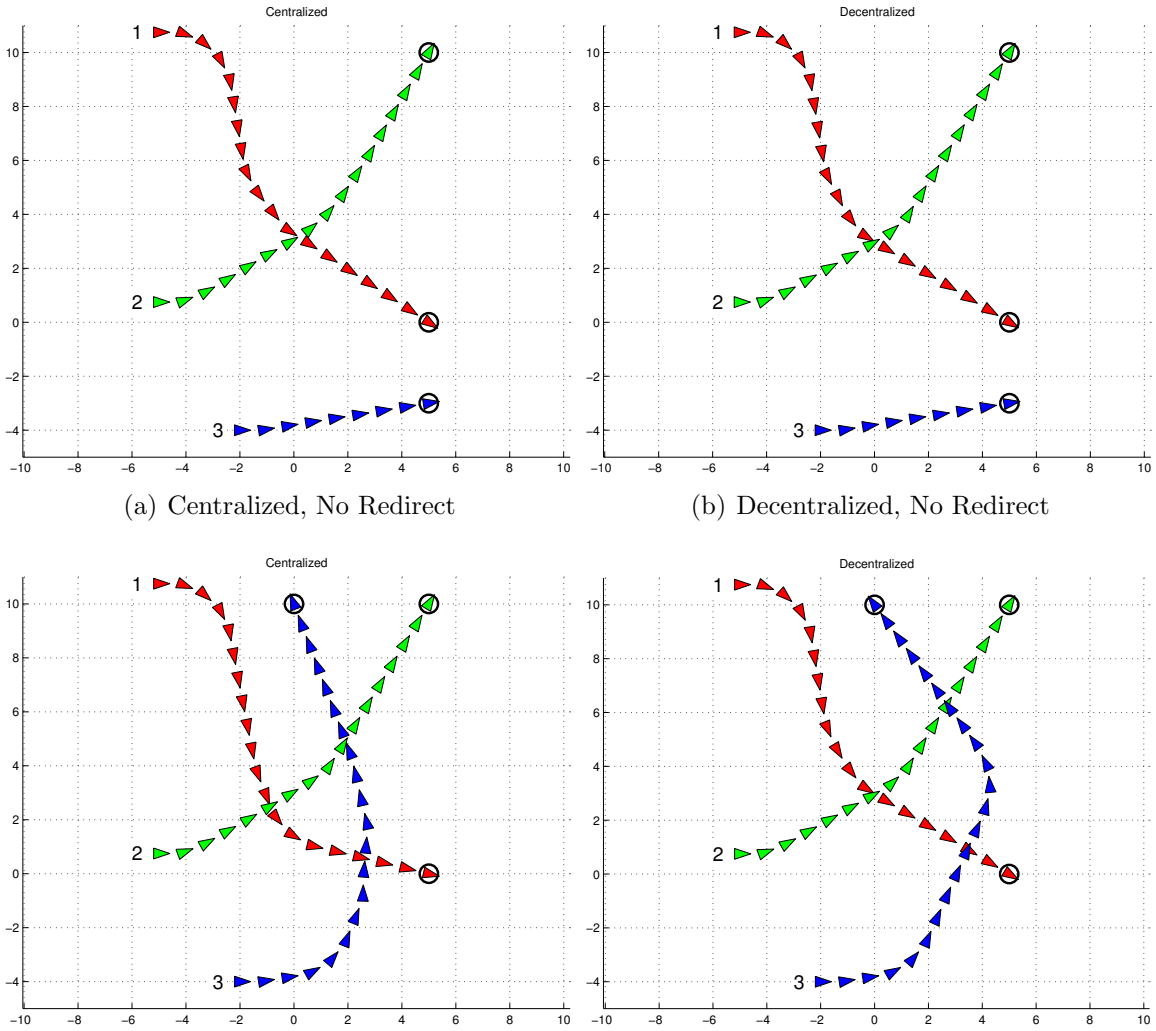


(b) Flight Time Difference for Each UAV

Figure 4-9: Differences in UAV Flight Times Between CMPC and DMPC. Positive difference indicates UAV reaches goal earlier using CMPC. Units of time are time steps, and typical flight times are 15 steps per vehicle. The plots show the mean difference (•) and range (gray bar) over fifty instances.

4.7.3 In-Flight Goal Change

This section shows an example involving a change of the goals part way through the flight. This class of uncertainty is not covered by Theorem 1, but the example shows that DMPC can potentially handle such a change and, crucially, does not require re-initializing using a centralized process. Figs. 4-10(a) and 4-10(b) show the trajectories for an example scenario with fixed goals using CMPC and DMPC, respectively. Observe that the trajectories are similar for both controllers and that UAV 3 does not interact with the other two. In the simulations shown in Figs. 4-10(c) and 4-10(d), the goals were initially the same as in Figs. 4-10(a) and 4-10(b), but at the third time step the goal for UAV 3 was moved to the location $(0, 10)$, requiring UAV 3 to interact with the other two. Comparing Figs. 4-10(a) and 4-10(c), it can be seen that the centralized controller diverts all UAVs to accommodate the changed goal. However, comparing Figs. 4-10(b) and 4-10(d), it can be seen that using DMPC, only the path of UAV 3 changes when its goal is changed. This is irrespective of the planning order: when UAV 3 replans for its new goal, it must choose a path consistent with the pre-existing intentions of UAVs 1 and 2. Therefore, UAV 3 takes a more circuitous route to its new goal and there is never any cause to change the paths of UAVs 1 and 2. However, the subproblems were still feasible and all UAVs reached their goals, suggesting that DMPC can handle broader classes of uncertainty than a simple disturbance action, such as ad-hoc team changes, adding or removing UAVs or goal changes.



(a) Centralized, No Redirect

(b) Decentralized, No Redirect

(c) Centralized, with Redirect

(d) Decentralized, with Redirect

Figure 4-10: Diverting a UAV in Mid-Mission

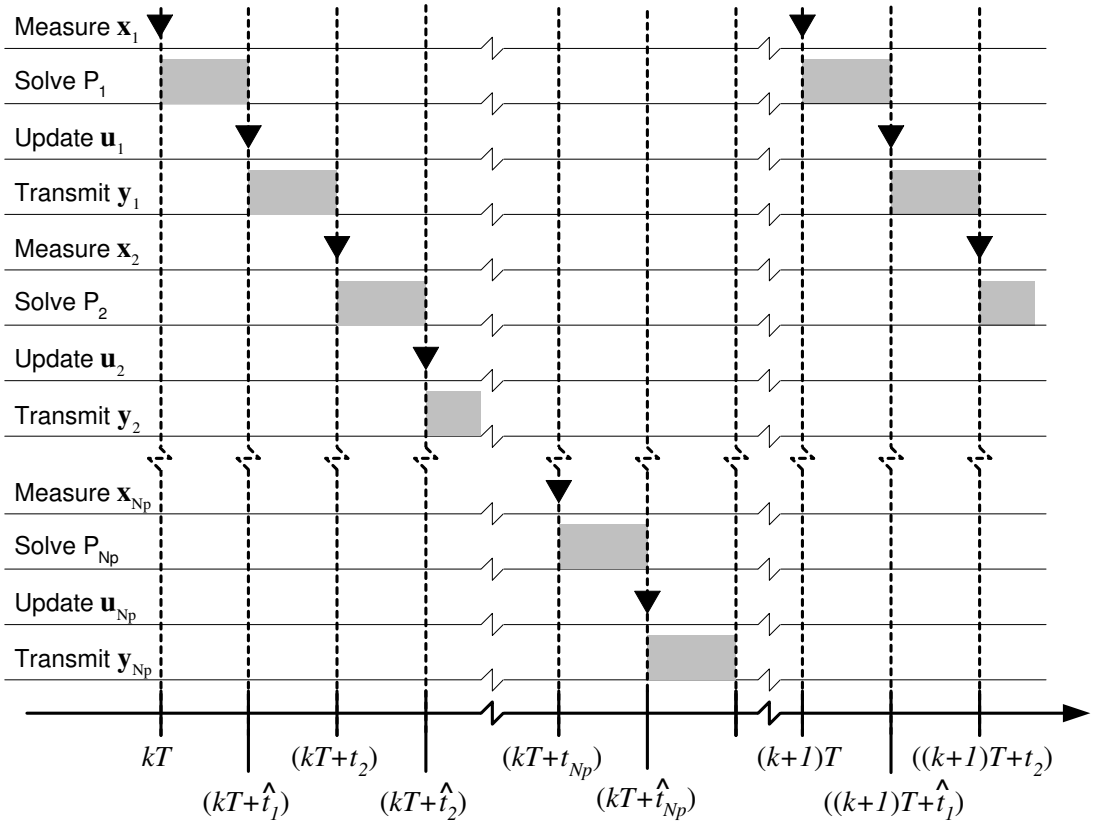


Figure 4-11: Timing of Decentralized Algorithm. Instantaneous events are marked by ▼ and delays are marked by shaded boxes.

4.8 Accounting for Delay

This section shows how to modify the decentralized MPC algorithm to explicitly account for communication and computation delays. The approach involves the estimate propagation concept, introduced in Section 3.2.5, to account for computation delay, and modification of the constraints to account for communication delay.

Fig. 4-11 shows a timing diagram for the modified algorithm. At each time step kT, the algorithm works through the subsystem sequence $p = 1, \ldots, N_p$. At time $kT + t_p$, subsystem p takes a measurement and forms an estimate of its state

$$\hat{\mathbf{x}}_p(kT + t_p) = \mathbf{x}_p(kT + t_p) + \mathbf{n}_p(k)$$

where $\mathbf{n}_p(k)$ is the estimation error. Then this estimate is propagated forward to form a prediction of its state at time $kT + \hat{t}_p$

$$\begin{aligned} \hat{\mathbf{x}}_p(k) &= \hat{\mathbf{x}}_p(kT + \hat{t}_p | kT + t_p) \\ &= \Phi_p(\hat{t}_p - t_p)\hat{\mathbf{x}}_p(kT + t_p) + \Gamma_p(\hat{t}_p - t_p)\mathbf{u}(k-1) \end{aligned} \tag{4.14}$$

where $\Phi_p(\cdot)$ and $\Gamma_p(\cdot)$ are state transition matrices. Subsystem p then solves its optimization. At time $kT+\hat{t}_p$, the computation is completed and the resulting control is implemented

$$\mathbf{u}_p(t) = \mathbf{u}_p(k), \quad kT + \hat{t}_p \leq t < (k+1)T + \hat{t}_p \tag{4.15}$$

The results of the computation are then transmitted to the other subsystems and not until time t_{p+1} does the next subsystem $p+1$ begin its computation. Therefore, $(\hat{t}_p - t_p)$ is the time available for computation of subproblem p and $(t_{(p+1)} - \hat{t}_p)$ is the time available for communicating data from subproblem p. As in Section 3.2.5, the dynamics of the state estimate $\hat{\mathbf{x}}_p$ can be written in the standard form of (4.1), as a perfect information system

$$\hat{\mathbf{x}}_p(k+1) = \mathbf{A}_p\hat{\mathbf{x}}_p(k) + \mathbf{B}_p\mathbf{u}_p(k) + \hat{\mathbf{w}}_p(k) \tag{4.16}$$

where the effective process noise $\hat{\mathbf{w}}_p$ is unknown but bounded

$$\hat{\mathbf{w}}_p(k) \in \hat{\mathcal{W}}_p \ \forall k \tag{4.17}$$

It is also necessary to modify the constraints to account for the offset between the sampling times for each subsystem, as shown in Fig. 4-11. Suppose the outputs, expressed in continuous time, are given by

$$\mathbf{y}_p(t) = \mathbf{C}_p\mathbf{x}_p(t) + \mathbf{D}_p\mathbf{u}_p(t) \tag{4.18}$$

with constraints to be enforced at common discrete time points $t = kT + \hat{t}_{N_p}$ for all

subsystems.

$$\sum_{p=1}^{N_p} \mathbf{y}_p(t) \in \mathcal{Y} \quad t = kT + \hat{t}_{N_p} \ \forall k \tag{4.19}$$

These time points are the ends of the last subproblem computations at each time-step. The state transition matrices $\Phi_p(\cdot)$ and $\Gamma_p(\cdot)$ are used again to propagate each state estimate $\mathbf{x}_p$ forward from its sampling time $kT + t_p$ to the common time $kT + \hat{t}_{N_p}$ for constraint enforcement

$$\begin{aligned} \hat{\mathbf{x}}_p(kT + \hat{t}_{N_p}|kT + t_p) &= \Phi_p(\hat{t}_{N_p} - \hat{t}_p)\hat{\mathbf{x}}_p(kT + \hat{t}_p|kT + t_p) + \Gamma_p(\hat{t}_{N_p} - \hat{t}_p)\mathbf{u}_p(k) \\ &= \Phi_p(\hat{t}_{N_p} - \hat{t}_p)\hat{\mathbf{x}}_p(k) + \Gamma_p(\hat{t}_{N_p} - \hat{t}_p)\mathbf{u}_p(k) \end{aligned} \tag{4.20}$$

Using the methods developed in Section 3.2.5, the error in this estimate can be determined and bounded.

$$\hat{\mathbf{e}}_p(k) = \hat{\mathbf{x}}_p(kT + \hat{t}_{N_p}|kT + t_p) - \mathbf{x}_p(kT + \hat{t}_{N_p}) \in \mathcal{E}_p \tag{4.21}$$

This set is used to modify the constraint set $\mathcal{Y}$ to account for the discrepancy between estimated and true outputs, similar to the modification in Section 3.2.1.

$$\hat{\mathcal{Y}} = \mathcal{Y} \sim \mathbf{C}_1\mathcal{E}_1 \sim \mathbf{C}_2\mathcal{E}_2 \sim \cdots \sim \mathbf{C}_{N_p}\mathcal{E}_{N_p} \tag{4.22}$$

The controls at time $kT + t_p$ are easily extracted from the zero-order hold expression (4.15)

$$\mathbf{u}_p(kT + \hat{t}_{N_p}) = \mathbf{u}_p(k) \tag{4.23}$$

Substituting from (4.20) and (4.23) into (4.18) gives the estimated outputs of subsystem p at time $kT + \hat{t}_{N_p}$ are given by

$$\begin{aligned} \hat{\mathbf{y}}_p(kT + \hat{t}_{N_p}) &= \mathbf{C}_p\hat{\mathbf{x}}_p(kT + \hat{t}_{N_p}|kT + t_p) + \mathbf{D}_p\mathbf{u}_p(kT + \hat{t}_{N_p}) \\ &= \hat{\mathbf{C}}_p\hat{\mathbf{x}}_p(k) + \hat{\mathbf{D}}_p\mathbf{u}_p(k) \end{aligned} \tag{4.24}$$

where

$$\hat{\mathbf{C}}_p = \mathbf{C}_p \Phi_p(\hat{t}_{N_p} - \hat{t}_p) \tag{4.25a}$$

$$\hat{\mathbf{D}}_p = \left(\mathbf{C}_p \Gamma_p(\hat{t}_{N_p} - \hat{t}_p) + \mathbf{D}_p\right) \tag{4.25b}$$

Finally, define modified optimizations $\hat{\mathsf{P}}_C$ and $\hat{\mathsf{P}}_p$, based on the original problems P_C and P_p defined in Sections 4.3 and 4.4, replacing $\mathbf{C}_p$, $\mathbf{D}_p$, $\mathcal{Y}_p$ and $\mathcal{W}_p$ in (4.6)–(4.7) and (4.11) with $\hat{\mathbf{C}}_p$, $\hat{\mathbf{D}}_p$, $\hat{\mathcal{Y}}_p$ and $\hat{\mathcal{W}}_p$, respectively. These problems are employed in the following algorithm, a modified form of Algorithm 4.1 using the timing shown in Fig. 4-11

Algorithm 4.2. (Decentralized MPC with Delays)

1. Find a solution to the initial centralized problem $\hat{\mathsf{P}}_C(\hat{\mathbf{x}}_1(0), \ldots, \hat{\mathbf{x}}_{N_p}(0))$. If a solution cannot be found, stop (problem is infeasible).
2. Set $k = 0$
3. Apply control $\mathbf{u}_p^*(0|0)$ to each subsystem p in order $1, \ldots, N_p$ at times $t = \hat{t}_p$
4. Increment k
5. For each subsystem p in order $1, \ldots, N_p$:
   (a) Gather, by communication, the plan data $\tilde{\mathbf{y}}_p(k, \ldots, k+N|k)$ from other subsystems, defined by (4.12)
   (b) (At time $t = kT + t_p$) Take measurement and form state prediction $\hat{\mathbf{x}}_p(k) = \hat{\mathbf{x}}_p(kT + \hat{t}_p | kT + t_p)$
   (c) Solve subproblem $\hat{\mathsf{P}}_p\left(\hat{\mathbf{x}}_p(k), \tilde{\mathbf{y}}_p(k, \ldots, k+N|k)\right)$
   (d) (At time $t = kT + \hat{t}_p$) Apply control $\mathbf{u}_p^*(k|k)$ to subsystem p
6. Go to 4

Theorem 4.2. ***(Robust Feasibility with Delays)*** *If a feasible solution to the initial centralized problem $\mathsf{P}_C(\mathbf{x}_1(0), \ldots, \mathbf{x}_{N_p}(0))$ can be found, and if the effective process noises and esimation errors obey (4.17) and (4.21) respectively, then the systems (4.1) controlled using Algorithm 4.2 are robustly feasible and satisfy constraints (4.18) and (4.19).*

Proof: Applying Theorem 4.1 to the dynamics of the estimates (4.16) guarantees robust feasibility and satisfaction of

$$\hat{\mathbf{y}}_p(kT + \hat{t}_{N_p}) \in \hat{\mathcal{Y}} \; \forall k \tag{4.26}$$

Combining the definitions of the true outputs (4.18), the estimated outputs (4.24) and the estimation error (4.21) gives the following relationship between the estimated outputs and the true outputs

$$\hat{\mathbf{y}}_p(kT + \hat{t}_{N_p}) = \mathbf{y}_p(kT + \hat{t}_{N_p}) + \sum_{p=1}^{N_p} \mathbf{C}_p \mathbf{e}_p(k) \tag{4.27}$$

Hence applying the property of the Pontryagin difference (2.11) to the sets in (4.22) and the relations (4.26) and (4.27) gives

$$\mathbf{y}_p(kT + \hat{t}_{N_p}) \in \mathcal{Y} \; \forall k$$

satifying the constraints (4.18) and (4.19). □

In summary, the effect of delay can be accommodated by propagating the state estimate forward and accounting for the additional uncertainty introduced, as shown for the centralized problem in Section 3.2.5. Importantly, in the decentralized case, it is not necessary to wait for the entire solution sequence to finish before updating the controls. Rather, each subsystem waits until its turn in the sequence to measure its state and can update its control as soon as its computation is finished. Therefore, each subproblem must only account for the delay of its own computation, not the whole sequence.

4.9 Summary

A formulation for Decentralized Model Predictive Control (DMPC) has been presented. It solves the problem of control of multiple subsystems subjected to coupled constraints but otherwise independent. Each subsystem solves a subproblem involving only its own state predictions. The scheme is proven to guarantee robust constraint satisfaction under the assumption of bounded disturbance action. Each subproblem is guaranteed to be feasible and no iteration between subsystems is required.

DMPC for teams of cooperating UAVs has been developed and demonstrated. Simulation results for the representative example of multi-UAV collision avoidance have been presented. DMPC guarantees constraint satisfaction, in this case avoidance, and offers significant computation improvement, compared to the equivalent centralized algorithm, for only a small degradation in performance, in this case UAV flight time.

4.A Proof of Theorem 4.1

Theorem 4.1 (Robust Feasibility) *If a feasible solution to the initial centralized problem* $\mathsf{P}_C(\mathbf{x}_1(0), \ldots, \mathbf{x}_{N_p}(0))$ *can be found, then the systems (4.1), subjected to disturbances obeying (4.2) and controlled using Algorithm 4.1, are robustly feasible and satisfy constraints (4.3) and (4.4).*

Proof The outline of the proof consists of four steps described immediately after Theorem 4.1 in Section 4.4. This section proves each step in detail.

Step 1 Assume a set of solutions is known for all subsystems at time k_0 satisfying the centralized problem $\mathsf{P}_C(k_0)$ (where the shortened index (k_0) denotes the problem for all states at that time).

Then the state sequences $\mathbf{x}_p^*(k_0 \ldots k_0 + N + 1|k_0)$, controls $\mathbf{u}_p^*(k_0 \ldots k_0 + N|k_0)$ and outputs $\mathbf{y}_p^*(k_0 \ldots k_0 + N|k_0)$ for all subsystems p satisfy centralized problem $\mathsf{P}_C(\mathbf{x}_1(k_0), \ldots, \mathbf{x}_{N_p}(k_0))$. This requires that the output sequences satisfy

$$\sum_{p=1}^{N_p} \mathbf{y}_p^*(k_0 + 1 + j|k_0) \in \mathcal{Y}_{N_p}(j+1) \quad j \in \{0 \ldots N-1\} \tag{4.28}$$

and

$$\mathbf{x}_p^*(k_0 + 1 + N|k_0) \in \mathcal{Q}_p \; \forall p \in \{1 \ldots N_p\} \tag{4.29}$$

Step 2 Show that given the assumption in Step 1, a feasible solution can be found to the first subproblem $\mathsf{P}_1(k_0+1)$ for all disturbances $\mathbf{w}_1(k_0)$ by showing feasibility of a candidate solution.

Define the candidate solution, denoted by $\hat{\cdot}$, by shifting the previous solution for subsystem 1 by one step, adding a single step of the control law κ_1 to the end, and adding a perturbation using the candidate controller $\mathbf{K}_1$ and its associated state transition matrx $\mathbf{L}_1$ defined in (4.8)

$$\begin{aligned}
\hat{\mathbf{u}}_1(k_0+1+j|k_0+1) &= \mathbf{u}_1^*(k_0+j|k_0)+\mathbf{K}_1(j)\mathbf{L}_1(j)\mathbf{w}_1(k_0)\ \forall j\in\{0\dots N-1\}\\
\hat{\mathbf{u}}_1(k_0+N+1|k_0+1) &= \kappa_1(\hat{\mathbf{x}}_1(k_0+N+1|k_0+1))\\
\\
\hat{\mathbf{x}}_1(k_0+1+j|k_0+1) &= \mathbf{x}_1^*(k_0+1+j|k_0)+\mathbf{L}_1(j)\mathbf{w}_1(k_0)\ \forall j\in\{0\dots N\}\\
\hat{\mathbf{x}}_1(k_0+N+2|k_0+1) &= \mathbf{A}_1\hat{\mathbf{x}}_1(k_0+N+1|k_0+1)+\mathbf{B}_1\kappa_1(\hat{\mathbf{x}}_1(k_0+N+1|k_0+1))\\
\\
\hat{\mathbf{y}}_1(k_0+j+1|k_0+1) &= \mathbf{C}_1\hat{\mathbf{x}}_1(k_0+j+1|k_0+1)+\mathbf{D}_1\hat{\mathbf{u}}_1(k_0+j+1|k_0+1)\\
&\qquad\qquad \forall j\in\{0\dots N\}
\end{aligned} \tag{4.30}$$

Theorem 2.1 has already shown that this candidate solution satisfies the dynamics, initial condition and terminal constraints (4.11a)–(4.11d). It remains to show that it satisfies the output constraints (4.11e) with $p=1$. Substituting the state and control sequences from (4.30) into the output sequence gives

$$\begin{aligned}
\hat{\mathbf{y}}_1(k_0+1+j|k_0+1) &= \mathbf{y}_1^*(k_0+1+j|k_0)+(\mathbf{C}_1+\mathbf{D}_1\mathbf{K}_1(j))\mathbf{L}_1(j)\mathbf{w}_1(k_0) \quad (4.31a)\\
&\qquad\qquad \forall j\in\{0\dots N-1\}\\
\hat{\mathbf{y}}_1(k_0+1+N|k_0+1) &= \mathbf{C}_1\mathbf{x}_1^*(k_0+1+N|k_0)+\mathbf{D}_1\kappa_1(\mathbf{x}_1^*(k_0+1+N|k_0)) \quad (4.31b)
\end{aligned}$$

Write the data representing other subsystems in subproblem 1, compiled using (4.12)

$$\tilde{\mathbf{y}}_1(k_0+1+j|k_0+1) = \sum_{p=2}^{N_p}\mathbf{y}_p^*(k_0+1+j|k_0)\ \forall j\in\{0\dots N-1\} \tag{4.32a}$$

$$\tilde{\mathbf{y}}_1(k_0+1+N|k_0+1) = \sum_{p=2}^{N_p}\mathbf{C}_p\mathbf{x}_p^*(k_0+1+N|k_0)+\mathbf{D}_p\kappa_p(\mathbf{x}_p^*(k_0+N|k_0)) \tag{4.32b}$$

Then adding (4.31) and (4.32) gives

$$\begin{aligned}&\hat{\mathbf{y}}_1(k_0+1+j|k_0+1)+\tilde{\mathbf{y}}_1(k_0+1+j|k_0+1)\\&=\sum_{p=1}^{N_p}\mathbf{y}_p^*(k_0+1+j|k_0)+(\mathbf{C}_1+\mathbf{D}_1\mathbf{K}_1(j))\mathbf{L}_1(j)\mathbf{w}_1(k_0)\ \ \forall j\in\{0\dots N-1\}\end{aligned} \tag{4.33a}$$

$$\begin{aligned}&\hat{\mathbf{y}}_1(k_0+1+N|k_0+1)+\tilde{\mathbf{y}}_1(k_0+1+N|k_0+1)\\&=\sum_{p=1}^{N_p}\mathbf{C}_p\mathbf{x}_p^*(k_0+1+N|k_0)+\mathbf{D}_p\kappa_p(\mathbf{x}_p^*(k_0+N|k_0))\end{aligned} \tag{4.33b}$$

Recall the set definition (2.8b)

$$\mathcal{Y}_{N_p}(j+1)=\mathcal{Y}_1(j)\sim(\mathbf{C}_1+\mathbf{D}_1\mathbf{K}_1(j))\,\mathbf{L}_1(j)\mathcal{W}_1\ \forall j\in\{0\dots N-1\}$$

then applying the property of the Pontryagin difference (2.11) to (4.28) and (4.33a) gives

$$\hat{\mathbf{y}}_1(k_0+1+j|k_0+1)+\tilde{\mathbf{y}}_1(k_0+1+j|k_0+1)\in\mathcal{Y}_1(j)\ \forall\mathbf{w}_1(k_0)\in\mathcal{W}_1\forall j\in\{0\dots N-1\} \tag{4.34}$$

Consider now the final step (4.33b). Using (4.29) and the definition of the terminal sets (4.10), (4.33b) gives

$$\hat{\mathbf{y}}_1(k_0+1+N|k_0+1)+\tilde{\mathbf{y}}_1(k_0+1+N|k_0+1)\in\mathcal{Y}_{N_p}(N) \tag{4.35}$$

Subtituting the nilpotency requirement $\mathbf{L}_p(N)=\mathbf{0}$ into the constraint tightening equation (4.7b) gives $\mathcal{Y}_{N_p}(N)=\mathcal{Y}_p(N)\forall p$. Then (4.34) and (4.35) combine to show

$$\hat{\mathbf{y}}_1(k_0+1+j|k_0+1)+\tilde{\mathbf{y}}_1(k_0+1+j|k_0+1)\in\mathcal{Y}_1(j)\ \forall\mathbf{w}_1(k_0)\in\mathcal{W}_1\forall j\in\{0\dots N\}$$

Therefore the output constraints (4.11e) are satisfied by the candidate sequence in (4.30). This completes Step 2 of the proof: feasibility of the centralized problem $\mathsf{P}_C(k_0)$ implies feasibility of the first subproblem at the subsequent step $\mathsf{P}_1(k_0{+}1)$.

Step 3 Show that, under the assumption in Step 1, given any solution to the subproblem $\mathsf{P}_{p_0}(k_0+1)$ for $p_0 \in \{1,\ldots,N_p-1\}$ then the next subproblem $\mathsf{P}_{(p_0+1)}(k_0+1)$ is feasible, again by showing feasibility of a candidate sequence;

Assume the solution for the subsystem p_0 has outputs satisfying

$$\mathbf{y}^*_{p_0}(k_0+1+j|k_0+1)+\tilde{\mathbf{y}}_{p_0}(k_0+1+j|k_0+1)\in\mathcal{Y}_{p_0}(j)\ \forall j\in\{0\ldots N\} \tag{4.36}$$

where the intentions of the other subsystems in subproblem p_0 are compiled according to (4.12) with $p=p_0$.

$$\begin{aligned}\tilde{\mathbf{y}}_{p_0}(k_0+1+j|k_0+1) &= \sum_{q=1}^{p_0-1}\mathbf{y}^*_q(k_0+1+j|k_0+1) &\text{(4.37a)}\\ &+ \sum_{q=p_0+1}^{N_p}\mathbf{y}^*_q(k_0+1+j|k_0)\quad \forall j\in\{0\ldots N-1\}\\ \tilde{\mathbf{y}}_{p_0}(k_0+1+N|k_0+1) &= \sum_{q=1}^{p_0-1}\mathbf{y}^*_q(k_0+1+N|k_0+1) &\text{(4.37b)}\\ &+ \sum_{q=p_0+1}^{N_p}\mathbf{C}_q\mathbf{x}^*_q(k_0+1+N|k_0)+\mathbf{D}_q\kappa_q(\mathbf{x}^*_q(k_0+1+N|k_0))\end{aligned}$$

Adding the outputs $\mathbf{y}^*_{p_0}(k_0+1+j|k_0+1)$ from the solution for p_0 at time k_0+1 gives the expression on the lefthand side of the constraint (4.11e)

$$\begin{aligned}&\mathbf{y}^*_{p_0}(k_0+1+j|k_0+1)+\tilde{\mathbf{y}}_{p_0}(k_0+1+j|k_0+1)\\ &=\sum_{q=1}^{p_0}\mathbf{y}^*_q(k_0+1+j|k_0+1)+\sum_{q=p_0+1}^{N_p}\mathbf{y}^*_q(k_0+1+j|k_0)\quad \forall j\in\{0\ldots N-1\}\end{aligned} \tag{4.38a}$$

$$\begin{aligned}&\mathbf{y}^*_{p_0}(k_0+1+N|k_0+1)+\tilde{\mathbf{y}}_{p_0}(k_0+1+N|k_0+1)\\ &=\sum_{q=1}^{p_0}\mathbf{y}^*_q(k_0+1+N|k_0+1)+\sum_{q=p_0+1}^{N_p}\mathbf{C}_q\mathbf{x}^*_q(k_0+1+N|k_0)+\mathbf{D}_q\kappa_q(\mathbf{x}^*_q(k_0+1+N|k_0))\end{aligned} \tag{4.38b}$$

so feasibility of problem p_0 gives bounds on these quantities

$$\mathbf{y}^*_{p_0}(k_0+1+j|k_0+1)+\tilde{\mathbf{y}}_{p_0}(k_0+1+j|k_0+1)\in\mathcal{Y}_{p_0}(j)\ \ \forall j\in\{0\ldots N\}$$

Define the candidate solution for system p_0+1, denoted by $\hat{\cdot}$, by shifting the previous solution for subsystem 1 by one step, adding a single step of the control law κ_{p_0+1} to the end, and adding a perturbation using the candidate controller $\mathbf{K}_{p_0+1}$ and its associated state transition matrx $\mathbf{L}_{p_0+1}$ defined in (4.8)

$$\begin{aligned}
\hat{\mathbf{u}}_{(p_0+1)}(k_0+1+j|k_0+1) &= \mathbf{u}^*_{(p_0+1)}(k_0+j|k_0) && \text{(4.39a)}\\
&+ \mathbf{K}_{(p_0+1)}(j)\mathbf{L}_{(p_0+1)}(j)\mathbf{w}_{(p_0+1)}(k_0)\ \forall j \in \{0\dots N-1\}\\
\hat{\mathbf{u}}_{(p_0+1)}(k_0+N+1|k_0+1) &= \kappa_{(p_0+1)}(\hat{\mathbf{x}}_{(p_0+1)}(k_0+N+1|k_0+1)) && \text{(4.39b)}\\
\hat{\mathbf{x}}_{(p_0+1)}(k_0+1+j|k_0+1) &= \mathbf{x}^*_{(p_0+1)}(k_0+1+j|k_0) && \text{(4.39c)}\\
&+ \mathbf{L}_{(p_0+1)}(j)\mathbf{w}_{(p_0+1)}(k_0)\ \forall j \in \{0\dots N\}\\
\hat{\mathbf{x}}_{(p_0+1)}(k_0+N+2|k_0+1) &= \mathbf{A}_{(p_0+1)}\hat{\mathbf{x}}_{(p_0+1)}(k_0+N+1|k_0+1) && \text{(4.39d)}\\
&+ \mathbf{B}_{(p_0+1)}\kappa_{(p_0+1)}(\hat{\mathbf{x}}_{(p_0+1)}(k_0+N+1|k_0+1))\\
\hat{\mathbf{y}}_{(p_0+1)}(k_0+j+1|k_0+1) &= \mathbf{C}_{(p_0+1)}\hat{\mathbf{x}}_{(p_0+1)}(k_0+j+1|k_0+1) && \text{(4.39e)}\\
&+ \mathbf{D}_{(p_0+1)}\hat{\mathbf{u}}_{(p_0+1)}(k_0+j+1|k_0+1)\ \forall j \in \{0\dots N\}
\end{aligned}$$

Theorem 2.1 has already shown that this candidate solution satisfies the dynamics, initial condition and terminal constraints (4.11a)–(4.11d). It remains to show that it satisfies the output constraints (4.11e) with $p = p_o + 1$. The output sequence associated with the candidate control (4.39) can be rewritten as

$$\begin{aligned}
\hat{\mathbf{y}}_{(p_0+1)}(k_0+1+j|k_0+1) &= \mathbf{y}^*_{(p_0+1)}(k_0+1+j|k_0) && \text{(4.40a)}\\
&+ (\mathbf{C}_{(p_0+1)} + \mathbf{D}_{(p_0+1)}\mathbf{K}_{(p_0+1)}(j))\mathbf{L}_{(p_0+1)}(j)\mathbf{w}_{(p_0+1)}(k_0)\\
&\qquad \forall j \in \{0\dots N-1\}\\
\hat{\mathbf{y}}_{(p_0+1)}(k_0+1+N|k_0+1) &= \mathbf{C}_{(p_0+1)}\mathbf{x}^*_{(p_0+1)}(k_0+1+N|k_0) && \text{(4.40b)}\\
&+ \mathbf{D}_{(p_0+1)}\kappa_{(p_0+1)}(\mathbf{x}^*_{(p_0+1)}(k_0+1+N|k_0))
\end{aligned}$$

Write the data for other subsystems, compiled using (4.12) with $p = p_0 + 1$.

$$\begin{aligned}\tilde{\mathbf{y}}_{(p_0+1)}(k_0+1+j|k_0+1) &= \sum_{q=1}^{p_0} \mathbf{y}_q^*(k_0+1+j|k_0+1) \qquad (4.41a)\\ &+ \sum_{q=p_0+2}^{N_p} \mathbf{y}_q^*(k_0+1+j|k_0) \quad \forall j \in \{0\dots N-1\}\\ \tilde{\mathbf{y}}_{(p_0+1)}(k_0+1+N|k_0+1) &= \sum_{q=1}^{p_0} \mathbf{y}_q^*(k_0+1+N|k_0+1) \qquad (4.41b)\\ &+ \sum_{q=p_0+2}^{N_p} \mathbf{C}_q\mathbf{x}_q^*(k_0+1+N|k_0) + \mathbf{D}_q\kappa_q(\mathbf{x}_q^*(k_0+1+N|k_0))\end{aligned}$$

Now add (4.40) and (4.41)

$$\begin{aligned}&\hat{\mathbf{y}}_{(p_0+1)}(k_0+1+j|k_0+1) + \tilde{\mathbf{y}}_{(p_0+1)}(k_0+1+j|k_0+1)\\ &= \sum_{q=1}^{p_0} \mathbf{y}_q^*(k_0+1+j|k_0+1) + \sum_{q=p_0+1}^{N_p} \mathbf{y}_q^*(k_0+1+j|k_0) \qquad (4.42a)\\ &+(\mathbf{C}_{(p_0+1)} + \mathbf{D}_{(p_0+1)}\mathbf{K}_{(p_0+1)}(j))\mathbf{L}_{(p_0+1)}(j)\mathbf{w}_{(p_0+1)}(k_0) \quad \forall j \in \{0\dots N-1\}\end{aligned}$$

$$\begin{aligned}&\mathbf{y}_{p_0}^*(k_0+1+N|k_0+1) + \tilde{\mathbf{y}}_{p_0}(k_0+1+N|k_0+1)\\ &= \sum_{q=1}^{p_0} \mathbf{y}_q^*(k_0+1+N|k_0+1) + \sum_{q=p_0+1}^{N_p} \mathbf{C}_q\mathbf{x}_q^*(k_0+1+N|k_0) + \mathbf{D}_q\kappa_q(\mathbf{x}_q^*(k_0+1+N|k_0))\end{aligned} \qquad (4.42b)$$

Now comparing (4.38) with (4.42) shows that the constrained quantities, *i.e.* the the left hand side of (4.11e), of the two subproblems p_0 and $p_0 + 1$ differ only by the perturbation term in the outputs

$$\begin{aligned}&\hat{\mathbf{y}}_{(p_0+1)}(k_0+1+j|k_0+1) + \tilde{\mathbf{y}}_{(p_0+1)}(k_0+1+j|k_0+1)\\ &= \mathbf{y}_{p_0}^*(k_0+1+j|k_0+1) + \tilde{\mathbf{y}}_{p_0}(k_0+1+j|k_0+1) \qquad (4.43a)\\ &+(\mathbf{C}_{(p_0+1)} + \mathbf{D}_{(p_0+1)}\mathbf{K}_{(p_0+1)}(j))\mathbf{L}_{(p_0+1)}(j)\mathbf{w}_{(p_0+1)}(k_0) \quad \forall j \in \{0\dots N-1\}\end{aligned}$$

$$\begin{aligned}&\mathbf{y}_{p_0}^*(k_0+1+N|k_0+1) + \tilde{\mathbf{y}}_{p_0}(k_0+1+N|k_0+1)\\ &= \mathbf{y}_{p_0}^*(k_0+1+N|k_0+1) + \tilde{\mathbf{y}}_{p_0}(k_0+1+N|k_0+1)\end{aligned} \qquad (4.43b)$$

Now the property (2.11) of the Pontryagin difference used in the set tightening recursion (4.7b) can be applied to (4.36) and (4.43) to show

$$\hat{\mathbf{y}}_{(p_0+1)}(k_0+1+j|k_0+1)+\tilde{\mathbf{y}}_{(p_0+1)}(k_0+1+j|k_0+1)\in\mathcal{Y}_{p_0+1}(j)$$
$$\forall j\in\{0\dots N-1\}\ \forall \mathbf{w}_{(p_0+1)}(k_0)\in\mathcal{W}_{(p_0+1)}$$

showing that the candidate solution (4.39) satisfies the constraints (4.11e) and hence is a feasible solution to problem $\mathsf{P}_{(p_0+1)}$.

Step 4 Show that if a feasible solution to the final subproblem $\mathsf{P}_{N_p}(k_0+1)$ can be found, then the set of solutions to all subproblems $\mathsf{P}_p(k_0+1)$ satisfies the constraints of the centralized problem $\mathsf{P}_C(k_0+1)$

The dynamics, initial and terminal constraints (4.6a)–(4.6d) are all satisfied for each subsystem since each subproblem contained the equivalent constraints (4.11a)–(4.11d) for each individual subsystem. Feasibility of the output constraints (4.11e) for the final subproblem $p=N_p$ implies the following

$$\mathbf{y}^*_{N_p}(k_0+j|k_0)+\tilde{\mathbf{y}}_{N_p}(k_0+j|k_0)=\sum_{p=1}^{N_p}\mathbf{y}^*_p(k_0+j|k_0)\in\mathcal{Y}_{N_p}(j)\ \ \forall j\in\{0,\dots,N\}\quad(4.44)$$

satisfying (4.6e). Therefore the combination of all the solutions to the subproblems satisfies all the constraints of the centralized problem.

Summary The overall recursion sequence is:

solutions at k_0 satisfy $\mathsf{P}_C(k_0)$ (Step 1 of Proof)
$\Downarrow$
subproblem $\mathsf{P}_1(k_0+1)$ is feasible (by Step 2)
$\Downarrow$
subproblem $\mathsf{P}_2(k_0+1)$ is feasible (by Step 3)
$\vdots$
$\Downarrow$
subproblem $\mathsf{P}_{N_p}(k_0+1)$ is feasible (by Step 3)
$\Downarrow$
solutions at k_0+1 satisfy $\mathsf{P}_C(k_0+1)$ (by Step 4)

Hence feasibility of the centralized problem at time 0, assumed in Theorem 4.1, implies feasibility of all subsequent subproblems. □

Chapter 5

Analytical Prediction of Performance

This chapter presents a new analysis tool for predicting the closed-loop performance of a system using the robust constrained Model Predictive Control (MPC) presented in Chapter 2. Currently, performance is typically evaluated by numerical simulation, leading to extensive computation when investigating the effect of controller parameters, such as horizon length, cost weightings, and constraint settings. The method in this chapter avoids this computational burden, enabling a rapid study of the trades between the design parameters and performance. This chapter shows that the expected performance can be predicted using a combination of the gains of the system in closed-loop with two linear controllers: the optimal control for the unconstrained system; and a candidate policy used in performing the constraint tightening. The method also accounts for possible mismatch between the predicted level of disturbance and the actual level encountered. The predictions are compared with simulation results for several examples.

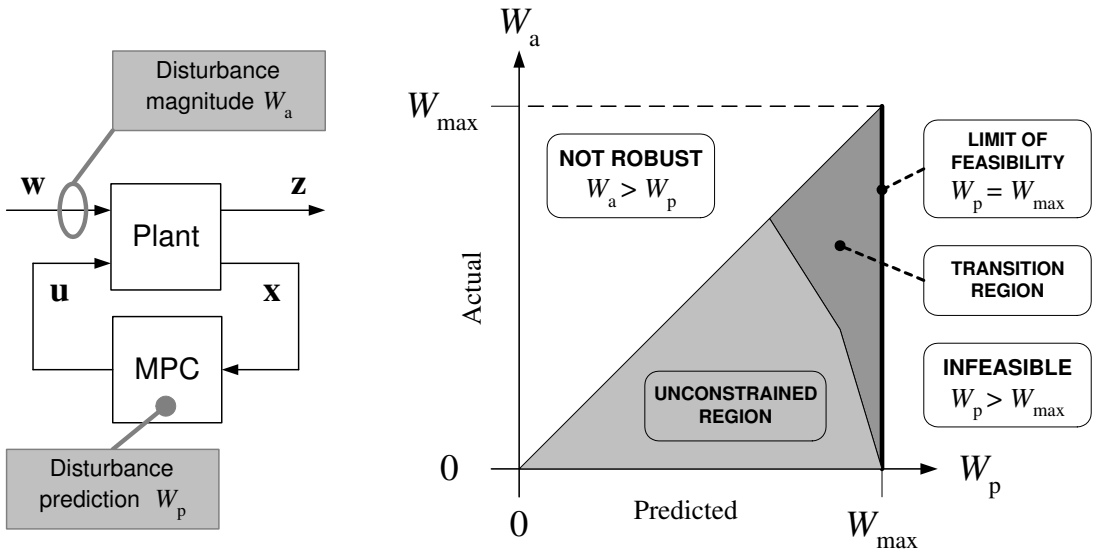
5.1 Introduction

In Chapter 2, a formulation for MPC was presented providing *robust feasibility*, guaranteeing feasibility of each optimization and satisfaction of the constraints in the

presence of an unknown but bounded disturbance. This method is an extension of an earlier work [13, 14] involving tightening constraints and retaining a "margin" in each plan for compensation against future disturbances. The margin is chosen such that a predetermined feedback policy can always be employed to counteract the disturbance, although in practice the on-line optimization usually finds a better solution than the application of that policy. While robust constraint satisfaction can imply bounds on the performance, *e.g.* the rate of fuel use of a spacecraft is clearly limited if its thrust at any time is limited, such bounds are very conservative. The new prediction method in this chapter considers the *expected* performance.

The expected robust performance of MPC is rarely considered analytically, but typically evaluated by numerical simulation instead. If various controller settings are to be investigated, this requires extensive computation. The contribution of this chapter is a method of analytically predicting the performance of a system controlled by MPC, enabling rapid investigation of the effect of controller parameters without recourse to extensive simulation.

Fig. 5-1 gives an overview of the performance prediction method. Fig. 5-1(a) shows the block diagram of the prediction method, in particular the roles of the actual and predicted disturbance levels. The predicted level is the designer's estimate of the actual disturbance level. It forms a disturbance model that is used to make the controller robust. The performance analysis is based on two observations of the variation of performance over the space of actual and predicted disturbance levels, shown in Fig. 5-1(b). The first observation is that, if the actual disturbance is sufficiently low, corresponding to the lightly-shaded *unconstrained region* in Fig. 5-1(b), the controller behaves like the optimal finite-horizon regulator for the unconstrained system. This is not surprising in itself, but one of the contributions of this chapter is a method of quantifying "sufficiently low" for this behavior. Observe that the upper limit of the unconstrained region varies with the predicted disturbance, as this determines the constraint tightening. The second observation is that, at the *limit of feasibility*, shown by the heavy line on the right of Fig. 5-1(b), the controller behaves like the predetermined candidate control used to determine the margin. Beyond the



(a) Framework for Performance Prediction

(b) Regions of Operation

Figure 5-1: Overview of Performance Prediction. In Fig. 5-1(a), **z** is the performance output. In Fig. 5-1(b), $W_{\max}$ is the greatest level of predicted disturbance that allows a feasible optimization. The two shaded areas are those in which performance predictions can be made.

limit of feasibility, the constraint tightening means the problem is always infeasible, but right at the limit, it has only one solution available and its behavior is therefore predictable using linear system tools. Between the end of the unconstrained region and the limit of feasibility lies the *transition region*, shaded dark gray in Fig. 5-1(b). In this region, we approximate the performance by a smooth interpolation between the unconstrained behavior and the predetermined control policy. The key technical challenges of performing the prediction are determining, analytically, the limit of feasibility and the upper limit of the unconstrained region of operation.

Section 5.2 defines the problem statement for performance prediction. Section 5.3 reviews the robust MPC algorithm. The analytical prediction of performance is described in Section 5.4. Section 5.5 presents modifications to the prediction algorithm to include the effect of estimation uncertainty. Finally, Section 5.6 compares the performance predictions with numerical simulations for several examples.

5.2 Problem Statement

This section presents the problem statement for robust MPC with analytical performance prediction. It is identical to the problem statement in Section 2.2 with some modified notation and the inclusion of a performance output. There is also an additional assumption of uniform distribution of the disturbance, whereas Section 2.2 permitted any distribution residing within the stated bounded set.

The aim is to control a linear system with discretized dynamics, a constrained output, performance output and disturbance input

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{E}\mathbf{w}(k) && \text{(5.1a)} \\ \mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) && \text{(5.1b)} \\ \mathbf{z}(k) &= \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k) && \text{(5.1c)} \end{aligned}$$

where $\mathbf{x}(k) \in \Re^{N_x}$ is the state vector, $\mathbf{u}(k) \in \Re^{N_u}$ is the controlled input, $\mathbf{w}(k) \in \Re^{N_w}$ is the disturbance input, $\mathbf{y}(k) \in \Re^{N_y}$ is the constrained output and $\mathbf{z}(k) \in \Re^{N_z}$ is the performance output. The pair $(\mathbf{A}, \mathbf{B})$ is assumed to be controllable. The matrices $\mathbf{C}$ and $\mathbf{D}$ are chosen by the designer to form the outputs, which are constrained to remain within a bounded set

$$\mathbf{y}(k) \in \mathcal{Y} \; \forall k \tag{5.2}$$

This form of output constraints can capture both input and state constraints, or mixtures thereof, such as a limited control magnitude or an error requirement. The set $\mathcal{Y}$, also chosen by the designer, is a polytope defined by N_p inequalities

$$\mathcal{Y} = \{\mathbf{y} \mid \mathbf{p}_n^T \mathbf{y} \leq q_n \; \forall n \in 1 \ldots N_p\} \tag{5.3}$$

The disturbance is unknown but bounded. It is to be assumed uncorrelated and uniformly distributed in a hypercube

$$\mathbf{w}(k) \in \mathcal{B}_\infty(W_a) \; \forall k \tag{5.4}$$

where

$$\mathcal{B}_\infty(W) = \{\mathbf{w} \mid \|\mathbf{w}\|_\infty \leq W\} \tag{5.5}$$

We do not assume that W_a is known *a priori*: the disturbance model for control design, described in Section 5.3, uses a different value of the bound. Thus the prediction method considers the implication of inaccurate disturbance modeling.

The robust MPC presented in Chapter 2 can be used to control this system and guarantee robust constraint satisfaction, provided the actual disturbance W_a is overbounded by the bound used in the controller. The aim of this chapter is to predict the steady-state RMS value of the performance output $\sqrt{E[\mathbf{z}(k)^T\mathbf{z}(k)]}$.

5.3 Robustly-Feasible MPC

This section presents the MPC optimization and algorithm for the problem with performance prediction. This is based on the controller from Section 2.3, with the following specializations to accommodate the performance prediction

- the optimization stage cost is quadratic

  $$\ell(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}$$

  with $\mathbf{Q}$ symmetric positive definite and $\mathbf{R}$ symmetric positive semi-definite;

- the candidate control policy $\mathbf{K}(j)$ is constant and makes the system nilpotent;

- the terminal constraint set is the origin;

- the control algorithms find the *optimal* solution to each problem (some results in Chapter 2 only required *feasible* solutions).

These assumptions are made so that familiar LQR tools can be used to find the optimal control for the problem, ignoring the constraints. The significance of this will be shown later.

Define robust MPC optimization problem $\mathsf{P}(\mathbf{x}(k), W_p)$, starting from state $\mathbf{x}(k)$ with predicted disturbance bound W_p, as follows

$$J^*(\mathbf{x}(k), W_p) = \min_{\mathbf{u},\mathbf{x},\mathbf{y}} \sum_{j=0}^{N} \left(\mathbf{x}^T(k+j|k)\mathbf{Q}\mathbf{x}(k+j|k) + \mathbf{u}^T(k+j|k)\mathbf{R}\mathbf{u}(k+j|k) \right) \quad (5.6)$$

subject to $\forall j \in \{0 \dots N\}$

$$\begin{aligned}
\mathbf{x}(k+j+1|k) &= \mathbf{A}\mathbf{x}(k+j|k) + \mathbf{B}\mathbf{u}(k+j|k) && (5.7a)\\
\mathbf{y}(k+j|k) &= \mathbf{C}\mathbf{x}(k+j|k) + \mathbf{D}\mathbf{u}(k+j|k) && (5.7b)\\
\mathbf{x}(k|k) &= \mathbf{x}(k) && (5.7c)\\
\mathbf{x}(k+N+1|k) &= \mathbf{0} && (5.7d)\\
\mathbf{y}(k+j|k) &\in \mathcal{Y}(j; W_p) && (5.7e)
\end{aligned}$$

where N is the planning horizon. The choice of N is the responsibility of the designer, but the analysis tool presented in this chapter enables an investigation of its effect on performance. The set $\mathcal{Y}(j; W_p)$ in (5.7e) denotes the constraints on the j^{th} predicted step, tightened to accommodate a predicted disturbance level W_p. Tightening is performed by the recursion

$$\begin{aligned}
\mathcal{Y}(0; W_p) &= \mathcal{Y} && (5.8a)\\
\mathcal{Y}(j+1; W_p) &= \mathcal{Y}(j; W_p) \sim (\mathbf{C} + \mathbf{D}\mathbf{K}_H)\, \mathbf{L}_H(j) \mathbf{E} \mathcal{B}_\infty(W_p) && (5.8b)\\
& \qquad \forall j \in \{0 \dots N-1\}
\end{aligned}$$

where the set $\mathcal{B}_\infty(W_p)$ is the *predicted* disturbance set. As discussed in Section 5.2, the predicted disturbance limit W_p is not assumed to be equal to the actual limit W_a. A Matlab routine is available [30] for calculating the Pontryagin difference between two polytopes, and the result is also a polytope, with only the right-hand side of the inequalities changed [31], *i.e.* of the form

$$\mathcal{Y}(j; W_p) = \{\mathbf{y} \mid \mathbf{p}_n^T \mathbf{y} \leq q_n(j; W_p) \; \forall n \in 1 \dots N_p\} \quad (5.9)$$

where the values of $q_n(j; W_p)$ are determined by the Pontryagin difference algorithm. The controller $\mathbf{K}_H$ in (5.8b) is chosen by the designer such that the static linear feedback control law $\mathbf{u} = \mathbf{K}_H \mathbf{x}$ makes the system nilpotent in at most N steps. Also in (5.8b), define $\mathbf{L}_H(j)$ as the state transition matrix for the closed-loop system under this control law

$$\mathbf{L}_H(0) = \mathbf{I} \tag{5.10}$$

$$\mathbf{L}_H(j+1) = (\mathbf{A} + \mathbf{B}\mathbf{K}_H)\,\mathbf{L}_H(j)\ \forall j \in \{0 \dots N-1\} \tag{5.11}$$

The problem defined above is employed in the following algorithm.

Algorithm 5.1. (Robustly-Feasible MPC)

1. Solve problem $\mathsf{P}(\mathbf{x}(k), W_p)$
2. Apply control $\mathbf{u}(k) = \mathbf{u}^*(k|k)$ from optimizing sequence
3. Go to 1

The origin $\mathbf{0}$ is an invariant terminal constraint set, equivalent to $\mathcal{X}_F$ in (2.7d), and, for $W_p \geq W_a$, $\mathbf{w}(k) \in \mathcal{B}_\infty(W_a) \subseteq \mathcal{B}_\infty(W_p)$ corresponding to the bound (2.2). Therefore, under the assumption that $W_p \geq W_a$, the robust feasibility result of Theorem 2.1 applies. Feasibility at time $k = 0$ implies feasibility at all future time steps and constraint satisfaction $\mathbf{y}(k) \in \mathcal{Y}$.

5.4 Analytical Performance Prediction

The performance prediction is expressed in the following conjecture. The expected steady-state performance of a system (5.1a) under the control of Algorithm 5.1, expressed as the RMS value of the performance output (5.1c), for predicted and actual disturbances obeying

$$0 \leq W_a \leq W_p \leq W_{\max}$$

(the upper limit $W_{\max}$ is defined below) is approximated by

$$\sqrt{E[\mathbf{z}(k)^T\mathbf{z}(k)]} \approx$$

$$\begin{cases} G(\mathbf{K}_L)W_a, & 0 \leq W_a \leq W_C(W_p) \\ \lambda(W_a,W_p)G(\mathbf{K}_L)W_a + [1-\lambda(W_a,W_p)]G(\mathbf{K}_H)W_a, & W_C(W_p) \leq W_a \leq W_p \end{cases} \tag{5.12}$$

where

- $G(\mathbf{K})$ is the gain, from the infinity-norm of the disturbance to the two-norm of the output, of the system under a static, stabilizing, linear control $\mathbf{u}(k) = \mathbf{K}\mathbf{x}(k)$

- $\mathbf{K}_L$ is the finite-horizon LQR controller for the unconstrained system

- $W_C(W_p)$ is the level of actual disturbance W_a at which the constraints, tightened for predicted disturbance W_p, begin to influence the operation, *i.e.* below this level, MPC behaves like the unconstrained optimal controller. This defines the line separating the transition region and unconstrained region in Fig. 5-1(b).

- $W_{\max}$ is the highest level of predicted disturbance W_p that has a non-empty set of feasible initial states for the optimization $\mathsf{P}(\mathbf{x}(k), W_p)$

- $\lambda(W_a, W_p)$ is an interpolation function such that $\lambda(W_C(W_p), W_p) = 0$ for any W_p (*i.e.* on the dividing line between constrained and unconstrained regions in Fig. 5-1(b)) and $\lambda(W_a, W_{\max}) = 1$ for any $W_a \neq 0$ (*i.e.* at the limit of feasibility). With this definition, the performance gain is $G(\mathbf{K}_L)$ at the edge of the unconstrained region and $G(\mathbf{K}_H)$ at the limit of feasibility.

The above is the mathematical expression of the observations expressed in Fig. 5-1(b). The following subsections describe how these quantities are calculated.

5.4.1 Gain under Linear Control

Under the assumptions concerning the disturbance made in Section 5.2, the covariance matrix of the disturbance for unit input level $W_a = 1$ is given by

$$E[\mathbf{w}(k)\mathbf{w}^T(j)] = \begin{cases} \frac{1}{3}\mathbf{I} & k = j \\ \mathbf{0} & \text{otherwise} \end{cases}$$

Then, under the assumption that the state and input can be treated as normal random variables, the system gain is given by

$$G(\mathbf{K}) = \sqrt{\text{tr}\left\{\mathbf{C}(\mathbf{K})\mathbf{X}(\mathbf{K})\mathbf{C}(\mathbf{K})^T\right\}} \tag{5.13}$$

with $\mathbf{C}$ and $\mathbf{X}$ satisfying

$$\mathbf{X}(\mathbf{K}) = (\mathbf{A}+\mathbf{B}\mathbf{K})\mathbf{X}(\mathbf{K})(\mathbf{A}+\mathbf{B}\mathbf{K})^T + \frac{1}{3}\mathbf{E}\mathbf{E}^T \tag{5.14}$$

$$\mathbf{C}(\mathbf{K}) = \mathbf{F} + \mathbf{G}\mathbf{K} \tag{5.15}$$

where $\mathbf{X}(\mathbf{K}) = E_k[\mathbf{x}(k)\mathbf{x}^T(k)]$ is the state covariance matrix under control $\mathbf{K}$ and can be found by solving the Lyapunov equation for its positive-definite solution. The validity of this analysis, *i.e.* that bounded disturbances can be treated like Gaussian distributions, will be demonstrated by examples in Section 5.6.

Remark 5.1. (Alternative Performance Metric) It may be desirable to express the performance in terms of a quadratic cost

$$\sqrt{\underset{k}{E}\left[\mathbf{u}(k)^T\tilde{\mathbf{R}}\mathbf{u}(k) + \mathbf{x}(k)^T\tilde{\mathbf{Q}}\mathbf{x}(k)\right]}$$

where $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{R}}$ are weighting matrices for the cost under investigation and need not match those used in the optimization cost (5.6). This can be achieved by setting

$$\mathbf{F} = \begin{bmatrix} \hat{\mathbf{Q}} \\ \mathbf{0} \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} \mathbf{0} \\ \hat{\mathbf{R}} \end{bmatrix}$$

using the Cholesky factors $\hat{\mathbf{Q}}^T\hat{\mathbf{Q}} = \tilde{\mathbf{Q}}$ and $\hat{\mathbf{R}}^T\hat{\mathbf{R}} = \tilde{\mathbf{R}}$.

5.4.2 Unconstrained Control Solution

The controller $\mathbf{K}_L$ is the unconstrained finite-horizon LQR solution, found by solving the following Ricatti equation

$$\begin{aligned}
\mathbf{P}(N+1) &= \infty\mathbf{I} && (5.16)\\
&\quad \forall j \in \{0 \dots (N+1)\} \\
\mathbf{P}(j-1) &= \mathbf{Q} + \mathbf{A}^T\mathbf{P}(j)\mathbf{A} - \mathbf{A}^T\mathbf{P}(j)\mathbf{B}\left[\mathbf{R} + \mathbf{B}^T\mathbf{P}(j)\mathbf{B}\right]^{-1}\mathbf{B}^T\mathbf{P}(j)\mathbf{A} && (5.17)\\
\mathbf{K}(j-1) &= -\left[\mathbf{R} + \mathbf{B}^T\mathbf{P}(j)\mathbf{B}\right]^{-1}\mathbf{B}^T\mathbf{P}(j)\mathbf{A} && (5.18)\\
\mathbf{K}_L &= \mathbf{K}(0) && (5.19)
\end{aligned}$$

It is also necessary to express the predicted outputs associated with the unconstrained solution as a function of the initial state $\mathbf{x}(k)$. Write

$$\mathbf{y}(k+j|k) = \mathbf{H}(j)\mathbf{x}(k) \tag{5.20}$$

where

$$\mathbf{H}(j) = (\mathbf{C} + \mathbf{D}\mathbf{K}(j))\mathbf{L}(j) \tag{5.21}$$

and the matrix $\mathbf{L}$ is the state transition matrix given by recursion

$$\begin{aligned}
\mathbf{L}(0) &= \mathbf{I} && (5.22)\\
\mathbf{L}(j+1) &= (\mathbf{A} + \mathbf{B}\mathbf{K}(j))\mathbf{L}(j) && (5.23)
\end{aligned}$$

These relations are used in Section 5.4.4 to find the limit of the unconstrained region.

5.4.3 Limit of Feasibility

The problem $\mathsf{P}(\mathbf{x}(k), W_p)$ becomes more constrained as the predicted disturbance W_p is increased, due to the constraint tightening in (5.8). The limit of feasibility $W_{\max}$ is

defined as the greatest value of W_p for which the set of feasible initial states for the problem $\mathsf{P}(\mathbf{x}(k), W_p)$ is non-empty. This is equivalent to the condition $\mathbf{0} \in \mathcal{Y}(N; W_p)$. If this condition holds, $\mathbf{0}$ is a feasible initial considition. If the condition does not hold, the terminal constraint and the final output constraint conflict and there can be no feasible solutions. Therefore the limit of feasibility is found by solving the following optimization

$$\begin{aligned} W_{\max} &= \max W_p \\ \text{s.t. } \mathbf{0} &\in \mathcal{Y}(N; W_p) \end{aligned} \tag{5.24}$$

(This is identical to the maximum disturbance parameter $\delta_{\max}$ in Remark 2.6, specialized for the problem with performance prediction.) By the definition of the Pontryagin difference (2.10), the condition $\mathbf{0} \in \mathcal{A} \sim \mathcal{B}$ is equivalent to $\mathcal{B} \subseteq \mathcal{A}$. Further, if $\mathcal{A}$ is convex and $\mathcal{B}$ is a mapping of a unit hypercube of dimension N_w through a matrix $\mathbf{M}$, this can be expressed in terms of the vertices of $\mathcal{B}$

$$\mathbf{M}\mathbf{v}_i \in \mathcal{A} \; \forall i \in \{1 \dots 2^{N_w}\}$$

where $\mathbf{v}_i$ are the vertices of the unit hypercube, which can be evaluated straightforwardly. Extending this principle to the recursion for $\mathcal{Y}(N; W_p)$ in (5.8b) gives the following optimization to find $W_{\max}$

$$\begin{aligned} W_{\max} = \quad & \max W_p \\ \text{s.t.} \quad & W_p \sum_{j=0}^{(N-1)} (\mathbf{C} + \mathbf{D}\mathbf{K}_H(j))\mathbf{L}_H(j)\mathbf{E}\mathbf{v}_{i_j} \in \mathcal{Y} \\ & \forall (i_0, i_1 \dots i_{(N-1)}) \in \left\{\{1 \dots 2^{N_w}\} \times \dots \times \{1 \dots 2^{N_w}\}\right\} \end{aligned} \tag{5.25}$$

Observe that the constraint has to be evaluated for all *combinations* of vertices of the unit hypercube. Since $\mathcal{Y}$ is a polytope (5.3), this can be rewritten as

$$W_{\max} = \min_{\substack{(n, i_0, i_1 \dots i_{(N-1)}) \in \\ \left\{\{1 \dots N_p\} \times \{1 \dots 2^{N_w}\} \times \dots \times \{1 \dots 2^{N_w}\}\right\}}} \frac{q_n}{\mathbf{p}_n^T \sum_{j=0}^{(N-1)} (\mathbf{C} + \mathbf{D}\mathbf{K}_H(j))\mathbf{L}_H(j)\mathbf{E}\mathbf{v}_{i_j}} \tag{5.26}$$

which can be readily found, despite the large numbers of constraints to check.

5.4.4 Limit of Unconstrained Operation

The quantity $W_C(W_p)$ is the greatest value of the actual disturbance W_a for which the trajectory of the state rarely interacts with the constraints, tightened for a predicted disturbance W_p. This corresponds to the division between the unconstrained region and the transition region, shown in Fig. 5-1(b). To calculate $W_C(W_p)$, we find the value of actual disturbance W_a for which, if the closed-loop system behaves like the unconstrained optimal control $\mathbf{u}(k) = \mathbf{K}_L\mathbf{x}(k)$, then 95% of the solutions to the optimal unconstrained problem (5.20) satisfy the constraints. The choice of the 95% threshhold is arbitrary and will be checked in simulations in Section 5.6.

If the system behaves as if controlled by the unconstrained optimal controller, and assuming that the state can be treated as a Gaussian random variable, then with 95% probability, the state resides in an ellipsoid

$$\frac{1}{W_a^2}\mathbf{x}^T(k)\mathbf{X}(\mathbf{K}_L)^{-1}\mathbf{x}(k) \leq 9 \tag{5.27}$$

where $\mathbf{X}(\mathbf{K}_L)$ is the state covariance matrix (5.14) under the action of the unconstrained control (5.19). This can be rewritten as a norm bound

$$\|\mathbf{Y}\mathbf{x}(k)\|_2 \leq 3W_a \tag{5.28}$$

where the matrix $\mathbf{Y}^T\mathbf{Y} = \mathbf{X}^{-1}(\mathbf{K}_L)$. Since $\mathbf{X}(\mathbf{K}_L)$ is taken to be the positive definite solution to the Lyapunov equation (5.14), the matrix $\mathbf{Y}$ exists and is invertible. Since the constraints (5.7e) are of the form

$$\mathbf{p}_n^T\mathbf{y}(k+j|k) \leq q_n(j;W_p)$$

we use (5.28) to derive a bound (for 95% probability) on the quantity $\mathbf{p}_n^T\mathbf{y}(k+j|k)$

as follows

$$\begin{aligned}\mathbf{p}_n^T\mathbf{y}(k+j|k) &= \mathbf{p}_n^T\mathbf{H}(j)\mathbf{x}(k)\\ &= \mathbf{p}_n^T\mathbf{H}(j)\mathbf{Y}^{-1}\mathbf{Y}\mathbf{x}(k)\\ &\leq 3W_a\|\mathbf{Y}^{-T}\mathbf{H}(j)^T\mathbf{p}_n\|_2\end{aligned}$$

where $\mathbf{H}(j)$ are given by (5.20). Therefore, using the limits $q_n(j;W_p)$ for the tightened constraints (5.9), the unconstrained solution is feasible, with 95% probability, if

$$3W_a\|\mathbf{Y}^{-T}\mathbf{H}(j)^T\mathbf{p}_n\| \leq q_n(j;W_p)\ \forall n \in \{1\dots N_P\}\ \forall j \in \{0\dots N\} \tag{5.29}$$

Then the value of $W_C(W_p)$ can be found by finding the greatest value of W_a for which all conditions (5.29) hold, which can be done by simply checking each constraint and taking the minimum value

$$W_C(W_p) = \min_{n\in\{1\dots N_p\}j\in\{0\dots N\}} \frac{q_n(j;W_p)}{3\|\mathbf{Y}^{-T}\mathbf{H}(j)^T\mathbf{p}_n\|_2} \tag{5.30}$$

5.4.5 Interpolation Function

The function $\lambda(W_a, W_p)$ is used in the transition region for interpolation between the unconstrained regime and the limit of feasibility

$$\lambda = \left(\frac{W_a - W_C(W_p)}{W_a}\right)^r \tag{5.31}$$

where $r > 2$ gives a smooth transition away from the unconstrained performance. As $W_p \to W_{\max}$ then $W_C(W_p) \to 0$ hence also $\lambda \to 1$ for $W_a \neq 0$. Also, $\lambda \to 0$ as $W_a \to W_C(W_p)$.

Finally, the calculations described in the subsections above are combined in the following algorithm for performance prediction.

Algorithm 5.2. Performance Prediction

Data: $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{G}, \mathcal{Y}, \mathbf{K}_H, N, \mathbf{Q}, \mathbf{R}, W_a, W_p$

1. If $W_a > W_p$, return 0 (not robust). Stop.

2. Calculate $W_{\max}$ using (5.25). If $W_p > W_{\max}$, return 0 (infeasible). Stop.

3. Calculate the unconstrained controller $\mathbf{K}_L$ using (5.19), gains $G(\mathbf{K}_L)$ using (5.13) and form the constraints $q_n(j; W_p)$ (5.9) using (5.8a) and (5.8b)

4. Calculate $W_C(W_p)$ using (5.30)

5. If $W_a \leq W_C(W_p)$, return $G(\mathbf{K}_L)W_a$ (unconstrained region). Stop.

6. Calculate gain $G(\mathbf{K}_H)$ and interpolation function λ using (5.31) and return

$$\lambda(W_p, W_a)G(\mathbf{K}_L)W_a + [1 - \lambda(W_p, W_a)]G(\mathbf{K}_H)W_a$$

5.5 Estimation Error

The performance prediction approximation presented in Section 5.4 can be combined with the approach to handling estimation error in Chapter 3 to provide a method of predicting the performance of robust MPC using Algorithm 3.1 in the presence of uncertain state estimates. Assume the state estimate includes an additive error

$$\hat{\mathbf{x}}(k) = \mathbf{x}(k) + \mathbf{M}\mathbf{e}(k)$$

where the error $\mathbf{e}(k) \in \Re^{N_e}$ is uniformly distributed in a hypercube with the same norm limit as the disturbance bound (5.4)

$$\mathbf{e}(k) \in \mathcal{B}_\infty(W_a)$$

Also assume that the estimation error is white and independent of the disturbance $\mathbf{w}(k)$. Differences in magnitude between the estimation error and disturbance are handled by varying the matrices $\mathbf{E}$ and $\mathbf{M}$. Note that Section 3.2.3 describes how the formulation can be further modified to account for correlation between process and estimation noises, which is likely to occur in practice.

The performance prediction algorithm for the problem with estimation error is identical in form to Algorithm 5.2 but with modifications to some of the quanities

involved. The unconstrained solution is unchanged from (5.19). The following subsections describe the modifications to the other calculations.

5.5.1 Gain under Linear Control

When the control is behaving like a linear feedback $\mathbf{u}(k) = \mathbf{K}\mathbf{x}(k)$ acting on the estimate, the dynamics of the estimate (3.4) can be rewritten as

$$\left[\begin{array}{c} \hat{\mathbf{x}}(k+1) \\ \mathbf{e}(k+1) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{A}+\mathbf{B}\mathbf{K} & -\mathbf{A}\mathbf{M} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \hat{\mathbf{x}}(k) \\ \mathbf{e}(k) \end{array}\right] + \left[\begin{array}{c|c} \mathbf{E} & \mathbf{M} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right] \left[\begin{array}{c} \mathbf{w}(k) \\ \mathbf{e}(k+1) \end{array}\right]$$

Note that this system appears unusual as it is driven by a "look-ahead" input $\mathbf{e}(k+1)$. This is permissible in this case as only the statistical properties of the signal $\mathbf{e}(k)$ are assumed known for the performance analysis. The performance gain under linear control from (5.13) can be rewritten for the estimation error case

$$G(\mathbf{K}) = \sqrt{\mathrm{tr}\left\{\hat{\mathbf{C}}(\mathbf{K})\hat{\mathbf{X}}(\mathbf{K})\hat{\mathbf{C}}(\mathbf{K})^T\right\}}$$

with $\hat{\mathbf{C}}$ and $\hat{\mathbf{X}}$ satisfying

$$\begin{aligned} \hat{\mathbf{X}}(\mathbf{K}) &= \hat{\mathbf{A}}\hat{\mathbf{X}}(\mathbf{K})\hat{\mathbf{A}}^T + \frac{1}{3}\mathbf{E}\mathbf{E}^T \\ \hat{\mathbf{C}}(\mathbf{K}) &= [\mathbf{F}+\mathbf{G}\mathbf{K} \mid -\mathbf{F}\mathbf{M}] \end{aligned}$$

where

$$\hat{\mathbf{A}} = \left[\begin{array}{c|c} \mathbf{A}+\mathbf{B}\mathbf{K} & -\mathbf{A}\mathbf{M} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right] \quad \hat{\mathbf{E}} = \left[\begin{array}{c|c} \mathbf{E} & \mathbf{M} \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right]$$

5.5.2 Limit of Feasibility

As described in Section 3.2.1, the constraints are tightened to accommodate the effective process noise

$$\hat{\mathbf{w}}(k) = \mathbf{E}\mathbf{w}(k) + \mathbf{M}\mathbf{e}(k+1) - \mathbf{A}\mathbf{M}\mathbf{e}(k)$$

which can be rewritten as

$$\hat{\mathbf{w}}(k) = \hat{\mathbf{E}}\bar{\mathbf{w}}(k)$$

where

$$\hat{\mathbf{E}} = [\mathbf{E} \mid \mathbf{M} \mid -\mathbf{AM}] \qquad \bar{\mathbf{w}}(k) = \begin{bmatrix} \mathbf{w}(k) \\ \mathbf{e}(k+1) \\ \mathbf{e}(k) \end{bmatrix}$$

Therefore, the calculation of the limit of feasibility is modified by replacing $\mathbf{Ew}(k)$ with $\hat{\mathbf{E}}\bar{\mathbf{w}}(k)$ so (5.26) becomes

$$W_{\max} = \min_{\substack{(n,i_0,i_1\ldots i_{(N-1)})\in \\ \left\{\{1\ldots N_p\}\times\{1\ldots 2^{N_{\hat{w}}}\}\times\ldots\times\{1\ldots 2^{N_{\hat{w}}}\}\right\}}} \frac{q_n}{\mathbf{p}_n^T \sum_{j=0}^{(N-1)} (\mathbf{C} + \mathbf{DK}_H(j))\mathbf{L}_H(j)\hat{\mathbf{E}}\hat{\mathbf{v}}_{i_j}}$$

where $\hat{\mathbf{v}}_{i_j}$ denotes the vertices of the hypercube containing $\bar{\mathbf{w}}(k)$.

5.5.3 Limit of Unconstrained Operation

The calculation of $W_C(W_p)$ involves both the constraint tightening and the deviation due to the actual uncertainty, so the modifications of both the previous subsections are combined. The expression for the limit of unconstrained operation (5.30) becomes

$$W_C(W_p) = \min_{n\in\{1\ldots N_p\}j\in\{0\ldots N\}} \frac{\hat{q}_n(j;W_p)}{3\|\hat{\mathbf{Y}}^{-T}\mathbf{H}(j)^T\mathbf{p}_n\|_2}$$

where $\hat{q}_n(\cdot)$ denotes the tightened constraint limits using the effective process noise $\hat{\mathbf{E}}\bar{\mathbf{w}}(k)$ and $\hat{\mathbf{Y}}$ is calculated from the covariance matrix for the state estimate $\hat{\mathbf{X}}(\mathbf{K})$ such that $\hat{\mathbf{Y}}^T\hat{\mathbf{Y}} = E[\hat{\mathbf{x}}(k)^T\hat{\mathbf{x}}(k)]$. The matrices $\mathbf{H}(j)$ depend only on the unconstrained LQR solution and are not changed by the inclusion of estimation error.

5.6 Examples

This section demonstrates the performance prediction approximation using Algorithm 5.2 by comparing analytical predictions with simulation results for a variety of systems. The first two examples in this section investigate the variation of performance with actual and predicted disturbances, exploring the space shown in Fig. 5-1(b). The third example shows the variation of performance with constraint limits, demonstrating the application of the analytical performance prediction to study trades between design parameters. The fourth example investigates the effect of horizon length on performance.

Fig. 5-2 shows results for neutrally-stable and unstable second-order systems. The respective system $\mathbf{A}$ matrices are

$$\mathbf{A}_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} 0.9 & 0.6 \\ 0.2 & 0.5 \end{bmatrix}$$

with the following parameters common to both examples

$$\mathbf{B} = \begin{bmatrix} 0.5 \\ 1.0 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix} \quad \mathbf{E} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}$$

The constraints are a unit bound on the infinity norm *i.e.* a unit magnitude bound on each element

$$\|\mathbf{y}(k)\|_\infty \leq 1$$

The optimization costs penalize the control energy with a smaller weighting on the state

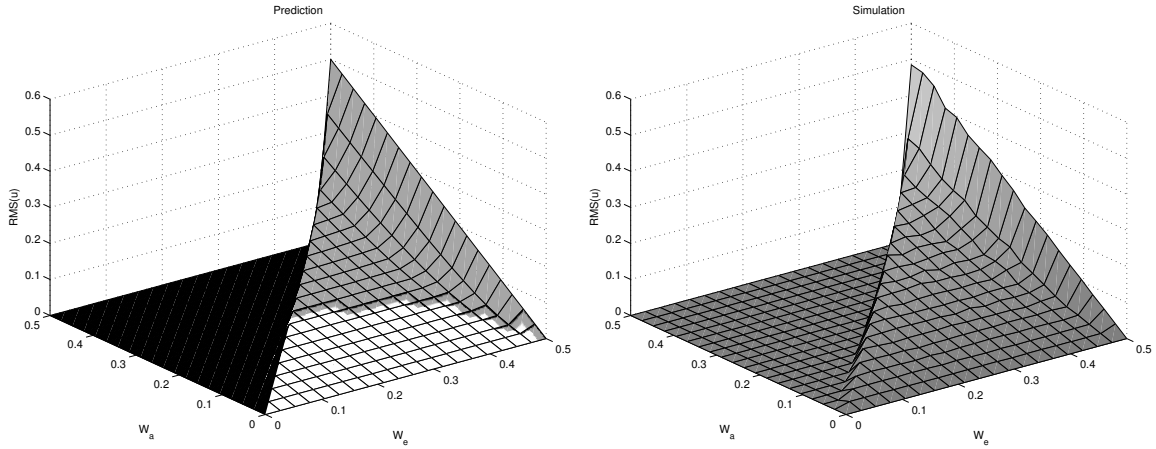
$$\mathbf{Q} = \mathbf{I}_2 \qquad \mathbf{R} = 100$$

The problem was solved with a horizon of $N = 6$ steps and the interpolation (5.31) used a power of $r = 4$.

5.6.1 Effect of Expected and Actual Disturbance Levels

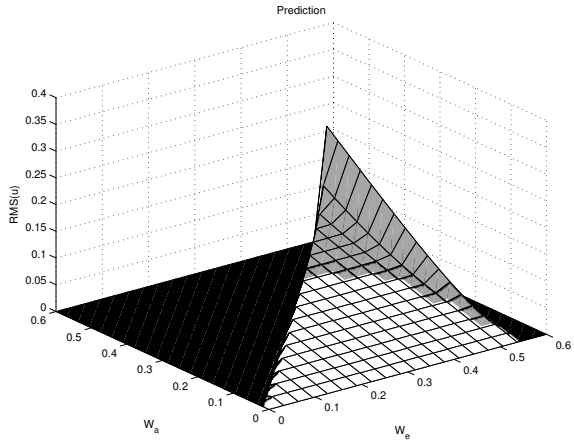
The expected disturbance W_p and actual disturbance W_a were varied between 0 and the appropriate $W_{\max}$, precalculated for each system. A simulation of 1000 time steps was performed at each setting (W_a, W_p). Fig. 5-2 compares performance predictions with simulation results for the metric of control effort. The predictions were found using Algorithm 5.2 with the performance output matrices $\mathbf{F} = [0\ 0]$ and $\mathbf{G} = 1$. In the prediction plots (Figs. 5-2(a) and 5-2(c)), the unconstrained region is white and the transition region shaded gray. In both cases, the predictions closely match the results of the simulations. At low disturbance levels the control RMS surface is flat as the controller behaves like the unconstrained LQR. Moving toward the top-right of the plots, the constraints become significant and the control effort begins to rise sharply away from the flat LQR "surface" as it becomes harder to satisfy the constraints. At the limit of feasibility, seen as the high ridge on the right-hand side of plots, the performance plot is a straight line again as the MPC behaves like the nilpotent candidate controller.

Fig. 5-3 compares the predicted RMS position errors with the simulation results using the matrices $\mathbf{F} = [1\ 0]$ and $\mathbf{G} = 0$. Again, predicted and simulated performances are in agreement. Notice that at high levels of actual disturbance, *i.e.* in the transition region (shaded gray), the RMS position error reduces as the expected disturbance level increases. This is because the MPC behaves more like the nilpotent regulator as W_p is increased and less like the unconstrained LQR, which penalizes control effort and therefore leads to large state deviations.

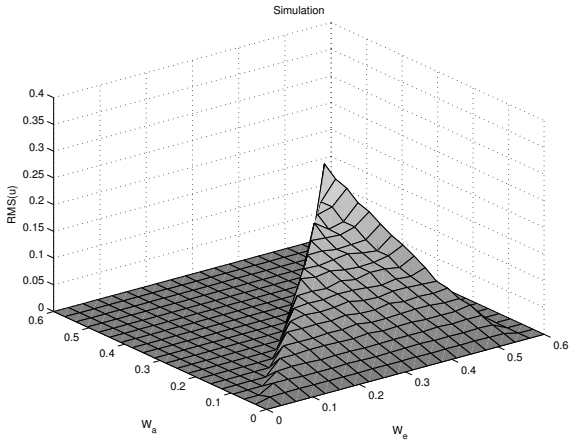


(a) Predicted, Neutrally-Stable $\mathbf{A}_1$

(b) Simulated, Neutrally-Stable $\mathbf{A}_1$

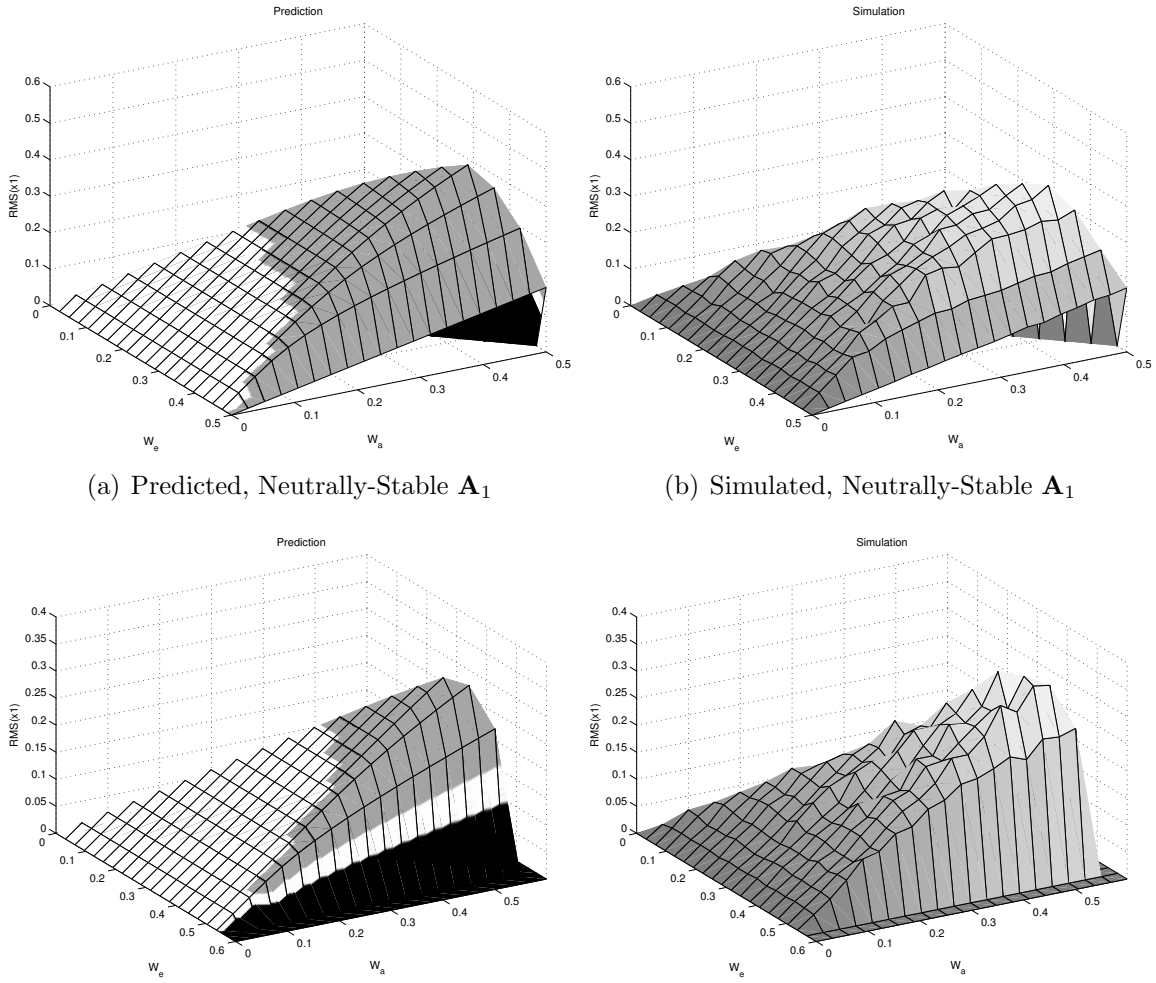


(c) Predicted, Unstable $\mathbf{A}_2$



(d) Simulated, Unstable $\mathbf{A}_2$

Figure 5-2: Comparison of Predicted Control Effort and Simulation Results for Two Systems with Varying Expected and Actual Disturbances. In Figs. 5-2(a) and 5-2(c), the unconstrained region is in white and the transition region in gray.



(a) Predicted, Neutrally-Stable $\mathbf{A}_1$

(b) Simulated, Neutrally-Stable $\mathbf{A}_1$

(c) Predicted, Unstable $\mathbf{A}_2$

(d) Simulated, Unstable $\mathbf{A}_2$

Figure 5-3: Comparison of Predicted Position RMS and Simulation Results for Two Systems with Varying Expected and Actual Disturbances. In Figs. 5-3(a) and 5-3(c), the unconstrained region is in white and the transition region in gray.

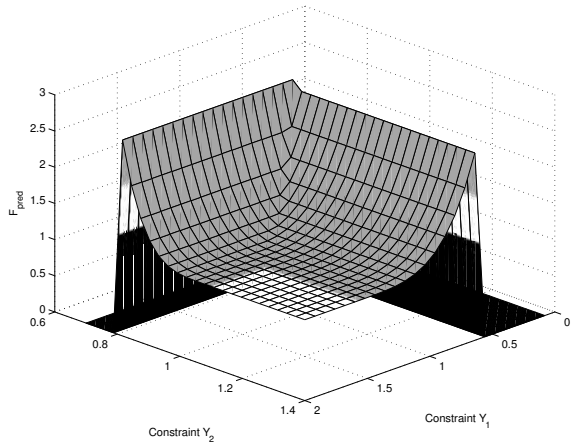
5.6.2 Effect of Constraint Settings

Fig. 5-4 shows an example in which the constraint levels are varied. The example uses the neutrally-stable system matrix $\mathbf{A}_1$. The predicted disturbance W_p is fixed at 80% of the maximum $W_{\max}$ and the actual disturbance W_a at 80% of W_p. The constraints are

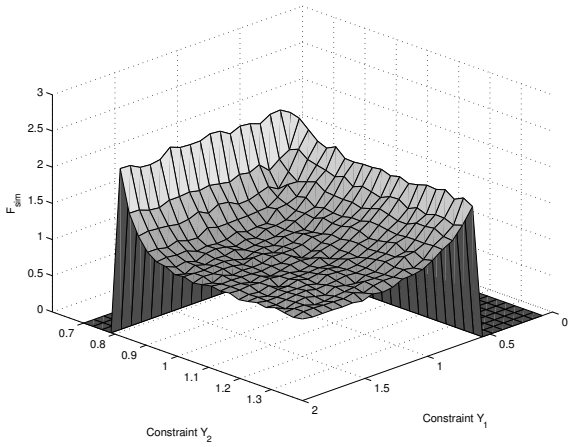
$$|y_1(k)| \leq Y_1 \quad |y_2(k)| \leq Y_2 \quad |y_3(k)| \leq 1$$

where Y_1 and Y_2 are variable limits on the position and velocity, respectively, whose effect is to be investigated. In the transition region, shaded gray in Fig. 5-4(a), the exact level of performance is underestimated by the prediction method. However, the predictions still capture important trends in the performance. The contour plots of the predictions (Fig. 5-4(c)) and the simulation data (Fig. 5-4(d)) are almost identical, apart from variations caused by randomness in the simulations. The "island" pattern in the top-right of Fig. 5-4(d) indicates that the performance surface is roughly flat in this region, with the small changes in simulation results dipping up and down across the contour line. The limiting values of Y_1 and Y_2 at which the constraints become significant are accurately predicted, as are the limits of feasibility.

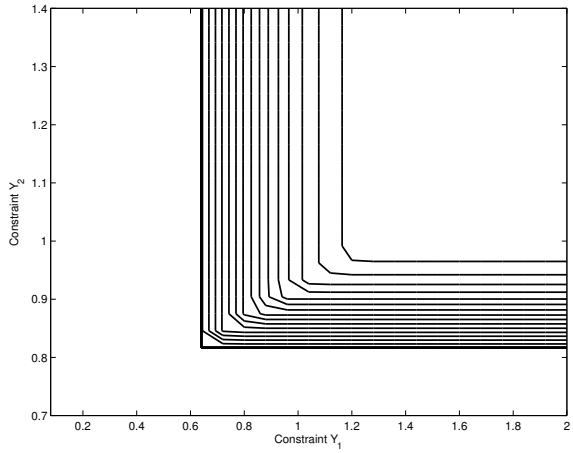
The performance prediction accurately identifies regions of different sensitivity, as shown in Fig. 5-4(e). In region A, the performance is sensitive to the position constraint setting Y_1 but not to the velocity constraint setting Y_2. In region B, the performance is insensitive to both Y_1 and Y_2. In region C, the performance is sensitive to the velocity constraint setting Y_2 but not to the position constraint setting Y_1. So, for example, if the initial controller design settings are in region B or C, the position control tolerance Y_1 can be tightened as far as the boundary with region A without incurring any penalty in control effort. Hence it can be seen that the prediction method allows the designer to assess the impact of design decisions such as constraint settings.



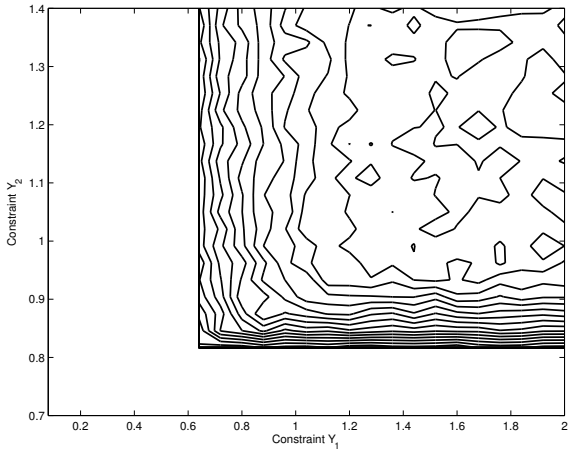
(a) Predicted



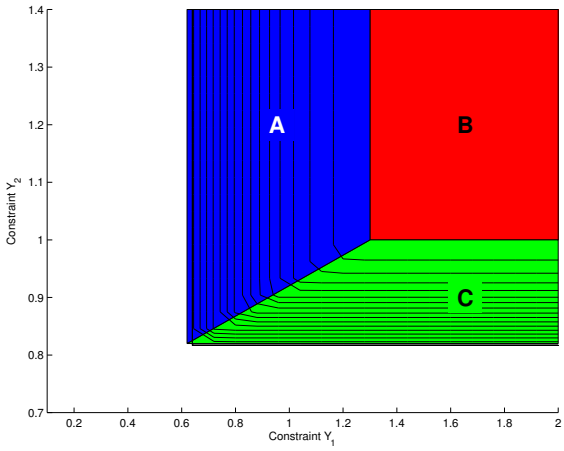
(b) Simulated



(c) Predicted



(d) Simulated



(e) Regions of Operation

Figure 5-4: Comparison of Predicted Performance and Simulation Results for a System with Varying Constraints

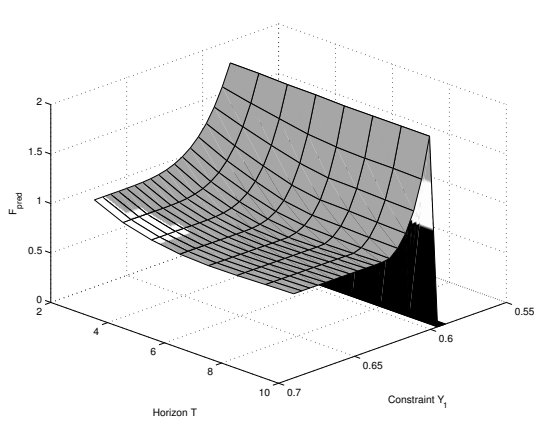
5.6.3 Effect of Horizon Length

Fig. 5-5 shows another trade study example, again involving the neutrally-stable system. The variables under investigation are the planning horizon N and the control magnitude constraint Y_3, with the state constraints held constant.

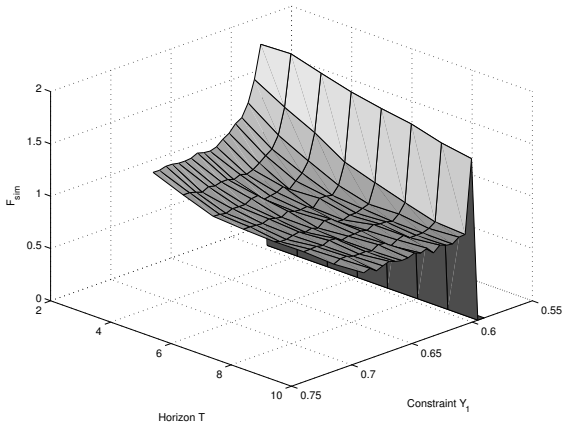
$$|y_1(k)| \leq 1 \quad |y_2(k)| \leq 1 \quad |y_3(k)| \leq Y_3$$

Again, there is some inaccuracy in the predicted performance levels in the transition region, but the contour plots are a close match. This result demonstrates how the performance prediction method can be employed to make a methodical choice of the prediction horizon. By definition, MPC requires the designer to choose a horizon, but this is commonly achieved by judgement, backed up by simulation.

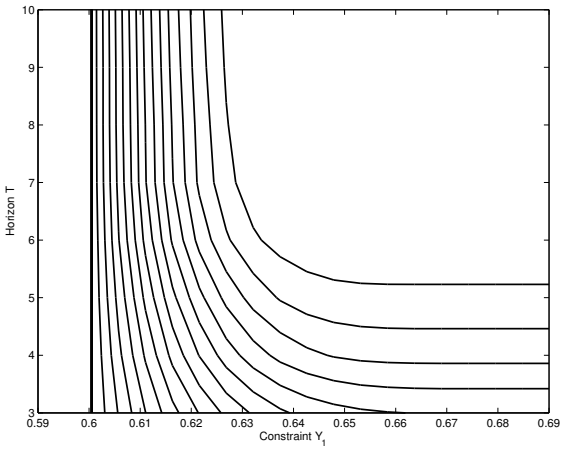
The contour plot, accurately determined by the performance prediction method, enables regions of operation to be identified based on sensitivity of the performance to the two design variables, shown in Fig. 5-5(e). These regions capture important trends in performance and can be used to indentify where savings can be made. For example, in regions A and B, the performance is insensitive to the horizon length N. Therefore, if the initial design settings are in either of these regions, the horizon length can be reduced as far as the edge of those regions, giving a shorter computation without incurring a performance penalty. Similarly, if the settings are in region C, where the performance is independent of control authority Y_1, the authority can be limited further, up to the boundary with region D, at no cost to performance.



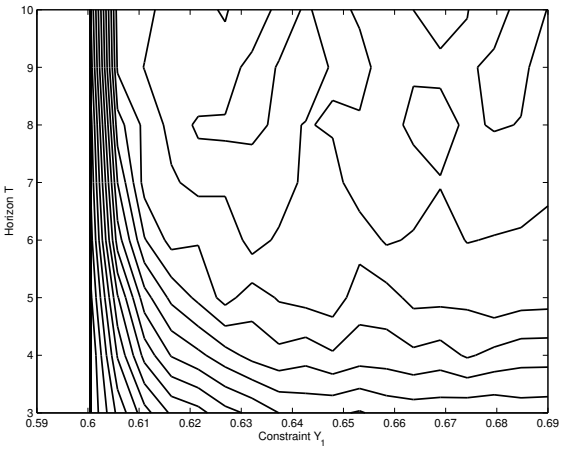
(a) Predicted



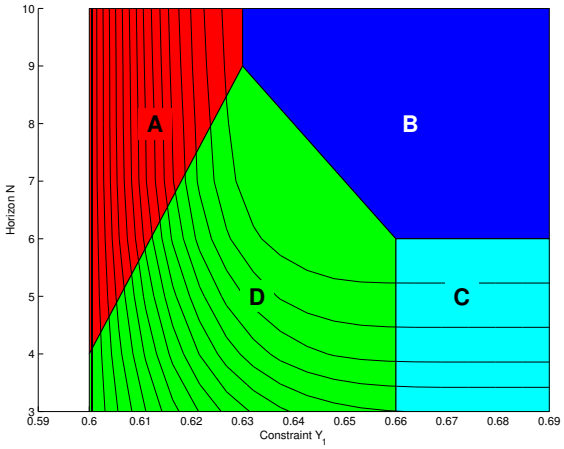
(b) Simulated



(c) Predicted



(d) Simulated



(e) Regions of Operation

Figure 5-5: Comparison of Predicted Performance and Simulation Results for a System with Varying Control Constraint and Horizon

5.6.4 Estimation Error

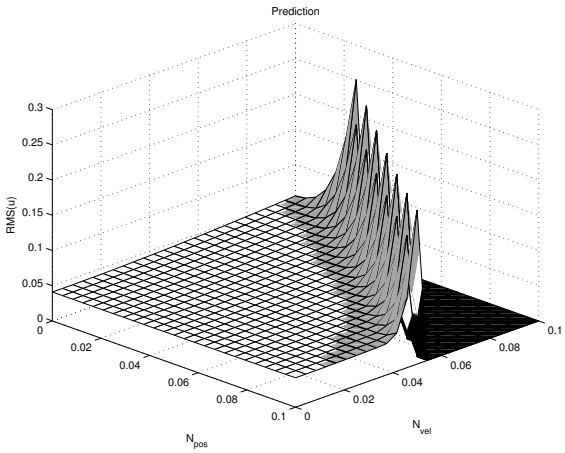
The example in this section investigates the effect of estimation error on the double-integrator control problem from Section 5.6.1. The disturbance and estimation errors were as follows

$$|e_1(k)| \leq N_{pos} \quad |e_2(k)| \leq N_{vel} \quad |w(k)| \leq 0.1$$

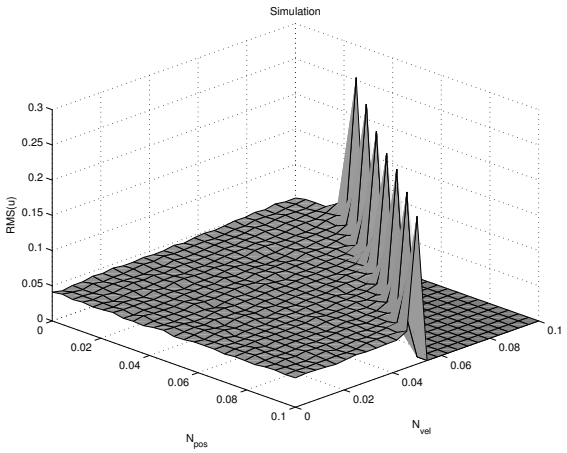
where the position and velocity errors, N_{pos} and N_{vel} respectively, were varied in the range $[0, 0.1]$. Fig. 5-6 compares predicted control RMS with results from simulations for different values of N_{pos} and N_{vel}. Fig. 5-6(e) shows the design space divided into regions of operation, based on sensitivity. In region A, the performance is independent of position noise but is sensitive to velocity noise. In region B, the performance becomes highly sensitive to both noise levels.

5.7 Summary

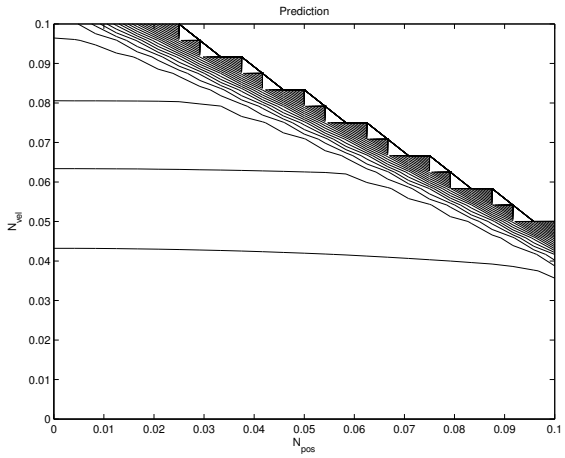
This chapter has presented a method of analytically predicting the expected closed-loop performance of a system using a form of robust Model Predictive Control. This enables trade studies on the effect of controller parameters to be performed without extensive numerical simulation. The prediction method has been demonstrated for several example systems and compared to simulation results. It has been shown to correctly identify trends in performance as a function of disturbance levels and constraint settings.



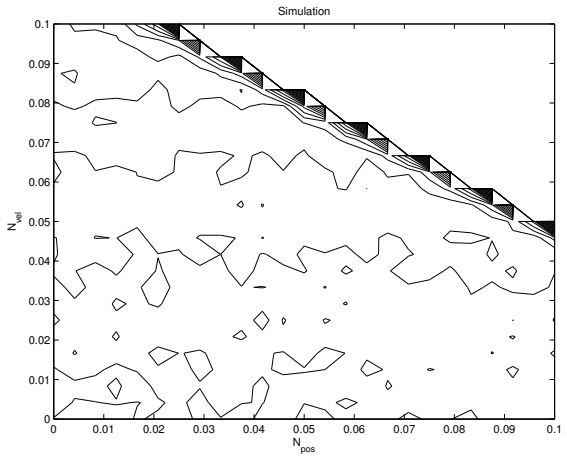
(a) Predicted



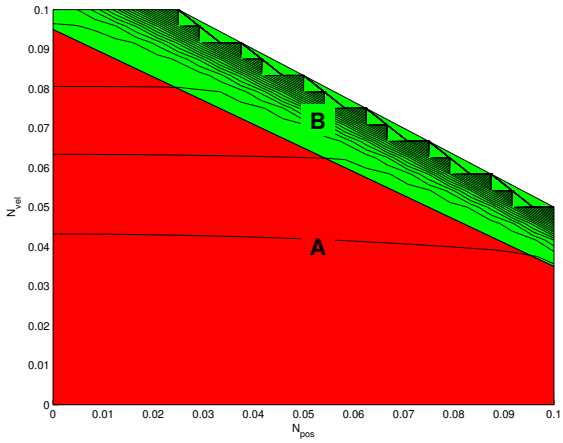
(b) Simulated



(c) Predicted



(d) Simulated



(e) Regions of Operation

Figure 5-6: Comparison of Predicted Performance and Simulation Results for a System with Varying Control Constraint and Horizon

Chapter 6

Hardware Demonstrations

This chapter describes three demonstrations using the methods developed in this thesis to control vehicles in two hardware testbeds, one involving spacecraft and the other using ground rovers. The first two demonstrations use the MPC presented in Chapters 2 and 3 to control a miniature satellite and a rover rendezvous, respectively. The third demonstration uses the decentralized MPC presented in Chapter 4 to control multiple rovers subject to collision avoidance constraints. The demonstrations serve two main purposes: first, to show that the controllers fulfill their requirements for control problems in the presence of realistic uncertainty; and second, to show that the controllers can be implemented in real time and in the presence of actual computation and communication delays.

6.1 Spheres

Fig. 6-1 shows a single "Sphere" indoor mini-satellite, part of the Synchronized Position Hold Engage and Reorient Experimental Satellites (SPHERES) [58] test facility at MIT's Space Systems Laboratory. Each Sphere has an on-board processor, an ultrasound position and attitude determination system, and twelve cold-gas thrusters providing control of six degrees-of-freedom. The complete testbed is comprised of multiple Spheres linked by wireless communications to a laptop base station. The system is designed for use in the International Space Station for formation-flying and

Figure 6-1: A "Sphere": a single satellite from the SPHERES testbed

docking experiments. This section describes experiments demonstrating robust MPC for position-holding using a single Sphere in a ground laboratory, floating on an air table to provide low friction movement in two dimensions.

6.1.1 Experiment Set-Up

The objective of the control is to keep the Sphere inside a 20 cm by 20 cm square "error box" in the presence of uncertainty, which arises from several sources. Disturbance forces include friction and unevenness of the operating surface. The ultrasonic position determination is subject to measurement inaccuracy. The dynamics of the Sphere are also uncertain, as gas depletion causes mass changes and the actual thrust applied for a given command varies due to pressure changes.

The robust MPC position controller runs every 5s and applies control in the form

of short pulses of thrust. The force exerted by each thruster is fixed but the firing pulse length is variable. We model this as an impulsive change in velocity ΔV, with up to ± 2.8 cm/s available at each step. In practice, there is a minimum firing time, but this is neglected in the controller and treated as uncertainty. The optimization, performed over a horizon of five steps, uses a point mass model moving in free-space and acted upon by impulsive velocity changes, allowing each of the X- and Y-directions to be handled as independent 1-D double-integrator systems, as in the example in Section 2.5.1. In between thruster firings for translational control, a pre-existing attitude controller, provided as part of the SPHERES testbed, is used to keep the Sphere axes aligned with the global reference frame. Thus the attitude and translational controllers are decoupled. Before applying MPC at each planning step, the state estimate is propagated forward to predict the state 0.3s ahead, accounting for delay as described in Section 3.2.5. Preliminary experiments identified a delay of 0.3s and prediction error bounds of ± 1 cm in position and ± 5 mm/s in velocity. The objective of the MPC optimization is to keep the Sphere inside the error box while minimizing fuel use.

The MPC optimization for the controller described above is a linear program (LP). Instead of implementing the LP on-line in the Sphere processor, the experiment employed the explicit MPC approach described in Ref. [59]. It can be shown that the solution of the MPC optimization for an LTI system with polytopic constraints is a piecewise affine (PWA) function of the initial condition [59]. The equivalent PWA control law for a single axis of the Sphere experiment was calculated in Matlab, using the Multi-Parametric Linear Programming (mp-LP) algorithm given in Ref. [59], and used to generate a C header file. A PWA look-up function to read this file was implemented in C within the SPHERES Guest Scientist programming interface [58]. This approach offers a rapid route from Matlab control design, analysis and simulation to implementation on Spheres for various MPC experiments, avoiding the need to implement new optimization problems in C code. The PWA law is identical to the solution of the optimization but much faster to evaluate in real-time. The calculation of the explicit solution can be extensive but it does not need to be done in real-time.

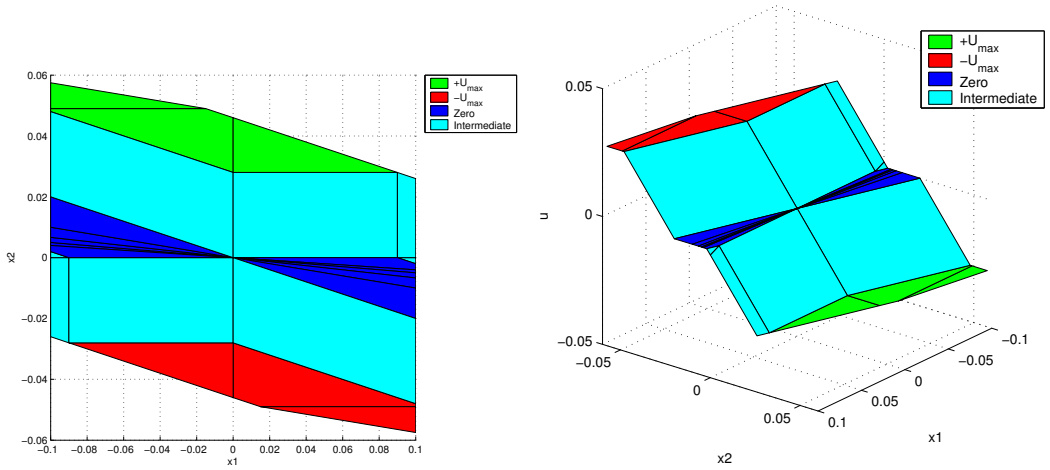


Figure 6-2: Precalculated Piecewise Affine Control Law for Spheres Experiment. This is equivalent to the online solution of the MPC optimization. (x_1 is position, x_2 velocity and u control, as ΔV)

Fig. 6-2 shows two views of the PWA control law for one axis.

6.1.2 Results

When the Sphere is released at rest on the air table without active control, the slight slope of the surface causes it to drift steadily, hitting the edge of the air table after roughly 20s. Fig. 6-3 shows the trajectory of the Sphere from an experiment run with MPC. Fig. 6-4 shows the position time histories for the same experiment. The trajectory remains inside the error box at all times. Starting near the center of the box, the Sphere moves in the $-X$ direction, due to the table slope, and shows some damped oscillation in the Y-axis. Eventually it reaches the left edge of the error box and spends the last half of the experiment duration "bouncing" off that constraint limit. The oscillation in the Y-axis is probably a result of thrust error or underestimated delay. The constraints are satisfied, as expected. It would be interesting future work to incorporate disturbance models, such as the slope seen here, to attempt to improve the performance when such information was available.

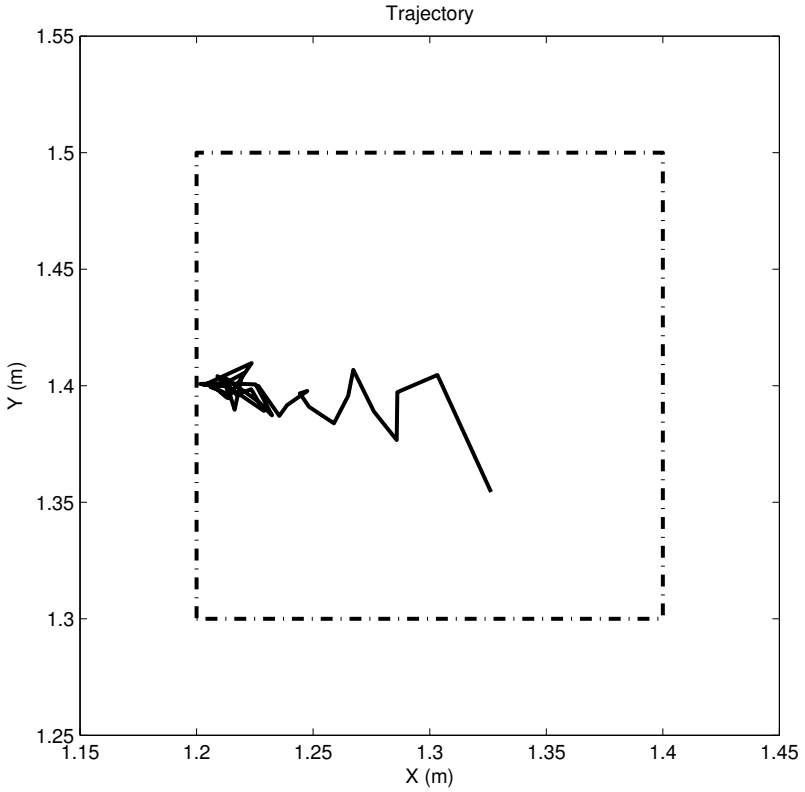
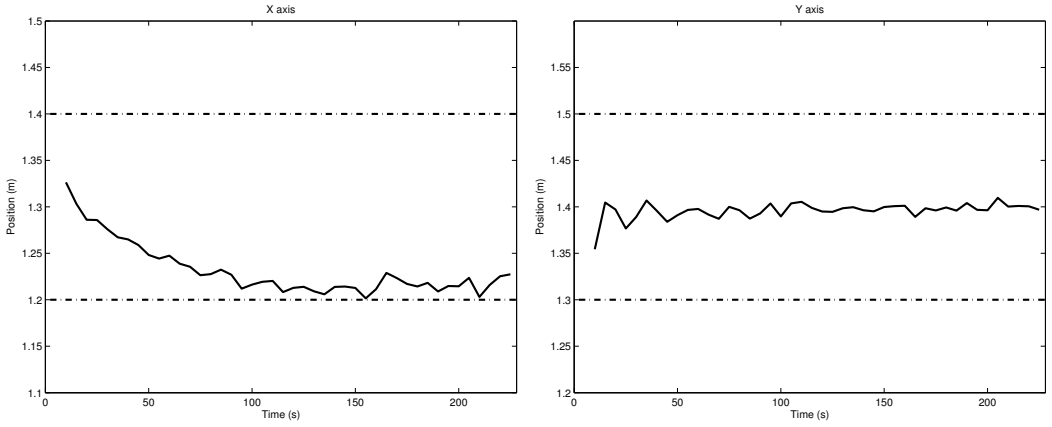
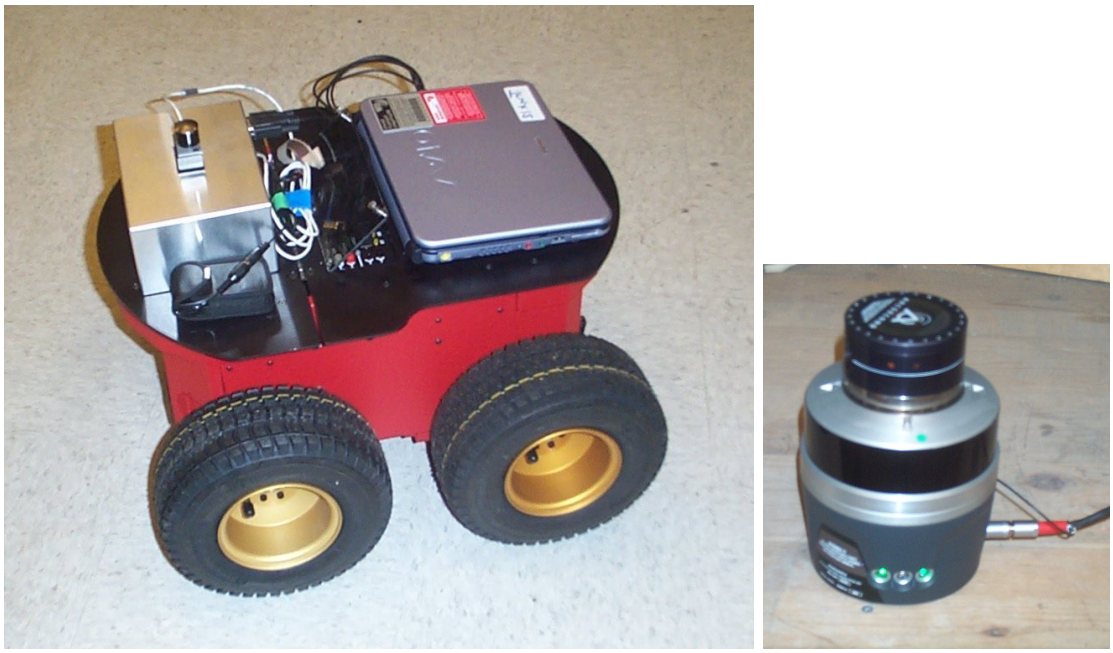


Figure 6-3: Trajectory Plot from Spheres Experiment. Error box limit shown by dashed line.



(a) X Position (b) Y Position

Figure 6-4: Position Time Histories from Spheres Experiment



(a) Rover (b) ArcSecond Transmitter

Figure 6-5: Rover Testbed

6.2 Rovers

Fig. 6-5(a) shows a vehicle from the multi-vehicle testbed at MIT's Aerospace Control Laboratory. The base vehicle is a Pioneer 3-AT from ActivMedia Robotics [60]. On the metal box at the front (toward the left in Fig. 6-5(a)) is the position sensor, part of the ArcSecond Constellation 3Di [61] system, along with laser transmitters, one of which is shown in Fig. 6-5(b). A Sony laptop completes the on-board equipment, connected to both sensor and rover and also to "ground" computers via an integrated wireless LAN adapter. A purpose-written application runs on the Sony laptop to provide low-level functions including estimation, communication handling and optional waypoint-following control. The ground computer runs Matlab, using TCP/IP sockets to communicate with the rover over wireless LAN.

This section describes demonstrations of robust MPC for a pair of rovers, one as a target and the other as a chaser, controlled by MPC to achieve either convergence to a target box or tracking. This could represent, for example, autonomous air-to-air

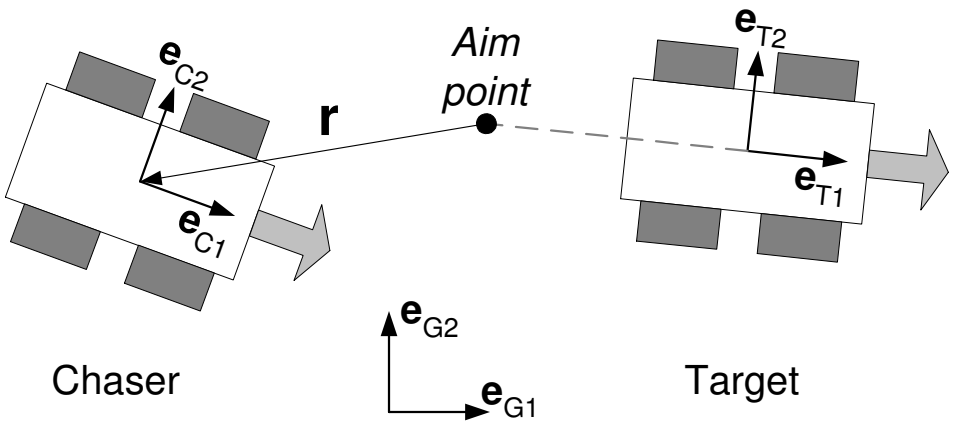


Figure 6-6: Reference frames and vectors for rover experiment. Vectors $\mathbf{e}_{C1}$ and $\mathbf{e}_{C2}$ are fixed to the chaser vehicle, $\mathbf{e}_{T1}$ and $\mathbf{e}_{T2}$ to the target vehicle. Vectors $\mathbf{e}_{G1}$ and $\mathbf{e}_{G2}$ represent the measurement reference frame of the Arcsecond system. The experiment controls the relative position vector $\mathbf{r}$.

refuelling of a UAV.

6.2.1 Experiment Set-Up

Fig. 6-6 shows a schematic of the rover experiments. MPC is used to control the position $\mathbf{r}$ of the chaser truck relative to the aim point. The aim point is fixed relative to the target vehicle. The target is commanded to drive straight at 0.2m/s. This is an open-loop command: no feedback control is applied to the target, and in some experiments an external disturbance was deliberately applied to the target to exercise the robustness of the chaser control. Under the assumption that the target moves with constant velocity, the dynamics of the vector $\mathbf{r}$ can be captured by a simple double integrator $\ddot{\mathbf{r}} = \mathbf{a}_T$ where $\mathbf{a}_T$ is the acceleration of the chaser. The simple aircraft dynamics model from the example in Section 2.7.2, using a point mass model with maximum speed limit and force (corresponding to a minimum turning radius), is used here to model the rover dynamics. The rover has the additional capability to stop and turn on the spot, but this facility is not exercised in this example, and

maneuvers feasible for the the aircraft model can be executed by the rover. The model employs a double integrator plant with a speed limit of 0.25m/s and acceleration limit of 0.18m/s^2. This corresponds to a minimum turning radius of 22cm at the nominal speed of 0.2m/s. Note that the speed limit applies to the velocity of the chaser relative to the target, not the absolute chaser motion. The system was discretized with a time step of 2s. Preliminary experiments, implementing a simple proportional feedback law and comparing with a software model, identified a delay of $D = 0.8$s in the loop between sending a steering command and observing its effect in received data, and a forward propagation of this length was applied to each state estimate according to Section 3.2.5. Further preliminary experiments identified prediction error levels of up to 0.1m in position and 0.06m/s in velocity. Sources of uncertainty in this experiment include the modeling assumptions and conversions involved, position measurement errors, disturbance forces (in some cases applied deliberately) and the effects of time delay.

The rover experiments involved two scenarios, one "approach" control, in which the chaser converges on the aim point from about 1m away, and one "tracking" control, in which the chaser must remain in a fixed error box around the aim point. Both controllers used robust MPC as presented in Section 2.3. The controllers have common system models and uncertainty sets, summarized below

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \\ 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathcal{W} = \left\{ \mathbf{w} \in \Re^4 \mid |w_1| \leq 0.1, |w_2| \leq 0.1, |w_3| \leq 0.06, |w_4| \leq 0.06 \right\}$$

The cost function and constraint differ between the two controllers, described individually in the following subsections.

Approach Control

The objective of the approach experiments is to get "close" to the aim point. The initial condition of the chaser is roughly a meter away from the aim point, otherwise chosen by the operator to provide variety in the experiments.

The MPC optimization $\mathsf{P}_{APPROACH}$ has no position constraint but a heavy weighting on position error in the cost function, which penalizes the one-norms of the states and controls. The velocity and force are constrained according to the dynamics limits described above.

$$\mathcal{Y} = \left\{ \mathbf{y} \in \Re^6 \mid ||(y_3, y_4)||_2 \leq 0.25, ||(y_5, y_6)||_2 \leq 0.18 \right\}$$

$$\ell(\mathbf{x}, \mathbf{u}) = 10|(x_1, x_2)| + |(x_3, x_4)| + 0.001|\mathbf{u}|$$

Tracking Control

The objective of the tracking experiments is to remain within a 50 cm by 50 cm box around the aim point. The chaser begins directly behind and with the same heading as the target, roughly at the center of the error box. Again, the initial conditions are set by the operator only roughly, to provide variety in the experiments. To exercise the robustness of the control further, a perturbation was applied to the target motion in each run, implemented by simply nudging the target to change its heading by a few degrees. The chaser control was not aware of this disturbance.

The MPC optimization P_{TRACK} minimizes acceleration subject to the error box constraint.

$$\mathcal{Y} = \left\{ \mathbf{y} \in \Re^6 \mid ||(y_1, y_2)||_\infty \leq 0.25, ||(y_3, y_4)||_2 \leq 0.25, ||(y_5, y_6)||_2 \leq 0.18 \right\}$$

$$\ell(\mathbf{x}, \mathbf{u}) = 0.1|(x_1, x_2)| + 0.01|(x_3, x_4)| + 100|\mathbf{u}|$$

Algorithm

The MPC algorithm for these experiments is based on Algorithm 3.2, incorporating the prediction to account for delay, with the addition of the conversions between reference frames and from acceleration commands to the speed and turn-rate commands.

Algorithm 6.1. (Rover Control)

1. Command both trucks to drive straight forward at nominal speed
2. Measure positions and velocities of chaser $(\mathbf{r}_C, \mathbf{v}_C)$ and target $(\mathbf{r}_T, \mathbf{v}_T)$ in global frame $\mathbf{e}_{G1}, \mathbf{e}_{G2}$
3. Find relative chaser state $\mathbf{x}(t)$ in target frame

$$\mathbf{x}(t) = \begin{bmatrix} (\mathbf{r}_C(t) - \mathbf{r}_T(t)) \cdot \mathbf{e}_{T1} - d \\ (\mathbf{r}_C(t) - \mathbf{r}_T(t)) \cdot \mathbf{e}_{T2} \\ (\mathbf{v}_C(t) - \mathbf{v}_T(t)) \cdot \mathbf{e}_{T1} \\ (\mathbf{v}_C(t) - \mathbf{v}_T(t)) \cdot \mathbf{e}_{T2} \end{bmatrix}$$

4. Propagate estimate ahead by the delay time D to form prediction $\hat{\mathbf{x}}(t + D|t)$
5. Solve MPC optimization $\mathsf{P}_{APPROACH}(\hat{\mathbf{x}}(t + D|t))$ or $\mathsf{P}_{TRACK}(\hat{\mathbf{x}}(t + D|t))$, using point mass model for relative dynamics of chaser, to generate acceleration command $\mathbf{u}$ for chaser, expressed in the target frame
6. Convert acceleration from target frame to chaser frame and separate components to form speed and turn rate commands

$$\begin{aligned} \text{speed command}: \quad V_{cmd} &= \|\mathbf{v}_C\| + \tfrac{1}{2}\mathbf{e}_{C1} \cdot (u_1\mathbf{e}_{T1} + u_2\mathbf{e}_{T2}) \\ \text{turn rate command}: \quad \Omega_{cmd} &= \frac{\mathbf{e}_{C2} \cdot (u_1\mathbf{e}_{T1} + u_2\mathbf{e}_{T2})}{V_{cmd}} \end{aligned}$$

7. Send commands to chaser
8. Store data. Wait for one time step. Go to 2

Algorithm 6.1 is employed in two different experiments, distinguished by the MPC problem solved in step 5.

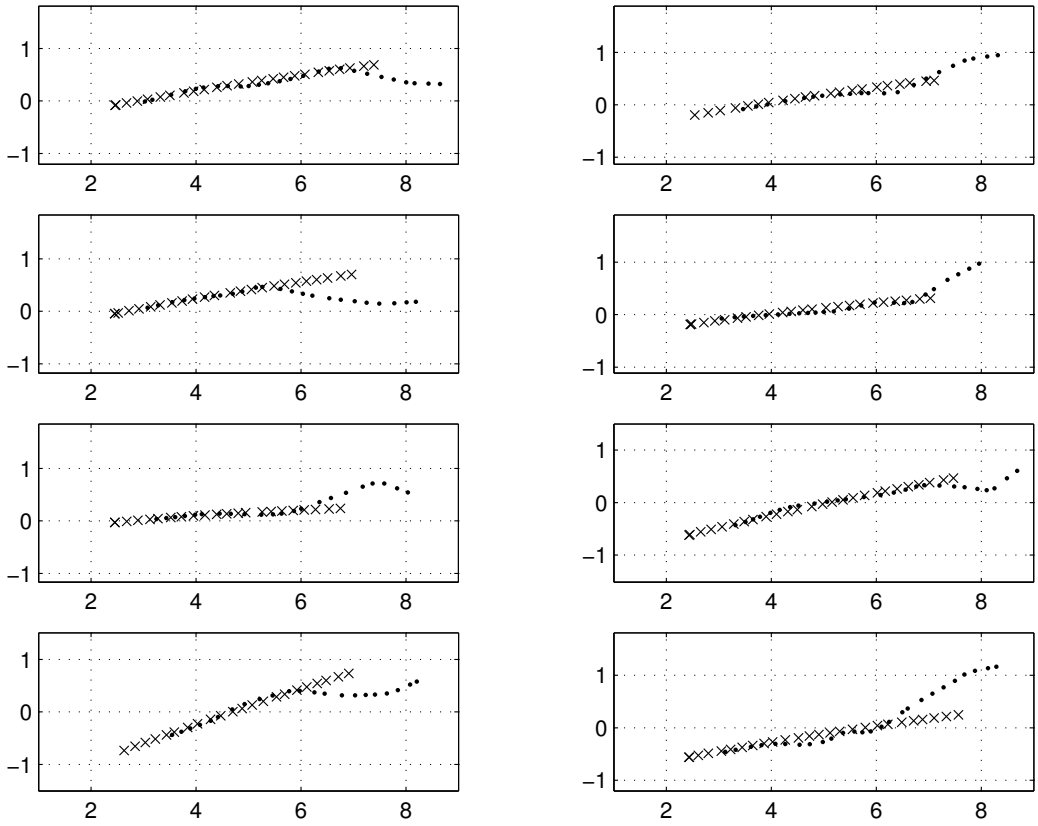


Figure 6-7: Absolute Trajectories for Eight Runs of Approach Control. Chaser marked by '•', target by '×'. Both rovers start at the right and move left. The chaser beings from various initial offsets and headings but always converges to follow the target, hence the dots end up overlapping the crosses as the two rovers follow the same path.

6.2.2 Results

Approach Control

Fig. 6-7 shows absolute trajectories of both rovers, *i.e.* in the global reference frame, for eight runs of the approach control experiment. It can be seen that the chaser starts in various positions and orientations but always converges to follow the target. Fig. 6-8 shows the time histories of the relative position $\mathbf{r}$, expressed in the target frame, for the same eight runs.

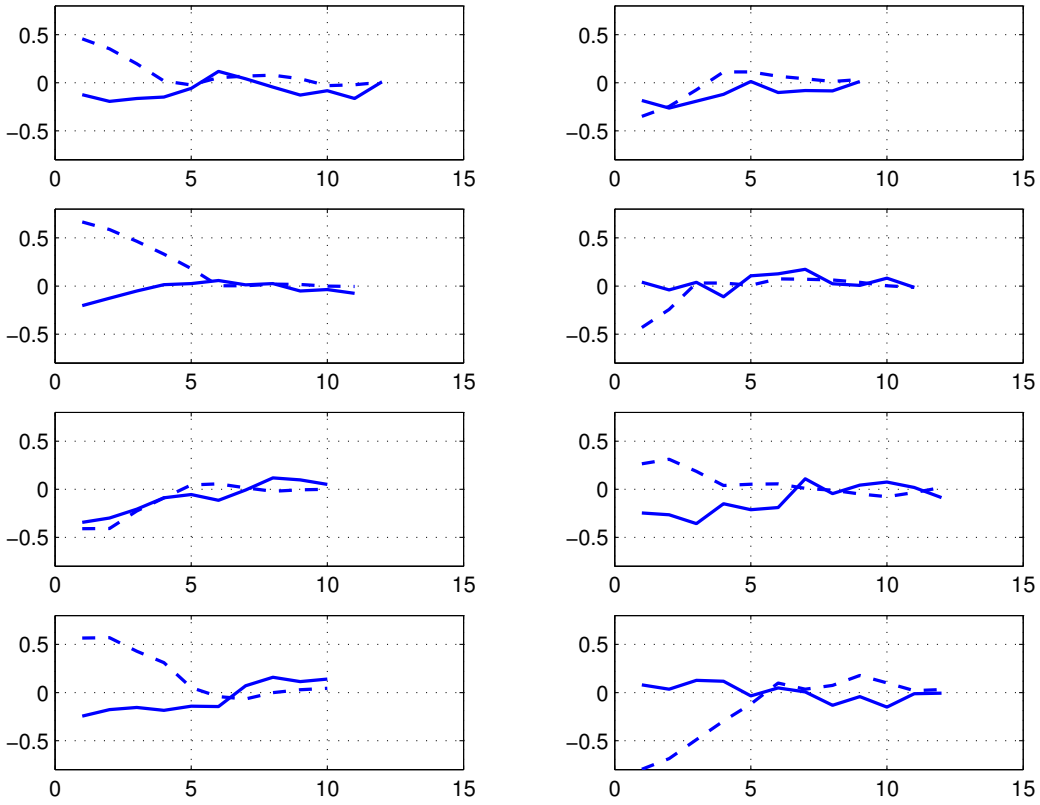


Figure 6-8: Time Histories of Relative Position Vector **r** for Eight Runs of Approach Control. (Solid: X, Dashed: Y)

Tracking Control

Fig. 6-9 shows the absolute trajectories of both rovers for 15 runs of the tracking control. In each case, the chaser follows the target closely. The turns made by the target when perturbed by the operator can be seen in these plots, and it is clear that the chaser turns to follow the target in each case. Fig. 6-10 shows time histories of the components of the relative position **r** for the same 15 runs. The vertical range of the plots in Fig. 6-10 ± 0.25 m, the same as the error box size, so it is clear that the vehicle remains inside the ± 0.25 m error box constraint.

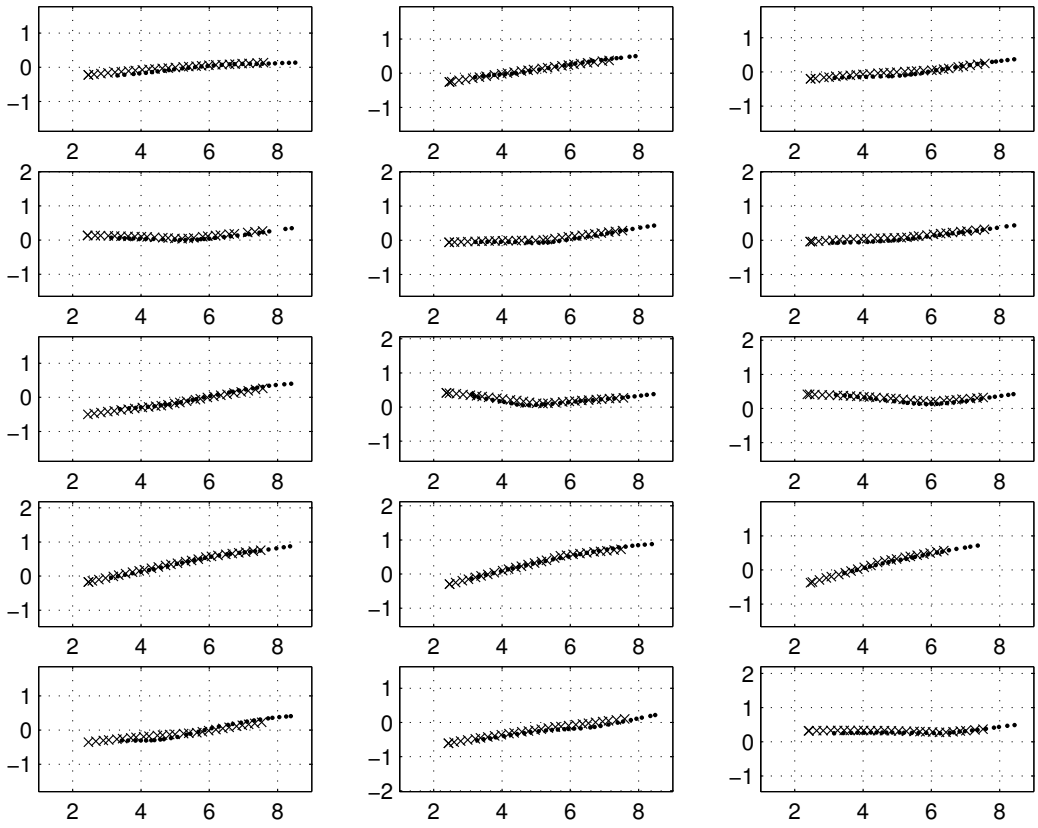


Figure 6-9: Absolute Trajectories for 15 Runs of Tracking Control. Chaser marked by '•', target by '×'. Both rovers start at the right and move left. Observe that the target changes direction, due to the impulse applied manually by the operator, and the chaser turns to follow.

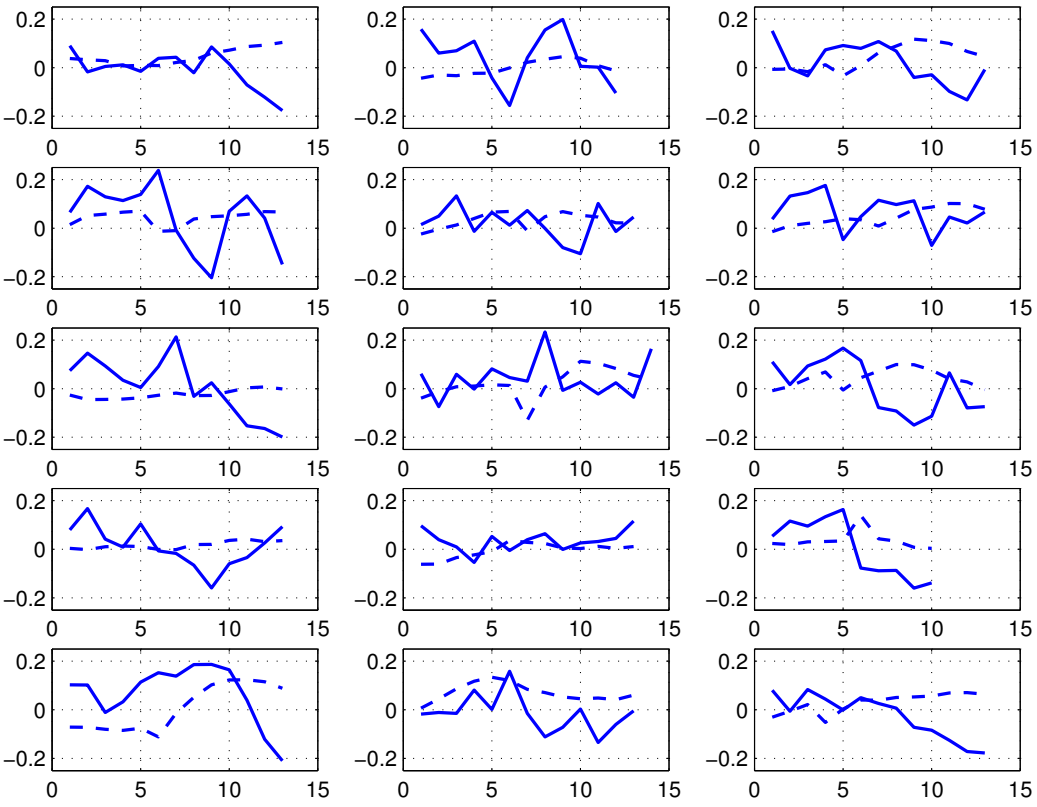


Figure 6-10: Time Histories of Relative Position Vector **r** for 15 Runs of Tracking Control. (Solid: X, Dashed: Y) The error box is ±0.25m, the same as the vertical range of the plots, so it is clear that the chaser remained in the error box at all times.

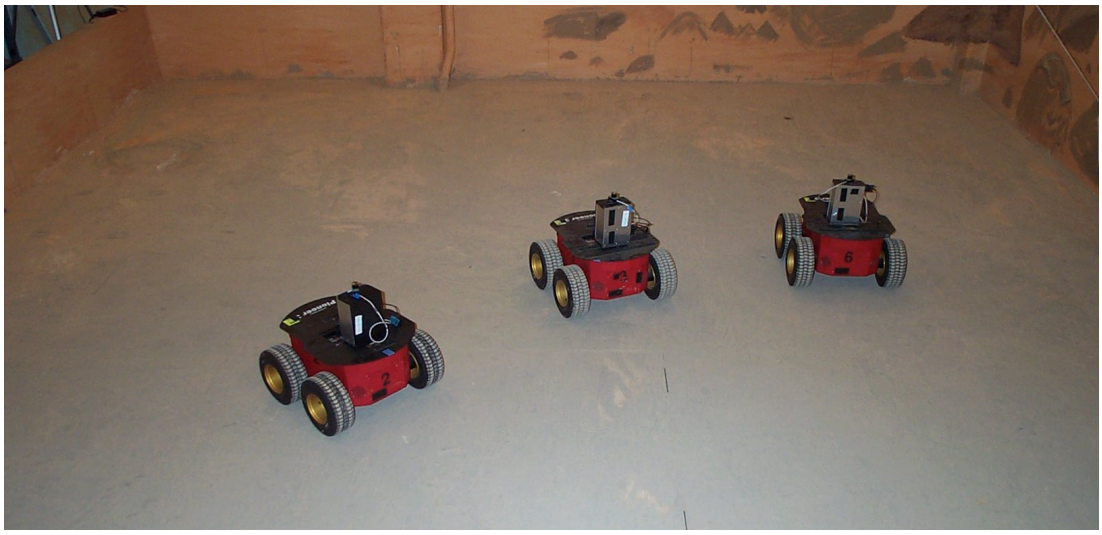


Figure 6-11: Three Rovers

6.3 Multi-Rover Collsion Avoidance using DMPC

This section describes demonstrations of the Decentralized Model Predictive Control (DMPC) algorithm developed in Chapter 4 for the guidance of the three rovers shown in Fig. 6-11. The rover testbed was described in detail in Section 6.2. In this DMPC demonstration, each rover is required to travel to a prespecified target while avoiding collisions with the others. This example, investigated in simulation in Section 4.6, exercises the cooperative capabilities of the DMPC algorithm, as the avoidance constraints couple the behaviors of the vehicle. The hardware demonstration extends the simulation by introducing realistic uncertainty and latency due to computation and communication.

6.3.1 Experiment Set-Up

The DMPC controller uses the same optimization problems P_C and P_p developed in Section 4.6.2, combined with the accommodation for delay described in Section 4.8 and the rover steering conversions presented in Section 6.2. For this demonstration, DMPC was implemented on a single computer, which is still consistent with the sequential solution procedure used in DMPC but simplifies the communication logistics.

The targets for each rover are prespecified in the controller, but the initial positions are taken from sensor measurements, so the scenario is varied by starting the rovers in different positions using the same software. The following algorithm includes the implementation details.

Algorithm 6.2. (Decentralized MPC for Trucks)

1. Rovers are stationary, positioned manually by operator. Gather states of all rovers. Replace velocities with default initial values based on known alignments.

2. Find a solution to the initial centralized problem P_C using CPLEX with a 1 minute computation time limit. If solution cannot be found, stop (problem is infeasible).

3. Set $k = 0$

4. For each rover p, convert each force command $\mathbf{u}_p^*(k|k)$ to equivalent speed and turn rate commands and transmit

5. Increment k. Store current time in t_{last}.

6. For each rover p in order $1, \ldots, P$:

   (a) Get state estimate of rover p and propagate forward to account for delay.

   (b) Compile the position plan data $\tilde{\mathbf{r}}_{pq}(k \ldots k+T|k)$ $\forall q \neq p$ from other subsystems

   (c) Propagate plans for other rovers $q \neq p$ to suit delays, finding the planned positions of each rover at the waypoint times of rover p.

   (d) Solve sub-problem $\mathsf{P}_p\left(\mathbf{x}_p(k), \tilde{\mathbf{r}}_{pq}(k \ldots k+T|k)\right)$

7. Wait until one time step length has elapsed since time t_{last}. Go to 4

The velocity assumptions in Step 1 are made to ease implementation. The problem assumes that the rovers are moving at the start, but to actually implement this would

require more operating space. Therefore the vehicles begin stationary and the initial speed command is adjusted to make up the difference in position.

Step 6(c) accounts for computation and communication delays as discussed in Section 4.8. All vehicles use the same discrete time step length, but due to the non-zero computation time, there are offsets between the sampling times. The plan propagations in Step 6(c) ensure that all data in each subproblem uses consistent sampling times.

For this demonstration, a maximum horizon of $\bar{N} = 12$ time steps each of 6s was used. The constraints specified a maximum speed of 0.2m/s and turning rate of 8deg/s. The rovers were required to remain more than 0.5m apart at all times. Preliminary experiments indicated a delay of 0.8s for each rover and prediction errors of $\pm$20cm in position and $\pm$4cm/s in velocity, and the constraints were tightened accordingly.

6.3.2 Results

Fig. 6-12 shows the trajectories from 12 initial experiments using only two rovers. Observe that the trajectories vary with the initial positions of the rovers. For example, in Fig. 6-12(c), rover 2 diverts to pass behind rover 1, whereas in Fig. 6-12(k), rover 1 passes behind rover 2.

Fig. 6-13 shows trajectories from a further 12 experiments using all three rovers. In the results in Figs. 6-13(a)–6-13(j), the target boxes were all at the bottom left of the figure. Again, the trajectories vary with the initial positions, and the vehicles deviate from their direct routes to avoid collisions. Figs. 6-13(k)–6-13(l) show a different scenario, in which rovers 1 and 3 must swap positions and rover 2 crosses both their paths. In these cases, all the rovers take indirect paths to avoid collisions.

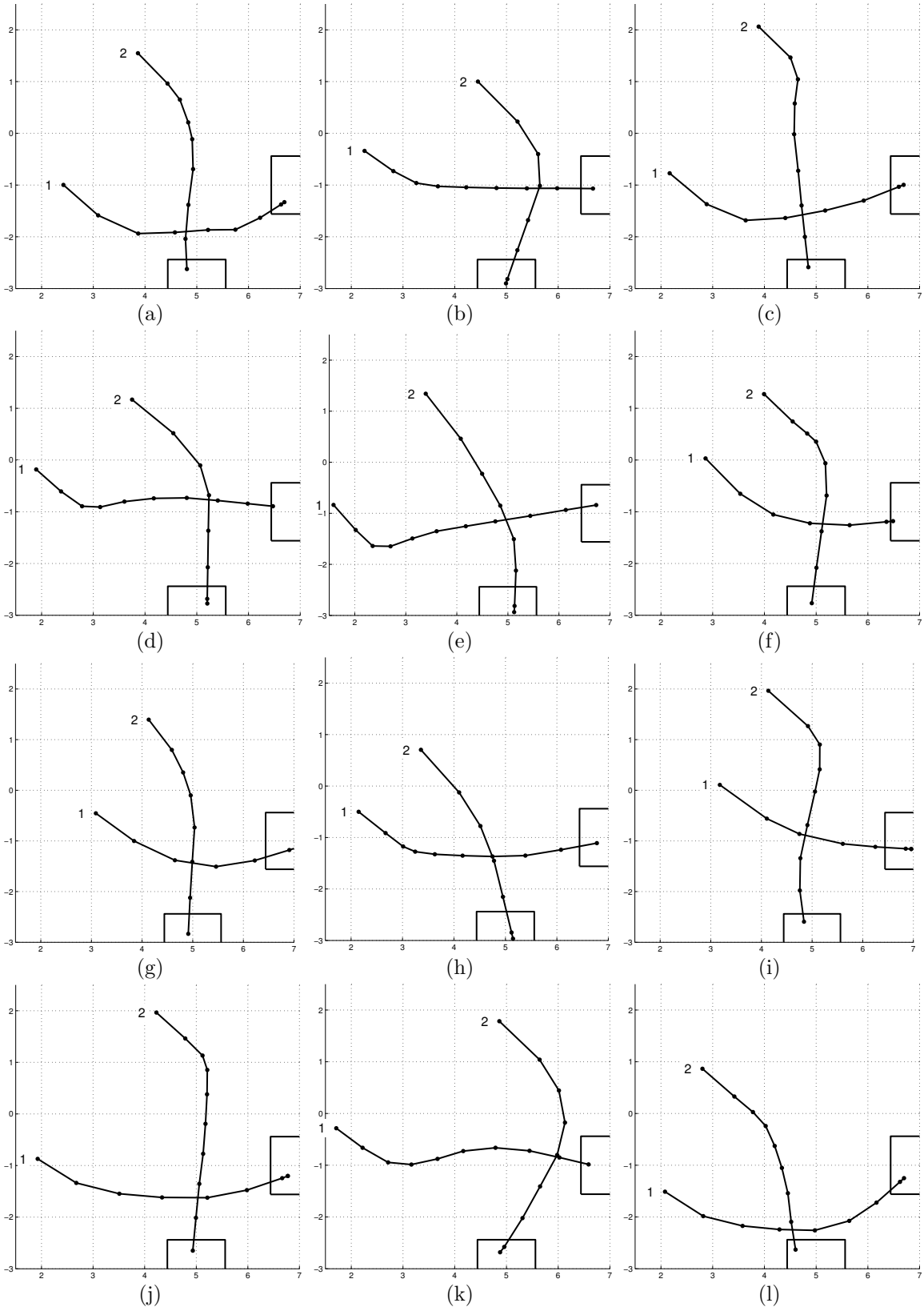


Figure 6-12: DMPC Results for Two Rovers. Numbers mark the starting points of each rover and the target regions are shown by boxes.

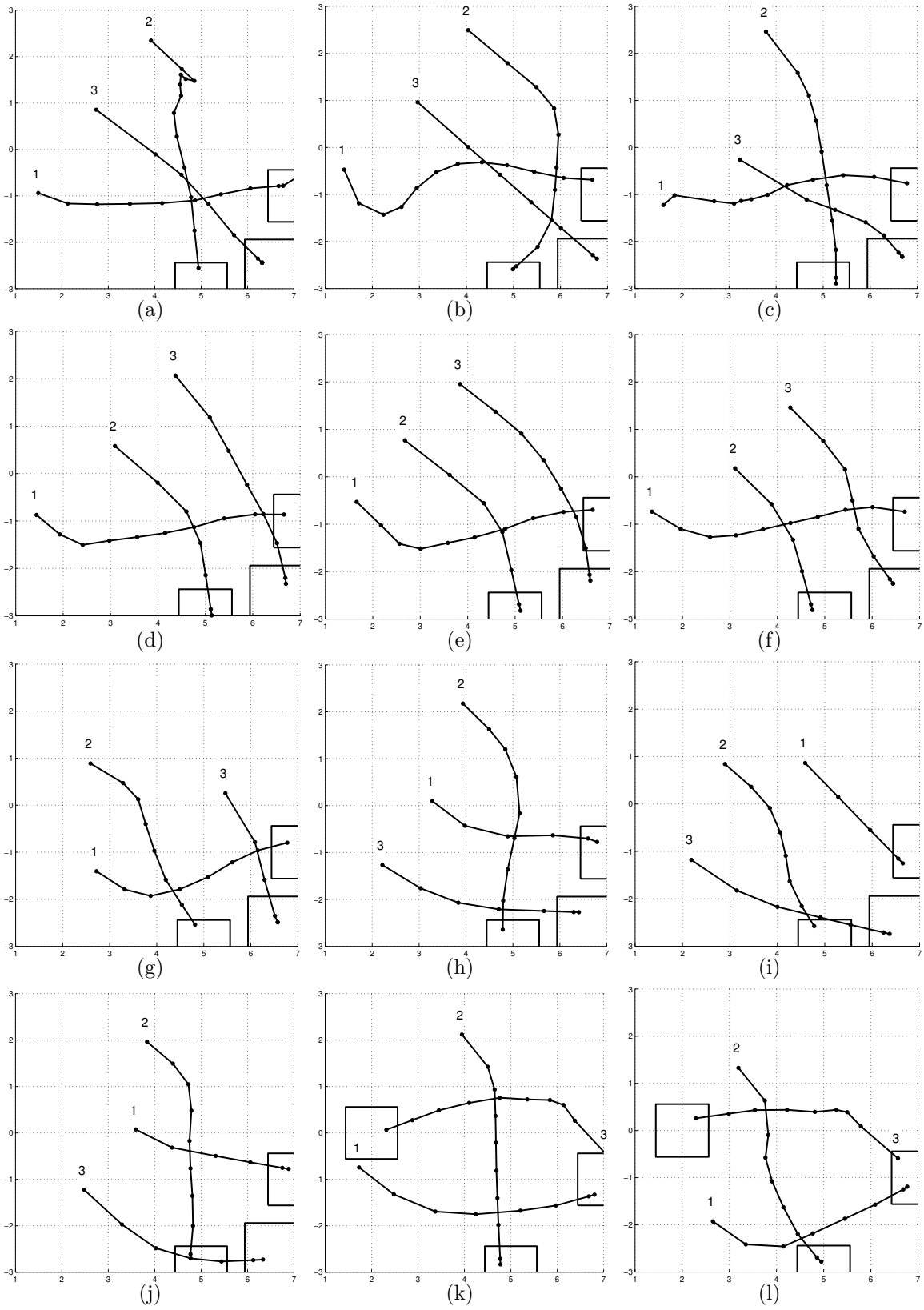


Figure 6-13: DMPC Results for Three Rovers. Numbers mark the starting points of each rover and the target regions are shown by boxes.

Chapter 7

Conclusion

7.1 Contributions

This thesis has made contributions in the area of robust MPC, with particular attention to problems involving uncertainty and multiple subsystems.

7.1.1 Robust MPC

The robust MPC formulations in Chapter 2 extend the constraint tightening approach [14] by (a) making the candidate control policy, used to perform the constraint tightening, time-varying and (b) employing either (i) a maximal robust control invariant terminal set or (ii) a variable horizon. These extensions include additional degrees of freedom in the controller design and lead to enlarged sets of feasible initial conditions. By plotting the feasible sets for some examples, robust feasibility has been verified, and we have shown that the controller has a good feasible region when compared to the theoretical maximum.

7.1.2 Output Feedback MPC

Section 3.2 developed a form of output feedback MPC combining a fixed-horizon estimator and the robust MPC from Chapter 2, with modified constraint tightening to account for the additional uncertainty in state information. In particular, it was

shown that overlooking correlation in the estimator can lead to conservatism in the control. Delay due to measurement, computation and actuation was also handled in the same framework.

7.1.3 Adaptive MPC

Section 3.3 developed an adaptive algorithm to estimate the uncertainty set from online measurements. The constraint tightening in robust MPC is then adjusted online as knowledge of the uncertainty level improves. This relaxes the requirement to know bounds on sources of uncertainty, such as disturbance and sensing noise, and enables performance improvements when such bounds are only known conservatively *a priori*.

7.1.4 Decentralized MPC

Chapter 4 developed a Decentralized MPC (DMPC) algorithm, using the constraint tightening principles from Chapter 2. The algorithm controls multiple subsystems and guarantees robust satisfaction of coupled constraints despite the action of unknown but bounded disturbances on each subsystem. The centralized planning problem is divided into smaller subproblems, each planning for the future actions of a single subsystem. Each subproblem is solved only once per time step, without iteration, and is guaranteed to be feasible. DMPC has been demonstrated for control of a team of cooperating UAVs subject to collision avoidance constraints. Extensive simulations show that the DMPC offers a significant reduction in computation time, compared to the corresponding centralized MPC, at the expense of only a small increase in flight time, which was the cost function in the controllers. The DMPC algorithm can account for computation and communication delays using a similar approach to that developed in Section 3.2.5 for centralized MPC.

7.1.5 Analytical performance prediction

Chapter 5 presented an approximate method to predict the performance, in terms of an RMS cost metric, for the robust MPC from Chapter 2. This enables rapid

trade studies of the effects of control parameters, such as cost weightings, constraint levels and prediction horizon length. The prediction algorithm accommodates possible mismatch between actual and predicted disturbance levels and between optimization cost and performance analysis metric.

7.1.6 Hardware Implementation

The algorithms developed in this thesis have been demonstrated in hardware for vehicle control. The robust MPC from Chapter 2, in conjunction with the estimation error and delay compensation from Chapter 3, have been implemented on both a ground rover testbed and a spacecraft simulator on air bearings. The DMPC from Chapter 4 has been implemented on multiple ground rovers for collision avoidance maneuvering.

7.2 Future Work

The core technologies developed in this thesis - constraint tightening for robustness and decentralized MPC - can be extended in various ways.

7.2.1 Continuous Time Formulations

The current formulation is applicable to discrete-time systems and involves replanning at regular intervals. It would be interesting to investigate a scheme in which replanning could occur at any time, subject to some limit. This would involve modelling the system in continuous time. The system would then compare its actual evolution to the predictions and plan only when deemed necessary.

7.2.2 Spontaneous Decentralized MPC

The combination of DMPC with a continuous time scheme outlined above would have some interesting properties. The current DMPC algorithm relies not only on periodic replanning but also on a predefined planning order, although that order can

be changed by concensus at particular instants. It would be interesting to consider a scheme in which any subsystem could replan at any time, and derive rules for such a scheme guaranteeing robust feasibility.

7.2.3 DMPC with Imperfect Communication

The D¡PC formulation developed in this thesis assumes perfect communication between all subsystems. An extension of this work would be to identify ways in which guarantees can be maintained if communicated information is corrupted, absent (*e.g.* due to lost communication link) or incomplete (*e.g.* if bandwidth requirements force some approximation of communicated data).

7.2.4 Including Better Disturbance Models

The model of an unknown-but-bounded disturbance model can often be applied, but in practice may ignore some additional information about the problem. For example, the Spheres hardware experiment in Section 6.1 was dominated by a constant disturbance force. It would be interesting to find ways to incorporate this and other information, such as frequency domain knowledge, into the robustness modifications.

7.2.5 Time varying Systems

Some model of spacecraft relative dynamics [65] lead to time-varying linear systems, in particular periodic time varying systems. Some approaches to MPC handle systems like this by treating the time-variation as a robustness issue. It would be interesting to pursue a less conservative technique that explicitly incorporated the time-varying model. Periodic time variation might be exploited - see the results concerning invariant sets based on periodic solutions in Proposition 2.1.

7.2.6 LPV/multi-model Uncertainty

The work in this thesis has considered affine disturances and state errors. Many other forms of robust MPC consider cases with model variation $\mathbf{A} \in \mathcal{A}$ but none of these

methods adopt the constraint tightening approach. It would be interesting to see if the constraint tightening could be coupled with multimodel uncertainty and what the features of such a scheme would be. A simple way of doing this would be to convert the mutlimodel uncertainty to an equivalent disturbance.

$$\mathbf{x}(k+1) = (\mathbf{A} + \Delta\mathbf{A})\mathbf{x}(k) = \mathbf{A}\mathbf{x}(k) + \mathbf{w}(k)$$

where bounds on $\Delta\mathbf{A}$ and $\mathbf{x}$ imply bounds on $\mathbf{w}$. The author suspects this would be conservative and better methods could be found by considering the form of uncertainty directly in the constraint tightening.

7.2.7 DMPC with Coupled Dynamics

The general form of the DMPC algorithm could be modified to include coupled dynamics as well as coupled constraints. The concepts of using nominal information for other subsystems in each subproblem would be unchanged. The key feature would be the determination of new constraint margins to account for the additional uncertainty effects. Weakly coupled systems could be handled as disturbances, but this might become conservative at stronger coupling levels.

7.2.8 Extensions to Performance Prediction

The performance prediction algorithm in Chapter 5 included several assumtpions needed to apply LQR tools. It might be possible to relax some of those restrictions if other cases could be identified in which the optimal control law can be found in closed form. Alternatively, parametric programming ideas could be applied to this problem [59]. Also, including disturbance filters and discount factors [64] would enhance this tool.

7.2.9 Other Applications

The tools and methods developed in this thesis have been motivated by aerospace applications and demonstrated using examples from this field, such as cooperative UAV guidance. However, the theoretical results could have applications in other fields. In particular, the DMPC algorithm could be applicable to problems in process control, such as temperature control in large plants or buildings, or general resource allocation problems in operations research.

Bibliography

[1] Predator Unmanned Aerial Vehicle project at Airforce Technology, `http://www.airforce-technology.com/projects/predator/`, visited November 2004.

[2] Terrestrial Planet Finder Mission Homepage, NASA, `http://planetquest.jpl.nasa.gov/TPF/tpf_index.html`, visited November 2004.

[3] J.M. Maciejowski, *Predictive Control with Constraints*, Prentice Hall, England, 2002.

[4] J. A. Rossiter, "Model-based Predictive Control – A Practical Approach," CRC Press, Florida, 2003.

[5] B. Mouvaritakis and M. Cannon, "Nonlinear Predictive Control – Theory and Practice," IEE, London, UK, 2001.

[6] H. Sunan, T. K. Kiong and L. T. Heng, "Applied Predictive Control," Springer, London, UK, 2002.

[7] V. Manikonda, P. O. Arambel, M. Gopinathan, R. K. Mehra and F. Y. Hadaegh, "A Model Predictive Control-based Approach for Spacecraft Formation Keeping and Attitude Control," *ACC,* San Diego CA, June 1999.

[8] G Inalhan, J. P. How and M. Tillerson, "Co-ordination and control of distributed spacecraft systems using convex optimization techniques," International Journal of Robust and Nonlinear Control, Vol.12, , John Wiley & Sons, 2002, p.207-242.

[9] W. B. Dunbar and R. M. Murray, "Model predictive control of coordinated multi-vehicle formations" IEEE Conference on Decision and Control, 2002.

[10] A. G. Richards and J. P. How, "Model Predictive Control of Vehicles Maneuvers with Guaranteed Completion Time and Robust Feasibility," American Control Conference, Denver CO, AACC, 2003.

[11] D. Q. Mayne, J. B. Rawlings, C. V. Rao, P. O. M. Scokaert, "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, 36(2000), Pergamon Press, UK, pp. 789–814.

[12] P. O. M. Scokaert and D. Q. Mayne, "Min-Max Feedback Model Predictive Control for Constrained Linear Systems," *IEEE Trans. on Automatic Control*, Vol. 43, No. 8, Aug. 1998, p 1136.

[13] J R Gossner, B Kouvaritakis and J A Rossiter, "Stable Generalized Predictive Control with Constraints and Bounded Disturbances," Automatica, Vol 33 No 4, Pergamon Press, UK, 1997, p.551.

[14] L. Chisci, J. A. Rossiter and G. Zappa, "Systems with Persistent Disturbances: Predictive Control with Restricted Constraints," Automatica, Vol. 37 No. 7, Pergamon Press, UK, 2001, pp. 1019-1028.

[15] S. S. Keerthi and E. G. Gilbert, "Optimal Infinite-Horizon Feedback Laws for a General Class of Constrained Discrete-Time Systems: Stability of Moving Horizon Approximations," Journal of Optimization Theory and Applications, Vol. 57 No. 2 May 1988, , Plenum Publishing Corp, 1988, p.265-293.

[16] P. O. M. Scokaert, D. Q. Mayne and J. B. Rawlings, "Suboptimal Model Predictive Control (Feasibility Implies Stability)," *IEEE Trans. on Automatic Control*, Vol. 44 No. 3, 1999, p. 648.

[17] E. C. Kerrigan and D. Q. Mayne, "Optimal Control of Constrained, Piecewise Affine Systems with Bounded Disturbances," IEEE Conference on Decision and Control, 2002.

[18] A Bemporad and M Morari, "Robust Model Predictive Control: A Survey," In Robustness in Identification and Control, A Garulli, A Tesi and A Vechino (Eds.), Springer Verlag, 1999, p.207-226.

[19] J A Primbs and V Nevistic, "A Framework for Robustness Analysis of Constrained Finite Receding Horizon Control," Transactions on Automatic Control, Vol 45 No 10, October 2000, IEEE, 2000, p.1828.

[20] E. C. Kerrigan and J. M. Maciejowski, "Robust Feasibility in Model Predictive Control: Necessary and Sufficient Conditions," *40th IEEE CDC*, Orlando FL, December 2001, pp. 728–733.

[21] L Chisci, P Falugi and G Zappa, "Predictive Control for Constrained Systems with Polytopic Uncertainty," American Control Conference, Arlington VA, AACC, 2001, p.3073.

[22] Y I Lee and B Kouvaritakis, "A linear Programming Approach to Constrained Robust Predictive Control," American Control Conference, Chicago IL, Jun-00, AACC, 2000, p.2814.

[23] F A Cuzzola, J C Geromel and M Morari, "An Improved Approach for Constrained Robust Model Predictive Control," Automatica, Vol 38 (2002), Pergamon Press, 2002, p.1183.

[24] J H Lee and Z Yu, "Worst-case Formulations of Model Predictive Control for Systems with Bounded Parameters," Automatica, Vol 33 No 5, Elsevier Science Ltd, 1997, p.763-781.

[25] M V Kothare, V Balakrishnan and M Morari, "Robust Constrained Model Predictive Control using Linear Matrix Inequalities," Automatica, Vol 32 No 10, Elsevier Science Ltd, UK, 1996, p.1361-1379.

[26] Y I Lee and B Kouvaritakis, "Constrained Receding Horizon Predictive Control for Systems with Disturbances," International Journal of Control, Vol 72 No 11, Taylor & Francis, 1999, p.1027-1032.

[27] A Bemporad, "Reducing Conservativeness in Predictive Control of Constrained Systems with Disturbances," Conference on Decision and Control, IEEE, 1998.

[28] I Kolmanovsky and E. G. Gilbert, "Maximal Output Admissible Sets for Discrete-Time Systems with Disturbance Inputs," American Control Conference, Seattle WA, , AACC, 1995, p.1995.

[29] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Belmont MA, 1997.

[30] E. Kerrigan, Invariant Set Toolbox for Matlab, available at `http://www-control.eng.cam.ac.uk/eck21`, July 2003.

[31] E. C. Kerrigan, "Robust Constraint Satisfaction: Invariant Sets and Predictive Control," PhD Thesis, Cambridge University, November 2000.

[32] F Fahroo and I M Ross, "Trajectory Optimization by Indirect Spectral Collocation Methods," Astrodynamics Specialist Conference, AIAA/AAS, AIAA-2000-4028, Denver CO, 123, 2000.

[33] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[34] R. Sedwick, D. Miller, E. Kong, "Mitigation of Differential Perturbations in Clusters of Formation Flying Satellites," *Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, Breckenridge, CO, Feb. 7-10, 1999. Pt. 1 (A99-39751 10-12), San Diego, CA, Univelt, Inc. (Advances in the Astronautical Sciences. Vol. 102, pt.1), 1999, p. 323-342.

[35] M. H. Kaplan, *Modern Spacecraft Dynamics and Control,* Wiley, New York NY, 1976 pp 108–115.

[36] A. G. Richards, T. Schouwenaars, J. How, E. Feron, "Spacecraft Trajectory Planning With Collision and Plume Avoidance Using Mixed-Integer Linear Programming," *AIAA JGCD*, Vol. 25 No. 4, July 2002.

[37] A. G. Richards, "Trajectory Control Using Mixed Integer Linear Programming," S.M. Thesis, MIT, 2002.

[38] A. G. Richards, J. How, "Aircraft Trajectory Planning with Collision Avoidance using Mixed Integer Linear Programming," American Control Conference, Anchorage AK, May 2002.

[39] D. P. Bertsekas and I. B. Rhodes, "On the Minmax Reachability of Target Sets and Target Tubes," *Automatica*, Vol 7, Pergamon Press, UK, 1971, pp. 233-247.

[40] R. Franz, M. Milam, and J. Hauser, " Applied Receding Horizon Control of the Caltech Ducted Fan," ACC 2002.

[41] Y. I. Lee and B. Kouvaritakis, "Receding Horizon Output Feedback Control for Linear Systems with Input Saturation" 39^{th} IEEE *Conference on Decision and Control* Sydney, Australia, 2000, p 656.

[42] R. Findeisen, L. Imsland, F. Allgower and B. A. Foss, "Output Feedback Stabilization of Constrained Systems with Nonlinear Predictive Control," *International Journal of Robust and Nonlinear Control* Vol. 13, John Wiley & Sons, 2003, pp. 211-227.

[43] L. Magni, G. De Nicolao and R. Scattolini, "Output Feedback Receding-Horizon Control of Discrete-Time Nonlinear Systems," IFAC NOLCOS 98.

[44] H. Michalska and D. Q. Mayne, "Moving Horizon Observers and Observer-Based Control," *IEEE Transactions on Automatic Control*, Vol. 40 No. 6, IEEE, June 1995, p. 995.

[45] K. H. Lee, J. H. Lee and W. H. Kwon, "A Stabilizing Low-Order Output Feedback Receding Horizon Control for Linear Discrete Time-Invariant Systems," *American Control Conference*, Anchorage AK, 2002, p. 2412.

[46] R. K. Mehra, "On the Identification of Variances and Adaptive Kalman Filtering," *IEEE Transactions of Automatic Control* Vol. 15 No. 2, IEEE, April 1970, p. 175.

[47] F. D. Busse, "Precise Formation-State Estimation in Low Earth Orbit Using Carrier Differential GPS," PhD Thesis, Massachusetts Institute of Technology, March 2003.

[48] A. Bemporad and M. Morari, "Control of Systems Integrating Logic, Dynamics, and Constraints," in *Automatica,* Pergamon / Elsevier Science, New York NY, Vol. 35, 1999, pp. 407–427.

[49] J. S. Bellingham, A. G. Richards and J. P. How, "Receding Horizon Control of Autonomous Aerial Vehicles," American Control Conference, Anchorage AK, AACC, 2002, pp. 3741-3746.

[50] A. G. Richards, J. S. Bellingham, M. Tillerson and J. P. How, "Co-ordination and Control of Multiple UAVs", AIAA paper no. 2002-4588, *AIAA Guidance, Navigation, and Control Conference and Exhibit,* 2002.

[51] D. Jia and B. Krogh, "Min-Max Feedback Model Predictive Control For Distributed Control with Communication," American Control Conference, Anchorage AK, AACC, 2002, pp.4507-45.

[52] Camponogara, E., Jia, D., Krogh, B.H. and Talukdar, S., "Distributed model predictive control", IEEE Control Systems Magazine, 2002.

[53] D. H. Shim, H. J. Kim and S. Sastry, "Decentralized Nonlinear Model Predictive Control of Multiple Flying Robots," *42nd IEEE CDC*, Maui, December 2003, p 3621.

[54] S. L. Waslander, G. Inalhan and C. J. Tomlin, "Decentralized Optimization via Nash Bargaining," Conference on Cooperative Control and Optimization, 2003.

[55] T. Keviczky, F. Borrelli and G. J. Balas, "Model Predictive Control for Decoupled Systems: A Study on Decentralized Schemes," *IEEE ACC*, Boston MA, 2004.

[56] T. Schouwenaars, B. DeMoor, E. Feron and J. How, "Mixed Integer Programming for Multi-Vehicle Path Planning," *European Control Conference,* Porto, Portugal, September 2001, pp. 2603-2608.

[57] L E Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," American Journal of Mathematics, Vol 79 No 3, 1957, , 497-516, 1957.

[58] SPHERES Web Site, MIT Space Systems Laboratory, `http://ssl.mit.edu/spheres/index.html`, visited August 2004.

[59] A Bemporad, F Borrelli and M Morari, "Model Predictive Control Based on Linear Programming - The Explicit Solution," Transacations on Autonatic Control, Vol 47 No 12, IEEE, 2002, p.1974.

[60] Pioneer 3-AT Web Site, ActivMedia Robotics `http://www.activrobots.com/ROBOTS/p2at.html`, visited August 2004.

[61] Constellation 3Di Web Site, `http://www.constellation3di.com`, visited August 2004.

[62] R D'Andrea and G E Dullerud, "Distributed Control Design for Spatially Interconnected Systems," IEEE Transactions on Automatic Control, Vol 48 No 9, September 2003, IEEE, 1478, 2003.

[63] L. Chisci and G. Zappa, "Feasibility in Predictive Control of Constrained Linear Systems: the Output Feedback Case," *International Jounral of Robust and Nonlinear Control,* Vol. 12, Wiley, 2002, pp. 465–487.

[64] J. E. Tierno and A Khalak, "Frequency Domain Control Synthesis for Discounted Markov Decision Processes," European Control Conference, 2003.

[65] D. Lawden, *Optimal Trajectories for Space Navigation,* Butterworths, London, 1963.